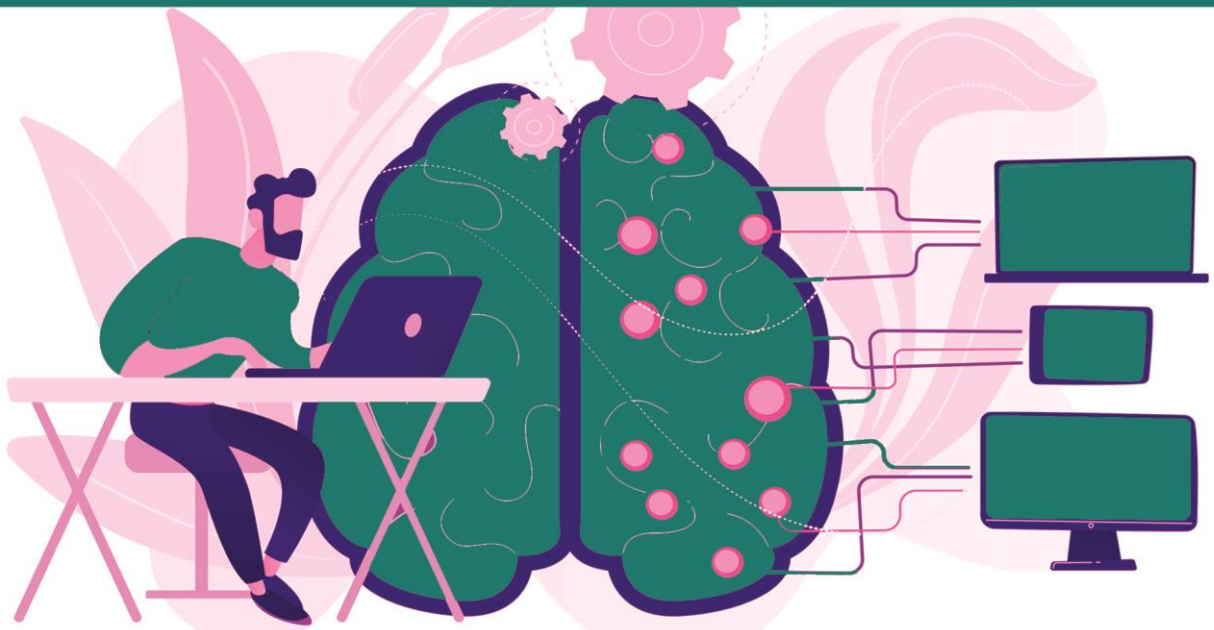
التعمق في التعلم العميق

الجزء الثالث: قابلية التوسعة والكفاءة والتطبيقات

تأليف : أنتون زانغ وآخرون

ترجمة : د. علاء طعيمة



بـــمه تعالى

التعمق فى التعلم العميق

الجزء الثالث

قابلية التوــع والكفاءة والتطبيقات

تأليف:

آــتون زانغ وآخرون

ترجمة:

د. علاء طعيمة

مقدمة المترجم

على مدى السنوات القليلة الماضية، طور فريق من علماء أمازون كتاباً " Dive into Deep Learning" يكتسب شعبية بين الطلاب والمطورين الذين ينجذبون إلى مجال التعلم العميق المزدهر، وهو مجموعة فرعية من التعلم الآلي تركز على الشبكات العصبية الاصطناعية واسعة النطاق.

عند انتهائي من قراءه هذا الكتاب، احببت ان اترجم هذا الكتاب وأشارككم هذه الترجمة لان هناك عدد من الأشياء الرائعة حول هذا الكتاب وأكثر ما يعجبني هو أنه يغطي كل مجالات التعلم العميق تقريبًا مثلاً للمبتدئين هناك فصول مثل الشبكات العصبية والبيرسيبترون متعدد الطبقات والانحدار والتصنيف بالإضافة الى المفاهيم الأساسية من الجبر الخطي، وحساب التفاضل والتكامل، والاحتمال الى فصول متقدمة مثل الشبكات العصبية الالتفافية CNN، تم تضمين الشبكات العصبية المتكررة RNN والرؤية الحاسوبية CV ومعالجة اللغات الطبيعية NLP أيضًا.

هذا كتاب تفاعلي مفتوح المصدر مقدمفي شكل فريد يدمج النص والرياضيات والكود، ويدعم الآن أطر برمجة TensorFlow وPyTorch وApache MXNet، والتي تمت صياغتها بالكامل من خلال Jupyter Notebook.

يمكن تقسيم الكتاب إلى ثلاثة أجزاء تقريبًا، لقد قمنافي الوقت الحالي بترجمة الجزء الأول والذي يشمل الأساسيات والمقدمات والجزء الثاني والذي يشمل التقنيات الحديثة للتعلم العميق والان نقدم لكم الجزء الثالث والذي يشمل مواضيع مثل الرؤية الحاسوبية والمعالجة اللغوية الطبيعية.

لقد اخترت كتاب "Dive into Deep Learning" لما رأيته من جودة هذا الكتاب، وللمنهجية التي اتبعها المؤلفونفي ترتيبه وبساطة شرحه. لقد حاولت قدر المستطاع ان اخرج بترجمة ذات جودة عالية، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذاكان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجال التعلم العميق ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال الذكاء الاصطناعي وتعلم الالة والتعلم العميق. ونرجو لك الاستمتاع مع التعلم العميق ولا تنسونا من صالح الدعاء.

د. علاء طعيمة/كلية علوم الحاسوب وتكنولوجيا المعلومات/جامعة القادسية/العراق

المحتويات

12 خوارزميات التحسين

12. خوارزميات التحسين Optimization Algorithms

إذا قرأت الكتاب بالتسلسل حتى هذه النقطة، فقد استخدمت بالفعل عددًا من خوارزميات التحسين optimization algorithms لتدريب نماذج التعلم العميق. لقد كانت الأدوات التي سمحت لنا بمواصلة تحديث معلمات النموذج وتقليل قيمة دالة الخطأ، كما تم تقييمهافي مجموعة التدريب.في الواقع، يمكن لأي شخص يتعامل مع التحسين كجهاز صندوق أسود لتقليل minimize دوال الهدف objective functionsفي إعداد بسيط أن يقنع نفسه بمعرفة أن هناك مجموعة من التعويذات لمثل هذا الإجراء (بأسماء مثل "SGD" و "Adam").

للقيام بعمل جيد، مع ذلك، هناك حاجة إلى بعض المعرفة الأعمق. تعد خوارزميات التحسين مهمة للتعلم العميق. من ناحية أخرى، قد يستغرق تدريب نموذج التعلم العميق المعقد ساعات أو أيامًا أو حتى أسابيع. يؤثر أداء خوارزمية التحسين بشكل مباشر على كفاءة تدريب النموذج. من ناحية أخرى، فإن فهم مبادئ خوارزميات التحسين المختلفة ودور معلماتها الفائقة hyperparameters سيمكننا من ضبط المعلمات الفائقة بطريقة مستهدفة لتحسين أداء نماذج التعلم العميق.

في هذا الفصل، نستكشف بعمق خوارزميات تحسين التعلم العميق الشائعة. تقريبًا جميع مشكلات التحسين التي تنشأفي التعلم العميق هي مشكلات غير محدبة nonconvex. ومع ذلك، فقد ثبت أن تصميم وتحليل الخوارزمياتفي سياق المشكلات المحدبة convex problems مفيد للغاية. ولهذا السبب يتضمن هذا الفصل كتابًا تمهيديًا عن التحسين المحدب convex optimization وإثبات خوارزمية التدرج الاشتقاقي العشوائي stochastic gradient descent البسيطة جدًا على دالة هدف محدبة convex objective function.

12.1. التحسين والتعلم العميق Optimization and Deep Learning

في هذا القسم، سنناقش العلاقة بين التحسين والتعلم العميق بالإضافة إلى تحديات استخدام التحسينفي التعلم العميق. بالنسبة لمشكلة التعلم العميق، سنحدد عادةً دالة الخطأ loss function أولاً. بمجرد أن نحصل على دالة الخطأ، يمكننا استخدام خوارزمية تحسينفي محاولة لتقليل الخطأ.في التحسين، غالبًا ما يشار إلى دالة الخطأ على أنها دالة الهدف لمشكلة التحسين. حسب التقاليد والأعراف، تهتم معظم خوارزميات التحسين بالتقليل minimization. إذا احتجنافي أي وقت إلى تعظيم maximize هدف، فهناك حل بسيط: فقط اقلب العلامة على الهدف.

12.1.1. هدف التحسين Goal of Optimization

على الرغم من أن التحسين يوفر طريقة لتقليل دالة الخطأ للتعلم العميق، إلا أن أهداف التحسين والتعلم العميق تختلف اختلافًا جوهريًا. الأول يهتمفي المقام الأول بتقليل الهدف بينما يهتم الأخير بإيجاد نموذج مناسب، بالنظر إلى كمية محدودة من البيانات.في القسم 3.6، ناقشنا الفرق بين هذين الهدفين بالتفصيل. على سبيل المثال، يختلف خطأ التدريب training error وخطأ التعميم generalization error بشكل عام: نظرًا لأن دالة الهدف لخوارزمية التحسين هي عادةً دالة خطأ (خسارة) تعتمد على مجموعة بيانات التدريب، فإن الهدف من التحسين هو تقليل خطأ التدريب. ومع ذلك، فإن الهدف من التعلم العميق (أو على نطاق أوسع، الاستدلال الإحصائي statistical inference) هو تقليل خطأ التعميم. لإنجاز هذا الأخير، نحتاج إلى الانتباه إلى الضبط الزائد overfitting بالإضافة إلى استخدام خوارزمية التحسين لتقليل خطأ التدريب.

```
%matplotlib inline
import numpy as np
import tensorflow as tf
from mpl_toolkits import mplot3d
from d2l import tensorflow as d2l
```

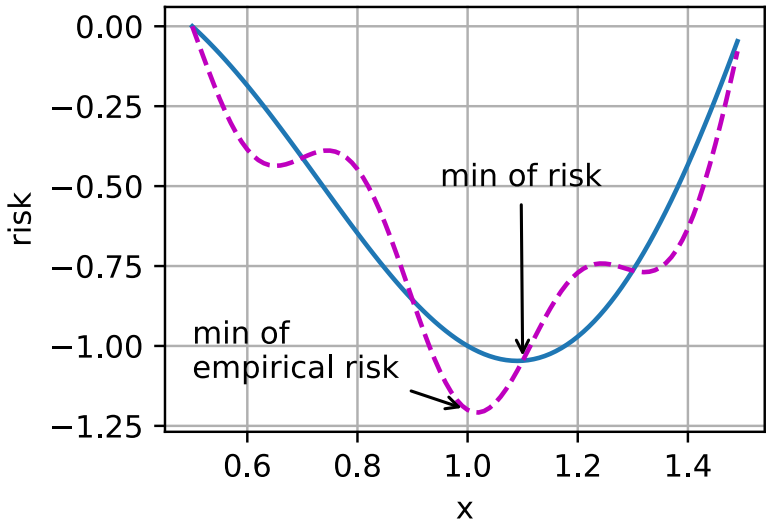
لتوضيح الأهداف المختلفة المذكورة أعلاه، دعونا ننظرفي المخاطر التجريبية empirical risk والمخاطر risk. كما هو موضحفي القسم 4.7.3.1، فإن الخطر التجريبي هو متوسط الخطأ في مجموعة بيانات التدريب بينما الخطر هو الخطأ المتوقع على مجموعة البيانات بأكملها. نحدد أدناه دالتين: دالة المخاطرة `f` ودالة المخاطرة التجريبية `g`. افترض أن لدينا كمية محدودة فقط من بيانات التدريب. نتيجة لذلك، هنا `g` أقل سلاسة من `f`.

```
def f(x):
    return x * tf.cos(np.pi * x)

def g(x):
    return f(x) + 0.2 * tf.cos(5 * np.pi * x)
```

يوضح الرسم البياني أدناه أن الحد الأدنى من المخاطر التجريبية على مجموعة بيانات التدريب قد يكونفي موقع مختلف عن الحد الأدنى من المخاطر (خطأ التعميم).

```
def annotate(text, xy, xytext):  #@save
    d2l.plt.gca().annotate(text, xy=xy, xytext=xytext,
                           arrowprops=dict(arrowstyle='-
>'))

x = tf.range(0.5, 1.5, 0.01)
d2l.set_figsize((4.5, 2.5))
```

```
d2l.plot(x, [f(x), g(x)], 'x', 'risk')
annotate('min of\nempirical risk', (1.0, -1.2), (0.5, -
1.1))
annotate('min of risk', (1.1, -1.05), (0.95, -0.5))
```



12.1.2. تحديات التحسين في التعلم العميق Optimization Challenges in Deep Learning

في هذا الفصل، سنركز بشكل خاص على أداء خوارزميات التحسينفي تقليل دالة الهدف بدلاً من خطأ التعميم الخاص بالنموذج.في القسم 3.1 ميزنا بين الحلول التحليلية والحلول العدديةفي مشاكل التحسين.في التعلم العميق، تكون معظم دوال الهدف معقدة ولا تحتوي على حلول تحليلية. بدلاً من ذلك، يجب أن نستخدم خوارزميات التحسين العددي. تقع جميع خوارزميات التحسينفي هذا الفصل ضمن هذه الفئة.

هناك العديد من التحدياتفي عملية تحسين التعلم العميق. بعض من أكثرها إزعاجًا هي الحدود الدنيا المحلية local minima، ونقاط السرج saddle points، والتدرجات المتلاشية vanishing gradients. دعونا نلقي نظرة عليهم.

12.1.2.1. الحد الأدنى المحلي Local Minima

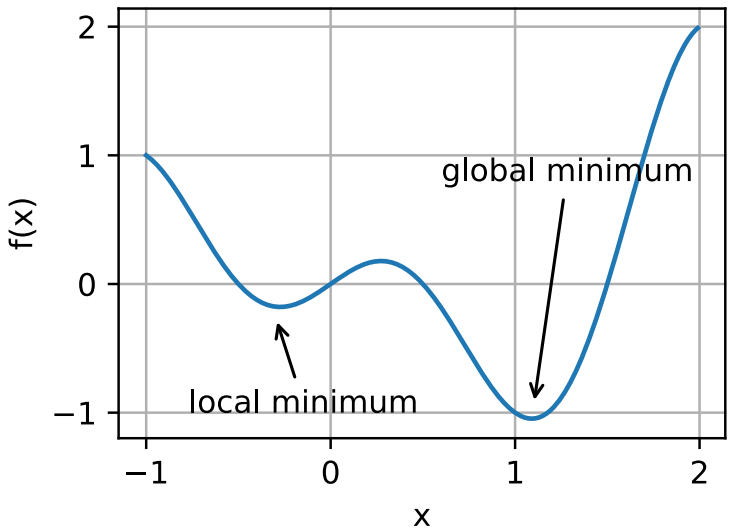
لأي دالة هدف $f(x)$ ، إذا كانت قيمة $f(x)$ عند x أصغر من قيم $f(x)$ عند أي نقاط أخرى في المنطقة المجاورة لـ x ، فيمكن أن تكون قيمة صغرى محلية local minimum. إذا كانت قيمة $f(x)$ عند x هي الحد الأدنى لدالة الهدف على النطاق بأكمله، فستكون $f(x)$ هي الحد الأدنى العالمي global minimum.

على سبيل المثال، بالنظر إلى الدالة:

$$f(x) = x \cdot \cos(\pi x) \text{ for } -1.0 \leq x \leq 2.0,$$

يمكننا تقريب الحد الأدنى المحلي والعالمي لهذه الدالة.

```
x = tf.range(-1.0, 2.0, 0.01)
d2l.plot(x, [f(x), ], 'x', 'f(x)')
annotate('local minimum', (-0.3, -0.25), (-0.77, -1.0))
annotate('global minimum', (1.1, -0.95), (0.6, 0.8))
```



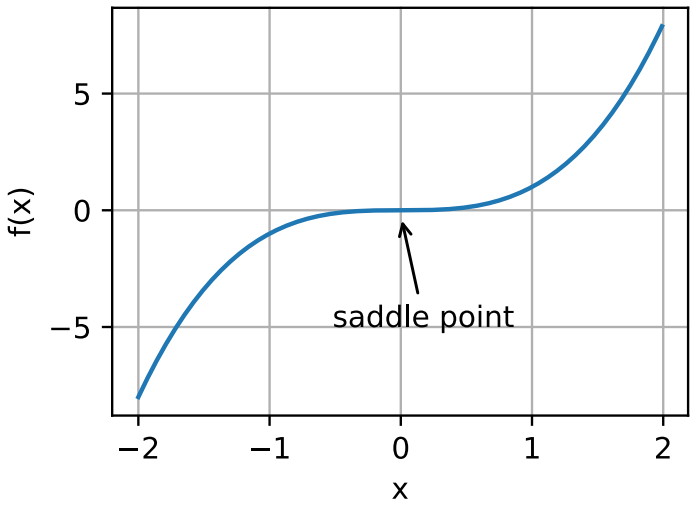
عادة ما يكون لدالة الهدف لنماذج التعلم العميق العديد من الخيارات المحلية local optima. عندما يكون الحل العددي لمشكلة التحسين قريبًا من المستوى المحلي الأمثل local optimum، فإن الحل العددي الذي تم الحصول عليه من خلال التكرار النهائي قد يقلل فقط دالة الهدف محليًا locally، وليس عالميًا globally، حيث يقترب تدرج حلول دالة الهدف أو يصبح صفرًا. قد تؤدي درجة معينة من الضوضاء فقط إلى إخراج المعلمة من الحد الأدنى المحلي.في الواقع، هذه هي إحدى الخصائص المفيدة للتدرج الاشتقاقي العشوائي المصغر minibatch stochastic gradient descent حيث يكون التباين الطبيعي للتدرجات على الدفعات الصغيرة minibatch قادرًا على إزاحة المعلمات من الحدود الدنيا المحلية.

12.1.2.2. نقاط السرج Saddle Points

إلى جانب الحدود الدنيا المحلية، تعتبر نقاط السرج saddle points سببًا آخر لتلاشي التدرجات. نقطة السرج هي أي مكان تتلاشى فيه جميع تدرجات الدالة ولكنه ليس حدًا أدنى

عالميًا ولا محليًا. ضعفي اعتبارك الدالة $f(x) = x^3$. مشتقها الأول والثاني يختفي لـ $x = 0$. قد يتوقف التحسينفي هذه المرحلة، على الرغم من أنه ليس بالحد الأدنى.

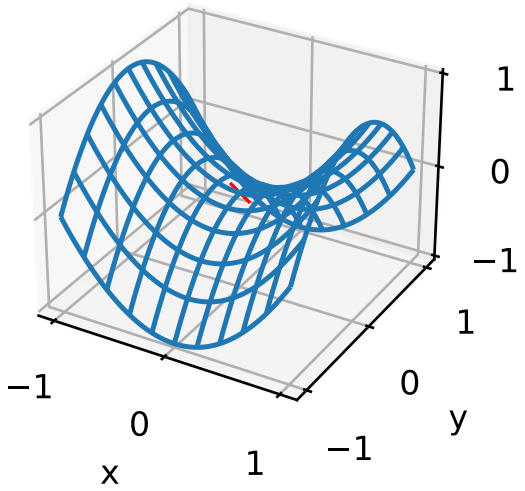
```
x = tf.range(-2.0, 2.0, 0.01)
d2l.plot(x, [x**3], 'x', 'f(x)')
annotate('saddle point', (0, -0.2), (-0.52, -5.0))
```



نقاط السرجفي الأبعاد الأعلى هي أكثر مكرًا، كما يوضح المثال أدناه. ضعفي اعتبارك الدالة $f(x,y) = x^2 - y^2$. لها نقطة السرجفي $(0,0)$. هذا هو الحد الأقصى فيما يتعلق بـ y والحد الأدنى فيما يتعلق بـ x. علاوة على ذلك، يبدو وكأنه سرج saddle، حيث حصلت هذه الخاصية الرياضية على اسمها.

```
x, y = tf.meshgrid(
    tf.linspace(-1.0, 1.0, 101), tf.linspace(-1.0, 1.0,
101))
z = x**2 - y**2

ax = d2l.plt.figure().add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z, **{'rstride': 10, 'cstride':
10})
ax.plot([0], [0], [0], 'rx')
ticks = [-1, 0, 1]
d2l.plt.xticks(ticks)
d2l.plt.yticks(ticks)
ax.set_zticks(ticks)
```

```
d2l.plt.xlabel('x')
d2l.plt.ylabel('y');
```



نحن نفترض أن مدخلات الدالة عبارة عن متجه ذي أبعاد k وأن ناتجها هو قيمة قياسية scalar، لذا فإن المصفوفة الهيسية Hessian سيكون لها k قيم ذاتية eigenvalues. يمكن أن يكون حل الدالة هو الحد الأدنى المحلي أو الحد الأقصى المحلي أو نقطة السرجفي موضع يكون فيه تدرج الدالة صفرًا:

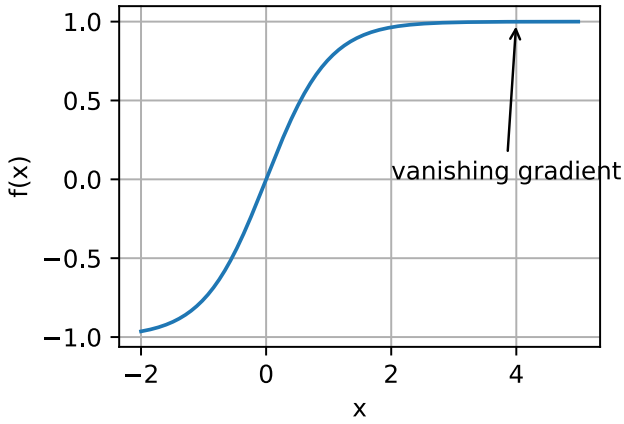
- عندما تكون القيم الذاتية للمصفوفة الهيسية للدالة عند موضع التدرج الصفري موجبة، يكون لدينا حد أدنى محلي local minimum للدالة.
- عندما تكون القيم الذاتية للمصفوفة الهيسية للدالة عند موضع التدرج الصفري كلها سالبة، يكون لدينا حد أقصى محلي local maximum للدالة.
- عندما تكون القيم الذاتية للمصفوفة الهيسية للدالة عند موضع التدرج الصفري سالبة وإيجابية، يكون لدينا نقطة سرج للدالة.

بالنسبة للمشكلات عالية الأبعاد، يكون احتمال أن تكون بعض القيم الذاتية سالبة على الأقل مرتفعًا جدًا. هذا يجعل نقاط السرج أكثر احتمالا من الحدود الدنيا المحلية. سنناقش بعض الاستثناءات من هذا الموقفي القسم التالي عند تقديم التحدب convexity. باختصار، الدوال المحدبة هي تلك التي لا تكون فيها القيم الذاتية لـ Hessian سلبية أبدًا. للأسف، على الرغم من ذلك، لا تقع معظم مشاكل التعلم العميقة ضمن هذه الفئة. ومع ذلك فهي أداة رائعة لدراسة خوارزميات التحسين.

12.1.2.3. تلاشي التدرجات Vanishing Gradients

ربما تكون المشكلة الأكثر مكراً التي يجب مواجهتها هي تلاشي التدرج vanishing gradient. تذكر دوال التنشيط شائعة الاستخدام ومشتقاتهافي القسم 5.1.2. على سبيل المثال، افترض أننا نريد تقليل الدالة $f(x) = \tanh(x)$ ويصادف أننا نبدأ من $x = 4$. كما نرى f، فإن انحدار f قريب من الصفر. بشكل أكثر تحديداً،$f'(x) = 1 - \tanh^2(x)$ وبالتالي $f'(4) =$ 0.0013. وبالتالي، سيتعطل التحسين لفترة طويلة قبل أن نحرز تقدمًا. تبين أن هذا هو أحد الأسباب التي جعلت تدريب نماذج التعلم العميق أمرًا صعبًا للغاية قبل إدخال دالة تنشيط ReLU.

```
x = tf.range(-2.0, 5.0, 0.01)
d2l.plot(x, [tf.tanh(x)], 'x', 'f(x)')
annotate('vanishing gradient', (4, 1), (2, 0.0))
```



كما رأينا، فإن التحسين من أجل التعلم العميق مليء بالتحديات. لحسن الحظ، توجد مجموعة قوية من الخوارزميات التي تعمل بشكل جيد والتي يسهل استخدامها حتى للمبتدئين. علاوة على ذلك، ليس من الضروري حقًا إيجاد الحل الأفضل. لا تزال الحلول المثلى المحلية Local optima أو حتى الحلول التقريبية approximate solutions مفيدة للغاية.

12.1.3. الملخص

- لا يضمن تقليل خطأ التدريب أننا نجد أفضل مجموعة من المعلمات لتقليل خطأ التعميم.
- قد يكون لمشاكل التحسين العديد من الحدود الدنيا المحلية.
- قد تحتوي المشكلة على المزيد من نقاط السرج، حيث إن المشكلات بشكل عام ليست محدبة.

- يمكن أن يؤدي تلاشي التدرجات إلى توقف التحسين.في كثير من الأحيان، تساعد معالجة المشكلة. يمكن أن تكون التهيئة الجيدة للمعلمات مفيدة أيضًا.

12.1.4. التمارين

1. ضعفي اعتبارك MLP بسيطًا مع طبقة مخفية واحدة، على سبيل المثال، أبعاد dفي الطبقة المخفية ومخرج واحد. أظهر أنه بالنسبة لأي حد أدنى محلي، هناك على الأقل $d!$حلول مكافئة تتصرف بشكل متماثل.
2. افترض أن لدينا مصفوفة عشوائية متماثلة M حيث $M_{ij} = M_{ji}$ يتم رسم كل مدخلات من بعض توزيعات الاحتمالات p_{ij}. علاوة على ذلك، افترض أن $p_{ij}(x) = p_{ij}(-x)$ توزيع متماثل symmetric (انظر على سبيل المثال، Wigner (1958) للحصول على التفاصيل).
   1. اثبت أن التوزيع على القيم الذاتية eigenvalues متماثل أيضًا. وهذا يعني، بالنسبة لأي متجه ذاتي (eigenvector) v، فإن احتمال أن قيمة λ تحقق $P(\lambda > 0) = P(\lambda < 0)$.
   2. لماذا ما سبق لا يشير إلى $P(\lambda > 0) = 0.5$؟
3. ما هي التحديات الأخرى التي ينطوي عليها تحسين التعلم العميق التي يمكنك التفكير فيها؟
4. افترض أنك تريد موازنة balance كرة (حقيقية) على سرج (حقيقي).
   1. لماذا هذا صعب؟
   2. هل يمكنك استغلال هذا التأثير أيضًافي خوارزميات التحسين؟

12.2. التحدب Convexity

يلعب التحدب Convexity دورًا حيويًافي تصميم خوارزميات التحسين. هذا يرجع إلى حد كبير إلى حقيقة أنه من الأسهل بكثير تحليل واختبار الخوارزمياتفي مثل هذا السياق. بعبارة أخرى، إذا كان أداء الخوارزمية ضعيفًا حتىفي الإعداد المحدب، فعادةً لا يجب أن نأملفي رؤية نتائج رائعة بخلاف ذلك. علاوة على ذلك، على الرغم من أن مشاكل التحسينفي التعلم العميق غير متشابكة بشكل عام، إلا أنها غالبًا ما تعرض بعض خصائص تلك المحدبة بالقرب من الحدود الدنيا المحلية. يمكن أن يؤدي ذلك إلى متغيرات تحسين جديدة ومثيرة مثل (Izmailov et al.، 2018).

```
%matplotlib inline
import numpy as np
import tensorflow as tf
from mpl_toolkits import mplot3d
```

```
from d2l import tensorflow as d2l
```

12.2.1. تعريفات Definitions

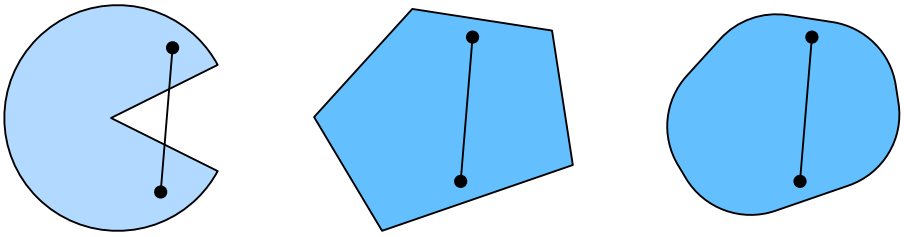
قبل التحليل المحدب convex analysis، نحتاج إلى تحديد المجموعات المحدبة والدوال المحدبة. إنها تؤدي إلى أدوات رياضية يتم تطبيقها بشكل شائع في التعلم الآلي.

12.2.1.1. مجموعات محدبة Convex Sets

المجموعات Sets هي أساس التحدب. ببساطة، المجموعة $\mathcal{X}$ في مساحة متجه محدبة إذا كانت لأي $a, b \in \mathcal{X}$ قطعة خط line segments متصلة a و b هي أيضًا موجودة. من الناحية الرياضية، هذا يعني ذلك لكل $\lambda \in [0,1]$ لدينا:

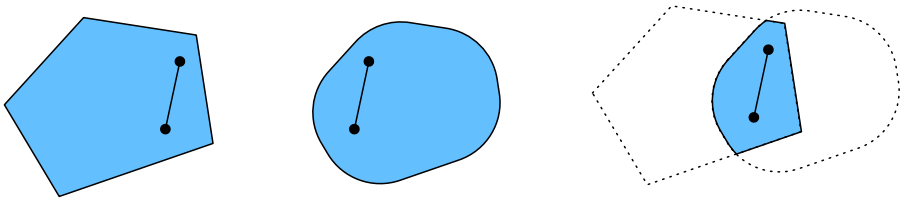
$$\lambda a + (1 - \lambda)b \in \mathcal{X} \text{ whenever } a, b \in \mathcal{X}.$$

هذا يبدو مجردا abstract بعض الشيء. لننظر في الشكل 12.2.1. المجموعة الأولى ليست محدبة nonconvex نظرًا لوجود مقاطع خطية غير متضمنة فيها. المجموعتان الأخريان لا تعانيان من مثل هذه المشكلة.



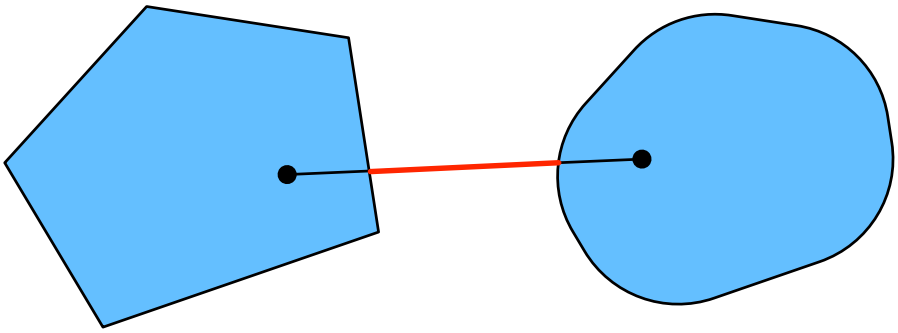
الشكل 12.2.1 المجموعة الأولى غير محدبة والمجموعات الأخرى محدبة.

التعريفات بحد ذاتها ليست مفيدة بشكل خاص إلا إذا كان بإمكانك فعل شيء بها. في هذه الحالة يمكننا النظر إلى التقاطعات كما هو موضح في الشكل 12.2.2. افترض أن $\mathcal{X}$ و $\mathcal{Y}$ مجموعات محدبة. ثم $\mathcal{X} \cap \mathcal{Y}$ هو أيضا محدب. لرؤية هذا، اعتبر كل $a, b \in \mathcal{X} \cap \mathcal{Y}$. نظرًا لكون $\mathcal{X}$ و $\mathcal{Y}$ محدبة، فإن مقاطع الخط a و b متصلة ومضمنة في كل من $\mathcal{X}$ و $\mathcal{Y}$. بالنظر إلى ذلك، يجب أيضًا احتواؤها في $\mathcal{X} \cap \mathcal{Y}$ ، وبالتالي إثبات نظريتنا.



الشكل 12.2.2 التقاطع بين مجموعتين محدبتين محدب.

يمكننا تقوية هذه النتيجة بجهد قليل: نظرًا للمجموعات المحدبة $\mathcal{X}_i$، يكون تقاطعها $\cap_i \mathcal{X}_i$ محدبًا. لترى أن العكس ليس صحيحًا، ضعفي اعتبارك مجموعتين منفصلتين $\mathcal{X} \cap \mathcal{Y} = \emptyset$. الآن اختر $a \in \mathcal{X}$ و$b \in \mathcal{Y}$. يجب أن يحتوي الجزء المستقيمفي الشكل 12.2.3 الذي يربط a وb على جزء ليسفي $\mathcal{X}$ ولافي $\mathcal{Y}$ ،، لأننا افترضنا $\mathcal{X} \cap \mathcal{Y} = \emptyset$. ومن ثم، فإن القطعة المستقيمة ليستفي $\mathcal{X} \cup \mathcal{Y}$ أيضًا، مما يثبت أنهفي الاتحادات العامة للمجموعات المحدبة لا يلزم أن تكون محدبة.



الشكل 12.2.3 لا يلزم أن يكون اتحاد مجموعتين محدبتين محدَّب.

عادةً ما يتم تحديد المشكلاتفي التعلم العميقفي مجموعات محدبة. على سبيل المثال،$\mathbb{R}^d$ مجموعة المتجهات ذات الأبعاد d للأرقام الحقيقية، هي مجموعة محدبة (بعد كل شيء، الخط الفاصل بين أي نقطتينفي $\mathbb{R}^d$ يبقىفي $\mathbb{R}^d$).في بعض الحالات، نتعامل مع متغيرات ذات طول محدد، مثل كرات نصف القطر r كما هو محدد بواسطة $\{\mathbf{x}|\mathbf{x} \in \mathbb{R}^d \text{ and } \|\mathbf{x}\| \leq r\}$.

12.2.1.2. دوال محدبة Convex Functions

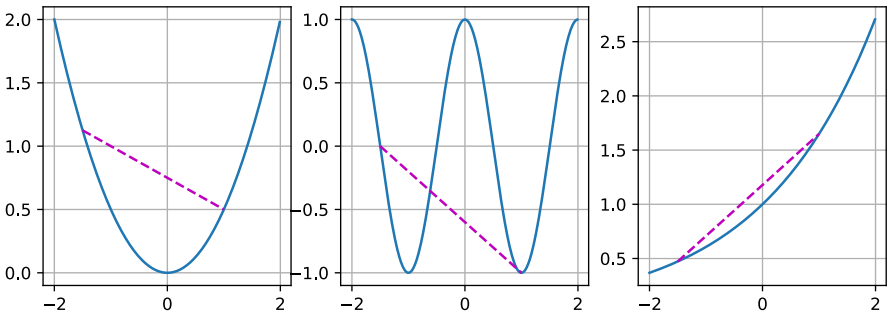
الآن بعد أن أصبح لدينا مجموعات محدبة، يمكننا تقديم دوال محدبة f. بالنظر إلى مجموعة محدبة $\mathcal{X}$، تكون الدالة $f: \mathcal{X} \to \mathbb{R}$ محدبة إذا كانت لكل $x, x' \in \mathcal{X}$ ولكل $\lambda \in [0,1]$ لدينا:

$$\lambda f(x) + (1-\lambda) f(x') \geq f(\lambda x + (1-\lambda)x').$$

لتوضيح ذلك، دعنا نرسم بعض الدوال ونتحقق من أي منها يلبي المتطلبات. نحدد أدناه بعض الدوال، المحدبة وغير المحدبة.

```
f = lambda x: 0.5 * x**2  # Convex
g = lambda x: tf.cos(np.pi * x)  # Nonconvex
h = lambda x: tf.exp(0.5 * x)  # Convex

x, segment = tf.range(-2, 2, 0.01), tf.constant([-1.5,
1])
d2l.use_svg_display()
```

```
_, axes = d2l.plt.subplots(1, 3, figsize=(9, 3))
for ax, func in zip(axes, [f, g, h]):
    d2l.plot([x, segment], [func(x), func(segment)],
axes=ax)
```



كما هو متوقع، فإن دالة جيب التمام هي غير محدبة nonconvex،في حين أن القطع المكافئ والدالة الأسية هما محدبتان. لاحظ أن المتطلبات التي تكون مجموعة $\mathcal{X}$ محدبة ضرورية لكي يكون الشرط منطقيًا. خلاف ذلك، قد لا يتم تحديد نتيجة $f(\lambda x + (1-\lambda)x')$ بشكل جيد.

12.2.1.3. متباينة ينسن Jensen's Inequality

بالنظر إلى دالة محدبة f، فإن إحدى أكثر الأدوات الرياضية فائدة هي متباينة ينسن Jensen's inequality. إنه يرقى إلى تعميم تعريف التحدب:

$$\sum_i \alpha_i f(x_i) \geq f(\sum_i \alpha_i x_i) \text{ and } E_X[f(X)] \geq f(E_X[X]),$$

حيث α_i هي أرقام حقيقية غير سالبة مثل $\sum_i \alpha_i = 1$ وX متغير عشوائي. وبعبارة أخرى، فإن توقع دالة محدبة لا يقل عن الدالة المحدبة للتوقع، حيث يكون الأخير عادة تعبيرًا أبسط. لإثبات المتباينة الأولى، طبقنا تعريف التحدب بشكل متكرر على مصطلح واحدفي المجموعفي المرة الواحدة.

أحد التطبيقات الشائعة لمتباينة ينسن هو ربط تعبير أكثر تعقيدًا بتعبير أبسط. على سبيل المثال، يمكن أن يكون تطبيقه فيما يتعلق باحتمالية log-likelihood المتغيرات العشوائية التي تمت ملاحظتها جزئيًا. هذا هو، نحن نستخدم:

$$E_{Y \sim P(Y)}[-\log P(X \mid Y)] \geq -\log P(X),$$

بسبب $\int P(Y)P(X \mid Y)dY = P(X)$. يمكن استخدام هذافي طرق التنويع variational methods. هنا Y هو عادةً المتغير العشوائي غير المرصود unobserved random

variable، $P(Y)$ هو أفضل تخمين لكيفية توزيعه، و $P(X)$ هو التوزيع مع الخارج المتكامل. على سبيل المثال،في التجميع (clustering) Y قد تكون تسميات المجموعة cluster labels و $P(X \mid Y)$ هو النموذج التوليدي عند تطبيق تسميات المجموعة.

12.2.2. الخصائص Properties

الدوال المحدبة لها العديد من الخصائص المفيدة. نصف عدد قليل منها شائع الاستخدام أدناه.

12.2.2.1. الحد الأدنى المحلي هو الحد الأدنى العالمي Local Minima Are Global Minima

أولاً وقبل كل شيء، تعتبر الحدود الدنيا المحلية لدوال المحدبة هي أيضًا الحدود الدنيا العالمية. يمكننا إثبات ذلك بالتناقض contradiction على النحو التالي.

ضعفي اعتبارك دالة محدبة f محددةفي مجموعة محدبة $\mathcal{X}$. افترض أن $x' \in \mathcal{X}$ هي حد أدنى محلي: توجد قيمة موجبة صغيرة p بحيث بالنسبة لـ $x \in \mathcal{X}$ التي تحقق $0 < |x - x^*| \leq p$ لدينا $f(x^*) < f(x)$.

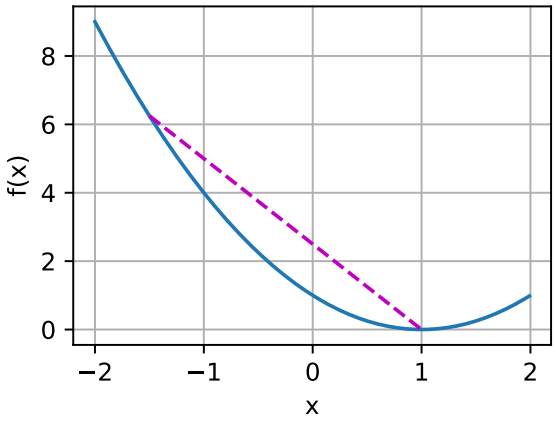
افترض أن الحد الأدنى المحلي x^* ليس هو الحد الأدنى العالمي لـ f: يوجد $x' \in \mathcal{X}$ لكل $f(x') < f(x^*)$. يوجد أيضًا $\lambda \in [0,1)$ مثل $\lambda = 1 - \frac{p}{|x^* - x'|}$ بحيث تكون $0 < |\lambda x^* + (1-\lambda)x' - x^*| \leq p$.

ومع ذلك، وفقًا لتعريف الدوال المحدبة، لدينا

$$\begin{aligned} f(\lambda x^* + (1-\lambda)x') &\leq \lambda f(x^*) + (1-\lambda) f(x') \\ &< \lambda f(x^*) + (1-\lambda) f(x^*) \\ &= f(x^*), \end{aligned}$$

وهو ما يتعارض مع بياننا بأن x^* هو الحد الأدنى المحلي. لذلك، لا يوجد $x' \in \mathcal{X}$ لـ $f(x') < f(x^*)$. الحد الأدنى المحلي x^* هو أيضًا الحد الأدنى العالمي.

```
f = lambda x: (x - 1) ** 2
d2l.set_figsize()
d2l.plot([x, segment], [f(x), f(segment)], 'x', 'f(x)')
```



حقيقة أن الحدود الدنيا المحلية للدوال المحدبة هي أيضًا الحدود الدنيا العالمية مريحة للغاية. هذا يعني أننا إذا قللنا من الدوال فلن نستطيع "أن نتعثر get stuck". لاحظ، مع ذلك، أن هذا لا يعني أنه لا يمكن أن يكون هناك أكثر من حد أدنى عالمي واحد أو أنه قد يوجد حد أدنى. على سبيل المثال، تصل الدالة $f(x) = \max(|x| - 1{,}0)$ إلى أدنى قيمة لها خلال الفترة $[-1{,}1]$. على العكس من ذلك، لا تحقق الدالة $f(x) = \exp(x)$ قيمة دنيا على: $\mathbb{R}$ لكل $x \to -\infty$ لها خطوط مقاربة إلى 0 ، ولكن لا يوجد x لـ $f(x) = 0$.

12.2.2.2. المجموعات التالية من دوال محدبة هي محدبة Below Sets of Convex Functions Are Convex

يمكننا تحديد المجموعات المحدبة بسهولة من خلال مجموعات الدوال المحدبة أدناه. بشكل ملموس، بالنظر إلى f دالة محدبة محددةفي مجموعة محدبة $\mathcal{X}$ ، أي المجموعة التالية below set:

$$\mathcal{S}_b := \{x | x \in \mathcal{X} \text{ and } f(x) \leq b\}$$

هي محدبة.

دعونا نثبت ذلك بسرعة. تذكر أنه بالنسبة لأي $x, x' \in \mathcal{S}_b$ ، نحتاج إلى إظهار أن $\lambda x + (1 - \lambda)x' \in \mathcal{S}_b$ طالما $\lambda \in [0,1]$. لان $f(x) \leq b$ و $f(x') \leq b$ ، من خلال تعريف التحدب convexity لدينا

$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda) f(x') \leq b.$$

12.2.2.3. التحدب والمشتقات الثانية Convexity and Second Derivatives

متى وجد المشتق الثاني للدالة $f: \mathbb{R}^n \to \mathbb{R}$ ، فمن السهل جدًا التحقق مما إذا كانت f محدبة. كل ما نحتاج إلى القيام به هو التحقق مما إذا كان هيسي Hessian لـ f موجبًا شبه محدد: $\nabla^2 f \geq 0$ ، أي الإشارة إلى المصفوفة الهيسية $\mathbf{x}^\top \mathbf{H}\mathbf{x} \geq 0$ بواسطة $\mathbf{x}^\top \mathbf{H}\mathbf{x} \geq 0$ ، لكل $\mathbf{x} \in \mathbb{R}^n$. على سبيل المثال، الدالة $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2$ محدبة لان $\nabla^2 f = \mathbf{1}$ ، أي Hessian هي مصفوفة الوحدة identity matrix.

بشكل رسمي، تكون الدالة $f: \mathbb{R} \to \mathbb{R}$ ذات البعد الواحد القابلة للتفاضل مرتين محدبة فقط إذا كان مشتقها الثاني $f'' \geq 0$. بالنسبة لأي دالة متعددة الأبعاد قابلة للتفاضل مرتين $f: \mathbb{R}^n \to \mathbb{R}$، تكون محدبة إذا وفقط إذا كانت $\nabla^2 f \geq 0$ هيسية.

أولًا، علينا إثبات الحالة أحادية البعد. لرؤية هذا التحدب لـ f تشير الى $f'' \geq 0$ أننا نستخدم حقيقة ان:

$$\frac{1}{2}f(x+\epsilon) + \frac{1}{2}f(x-\epsilon) \geq f\left(\frac{x+\epsilon}{2} + \frac{x-\epsilon}{2}\right) = f(x).$$

بما أن المشتق الثاني يُعطى بالغاية limit على الفروق المحدودة فإنه يتبع :

$$f''(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) + f(x-\epsilon) - 2f(x)}{\epsilon^2} \geq 0.$$

لنرى أن $f'' \geq 0$ تشير إلى أن f محدب، فإننا نستخدم حقيقة أن $f'' \geq 0$ تشير إلى أن f' هي دالة رتيبة لا تنقص. لنفترض أن $a < x < b$ تساوي ثلاث نقاطفي $\mathbb{R}$ ، حيث $x = (1-\lambda)a + \lambda b$ و $\lambda \in (0,1)$. وفقًا لنظرية القيمة المتوسطة mean value theorem، يوجد $\alpha \in [a,x]$ و $\beta \in [x,b]$ مثل:

$$f'(\alpha) = \frac{f(x) - f(a)}{x - a} \text{ and } f'(\beta) = \frac{f(b) - f(x)}{b - x}.$$

عن طريق رتابة $f'(\beta) \geq f'(\alpha)$ ، وبالتالي

$$\frac{x-a}{b-a}f(b) + \frac{b-x}{b-a}f(a) \geq f(x).$$

لان $x = (1-\lambda)a + \lambda b$ ، لدينا

$$\lambda f(b) + (1-\lambda)f(a) \geq f((1-\lambda)a + \lambda b),$$

مما يثبت التحدب.

ثانيًا، نحتاج إلى lemma قبل إثبات الحالة متعددة الأبعاد: $f: \mathbb{R}^n \to \mathbb{R}$ محدب إذا وفقط إذا كان لكل $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$g(z) \overset{\text{def}}{=} f(z\mathbf{x} + (1-z)\mathbf{y}) \text{ where } z \in [0,1]$$

هو محدب.

لإثبات أن تحدب f يشير الى g انها محدبة، يمكننا إظهار ذلك لكل $a, b, \lambda \in [0,1]$ (وبالتالي $0 \leq \lambda a + (1-\lambda)b \leq 1$)

$$\begin{aligned} & g(\lambda a + (1-\lambda)b) \\ = \; & f((\lambda a + (1-\lambda)b)\mathbf{x} + (1 - \lambda a - (1-\lambda)b)\mathbf{y}) \\ = \; & f(\lambda(a\mathbf{x} + (1-a)\mathbf{y}) + (1-\lambda)(b\mathbf{x} + (1-b)\mathbf{y})) \\ \leq \; & \lambda f(a\mathbf{x} + (1-a)\mathbf{y}) + (1-\lambda) f(b\mathbf{x} + (1-b)\mathbf{y}) \\ = \; & \lambda g(a) + (1-\lambda) g(b). \end{aligned}$$

لإثبات العكس، يمكننا إظهار ذلك لكل $\lambda \in [0,1]$

$$\begin{aligned} & f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \\ = \; & g(\lambda \cdot 1 + (1-\lambda) \cdot 0) \\ \leq \; & \lambda g(1) + (1-\lambda) g(0) \\ = \; & \lambda f(\mathbf{x}) + (1-\lambda) g(\mathbf{y}). \end{aligned}$$

أخيرًا، باستخدام lemma أعلاه ونتيجة الحالة أحادية البعد، يمكن إثبات الحالة متعددة الأبعاد على النحو التالي. تكون الدالة $f: \mathbb{R}^n \to \mathbb{R}$ متعددة الأبعاد محدبة إذا وفقط إذا لكل $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $g(z) \overset{\text{def}}{=} f(z\mathbf{x} + (1-z)\mathbf{y})$ حيث $z \in [0,1]$ محدب. وفقًا للحالة أحادية البعد، فإن هذا ينطبق فقط إذا $g'' = (\mathbf{x}-\mathbf{y})^\top \mathbf{H}(\mathbf{x}-\mathbf{y}) \geq 0$ لكل $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ وهو ما يعادل $\mathbf{H} \succeq 0$ حسب تعريف المصفوفات الموجبة شبه المحددة.

12.2.3. القيود Constraints

إحدى الخصائص الرائعة للتحسين المحدب هي أنه يتيح لنا التعامل مع القيود constraints بكفاءة. أي أنه يسمح لنا بحل مشاكل التحسين المقيدة constrained optimization problems بالشكل:

$$\begin{aligned} \text{minimize} \quad & \underset{\mathbf{x}}{f(\mathbf{x})} \\ \text{subject to} \quad & c_i(\mathbf{x}) \leq 0 \text{ for all } i \in \{1, \ldots, n\}, \end{aligned}$$

حيث f هو الهدف والدوال c_i هي دوال قيد constraint functions. لمعرفة ما هذا، يجب مراعاة الحالة التي يكون فيها $c_1(\mathbf{x}) = \|\mathbf{x}\|_2 - 1$. في هذه الحالة، تكون المعلمات $\mathbf{x}$ مقيدة

بوحدة الكرة. إذا كان القيد الثاني هو $c_2(\mathbf{x}) = \mathbf{v}^\top \mathbf{x} + b$ ، فهذا يتوافق مع جميع $\mathbf{x}$ تقع على نصف مساحة. تحقيق كلا الشرطينفي وقت واحد يعني اختيار قطعة من الكرة.

12.2.3.1. لاغرانج Lagrangian

بشكل عام، يعد حل مشكلة التحسين المقيدة أمرًا صعبًا. تنبع إحدى طرق معالجتها من الفيزياء بحدس بسيط نوعًا ما. تخيل كرة داخل صندوق. سوف تتدحرج الكرة إلى المكان الأدنى وسيتم موازنة قوى الجاذبية بالقوى التي يمكن أن تفرضها جوانب الصندوق على الكرة. وباختصار، فإن الانحدار gradient لدالة الهدف (أي الجاذبية) سوف يقابله انحدار دالة القيد (يجب أن تظل الكرة داخل الصندوق بحكم الجدران "التي تدفع للخلف"). لاحظ أن بعض القيود قد لا تكون نشطة: الجدران التي لا تلمسها الكرة لن تكون قادرة على ممارسة أي قوة على الكرة.

تخطي اشتقاق لاغرانج L، يمكن التعبير عن المنطق أعلاه من خلال مشكلة تحسين نقطة السرج saddle point optimization problem التالية:

$$L(\mathbf{x}, \alpha_1, \ldots, \alpha_n) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i c_i(\mathbf{x}) \text{ where } \alpha_i \geq 0.$$

هنا المتغيرات $\alpha_i (i = 1, \ldots, n)$ هي ما يسمى بمضاعفات لاغرانج Lagrange multipliers التي تضمن فرض القيود بشكل صحيح. يتم اختيارهم بحجم كبير بما يكفي لضمان $c_i(\mathbf{x}) \leq 0$ لكل i. على سبيل المثال، لكل $\mathbf{x}$ $c_i(\mathbf{x}) < 0$ طبيعي، سننتهي باختيار $\alpha_i = 0$. علاوة على ذلك، فهذه مشكلة تحسين نقطة السرج حيث يريد المرء تعظيم (maximize) L فيما يتعلق بكل α_i وتقليلها(minimize)في نفس الوقت فيما يتعلق بـ $\mathbf{x}$. هناك مجموعة غنية من الأدبيات التي تشرح كيفية الوصول إلى الدالة $L(\mathbf{x}, \alpha_1, \ldots, \alpha_n)$. من اجل اهدافنا، يكفي معرفة أن نقطة السرج هي المكان الذي يتم فيه حل مشكلة التحسين المقيدة الأصلية على النحو الأمثل.

12.2.3.2. العقوبات Penalties

طريقة واحدة لتلبية مشاكل التحسين المقيدة على الأقل تقريبًا approximately هي تكييف لاغرانج L. بدلاً من تحقيق $c_i(\mathbf{x}) \leq 0$ ، نضيف ببساطة إلى دالة الهدف $f(x)$. هذا يضمن عدم انتهاك القيود بشكل سيء للغاية.

في الواقع، لقد استخدمنا هذه الحيلة طوال الوقت. ضعفي اعتبارك تناقص الوزن weight decay في القسم 3.7. نضيف فيه $\frac{\lambda}{2}\|\mathbf{w}\|^2$ إلى دالة الهدف للتأكد من أنها لا تنمو بشكل كبير جدًا. من وجهة نظر التحسين المقيدة، يمكننا أن نرى أن هذا سيضمن $\|\mathbf{w}\|^2 - r^2 \leq 0$لبعض نصف القطر. يتيح لنا ضبط قيمة λ لتغيير حجم $\mathbf{w}$.

بشكل عام، تعد إضافة العقوبات penalties طريقة جيدة لضمان الرضا التقريبي للقيد approximate constraint satisfaction. من الناحية العملية، يتبين أن هذا أكثر قوة من الرضا الدقيق exact satisfaction. علاوة على ذلك، بالنسبة للمشكلات غير المحدبة، فإن العديد من الخصائص التي تجعل النهج الدقيق جذابًا للغايةفي الحالة المحدبة (على سبيل المثال، الأمثلية optimality) لم تعد صالحة.

12.2.3.3. الاسقاطات Projections

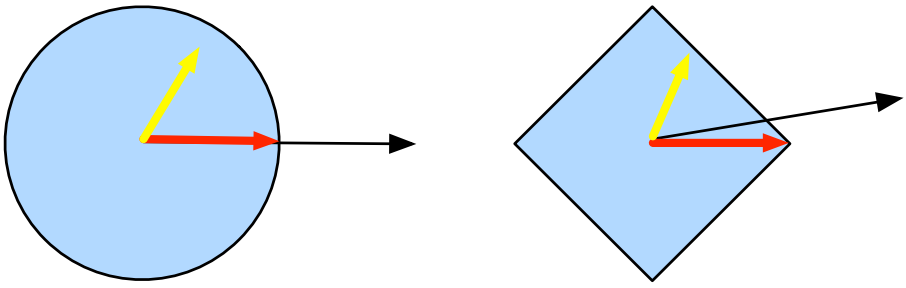
استراتيجية بديلة لتحقيق القيود هي الاسقاطات projections. مرة أخرى، واجهناها من قبل، على سبيل المثال، عند التعامل مع قص التدرج gradient clippingفي القسم 9.5. هناك تأكدنا من أن طول التدرج يحده θ عبر

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min(1, \theta/\|\mathbf{g}\|).$$

تبين أن هذا هو إسقاط $\mathbf{g}$ على كرة نصف قطرها θ. بشكل عام، يتم تعريف الإسقاط على مجموعة محدبة $\mathcal{X}$ على أنه

$$\text{Proj}_{\mathcal{X}}(\mathbf{x}) = \underset{\mathbf{x}' \in \mathcal{X}}{\text{argmin}} \ \|\mathbf{x} - \mathbf{x}'\|,$$

وهي أقرب نقطةفي $\mathcal{X}$ الى $\mathbf{x}$



الشكل 12.2.4 الإسقاطات المحدبة Convex Projections.

قد يبدو التعريف الرياضي للإسقاطات مجردة بعض الشيء. يوضح الشكل 12.2.4 ذلك إلى حد ما بشكل أكثر وضوحًا. لدينا مجموعتان محدبتان، دائرة circle وماسة diamond. تظل النقاط الموجودة داخل كلتا المجموعتين (الأصفر) دون تغيير أثناء الإسقاطات. يتم إسقاط النقاط الموجودة خارج كلتا المجموعتين (أسود) على النقاط الموجودة داخل المجموعات (باللون الأحمر) والتي تكون قريبة من النقاط الأصلية (سوداء). بينما يترك هذا الاتجاه للكرات ℓ_2 دون تغيير، لا يجب أن يكون هذا هو الحال بشكل عام، كما يمكن رؤيتهفي حالة الماسة.

أحد استخدامات الإسقاطات المحدبة هو حساب متجهات الوزن المتناثرة sparse weight vectors.في هذه الحالة نقوم بإسقاط متجهات الوزن على ℓ_1 كرة، وهي نسخة معممة من علبة الألماس في الشكل 12.2.4.

12.2.4. الملخص

في سياق التعلم العميق، الغرض الرئيسي من الدوال المحدبة هو تحفيز خوارزميات التحسين ومساعدتنا على فهمها بالتفصيل. فيما يلي سنرى كيف يمكن اشتقاق الانحدار التدريجي والانحدار التدريجي العشوائي وفقًا لذلك.

- تقاطعات المجموعات المحدبة محدبة. الاتحادات ليست كذلك.
- لا يقل توقع دالة محدبة عن دالة محدبة لتوقع (متباينة ينسن).
- تكون الدالة القابلة للتفاضل مرتين محدبة إذا وفقط إذا كانت Hessian (مصفوفة من المشتقات الثانية) موجبة شبه محددة.
- يمكن إضافة قيود محدبة عبر لاغرانج.في الممارسة العملية، يمكننا ببساطة إضافتها مع عقوبة إلى دالة الهدف.
- تعيين الإسقاطات لنقاطفي المجموعة المحدبة الأقرب إلى النقاط الأصلية.

12.2.5. التمارين

1. افترض أننا نريد التحقق من تحدب مجموعة عن طريق رسم جميع الخطوط بين النقاط داخل المجموعة والتحقق مما إذا كانت الخطوط محتواة.
   1. اثبت أنه يكفي التحقق فقط من النقاط الموجودة على الحدود.
   2. أثبت أنه يكفي فحص رؤوس المجموعة فقط.
2. للدلالة به $\mathcal{B}_p[r] \stackrel{\text{def}}{=} \{\mathbf{x}|\mathbf{x} \in \mathbb{R}^d \text{ and } \|\mathbf{x}\|_p \leq r\}$ كرة نصف قطرها r باستخدام $p - norm$. إثبت أن $\mathcal{B}_p[r]$ هو محدب لكل $p \geq 1$.
3. بالنظر إلى الدوال المحدبة f ، أظهر أن $\max(f, g)$ محدب أيضًا. إثبت أن $\min(f, g)$ ليس محدب.
4. إثبت أن تسوية دالة softmax محدب. بشكل أكثر تحديداً إثبت تحدب $f(x) = \log \sum_i \exp(x_i)$.
5. إثبت أن المسافات الجزئية الخطية linear subspaces، أي $\mathcal{X} = \{\mathbf{x}|\mathbf{W}\mathbf{x} = \mathbf{b}\}$ مجموعات محدبة.
6. إثبت أنهفي حالة المساحات الجزئية الخطية مع $\mathbf{b} = \mathbf{0}$ الإسقاط $\text{Proj}_{\mathcal{X}}$ يمكن كتابتها كمافي $\mathbf{M}\mathbf{x}$ لبعض المصفوفات $\mathbf{M}$.

7. أظهر أنه بالنسبة إلى الدوال المحدبة f القابلة للتفاضل مرتين، يمكننا الكتابة $f(x+\epsilon) = f(x) + \epsilon f'(x) + \frac{1}{2}\epsilon^2 f''(x+\xi)$ بالنسبة للبعض $\xi \in [0, \epsilon]$.
8. متجه معطى $\mathbf{w} \in \mathbb{R}^d$ مع حساب الإسقاط على ℓ_1 وحدة الكرة.
   1. كخطوة وسيطة، اكتب الهدف المعاقب $\|\mathbf{w} - \mathbf{w}'\|^2 + \lambda\|\mathbf{w}'\|_1$ واحسب الحل لمعطى معين $\lambda > 0$.
   2. هل يمكنك العثور على القيمة "الصحيحة" لـ λ بدون الكثير من التجربة والخطأ trial and error؟
9. بالنظر إلى مجموعة محدبة $\mathcal{X}$ ومتجهين $\mathbf{x}$ و $\mathbf{y}$ ، إثبت أن الإسقاطات لا تزيد المسافات أبدًا، أي $\|\mathbf{x} - \mathbf{y}\| \geq \|\text{Proj}_{\mathcal{X}}(\mathbf{x}) - \text{Proj}_{\mathcal{X}}(\mathbf{y})\|$.

12.3. الانحدار الاشتقاقي Gradient Descent

في هذا القسم سوف نقدم المفاهيم الأساسية الكامنة وراء الانحدار الاشتقاقي gradient descent. على الرغم من أنه نادرًا ما يستخدم مباشرةفي التعلم العميق، إلا أن فهم الانحدار الاشتقاقي هو المفتاح لفهم خوارزميات الانحدار الاشتقاقي العشوائي stochastic gradient descent. على سبيل المثال، قد تتباعد مشكلة التحسين بسبب معدل التعلم الكبير للغاية. يمكن رؤية هذه الظاهرة بالفعلفي الانحدار الاشتقاقي. وبالمثل، فإن التكييف المسبق preconditioning هو أسلوب شائعفي الانحدار الاشتقاقي وينتقل إلى خوارزميات أكثر تقدمًا. لنبدأ بحالة خاصة بسيطة.

12.3.1. الانحدار الاشتقاقي أحادي البعد One-Dimensional Gradient Descent

يعد الانحدار الاشتقاقيفي بُعد واحد مثالًا ممتازًا لشرح السببفي أن خوارزمية الانحدار الاشتقاقي قد تقلل من قيمة دالة الهدف. ضعفي اعتبارك بعض الدوال ذات القيمة الحقيقية القابلة للتفاضل باستمرار $f: \mathbb{R} \to \mathbb{R}$. باستخدام توسع تايلور Taylor expansion نحصل عليه

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \mathcal{O}(\epsilon^2).$$

أي،في التقريب من الدرجة الأولى $f(x+\epsilon)$، يتم إعطاء قيمة الدالة $f(x)$ والمشتق الأول $f'(x)$ عند x. ليس من غير المعقول افتراض أن الحركة الصغيرة ϵفي اتجاه الانحدار الاشتقاقي ستقلل f. لتبسيط الأمور، نختار حجم خطوة ثابتًا $\eta > 0$ ونختار $\epsilon = -\eta f'(x)$. توصيل هذا في توسيع تايلور أعلاه نحصل عليه

$$f(x - \eta f'(x)) = f(x) - \eta f'^2(x) + \mathcal{O}(\eta^2 f'^2(x)).$$

إذا لم يتلاشى المشتق $f'(x) \neq 0$، فإننا نحرز تقدمًا لان $\eta f'^2(x) > 0$. علاوة على ذلك، يمكننا دائمًا اختيار η صغيرة بما يكفي لأن تصبح مصطلحات الترتيب الأعلى غير ذات صلة. ومن هنا وصلنا إلى

$$f(x - \eta f'(x)) \lessapprox f(x).$$

هذا يعني أنه إذا استخدمنا

$$x \leftarrow x - \eta f'(x)$$

لتكرار x، قد تنخفض قيمة الدالة $f(x)$. لذلك،في الانحدار الاشتقاقي نختار أولاً قيمة أولية x وثابت $\eta > 0$ ثم نستخدمهما للتكرار المستمر لـ x حتى يتم الوصول إلى حالة التوقف، على سبيل المثال، عندما يكون حجم الانحدار الاشتقاقي $|f'(x)|$ صغيرًا بدرجة كافية أو يصل عدد التكرارات إلى قيمة معينة.

من أجل البساطة، نختار دالة الهدف $f(x) = x^2$ لتوضيح كيفية تنفيذ الانحدار الاشتقاقي. على الرغم من أننا نعلم أن $x = 0$هو الحل لتقليل $f(x)$، إلا أننا ما زلنا نستخدم هذه الدالة البسيطة لمراقبة كيفية يتغير x.

```
%matplotlib inline
import numpy as np
import tensorflow as tf
from d2l import tensorflow as d2l

def f(x):  # Objective function
    return x ** 2

def f_grad(x):  # Gradient (derivative) of the objective
function
    return 2 * x
```

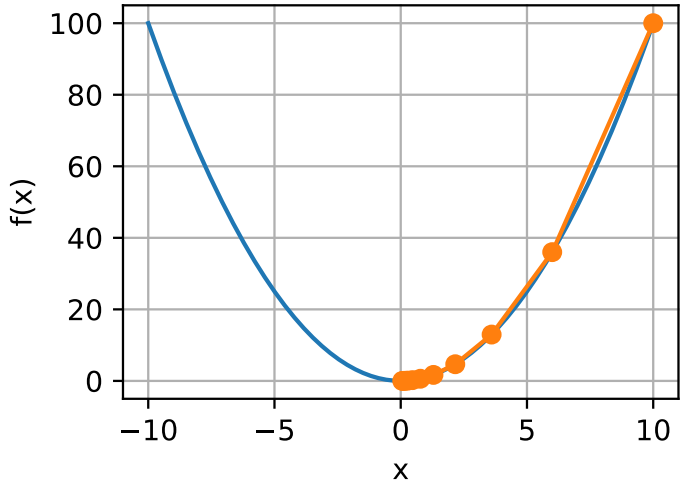
بعد ذلك، نستخدم $x = 10$ كقيمة أولية ونفترض $\eta = 0.2$. باستخدام الانحدار الاشتقاقي لتكرار x لعشر مرات يمكننا أن نرى،في النهاية، قيمة x تقترب من الحل الأمثل.

```
def gd(eta, f_grad):
    x = 10.0
    results = [x]
    for i in range(10):
        x -= eta * f_grad(x)
        results.append(float(x))
    print(f'epoch 10, x: {x:f}')
    return results
```

```
results = gd(0.2, f_grad)
```

يمكن رسم تقدم تحسين x على النحو التالي.

```
def show_trace(results, f):
    n = max(abs(min(results)), abs(max(results)))
    f_line = tf.range(-n, n, 0.01)
    d2l.set_figsize()
    d2l.plot([f_line, results], [[f(x) for x in f_line],
[
        f(x) for x in results]], 'x', 'f(x)', fmts=['-',
'-o'])

show_trace(results, f)
```

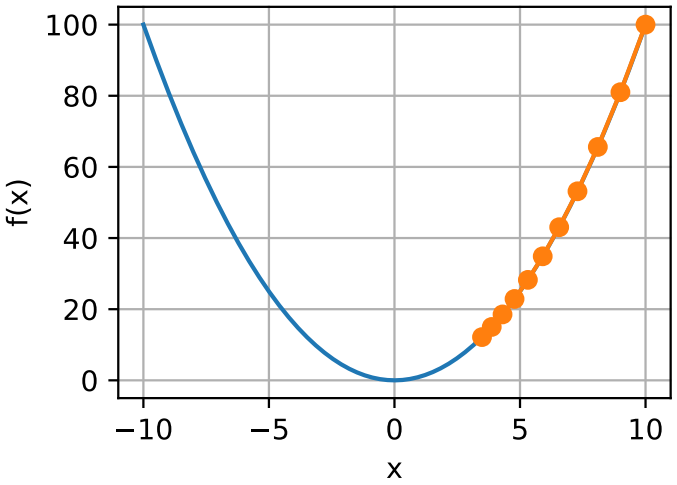


12.3.1.1. معدل التعلم Learning Rate

يمكن تحديد معدل التعلم η بواسطة مصمم الخوارزمية. إذا استخدمنا معدل تعلم صغيرًا جدًا، فسيؤدي ذلك إلى تحديث x ببطء شديد، مما يتطلب المزيد من التكرارات للحصول على حل أفضل. لإظهار ما يحدثفي مثل هذه الحالة، ضعفي اعتبارك التقدمفي نفس مشكلة التحسين لـ $\eta = 0.05$. كما نرى، حتى بعد 10 خطوات ما زلنا بعيدين جدًا عن الحل الأمثل.

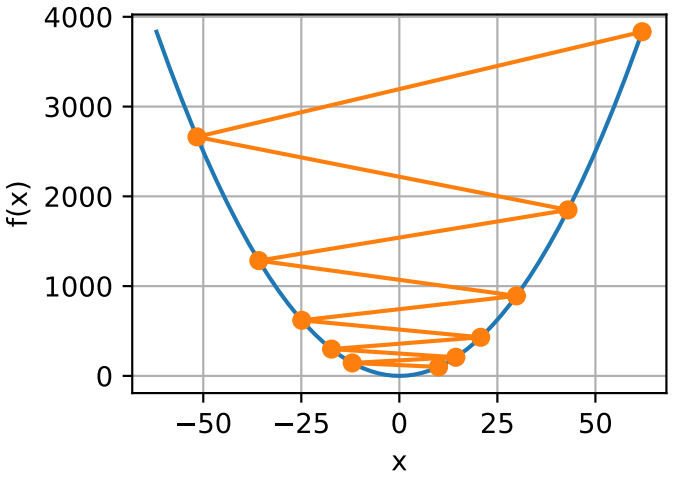
```
show_trace(gd(0.05, f_grad), f)
```

```
epoch 10, x: 3.486784
```



على العكس من ذلك، إذا استخدمنا معدل تعلم مرتفعًا بشكل مفرط، $|\eta f'(x)|$ فقد يكون كبيرًا جدًا بالنسبة لصيغة توسيع تايلور من الدرجة الأولى. أي أن المصطلح $\mathcal{O}(\eta^2 f'^2(x))$ في (12.3.2) قد يصبح مهمًا.في هذه الحالة، لا يمكننا ضمان أن تكرار x سيكون قادرًا على خفض قيمة $f(x)$. على سبيل المثال، عندما نضبط معدل التعلم على $\eta = 1.1$، x يتجاوز الحل الأمثل $x = 0$ ويتباعد تدريجيًا.

```
show_trace(gd(1.1, f_grad), f)
```
```
epoch 10, x: 61.917364
```

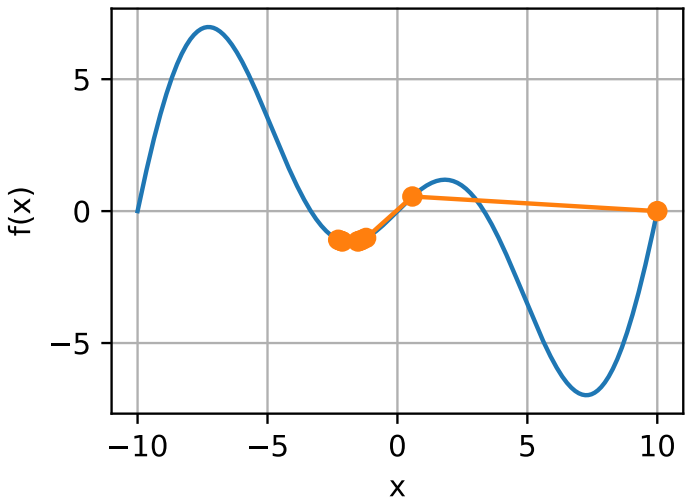


12.3.1.2. الحد الأدنى المحلي Local Minima

لتوضيح ما يحدث للدوال غير المحدبة، ضعفي اعتبارك حالة $f(x) = x \cdot \cos(cx)$ لبعض الثابت c. هذه الدالة لها عدد لا نهائي من الحدود الدنيا المحلية local minima. اعتمادًا على اختيارنا لمعدل التعلم واعتمادًا على مدى جودة المشكلة، قد ننتهي بواحد من العديد من الحلول. يوضح المثال أدناه كيف أن معدل التعلم العالي (غير الواقعي) سيؤدي إلى حد أدنى محلي ضعيف.

```
c = tf.constant(0.15 * np.pi)

def f(x):  # Objective function
    return x * tf.cos(c * x)

def f_grad(x):  # Gradient of the objective function
    return tf.cos(c * x) - c * x * tf.sin(c * x)

show_trace(gd(2, f_grad), f)
```

```
epoch 10, x: -1.528165
```



12.3.2. الانحدار الاشتقاقي متعدد المتغيرات Multivariate Gradient Descent

الآن بعد أن أصبح لدينا حدس أفضل للحالة أحادية المتغير univariate case، دعنا نفكر في الموقف حيث $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top$. وهذا يعني أن دالة الهدف $f: \mathbb{R}^d \to \mathbb{R}$ ترسم

المتجهات إلى قيم قياسية scalars.في المقابل، يكون تدرجها متعدد المتغيرات multivariate أيضًا. هو متجه يتكون من d مشتقات جزئية:

$$\nabla f(\mathbf{x}) = [\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}]^\top.$$

يشير كل عنصر مشتق جزئي $\partial f(\mathbf{x})/\partial x_i$في الانحدار الاشتقاقي إلى معدل تغيير f عند $\mathbf{x}$ بالنسبة للإدخال x_i. كما كان من قبلفي الحالة أحادية المتغير، يمكننا استخدام تقريب تايلور المقابل للدوال متعددة المتغيرات للحصول على فكرة عما يجب أن نفعله. على وجه الخصوص، لدينا ذلك

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^\top \nabla f(\mathbf{x}) + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2).$$

بعبارة أخرى، ما يصل إلى الحد من الدرجة الثانيةفي $\boldsymbol{\epsilon}$ اتجاه أشد انحدارًا يتم الحصول عليه من خلال الانحدار الاشتقاقي السالب $-\nabla f(\mathbf{x})$. ينتج عن اختيار معدل التعلم $\eta > 0$ المناسب خوارزمية الانحدار الاشتقاقي النموذجي:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x}).$$

لنرى كيف تتصرف الخوارزميةفي الممارسة العملية، دعونا نبني دالة هدف $f(\mathbf{x}) = x_1^2 + 2x_2^2$ مع متجه ثنائي الأبعاد $\mathbf{x} = [x_1, x_2]^\top$ كمدخلات وقيمة قياسية scalar كمخرج. يتم إعطاء الانحدار الاشتقاقي بواسطة$\nabla f(\mathbf{x}) = [2x_1, 4x_2]^\top$. سوف نلاحظ مسار $\mathbf{x}$ بواسطة الانحدار الاشتقاقي من الموضع الأولي $[-5, -2]$.

بادئ ذي بدء، نحتاج إلى دالتين مساعدتين إضافيتين. الأول يستخدم دالة التحديث ويطبقها 20 مرة على القيمة الأولية. المساعد الثاني يرسم مسار $\mathbf{x}$.

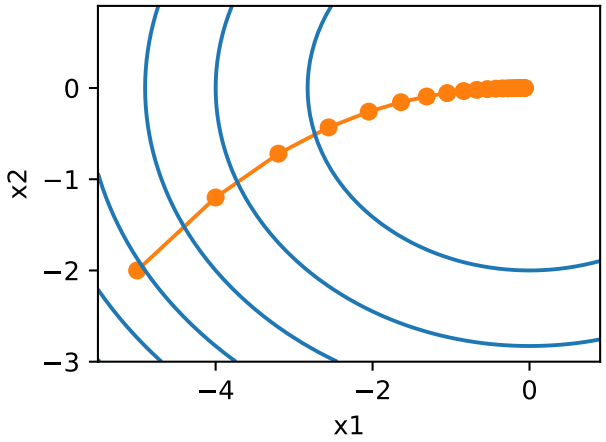
```
def train_2d(trainer, steps=20, f_grad=None):  #@save
    """Optimize a 2D objective function with a
customized trainer."""
    # `s1` and `s2` are internal state variables that
will be used in Momentum, adagrad, RMSProp
    x1, x2, s1, s2 = -5, -2, 0, 0
    results = [(x1, x2)]
    for i in range(steps):
        if f_grad:
            x1, x2, s1, s2 = trainer(x1, x2, s1, s2,
f_grad)
        else:
            x1, x2, s1, s2 = trainer(x1, x2, s1, s2)
```

```
        results.append((x1, x2))
    print(f'epoch {i + 1}, x1: {float(x1):f}, x2:
{float(x2):f}')
    return results

def show_trace_2d(f, results):  #@save
    """Show the trace of 2D variables during
optimization."""
    d2l.set_figsize()
    d2l.plt.plot(*zip(*results), '-o', color='#ff7f0e')
    x1, x2 = tf.meshgrid(tf.range(-5.5, 1.0, 0.1),
                         tf.range(-3.0, 1.0, 0.1))
    d2l.plt.contour(x1, x2, f(x1, x2), colors='#1f77b4')
    d2l.plt.xlabel('x1')
    d2l.plt.ylabel('x2')
```

بعد ذلك، نلاحظ مسار متغير التحسين $\mathbf{x}$ لمعدل التعلم $\eta = 0.1$. يمكننا أن نرى أنه بعد 20 خطوة، فإن قيمة $\mathbf{x}$ تقترب من الحد الأدنى لها عند $[0,0]$. التقدم حسن السلوك إلى حد ما وإن كان بطيئًا إلى حد ما.

```
def f_2d(x1, x2):  # Objective function
    return x1 ** 2 + 2 * x2 ** 2

def f_2d_grad(x1, x2):  # Gradient of the objective
function
    return (2 * x1, 4 * x2)

def gd_2d(x1, x2, s1, s2, f_grad):
    g1, g2 = f_grad(x1, x2)
    return (x1 - eta * g1, x2 - eta * g2, 0, 0)

eta = 0.1
show_trace_2d(f_2d, train_2d(gd_2d, f_grad=f_2d_grad))
```

```
epoch 20, x1: -0.057646, x2: -0.000073
```



12.3.3. طرق التكيف Adaptive Methods

كما يمكن أن نرىفي القسم 12.3.1.1، فإن الحصول على معدل التعلم η "الصحيح تمامًا" يعد أمرًا صعبًا. إذا اخترناها صغيرة جدًا، فإننا نحرز تقدمًا ضئيلًا. إذا اخترناها كبيرة جدًا، فإن الحل يتذبذب وفي أسوأ الحالات قد يتباعد. ماذا لو تمكنا من التحديد تلقائيًا أو التخلص من الاضطرار إلى تحديد معدل التعلم η على الإطلاق؟ يمكن أن تساعد طرق الدرجة الثانية التي لا تنظر فقط إلى قيمة وتدرج (انحدار) دالة الهدف ولكن أيضًافي تقوسهاcurvatureفي هذه الحالة.في حين أن هذه الأساليب لا يمكن تطبيقها على التعلم العميق مباشرة بسبب التكلفة الحسابية، فإنها توفر حدسًا مفيدًا حول كيفية تصميم خوارزميات التحسين المتقدمة التي تحاكي العديد من الخصائص المرغوبة للخوارزميات الموضحة أدناه.

12.3.3.1. طريقة نيوتن Newton's Method

مراجعة توسيع تايلور لبعض الدوال $f: \mathbb{R}^d \to \mathbb{R}$ ليست هناك حاجة للتوقف بعد المصطلح الأول.في الواقع، يمكننا كتابتها كـ

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^\top \nabla f(\mathbf{x}) + \frac{1}{2}\boldsymbol{\epsilon}^\top \nabla^2 f(\mathbf{x})\boldsymbol{\epsilon} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^3).$$

لتجنب التدوين المتعب نحدد $\mathbf{H} \stackrel{\text{def}}{=} \nabla^2 f(\mathbf{x})$ ليكون من Hessian لـ f، وهي مصفوفة $d \times d$. للمشاكل الصغيرة d والبسيطة $\mathbf{H}$ من السهل حسابها. بالنسبة للشبكات العصبية العميقة، من ناحية أخرى، $\mathbf{H}$ قد تكون كبيرة، بسبب تكلفة تخزين الإدخالات $\mathcal{O}(d^2)$. علاوة على ذلك، قد يكون الحساب عن طريق الانتشار الخلفي backpropagation مكلفًا للغاية. دعونا الآن نتجاهل مثل هذه الاعتبارات وننظر إلى الخوارزمية التي سنحصل عليها.

بعد كل شيء، الحد الأقل من f يحقق $\nabla f = 0$. باتباع قواعد حساب التفاضل والتكامل في القسم 2.4.3، من خلال أخذ مشتقات (12.3.8) فيما يتعلق بـ $\boldsymbol{\epsilon}$ وتجاهل المصطلحات ذات الترتيب الأعلى، نصل إلى

$$\nabla f(\mathbf{x}) + \mathbf{H}\boldsymbol{\epsilon} = 0 \text{ and hence } \boldsymbol{\epsilon} = -\mathbf{H}^{-1}\nabla f(\mathbf{x}).$$

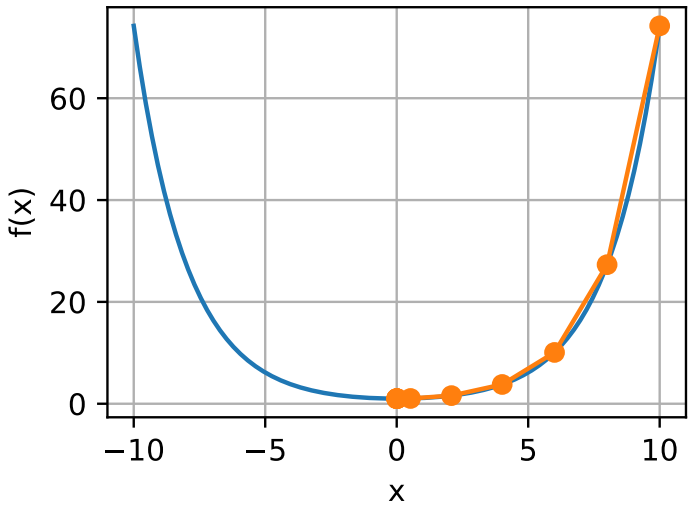
وهذا يعني أننا نحتاج إلى قلب Hessian $\mathbf{H}$ كجزء من مشكلة التحسين.

كمثال بسيط، لـ $f(x) = \frac{1}{2}x^2$ لدينا $\nabla f(x) = x$ و $\mathbf{H} = 1$. ومن ثم لأي x نحصل على $\epsilon = -x$. بمعنى آخر، الخطوة الواحدة تكفي للتقارب بشكل مثالي دون الحاجة إلى أي تعديل! للأسف، لقد حالفنا بعض الشيء هنا: كان توسع تايلور دقيقًا لان $f(x + \epsilon) = \frac{1}{2}x^2 + \epsilon x + \frac{1}{2}\epsilon^2$

دعونا نرى ما يحدث في المشاكل الأخرى. بالنظر إلى دالة جيب التمام المحدبة الزائدية $f(x) = \cosh(cx)$ لبعض الثوابت c ، يمكننا أن نرى أنه تم الوصول إلى الحد الأدنى العالمي عند $x = 0$ بعد عدة تكرارات.

```
c = tf.constant(0.5)

def f(x):  # Objective function
    return tf.cosh(c * x)

def f_grad(x):  # Gradient of the objective function
    return c * tf.sinh(c * x)

def f_hess(x):  # Hessian of the objective function
    return c**2 * tf.cosh(c * x)

def newton(eta=1):
    x = 10.0
    results = [x]
    for i in range(10):
        x -= eta * f_grad(x) / f_hess(x)
        results.append(float(x))
    print('epoch 10, x:', x)
    return results

show_trace(newton(), f)
```

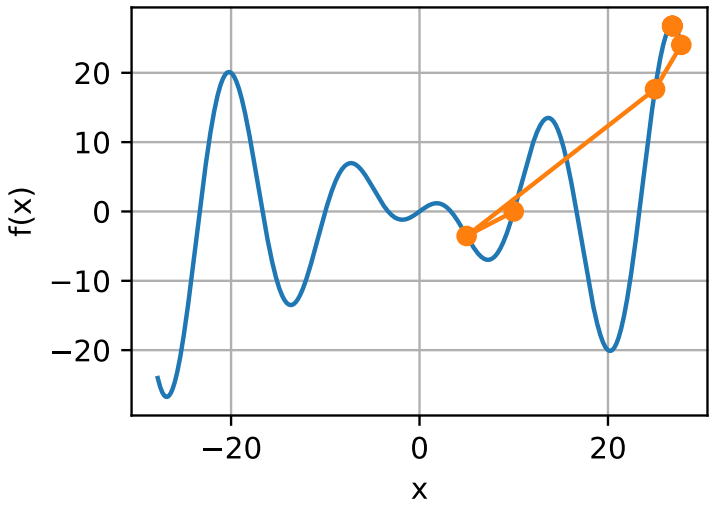
```
epoch 10, x: tf.Tensor(0.0, shape=(), dtype=float32)
```



دعنا الآن نفكرفي دالة غير محدبة، مثل $f(x) = x\cos(cx)$ لبعض الثوابت c. بعد كل شيء، لاحظ أنهفي طريقة نيوتن ينتهي بنا الأمر بالقسمة على Hessian. هذا يعني أنه إذا كان المشتق الثاني سالبًا، فقد نسيرفي اتجاه زيادة قيمة f. هذا عيب فادحفي الخوارزمية. دعونا نرى ما يحدث في الممارسة العملية.

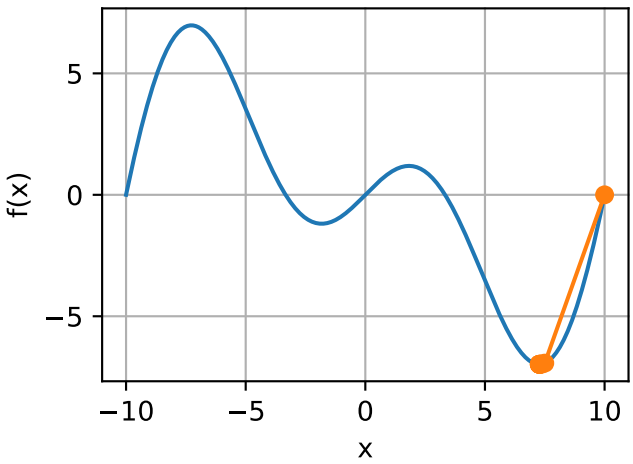
```
c = tf.constant(0.15 * np.pi)

def f(x):  # Objective function
    return x * tf.cos(c * x)

def f_grad(x):  # Gradient of the objective function
    return tf.cos(c * x) - c * x * tf.sin(c * x)

def f_hess(x):  # Hessian of the objective function
    return - 2 * c * tf.sin(c * x) - x * c**2 * tf.cos(c
* x)

show_trace(newton(), f)
```

```
epoch 10, x: tf.Tensor(26.834133, shape=(),
dtype=float32)
```



حدث هذا بشكل خاطئ بشكل مذهل. كيف يمكننا إصلاحه؟ إحدى الطرق هي "إصلاح" Hessian بأخذ قيمته المطلقة بدلاً من ذلك. استراتيجية أخرى هي إعادة معدل التعلم. يبدو أن هذا يلغي الهدف، لكن ليس تمامًا. يتيح لنا الحصول على معلومات من الدرجة الثانية توخي الحذر كلما كان الانحناء كبيرًا واتخاذ خطوات أطول كلما كانت دالة الهدف أكثر انبساطًا flatter. دعونا نرى كيف يعمل هذا مع معدل تعلم أصغر قليلاً، على سبيل المثال $\eta = 0.5$. كما نرى، لدينا خوارزمية فعالة تمامًا.

```
show_trace(newton(0.5), f)
epoch 10, x: tf.Tensor(7.26986, shape=(), dtype=float32)
```



12.3.3.2. تحليل التقارب Convergence Analysis

نحن نحلل فقط معدل التقارب لطريقة نيوتن لبعض دوال الهدف المحدبة وثلاث مرات قابلة للتفاضل f ، حيث يكون المشتق الثاني غير صفري، أي $f'' > 0$. يعد الدليل متعدد المتغيرات امتدادًا مباشرًا للوسيطة أحادية البعد أدناه ويتم حذفه لأنه لا يساعدنا كثيرًا من حيث الحدس.

قم بالإشارة إلى $x^{(k)}$ حيث قيمة x عند التكرار k^{th} وليكن $e^{(k)} \stackrel{\text{def}}{=} x^{(k)} - x^*$ تكون المسافة من المثالية عند التكرار k^{th}. من خلال توسع تايلور، لدينا أن الشرط $f'(x^*) = 0$ يمكن كتابته كـ

$$0 = f'(x^{(k)} - e^{(k)}) = f'(x^{(k)}) - e^{(k)} f''(x^{(k)}) + \frac{1}{2}(e^{(k)})^2 f'''(\xi^{(k)}),$$

وهو ما يحمله لبعض $\xi^{(k)} \in [x^{(k)} - e^{(k)}, x^{(k)}]$. قسمة التوسع أعلاه على $f''(x^{(k)})$ ينتج:

$$e^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})} = \frac{1}{2}(e^{(k)})^2 \frac{f'''(\xi^{(k)})}{f''(x^{(k)})}.$$

تذكر أن لدينا التحديث $x^{(k+1)} = x^{(k)} - f'(x^{(k)})/f''(x^{(k)})$. بالتعويضفي معادلة التحديث هذه وأخذ القيمة المطلقة لكلا الجانبين، لدينا

$$|e^{(k+1)}| = \frac{1}{2}(e^{(k)})^2 \frac{|f'''(\xi^{(k)})|}{f''(x^{(k)})}.$$

وبالتالي، كلما كنافي منطقة محدودة $|f'''(\xi^{(k)})|/(2f''(x^{(k)})) \leq c$ ، لدينا خطأ تنازلي تربيعي

$$|e^{(k+1)}| \leq c(e^{(k)})^2.$$

جانبا، يسمي باحثو التحسين هذا بالتقارب الخطي linear convergence،في حين أن حالة مثل $|e^{(k+1)}| \leq \alpha|e^{(k)}|$ يسمى معدل التقارب الثابت constant rate of convergence. لاحظ أن هذا التحليل يأتي مع عدد من المحاذير. أولاً، ليس لدينا الكثير من الضمانات عندما نصل إلى منطقة التقارب السريع. بدلاً من ذلك، نعلم فقط أنه بمجرد أن نصل إليه، سيكون التقارب سريعًا جدًا. ثانيًا، يتطلب هذا التحليل أن يكون f حسن التصرف حتى المشتقات عالية الرتبة. يتعلق الأمر بضمان f لا تحتوي أي خصائص "مفاجئة" من حيث كيفية تغيير قيمها.

12.3.3.3. التكييف المسبق Preconditioning

ليس من المستغرب تمامًا حساب وتخزين Hessian بالكامل مكلف للغاية. لذلك من المستحسن إيجاد بدائل. طريقة واحدة لتحسين الأمور هي التكييف المسبق

Preconditioning. يتجنب حساب Hessian بالكامل ولكنه يحسب فقط الإدخالات القطرية diagonal entries. هذا يؤدي إلى تحديث خوارزميات النموذج

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \text{diag}(\mathbf{H})^{-1} \nabla f(\mathbf{x}).$$

في حين أن هذا ليس جيدًا تمامًا مثل طريقة نيوتن الكاملة، إلا أنه لا يزال أفضل بكثير من عدم استخدامه. لمعرفة سبب كون هذه فكرة جيدة، ضع في اعتبارك موقفًا يشير فيه أحد المتغيرات إلى الارتفاع بالمليمترات بينما يشير الآخر إلى الارتفاع بالكيلومترات. بافتراض أن كلا المقياس الطبيعي بالأمتار، لدينا عدم تطابق رهيب في المعلمات. لحسن الحظ، فإن استخدام التكييف المسبق يزيل ذلك. إن التكييف المسبق الفعال مع الانحدار الاشتقاقي يعني اختيار معدل تعلم مختلف لكل متغير (تنسيق متجه $\mathbf{x}$). كما سنرى لاحقًا، يقود التكييف المسبق بعض الابتكارات في خوارزميات تحسين الانحدار الاشتقاقي العشوائي.

12.3.3.4. الانحدار الاشتقاقي مع البحث الخطي Gradient Descent with Line Search

تتمثل إحدى المشكلات الرئيسية في الانحدار الاشتقاقي في أننا قد نتجاوز الهدف أو نحرز تقدمًا غير كاف. حل بسيط لهذه المشكلة هو استخدام البحث الخطي line search مع الانحدار الاشتقاقي. أي أننا نستخدم الاتجاه الذي قدم بواسطة $\nabla f(\mathbf{x})$ ثم نجري بحثًا ثنائيًا عن معدل التعلم η الذي يقلل $f(\mathbf{x} - \eta \nabla f(\mathbf{x}))$.

تتقارب هذه الخوارزمية بسرعة (للاطلاع على التحليل والإثبات، على سبيل المثال، Boyd and Vandenberghe(2004)). ومع ذلك، لغرض التعلم العميق، هذا ليس ممكنًا تمامًا، لأن كل خطوة من خطوات البحث الخطي تتطلب منا تقييم دالة الهدف في مجموعة البيانات بأكملها. هذه طريقة مكلفة للغاية لإنجازها.

12.3.4. الملخص

- معدلات التعلم مهمة. كبير جدًا ونحن متباعدون، صغيرة جدًا ولا نحرز تقدمًا.
- يمكن أن يعلق الانحدار الاشتقاقي في الحدود الدنيا المحلية.
- في الأبعاد العالية، يعد ضبط معدل التعلم أمرًا معقدًا.
- يمكن أن يساعد التكييف المسبق Preconditioning في تعديل المقياس.
- تكون طريقة نيوتن أسرع كثيرًا بمجرد أن تبدأ في العمل بشكل صحيح في المشكلات المحدبة.
- احذر من استخدام طريقة نيوتن دون أي تعديلات للمشكلات غير المحدبة.

12.3.5. التمارين

1. جرب مع معدلات التعلم المختلفة ودوال الهدف للانحدار الاشتقاقي.
2. نفذ البحث الخطي لتقليل دالة محدبةفي الفاصل $[a, b]$.
   1. هل تحتاج إلى مشتقات للبحث الثنائي، أي لتحديد ما إذا كنت تريد اختيار $[a, (a+b)/2]$ أم $[(a+b)/2, b]$.
   2. ما مدى سرعة معدل التقارب للخوارزمية؟
   3. نفذ الخوارزمية وطبقها على تقليل $\log(\exp(x) + \exp(-2x-3))$.
3. صمم دالة هدف محددةفي المكان الذي يكون فيه الانحدار بطيئًا للغاية. ملحوظة: مقياس إحداثيات مختلفة بشكل مختلف.
4. نفذ النسخة الخفيفة من طريقة نيوتن باستخدام التكييف المسبق preconditioning:
   1. استخدم Hessian قطري كمكيف مسبق.
   2. استخدم القيم المطلقة لذلك بدلاً من القيم الفعلية (ربما موقعة signed).
   3. قم بتطبيق هذا على المشكلة أعلاه.
5. قم بتطبيق الخوارزمية أعلاه على عدد من دوال الهدف (محدبة أم لا). ماذا يحدث إذا قمت بتدوير الإحداثيات 45 درجة؟

12.4. الانحدار الاشتقاقي العشوائي Stochastic Gradient Descent

في الفصول السابقة، واصلنا استخدام الانحدار الاشتقاقي العشوائيفي إجراءات التدريب لدينا، ومع ذلك، دون توضيح سبب نجاحه. لإلقاء بعض الضوء عليها، قمنا للتو بوصف المبادئ الأساسية للانحدار الاشتقاقيفي القسم 12.3. في هذا القسم، ننتقل لمناقشة الانحدار الاشتقاقي العشوائي بمزيد من التفصيل.

```
%matplotlib inline
import math
import tensorflow as tf
from d2l import tensorflow as d2l
```

12.4.1. تحديثات الانحدار العشوائي Stochastic Gradient Updates

في التعلم العميق، عادةً ما تكون دالة الهدف هي متوسط دوال الخسارة (الخطأ) لكل مثالفي مجموعة بيانات التدريب. بالنظر إلى مجموعة بيانات التدريب من الأمثلة n ، نفترض أن $f_i(\mathbf{x})$ هي دالة الخسارة فيما يتعلق بمثال التدريب للفهرس i ، حيث $\mathbf{x}$ يكون متجه المعلمة. ثم نصل إلى دالة الهدف:

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

يتم حساب انحدار دالة الهدف في كـ

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}).$$

إذا تم استخدام الانحدار الاشتقاقي، فإن التكلفة الحسابية لكل تكرار متغير مستقل هي $\mathcal{O}(n)$ والتي تنمو خطيًا مع n. لذلك، عندما تكون مجموعة بيانات التدريب أكبر، ستكون تكلفة الانحدار لكل تكرار أعلى.

يعمل الانحدار الاشتقاقي العشوائي (SGD) على تقليل التكلفة الحسابية عند كل تكرار. في كل تكرار لنسب التدرج العشوائي، نقوم بتجربة فهرس $i \in \{1, \ldots, n\}$ بشكل موحد لأمثلة البيانات بشكل عشوائي، ونحسب الانحدار $\nabla f_i(\mathbf{x})$ لتحديثه:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x}),$$

حيث η هو معدل التعلم. يمكننا أن نرى أن التكلفة الحسابية لكل تكرار تنخفض من $\mathcal{O}(n)$ للانحدار الاشتقاقي إلى الثابت $\mathcal{O}(1)$. علاوة على ذلك، نريد التأكيد على أن الانحدار الاشتقاقي العشوائي $\nabla f_i(\mathbf{x})$ هو تقدير غير متحيز للانحدار الكامل $\nabla f(\mathbf{x})$ لأن

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

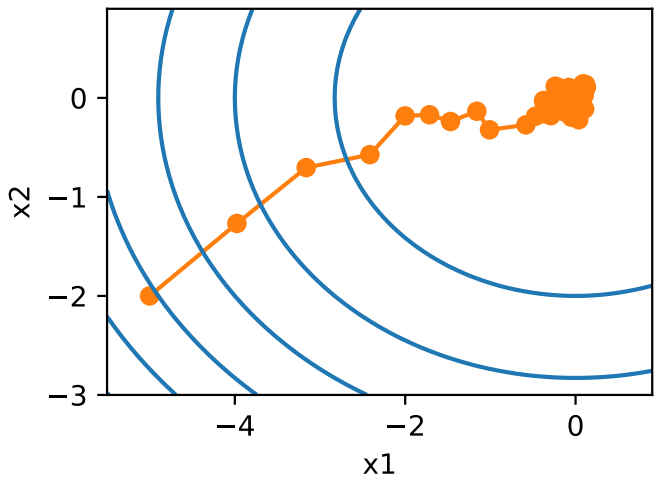
هذا يعني، في المتوسط، أن الانحدار الاشتقاقي العشوائي يعد تقديرًا جيداً للانحدار.

الآن، سنقارنه مع الانحدار الاشتقاقي عن طريق إضافة ضوضاء عشوائية بمتوسط 0 وتباين 1 إلى التدرج لمحاكاة الانحدار الاشتقاقي العشوائي.

```
def f(x1, x2):  # Objective function
    return x1 ** 2 + 2 * x2 ** 2

def f_grad(x1, x2):  # Gradient of the objective
function
    return 2 * x1, 4 * x2

def sgd(x1, x2, s1, s2, f_grad):
    g1, g2 = f_grad(x1, x2)
    # Simulate noisy gradient
```

```
    g1 += tf.random.normal([1], 0.0, 1)
    g2 += tf.random.normal([1], 0.0, 1)
    eta_t = eta * lr()
    return (x1 - eta_t * g1, x2 - eta_t * g2, 0, 0)

def constant_lr():
    return 1

eta = 0.1
lr = constant_lr  # Constant learning rate
d2l.show_trace_2d(f, d2l.train_2d(sgd, steps=50,
f_grad=f_grad))
```

```
epoch 50, x1: 0.037351, x2: -0.220945
/home/d2l-worker/miniconda3/envs/d2l-en-release-
0/lib/python3.9/site-
packages/numpy/core/shape_base.py:65:
VisibleDeprecationWarning: Creating an ndarray from
ragged nested sequences (which is a list-or-tuple of
lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must
specify 'dtype=object' when creating the ndarray.
  ary = asanyarray(ary)
```



كما نرى، فإن مسار المتغيراتفي الانحدار الاشتقاقي العشوائي أكثر ضوضاءً من ذلك الذي لاحظناهفي الانحدار الاشتقاقيفي القسم 12.3. هذا يرجع إلى الطبيعة العشوائية للانحدار. وهذا يعني أنه حتى عندما نقترب من الحد الأدنى، فإننا لا نزال خاضعين لعدم اليقين الذي يضخه

الانحدار اللحظي عبر $\eta\nabla f_i(\mathbf{x})$. حتى بعد 50 خطوة، لا تزال الجودة غير جيدة. والأسوأ من ذلك، أنه لن يتحسن بعد خطوات إضافية (نشجعك على تجربة عدد أكبر من الخطوات لتأكيد ذلك). هذا يترك لنا البديل الوحيد: تغيير معدل التعلم η. ومع ذلك، إذا اخترنا هذا صغيرًا جدًا، فلن نحرز أي تقدم ذي مغزىفي البداية. من ناحية أخرى، إذا اخترناها كبيرة جدًا، فلن نحصل على حل جيد، كما رأينا أعلاه. الطريقة الوحيدة لحل هذه الأهداف المتضاربة هي تقليل معدل التعلم ديناميكيًا dynamically مع تقدم التحسين.

وهذا أيضًا سبب إضافة دالة معدل التعلم `lr` إلى الدالة الخطوية `sgd`.في المثال أعلاه، تكون أي دالة لجدولة معدل التعلم تكمنفي سبات حيث قمنا بتعيين دالة `lr` المرتبطة لتكون ثابتة.

12.4.2. معدل التعلم الديناميكي Dynamic Learning Rate

يؤدي استبدال η بمعدل التعلم المعتمد على الوقت إلى زيادة تعقيد التحكمفي تقارب خوارزمية التحسين. على وجه الخصوص، نحتاج إلى معرفة السرعة التي يجب أن تتحلل بها. إذا كان الأمر سريعًا جدًا، فسوف نتوقف عن التحسين قبل الأوان. إذا قللناها ببطء شديد، فإننا نضيع الكثير من الوقتفي التحسين. فيما يلي بعض الاستراتيجيات الأساسية المستخدمةفي تعديل η بمرور الوقت (سنناقش المزيد من الاستراتيجيات المتقدمة لاحقًا):

$$\begin{aligned} \eta(t) &= \eta_i \text{ if } t_i \leq t \leq t_{i+1} && \text{piecewise constant} \\ \eta(t) &= \eta_0 \cdot e^{-\lambda t} && \text{exponential decay} \\ \eta(t) &= \eta_0 \cdot (\beta t + 1)^{-\alpha} && \text{polynomial decay} \end{aligned}$$

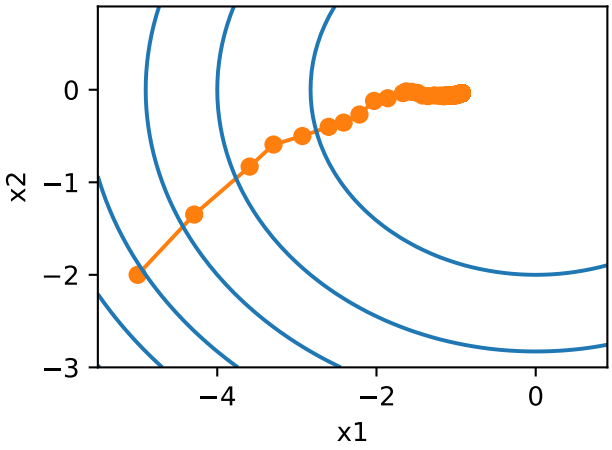
في السيناريو الثابت متعدد الأجزاء الأول، نقوم بتقليل معدل التعلم، على سبيل المثال، كلما توقف التقدمفي تحسين الأداء. هذه استراتيجية شائعة لتدريب الشبكات العميقة. بدلاً من ذلك، يمكننا تقليله بقوة أكبر من خلال الاضمحلال الأسي exponential decay. لسوء الحظ، يؤدي هذا غالبًا إلى التوقف المبكر قبل أن تتقارب الخوارزمية. الاختيار الشائع هو الاضمحلال متعدد الحدود polynomial decay مع $\alpha = 0.5$.في حالة التحسين المحدب، هناك عدد من البراهين التي تظهر أن هذا المعدل حسن التصرف.

دعونا نرى كيف يبدو الانحلال الأسيفي الممارسة العملية.

```
def exponential_lr():
    # Global variable that is defined outside this
function and updated inside
    global t
    t += 1
    return math.exp(-0.1 * t)
```

```
t = 1
lr = exponential_lr
d2l.show_trace_2d(f, d2l.train_2d(sgd, steps=1000,
f_grad=f_grad))
```

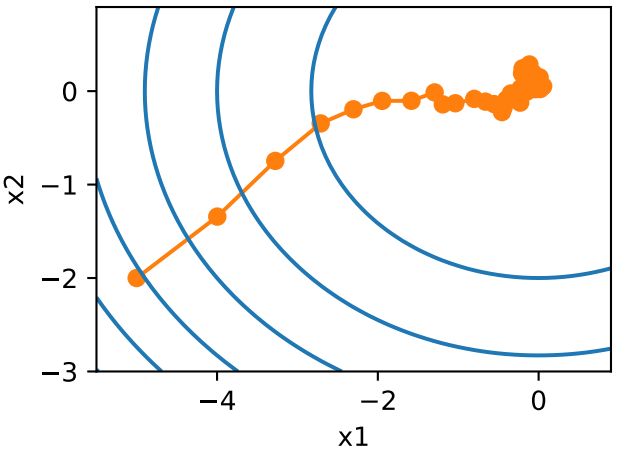
```
epoch 1000, x1: -0.928472, x2: -0.037240
```



كما هو متوقع، يتم تقليل التباينفي المعلمات بشكل كبير. ومع ذلك، يأتي هذا على حساب الفشلفي التقارب بالحل الأمثل $\mathbf{x} = (0,0)$. حتى بعد 1000 خطوة من التكرار، ما زلنا بعيدين جدًا عن الحل الأمثل.في الواقع، فشلت الخوارزميةفي التقارب على الإطلاق. من ناحية أخرى، إذا استخدمنا الانحلال متعدد الحدود حيث يتحلل معدل التعلم مع الجذر التربيعي العكسي لعدد الخطوات، فإن التقارب يتحسن بعد 50 خطوة فقط.

```
def polynomial_lr():
    # Global variable that is defined outside this
function and updated inside
    global t
    t += 1
    return (1 + 0.1 * t) ** (-0.5)

t = 1
lr = polynomial_lr
d2l.show_trace_2d(f, d2l.train_2d(sgd, steps=50,
f_grad=f_grad))
```

```
epoch 50, x1: -0.066644, x2: 0.113844
```



يوجد العديد من الخيارات حول كيفية ضبط معدل التعلم. على سبيل المثال، يمكننا أن نبدأ بمعدل صغير، ثم نزيده بسرعة ثم نخفضه مرة أخرى، وإن كان ذلك بشكل أبطأ. يمكننا حتى التناوب بين معدلات التعلم الأصغر والأكبر. توجد مجموعة كبيرة ومتنوعة مثل هذه الجدولة. دعونا الآن نركز على جداول معدل التعلم التي يمكن إجراء تحليل نظري شامل لها، أي على معدلات التعلمفي وضع محدب. بالنسبة للمشكلات العامة غير المحدبة، من الصعب جدًا الحصول على ضمانات تقارب ذات مغزى، نظرًا لأن التقليل من المشكلات غير الخطية غير المعقدة بشكل عام مسائل NP صعبة NP hard. للاستطلاع، انظر على سبيل المثال، ملاحظات المحاضرة (lecture notes) الممتازة لـ Tibshirani 2015.

12.4.3. تحليل التقارب للأهداف المحدبة Convergence Analysis for Convex Objectives

يعد تحليل التقارب convergence analysis التالي للانحدار الاشتقاقي العشوائي لدوال الهدف المحدبة اختياريًا ويعمل بشكل أساسي على نقل المزيد من الحدس حول المشكلة. نحن نقتصر على واحد من أبسط البراهين (Nesterov and Vial، 2000). توجد تقنيات إثبات أكثر تقدمًا بشكل ملحوظ، على سبيل المثال، عندما تكون دالة الهدف حسنة التصرف بشكل خاص.

افترض أن دالة الهدف $f(\boldsymbol{\xi}, \mathbf{x})$ محدبة للجميع. بشكل أكثر تحديدا، نحن نعتبر تحديث الانحدار الاشتقاقي العشوائي:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \partial_{\mathbf{x}} f(\boldsymbol{\xi}_t, \mathbf{x}),$$

حيث $f(\boldsymbol{\xi}_t, \mathbf{x})$ هي دالة الهدف فيما يتعلق بمثال التدريب $\boldsymbol{\xi}_t$ المأخوذ من بعض التوزيعاتفي الخطوة t و$\mathbf{x}$ هي معلمة النموذج. يشار اليه

$$R(\mathbf{x}) = E_{\xi}[f(\xi, \mathbf{x})]$$

المخاطر المتوقعة R^* وبأدنى حد لها فيما يتعلق $\mathbf{x}$. لتكن $\mathbf{x}^*$ المصغر minimizer (نفترض أنه موجود داخل المجال الذي تم تعريف $\mathbf{x}$ فيه).في هذه الحالة، يمكننا تتبع المسافة بين المعلمة الحالية $\mathbf{x}_t$في الوقت المحدد t ومُقلِّل المخاطر $\mathbf{x}^*$ ومعرفة ما إذا كانت تتحسن بمرور الوقت:

$$\begin{aligned} & \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \\ = & \|\mathbf{x}_t - \eta_t \partial_{\mathbf{x}} f(\xi_t, \mathbf{x}) - \mathbf{x}^*\|^2 \\ = & \|\mathbf{x}_t - \mathbf{x}^*\|^2 + \eta_t^2 \|\partial_{\mathbf{x}} f(\xi_t, \mathbf{x})\|^2 - 2\eta_t \langle \mathbf{x}_t - \mathbf{x}^*, \partial_{\mathbf{x}} f(\xi_t, \mathbf{x}) \rangle. \end{aligned}$$

نحن نفترض أن معيار ℓ_2 للانحدار الاشتقاقي العشوائي $\partial_{\mathbf{x}} f(\xi_t, \mathbf{x})$ يحده بعض الثوابت L ، ومن ثم لدينا ذلك

$$\eta_t^2 \|\partial_{\mathbf{x}} f(\xi_t, \mathbf{x})\|^2 \leq \eta_t^2 L^2. \tag{12.4.9}$$

نحن مهتمونفي الغالب بكيفية تغير المسافة بين $\mathbf{x}_t$ و$\mathbf{x}^*$ تتغيرفي التوقعات.في الواقع، بالنسبة لأي تسلسل محدد من الخطوات، قد تزيد المسافة بشكل جيد، اعتمادًا على ξ_t الذي نواجهه. ومن ثم نحتاج إلى ربط حاصل الضرب النقطي. نظرًا لأنه بالنسبة لأي دالة محدبة f ، فإنها تحمل $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle f'(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$ لكل $\mathbf{x}$ و $\mathbf{y}$ ، ومن خلال التحدب لدينا

$$f(\xi_t, \mathbf{x}^*) \geq f(\xi_t, \mathbf{x}_t) + \langle \mathbf{x}^* - \mathbf{x}_t, \partial_{\mathbf{x}} f(\xi_t, \mathbf{x}_t) \rangle. \tag{12.4.10}$$

بإدخال كل من المتباينات (12.4.9) و (12.4.10)في (12.4.8) نحصل على حد للمسافة بين المعلماتفي الوقت $t + 1$ على النحو التالي:

$$\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \geq 2\eta_t (f(\xi_t, \mathbf{x}_t) - f(\xi_t, \mathbf{x}^*)) - \eta_t^2 L^2. \tag{12.4.11}$$

هذا يعني أننا نحرز تقدمًا طالما أن الفرق بين الخسارة الحالية والخسارة المثلى يفوق $\eta_t L^2/2$. نظرًا لأن هذا الاختلاف لا بد أن يتقارب مع الصفر، فهذا يعني أن معدل التعلم η_t يحتاج أيضًا إلى التلاشي vanish.

بعد ذلك نأخذ التوقعاتفي (12.4.11). هذا ينتج

$$E[\|\mathbf{x}_t - \mathbf{x}^*\|^2] - E[\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2] \geq 2\eta_t [E[R(\mathbf{x}_t)] - R^*] - \eta_t^2 L^2.$$

تتضمن الخطوة الأخيرة جمع المتباينات لـ $t \in \{1, \ldots, T\}$. لان مجموع التلسكوبات وإسقاط الحد الأدنى نحصل على

$$\|\mathbf{x}_1 - \mathbf{x}^*\|^2 \geq 2\left(\sum_{t=1}^{T} \eta_t\right)[E[R(\mathbf{x}_t)] - R^*] - L^2 \sum_{t=1}^{T} \eta_t^2.$$

لاحظ أننا استغلنا $\mathbf{x}_1$المعطى وبالتالي يمكن إسقاط التوقع. آخر تعريف

$$\overline{\mathbf{x}} \stackrel{\text{def}}{=} \frac{\sum_{t=1}^{T} \eta_t \mathbf{x}_t}{\sum_{t=1}^{T} \eta_t}.$$

لان

$$E\left(\frac{\sum_{t=1}^{T} \eta_t R(\mathbf{x}_t)}{\sum_{t=1}^{T} \eta_t}\right) = \frac{\sum_{t=1}^{T} \eta_t E[R(\mathbf{x}_t)]}{\sum_{t=1}^{T} \eta_t} = E[R(\mathbf{x}_t)],$$

من خلال متباينة ينسن (إعداد $i = t$، في (12.2.3)) وتحدب R يتبع $E[R(\mathbf{x}_t)] \geq$ $E[R(\overline{\mathbf{x}})]$، لذا

$$\sum_{t=1}^{T} \eta_t E[R(\mathbf{x}_t)] \geq \sum_{t=1}^{T} \eta_t E[R(\overline{\mathbf{x}})].$$

بالتعويض في المتباينة (12.4.13) نحصل على الحد

$$[E[\overline{\mathbf{x}}]] - R^* \leq \frac{r^2 + L^2 \sum_{t=1}^{T} \eta_t^2}{2 \sum_{t=1}^{T} \eta_t},$$

حيث $r^2 \stackrel{\text{def}}{=} \|\mathbf{x}_1 - \mathbf{x}^*\|^2$ هي حدود المسافة بين الاختيار الأولي للمعلمات والنتيجة النهائية. باختصار، تعتمد سرعة التقارب على كيفية تقييد معيار الانحدار الاشتقاقي العشوائي (L) ومدى بُعد قيمة المعلمة الأولية (r) عن المثالية. لاحظ أن الحد من حيث $\overline{\mathbf{x}}$ عوضا عن $\mathbf{x}_T$. هذا هو الحال لان $\overline{\mathbf{x}}$ هي نسخة متجانسة من مسار التحسين. كلما كانت r, L وT معروفة، يمكننا اختيار معدل التعلم $\eta = r/(L\sqrt{T})$. ينتج هذا الحد الأعلى $rL/\sqrt{T}$. أي أننا نتقارب مع المعدل $\mathcal{O}(1/\sqrt{T})$ إلى الحل الأمثل.

12.4.4. الانحدارات العشوائية والعينات النهائية Stochastic Gradients and Finite Samples

لقد لعبنا حتى الآن بعض الشيء بسرعة وبشكل فضفاض عندما يتعلق الأمر بالحديث عن الانحدار الاشتقاقي العشوائي. افترضنا أننا نرسم أمثلة x_i، عادةً مع تسميات y_i من بعض التوزيعات $p(x, y)$ وأننا نستخدم هذا لتحديث معلمات النموذج بطريقة ما. على وجه الخصوص، بالنسبة لحجم العينة المحدود، ناقشنا ببساطة بأن التوزيع المتقطع $p(x, y) = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}(x)\delta_{y_i}(y)$ لبعض الدوال δ_{x_i} وδ_{y_i} يسمح لنا بأداء اتحدار اشتقاقي عشوائي عليه.

ومع ذلك، فهذا ليس ما فعلناه حقًا.في أمثلة اللعبةفي القسم الحالي، أضفنا ضوضاء ببساطة إلى انحدار غير عشوائي، أي تظاهرنا بوجود أزواج (x_i, y_i). اتضح أن هذا مبرر هنا (انظر التمارين للحصول على مناقشة مفصلة). الأمر الأكثر إثارة للقلق هو أنه من الواضح أننا لم نفعل ذلكفي جميع المناقشات السابقة. بدلاً من ذلك، قمنا بالتكرارفي جميع الحالات مرة واحدة بالضبط. لمعرفة سبب تفضيل ذلك، ضعفي اعتبارك العكس، أي أننا نأخذ عينات من n مشاهدات من التوزيع المنفصل مع الاستبدال replacement. احتمال اختيار عنصر عشوائيًا هو $1/n$. وبالتالي اختياره مرة واحدة على الأقل

$$P(\text{choose } i) = 1 - P(\text{omit } i) = 1 - (1 - 1/n)^n \approx 1 - e^{-1} \approx 0.63.$$

يوضح المنطق المماثل أن احتمالية اختيار عينة ما (أي مثال تدريبي) مرة واحدة بالضبط يتم تقديمها من قبل

$$\binom{n}{1}\frac{1}{n}(1-\frac{1}{n})^{n-1} = \frac{n}{n-1}(1-\frac{1}{n})^n \approx e^{-1} \approx 0.37.$$

يؤدي أخذ العينات Sampling مع الاستبدال replacement إلى زيادة التباين وانخفاض كفاءة البيانات بالنسبة لأخذ العينات دون استبدال. ومن ثم، فإننافي الواقع نقوم بتنفيذ الخيار الأخير (وهذا هو الخيار الافتراضيفي جميع أنحاء هذا الكتاب). لاحظ أخيرًا أن التمريرات المتكررة عبر مجموعة بيانات التدريب تجتازها بترتيب عشوائي مختلف.

12.4.5. الملخص

- بالنسبة للمشكلات المحدبة، يمكننا إثبات أنه بالنسبة لمجموعة واسعة من معدلات التعلم، فإن الانحدار الاشتقاقي العشوائي سوف يتقارب مع الحل الأمثل.
- بالنسبة للتعلم العميق، هذا ليس هو الحال بشكل عام. ومع ذلك، فإن تحليل المشكلات المحدبة يعطينا نظرة ثاقبة مفيدة حول كيفية التعامل مع التحسين، أي تقليل معدل التعلم تدريجياً، وإن لم يكن بسرعة كبيرة.
- تحدث المشكلات عندما يكون معدل التعلم صغيرًا جدًا أو كبيرًا جدًا. من الناحية العملية، غالبًا ما يتم العثور على معدل التعلم المناسب فقط بعد تجارب متعددة.
- عندما يكون هناك المزيد من الأمثلةفي مجموعة بيانات التدريب، فإن حساب كل تكرار للانحدار الاشتقاقي يكلف أكثر، لذلك يفضل الانحدار الاشتقاقي العشوائيفي هذه الحالات.
- الضمانات الامثلية للانحدار الاشتقاقي العشوائي غير متوفرة بشكل عامفي الحالات غير المحدبة نظرًا لأن عدد الحدود الدنيا المحلية التي تتطلب التحقق قد يكون أسيًا.

12.4.6. التمارين

1. جرب جداول معدل التعلم المختلفة للانحدار الاشتقاقي العشوائي وبأعداد مختلفة من التكرارات. على وجه الخصوص، ارسم المسافة من الحل الأمثل (0,0) كدالة لعدد التكرارات.
2. اثبت ذلك للدالة $f(x_1, x_2) = x_1^2 + 2x_2^2$ إن إضافة ضوضاء عادية إلى الانحدار يعادل تقليل دالة الخسارة $f(\mathbf{x}, \mathbf{w}) = (x_1 - w_1)^2 + 2(x_2 - w_2)^2$ حيث $\mathbf{x}$ يتم استخلاصه من التوزيع الطبيعي.
3. قارن تقارب الانحدار الاشتقاقي العشوائي عند أخذ العينة من $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ مع الاستبدال وعند أخذ العينة بدون استبدال.
4. كيف يمكنك تغيير أداة حل الانحدار الاشتقاقي العشوائي إذا كان بعض الانحدار (أو بالأحرى بعض الإحداثيات المرتبطة به) أكبر باستمرار من جميع الانحدارات الأخرى؟
5. افترض أن $f(x) = x^2(1 + \sin x)$. كم عدد الحد الأدنى المحلي local minima لدى f؟ هل يمكنك تغيير f بطريقة تجعل المرء يحتاج إلى تقييم كل الحدود الدنيا المحلية لتقليله؟

12.5. الانحدار الاشتقاقي ذو الدفعات الصغيرة Minibatch Stochastic Gradient Descent

لقد واجهنا حتى الآن طرفي نقيضفي نهج التعلم القائم على التدرج (الانحدار): يستخدم القسم 12.3 مجموعة البيانات الكاملة لحساب التدرجات وتحديث المعلمات، مرة واحدةفي كل مرة. على العكس من ذلك، يعالج القسم 12.4 مثالًا تدريبيًا واحدًافي كل مرة لإحراز تقدم. كلاهما له عيوبه الخاصة. لا يعتبر الانحدار الاشتقاقي Gradient descent فعالًافي استخدام البيانات بشكل خاص عندما تكون البيانات متشابهة جدًا. لا يعتبر الانحدار الاشتقاقي العشوائي Stochastic gradient descent فعالاً من الناحية الحسابية بشكل خاص لأن وحدات المعالجة المركزية (CPU) ووحدات معالجة الرسومات (GPU) لا يمكنها استغلال القوة الكاملة للفيكتورايزيشن vectorization. يشير هذا إلى أنه قد يكون هناك شيء ما بينهما، وفي الواقع، هذا هو ما استخدمناه حتى الآنفي الأمثلة التي ناقشناها.

12.5.1. الفيكتورايزشن والذواكر المخبئية Vectorization and Caches

تعد الكفاءة الحسابيةفي صميم قرار استخدام الدفعات الصغيرة minibatches. يمكن فهم ذلك بسهولة أكبر عند التفكيرفي الموازاة مع وحدات معالجة رسومات متعددة وخوادم متعددة.في هذه الحالة، نحتاج إلى إرسال صورة واحدة على الأقل إلى كل وحدة معالجة رسومات. مع وجود 8

وحدات معالجة رسومات لكل خادم و16 خادمًا، وصلنا بالفعل إلى حجم صغير لا يقل عن 128.

تكون الأمور أكثر دقة قليلاً عندما يتعلق الأمر بوحدات معالجة الرسومات الفردية أو حتى وحدات المعالجة المركزية. تحتوي هذه الأجهزة على أنواع متعددة من الذاكرة، وغالبًا ما تكون أنواعًا متعددة من الوحدات الحسابية وقيود عرض نطاق مختلفة فيما بينها. على سبيل المثال، تحتوي وحدة المعالجة المركزية على عدد صغير من السجلات registers ثم L1 و L2 وفي بعض الحالات حتى ذاكرة التخزين المخبئية L3 cache (والتي يتم مشاركتها بين نوى المعالج processor cores المختلفة). هذه ذاكرات التخزين المخبئية ذات حجم ووقت استجابة متزايد (وفي نفس الوقت تتناقص من عرض النطاق الترددي bandwidth). يكفي القول، المعالج قادر على تنفيذ العديد من العمليات أكثر مما تستطيع واجهة الذاكرة الرئيسية توفيره.

أولاً، يمكن لوحدة المعالجة المركزية بسرعة 2 جيجاهرتز مع 16 نواة و AVX-512 معالجة ما يصل إلى $2 \cdot 10^9 \cdot 16 \cdot 32 = 10^{12}$ بايتفي الثانية. تتجاوز قدرة وحدات معالجة الرسومات هذا الرقم بسهولة بمعامل 100. ومن ناحية أخرى، قد لا يحتوي معالج الخادم متوسط المدى على عرض نطاق ترددي يزيد عن 100 غيغابايت / ثانية، أي أقل من عُشر ما هو مطلوب للحفاظ على المعالج تغذيها. لجعل الأمور أسوأ، لا يتم إنشاء جميع عمليات الوصول إلى الذاكرة على قدم المساواة: عادةً ما تكون واجهات الذاكرة بعرض 64 بت أو أوسع (على سبيل المثال، على وحدات معالجة الرسومات حتى 384 بت)، وبالتالي فإن قراءة بايت واحد تتحمل تكلفة وصول أوسع بكثير.

ثانيًا، هناك عبء كبير للوصول الأولفي حين أن الوصول التسلسلي رخيص نسبيًا (يُسمى هذا غالبًا قراءة الاندفاع burst read). هناك العديد من الأشياء التي يجب وضعهافي الاعتبار، مثل التخزين المؤقت عندما يكون لدينا مقابس متعددة، وشرائح، وهياكل أخرى. راجع مقالة ويكيبيديا هذه لمزيد من المناقشة المتعمقة.

تتمثل طريقة تخفيف هذه القيودفي استخدام تسلسل هرمي لذاكرة التخزين المخبئية لوحدة المعالجة المركزية التي تكونفي الواقع سريعة بما يكفي لتزويد المعالج بالبيانات. هذه هي القوة المحركة driving force وراء التجميع batchingفي التعلم العميق. لتبسيط الأمور، ضعفي اعتبارك ضرب مصفوفةـمصفوفة، على سبيل المثال $\mathbf{A} = \mathbf{BC}$. لدينا عدد من الخيارات لحساب $\mathbf{A}$. على سبيل المثال، يمكننا تجربة ما يلي:

1. يمكننا حساب $\mathbf{A}_{ij} = \mathbf{B}_{i,:}\mathbf{C}_{:,j}$ ، أي يمكننا حسابها بطريقة عنصرية elementwise عن طريق حاصل الضرب النقطي.

2. يمكننا حساب $\mathbf{A}_{:,j} = \mathbf{BC}_{:,j}$ ، أي يمكننا حسابه بعمود واحدفي كل مرة. وبالمثل يمكننا حساب $\mathbf{A}$ صف واحد $\mathbf{A}_{i,:}$في كل مرة.
3. يمكننا ببساطة حساب $\mathbf{A} = \mathbf{BC}$.
4. يمكننا تقسيم $\mathbf{B}$ و $\mathbf{C}$ إلى مصفوفات كتلة أصغر وحساب $\mathbf{A}$ كتلة واحدةفي كل مرة.

إذا اتبعنا الخيار الأول، فسنحتاج إلى نسخ متجه صف واحد وعمود واحدفي وحدة المعالجة المركزيةفي كل مرة نريد حساب عنصر $\mathbf{A}_{ij}$. والأسوأ من ذلك، نظرًا لحقيقة أن عناصر المصفوفة تتم محاذاتها بالتتابع، فإننا مطالبون بالوصول إلى العديد من المواقع المنفصلة لأحد المتجهين أثناء قراءتها من الذاكرة. الخيار الثاني أكثر ملاءمة.في ذلك، يمكننا الاحتفاظ بمتجه العمود $\mathbf{C}_{:,j}$ في ذاكرة التخزين المؤقت لوحدة المعالجة المركزية بينما نستمرفي عبور $\mathbf{B}$. هذا يقلل من متطلبات عرض النطاق الترددي للذاكرة إلى النصف مع وصول أسرعفي المقابل. بالطبع، الخيار الثالث مرغوب فيه للغاية. لسوء الحظ، قد لا تتناسب معظم المصفوفات تمامًا مع ذاكرة التخزين المؤقت (هذا ما نناقشه بعد كل شيء). ومع ذلك، يقدم الخيار الرابع بديلاً مفيدًا عمليًا: يمكننا نقل كتل المصفوفة إلى ذاكرة التخزين المؤقت ومضاعفتها محليًا. المكتبات المُحسَّنة تعتني بهذا الأمر من أجلنا. دعونا نلقي نظرة على مدى كفاءة هذه العملياتفي الممارسة العملية.

إلى جانب الكفاءة الحسابية computational efficiency، فإن الحمل overhead التي تقدمها بايثون وإطار التعلم العميق نفسه كبيرة. تذكر أنهفي كل مرة ننفذ فيها أمرًا، يرسل مترجم بايثون أمرًا إلى محرك TensorFlow والذي يحتاج إلى إدراجهفي الرسم البياني الحسابي والتعامل معه أثناء الجدولة. يمكن أن تكون هذه overhead ضارة للغاية. باختصار، يُنصح بشدة باستخدام الفيكتورايزيشن (والمصفوفات) كلما أمكن ذلك.

```
%matplotlib inline
import time
import numpy as np
import tensorflow as tf
from d2l import tensorflow as d2l

A = tf.Variable(tf.zeros((256, 256)))
B = tf.Variable(tf.random.normal([256, 256], 0, 1))
C = tf.Variable(tf.random.normal([256, 256], 0, 1))
```

نظرًا لأننا سنقوم بقياس وقت التشغيل بشكل متكررفي بقية الكتاب، فلنحدد مؤقتًا timer.

```
class Timer:  #@save
    """Record multiple running times."""
    def __init__(self):
        self.times = []
```

```
        self.start()

    def start(self):
        """Start the timer."""
        self.tik = time.time()

    def stop(self):
        """Stop the timer and record the time in a
list."""
        self.times.append(time.time() - self.tik)
        return self.times[-1]

    def avg(self):
        """Return the average time."""
        return sum(self.times) / len(self.times)

    def sum(self):
        """Return the sum of time."""
        return sum(self.times)

    def cumsum(self):
        """Return the accumulated time."""
        return np.array(self.times).cumsum().tolist()

timer = Timer()
```

يتكرر التخصيص حسب العنصر Element-wise assignment ببساطةفي جميع الصفوف والأعمدة لـ **B** و **C** على التوالي لتعيين القيمة الى **A**.

```
# Compute A = BC one element at a time
timer.start()
for i in range(256):
    for j in range(256):
        A[i, j].assign(tf.tensordot(B[i, :], C[:, j],
axes=1))
timer.stop()
```

```
113.0004358291626
```

تتمثل الإستراتيجية الأسرعفي إجراء التخصيص حسب العمود column-wise assignment.

```
timer.start()
for j in range(256):
    A[:, j].assign(tf.tensordot(B, C[:, j], axes=1))
timer.stop()
```

```
0.35082268714904785
```

أخيرًا، الطريقة الأكثر فاعلية هي إجراء العملية بأكملهافي كتلة واحدة. لاحظ أن ضرب أي مصفوفتين $\mathbf{B} \in \mathbb{R}^{m \times n}$ يأخذ عمليات الفاصلة العائمة تقريبًا $2mnp$ ، عندما يتم احتساب الضرب القياسي والإضافة كعمليات منفصلة (مدمجةفي الممارسة). وبالتالي، فإن ضرب مصفوفتين 256×256 يتطلب 0.03 مليار عملية فاصلة عائمة. دعونا نرى ما هي السرعة ذات الصلة للعمليات.

```
timer.start()
A.assign(tf.tensordot(B, C, axes=1))
timer.stop()

gigaflops = [0.03 / i for i in timer.times]
print(f'performance in Gigaflops: element
{gigaflops[0]:.3f}, '
      f'column {gigaflops[1]:.3f}, full
{gigaflops[2]:.3f}')
```

```
performance in Gigaflops: element 0.000, column 0.086,
full 1.254
```

12.5.2. الدفعات الصغيرة Minibatches

في الماضي كنا نعتبر أنه من المسلم به أننا سنقرأ الدفعات الصغيرة minibatches من البيانات بدلاً من المشاهدات الفردية لتحديث المعلمات. نقدم الآن تبريرا موجزا لذلك. تتطلب منا معالجة المشاهدات الفردية إجراء العديد من ضرب متجه–المصفوفة الفردية (أو حتى ضرب المتجه–المتجه)، وهي مكلفة للغاية وتتحمل overhead كبيرة نيابة عن إطار التعلم العميق الأساسي. ينطبق هذا على كل من تقييم الشبكة عند تطبيقها على البيانات (يشار إليها غالبًا بالاستدلال inference) وعند حساب الانحدارات لتحديث المعلمات. هذا هو، هذا ينطبق كلما أدينا $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \mathbf{g}_t$ عندما

$$\mathbf{g}_t = \partial_{\mathbf{w}} f(\mathbf{x}_t, \mathbf{w})$$

يمكننا زيادة الكفاءة الحسابية لهذه العملية من خلال تطبيقها على دفعة صغيرة من المشاهدات في كل مرة. أي أننا نستبدل الانحدار $\mathbf{g}_t$ على مشاهدة واحدة بواحد على دفعة صغيرة

$$\mathbf{g}_t = \partial_{\mathbf{w}} \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} f(\mathbf{x}_i, \mathbf{w})$$

دعنا نرى ما يفعله هذا بالخصائص الإحصائية لـ $\mathbf{g}_t$: نظرًا لأن $\mathbf{x}_t$ وكل من عناصر الدفعات الصغيرة $\mathcal{B}_t$ يتم رسمها بشكل منتظم عشوائيًا من مجموعة التدريب، فإن توقع الانحدار يظل

دون تغيير. من ناحية أخرى، يتم تقليل التباين بشكل كبير. نظرًا لأن الانحدار الاشتقاقي المصغر minibatch gradient يتكون من $b := |\mathcal{B}_t|$ انحدارات مستقلة يتم حساب متوسطها، ويتم تقليل انحرافها المعياري بعامل $b^{-\frac{1}{2}}$. هذا،في حد ذاته، يعد أمرًا جيدًا، لأنه يعني أن التحديثات تتوافق بشكل أكثر موثوقية مع الانحدار الاشتقاقي الكامل.

قد يشير هذا بسذاجة إلى أن اختيار الدفعات الصغيرة $\mathcal{B}_t$ كبير سيكون مرغوبًا عالميًا. للأسف، بعد نقطة ما، يكون الانخفاض الإضافيفي الانحراف المعياري ضئيلًا عند مقارنته بالزيادة الخطية في التكلفة الحسابية. من الناحية العملية، نختار الدفعات الصغيرة الكبير بما يكفي لتقديم كفاءة حسابية جيدة مع الاستمرارفي تركيبهفي ذاكرة وحدة معالجة الرسومات. لتوضيح المدخرات، دعونا نلقي نظرة على بعض التعليمات البرمجية. نقوم فيه بنفس عملية ضرب المصفوفة ـ المصفوفة، ولكن هذه المرة تم تقسيمها إلى "دفعات صغيرة" تتكون من 64 عمودًافي المرة الواحدة.

```
timer.start()
for j in range(0, 256, 64):
    A[:, j:j+64].assign(tf.tensordot(B, C[:, j:j+64],
axes=1))
timer.stop()
print(f'performance in Gigaflops: block {0.03 /
timer.times[3]:.3f}')
```

```
performance in Gigaflops: block 3.947
```

كما نرى، فإن الحساب على الدفعات الصغيرة فعال بشكل أساسي كما هو الحالفي المصفوفة الكاملة. وهناك كلمة تحذيرفي محله.في القسم 8.5، استخدمنا نوعًا من التنظيم الذي كان يعتمد بشكل كبير على مقدار التباينفي الدفعات الصغيرة. كلما زدنا الأخير، قل التباين ومعه فائدة حقن الضوضاء بسبب تسوية الدُفعات. انظر على سبيل المثال، Ioffe (2017) للحصول على تفاصيل حول كيفية إعادة قياس المصطلحات المناسبة وحسابها.

12.5.3. قراءة مجموعة البيانات Reading the Dataset

دعونا نلقي نظرة على كيفية إنشاء الدفعات الصغيرة بكفاءة من البيانات. فيما يلي نستخدم مجموعة بيانات طورتها وكالة ناسا لاختبار ضوضاء الجناح من طائرات مختلفة لمقارنة خوارزميات التحسين هذه. للسهولة، نستخدم اول 1,500 مثال فقط. يتم تبييض البيانات للمعالجة المسبقة، أي نزيل المتوسط ونعيد قياس التباين لكل 1 تنسيق.

```
#@save
d2l.DATA_HUB['airfoil'] = (d2l.DATA_URL +
'airfoil_self_noise.dat',
```

```
'76e5be1548fd8222e5074cf0faae75edff8cf93f')

#@save
def get_data_ch11(batch_size=10, n=1500):
    data = np.genfromtxt(d2l.download('airfoil'),
                         dtype=np.float32,
delimiter='\t')
    data = (data - data.mean(axis=0)) / data.std(axis=0)
    data_iter = d2l.load_array((data[:n, :-1], data[:n,
-1]),
                               batch_size,
is_train=True)
    return data_iter, data.shape[1]-1
```

12.5.4. التنفيذ من البداية Implementation from Scratch

تذكر تنفيذ الانحدار الاشتقاقي العشوائي المصغر minibatch stochastic gradient descent من القسم 3.4. فيما يلي نقدم تطبيقًا أكثر عمومية قليلاً. للسهولة، يحتوي على نفس توقيع الاستدعاء مثل خوارزميات التحسين الأخرى التي تم تقديمها لاحقًافي هذا الفصل. على وجه التحديد، نضيف حالات إدخال `states` ونضع المعلمة الفائقةفي قاموس المعامِلات الفائقة `hyperparams`. بالإضافة إلى ذلك، سنقوم بمتوسط خسارة كل مثال من امثلة الدفعات المصغرةفي دالة التدريب، وبالتالي لا يحتاج الانحدارفي خوارزمية التحسين إلى تقسيمه على حجم الدُفعة.

```
def sgd(params, grads, states, hyperparams):
    for param, grad in zip(params, grads):
        param.assign_sub(hyperparams['lr']*grad)
```

بعد ذلك، نقوم بتنفيذ دالة تدريب عامة لتسهيل استخدام خوارزميات التحسين الأخرى المقدمة لاحقًافي هذا الفصل. يقوم بتهيئة نموذج الانحدار الخطي ويمكن استخدامه لتدريب النموذج باستخدام الانحدار الاشتقاقي العشوائي المصغر والخوارزميات الأخرى التي تم تقديمها لاحقًا.

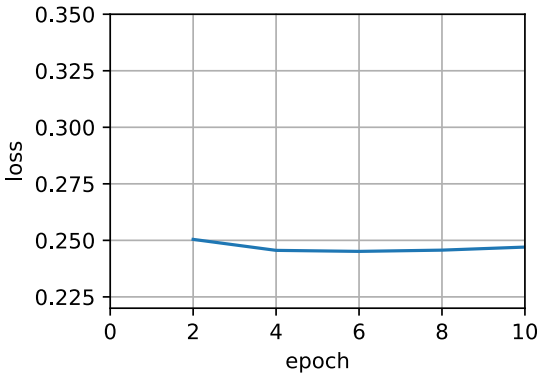
```
#@save
def train_ch11(trainer_fn, states, hyperparams,
data_iter,
               feature_dim, num_epochs=2):
    # Initialization
    w = tf.Variable(tf.random.normal(shape=(feature_dim,
1),
                                     mean=0,
stddev=0.01),trainable=True)
```

```
    b = tf.Variable(tf.zeros(1), trainable=True)

    # Train
    net, loss = lambda X: d2l.linreg(X, w, b),
d2l.squared_loss
    animator = d2l.Animator(xlabel='epoch',
ylabel='loss',
                            xlim=[0, num_epochs],
ylim=[0.22, 0.35])
    n, timer = 0, d2l.Timer()

    for _ in range(num_epochs):
        for X, y in data_iter:
          with tf.GradientTape() as g:
            l = tf.math.reduce_mean(loss(net(X), y))

          dw, db = g.gradient(l, [w, b])
          trainer_fn([w, b], [dw, db], states,
hyperparams)
          n += X.shape[0]
          if n % 200 == 0:
              timer.stop()
              p = n/X.shape[0]
              q =
p/tf.data.experimental.cardinality(data_iter).numpy()
              r = (d2l.evaluate_loss(net, data_iter,
loss),)
              animator.add(q, r)
              timer.start()
    print(f'loss: {animator.Y[0][-1]:.3f},
{timer.avg():.3f} sec/epoch')
    return timer.cumsum(), animator.Y[0]
```

دعونا نرى كيف تستمر عملية التحسينفي الانحدار الاشتقاقي العشوائي المصغر. يمكن تحقيق ذلك عن طريق ضبط حجم الدفعات الصغيرة على 1500 (أي إلى العدد الإجمالي للأمثلة). ونتيجة لذلك، يتم تحديث معلمات النموذج مرة واحدة فقطفي كل فترة. هناك تقدم ضئيل.في الواقع، توقف التقدم بعد 6 خطوات.

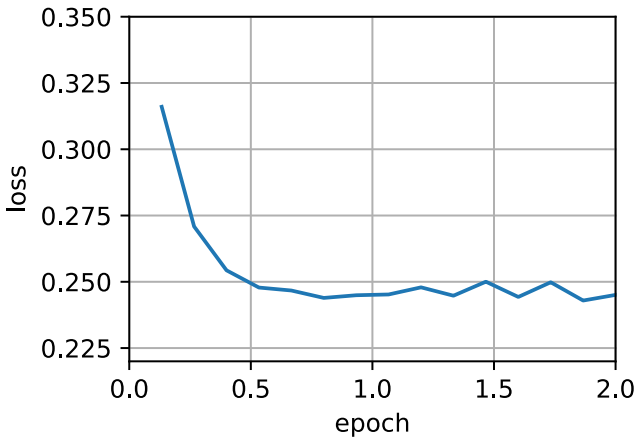
```
def train_sgd(lr, batch_size, num_epochs=2):
    data_iter, feature_dim = get_data_ch11(batch_size)
    return train_ch11(
```

```
        sgd, None, {'lr': lr}, data_iter, feature_dim,
num_epochs)

gd_res = train_sgd(1, 1500, 10)
loss: 0.247, 0.032 sec/epoch
```



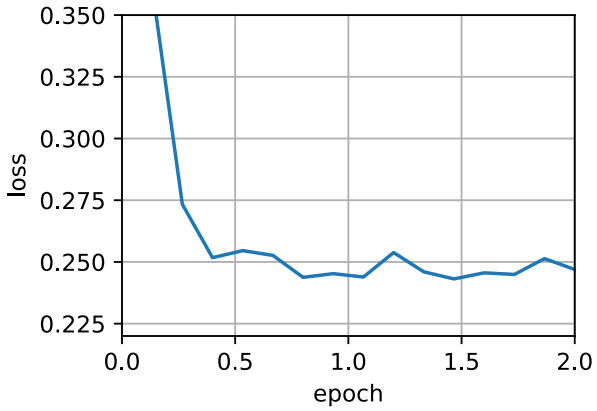
عندما يساوي حجم الدُفعة 1، فإننا نستخدم الانحدار الاشتقاقي العشوائي للتحسين. لتبسيط التنفيذ، اخترنا معدل تعلم ثابتًا (وإن كان صغيرًا).في الانحدار الاشتقاقي العشوائي، يتم تحديث معلمات النموذج كلما تمت معالجة أحد الأمثلة.في حالتنا هذا يصل إلى 1500 تحديث لكل فترة. كما نرى، يتباطأ الانخفاضفي قيمة دالة الهدف بعد فترة واحدة. على الرغم من أن كلا الإجرائين عالجتا 1500 مثالفي فترة واحدة، فإن الانحدار الاشتقاقي العشوائي يستهلك وقتًا أطول من الانحدار الاشتقاقيفي تجربتنا. وذلك لأن الانحدار الاشتقاقي العشوائي قام بتحديث المعلمات بشكل متكرر ولأنه أقل كفاءةفي معالجة الملاحظات الفردية واحدة تلو الأخرى.

```
sgd_res = train_sgd(0.005, 1)
loss: 0.245, 0.543 sec/epoch
```



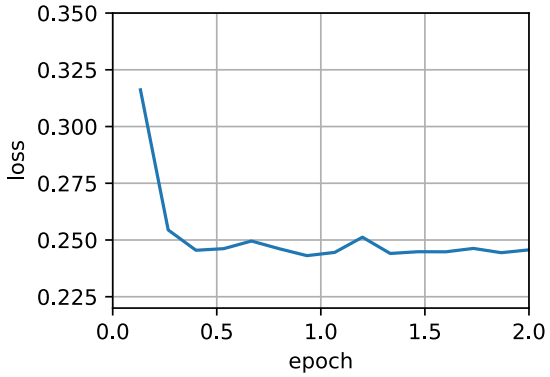
أخيرًا، عندما يساوي حجم الدُفعة 100، فإننا نستخدم الانحدار الاشتقاقي العشوائي المصغر للتحسين. الوقت المطلوب لكل فترة أقصر من الوقت اللازم للانحدار الاشتقاقي العشوائي ووقت الانحدار الاشتقاقي العشوائي المصغر.

```
mini1_res = train_sgd(.4, 100)
```
```
loss: 0.247, 0.008 sec/epoch
```



تقليل حجم الدُفعة إلى 10، يزداد الوقت لكل فترة لأن عبء العمل لكل دفعة أقل كفاءةفي التنفيذ.

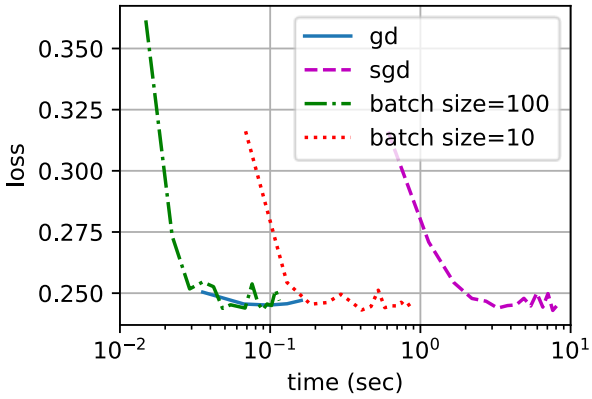
```
mini2_res = train_sgd(.05, 10)
```
```
loss: 0.246, 0.057 sec/epoch
```



الآن يمكننا مقارنة الوقت مقابل الضياعفي التجارب الأربع السابقة. كما يمكن رؤيته، على الرغم من أن الانحدار الاشتقاقي العشوائي SGD يتقارب بشكل أسرع من الانحدار الاشتقاقي GD من حيث عدد الأمثلة التي تتم معالجتها، إلا أنه يستخدم وقتًا أطول للوصول إلى نفس الخسارة مقارنة بـ GD لأن حساب مثال التدرج على سبيل المثال ليس بنفس الكفاءة. إن الانحدار الاشتقاقي العشوائي لـ Minibatch قادر على مفاضلة سرعة التقارب وكفاءة الحساب. يعتبر

حجم الدفعات الصغيرة البالغ 10 أكثر كفاءة من SGD؛ يتفوق حجم الدفعات الصغيرة البالغ 100 حتى على GD من حيث وقت التشغيل.

```
d2l.set_figsize([6, 3])
d2l.plot(*list(map(list, zip(gd_res, sgd_res, mini1_res,
mini2_res))),
         'time (sec)', 'loss', xlim=[1e-2, 10],
         legend=['gd', 'sgd', 'batch size=100', 'batch
size=10'])
d2l.plt.gca().set_xscale('log')
```



12.5.5. التنفيذ المختصر Concise Implementation

في Gluon، يمكننا استخدام فئة `Trainer` لاستدعاء خوارزميات التحسين. يستخدم هذا لتنفيذ دالة تدريب عامة. سوف نستخدم هذا خلال الفصل الحالي.

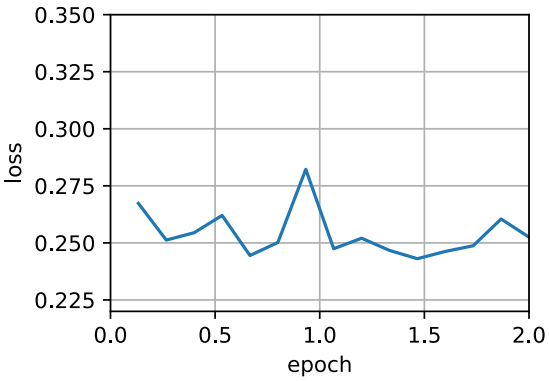
```
#@save
def train_concise_ch11(trainer_fn, hyperparams,
data_iter, num_epochs=2):
    # Initialization
    net = tf.keras.Sequential()
    net.add(tf.keras.layers.Dense(1,

kernel_initializer=tf.random_normal_initializer(stddev=0
.01)))
    optimizer = trainer_fn(**hyperparams)
    loss = tf.keras.losses.MeanSquaredError()
    animator = d2l.Animator(xlabel='epoch',
ylabel='loss',
```

```
                                xlim=[0, num_epochs],
ylim=[0.22, 0.35])
    n, timer = 0, d2l.Timer()
    for _ in range(num_epochs):
        for X, y in data_iter:
            with tf.GradientTape() as g:
                out = net(X)
                l = loss(y, out)
                params = net.trainable_variables
                grads = g.gradient(l, params)
            optimizer.apply_gradients(zip(grads,
params))
            n += X.shape[0]
            if n % 200 == 0:
                timer.stop()
                p = n/X.shape[0]
                q =
p/tf.data.experimental.cardinality(data_iter).numpy()
                # `MeanSquaredError` computes squared
error without the 1/2
                # factor
                r = (d2l.evaluate_loss(net, data_iter,
loss) / 2,)
                animator.add(q, r)
                timer.start()
    print(f'loss: {animator.Y[0][-1]:.3f},
{timer.avg():.3f} sec/epoch')
```

يُظهر استخدام Gluon لتكرار التجربة الأخيرة سلوكًا متطابقًا.

```
data_iter, _ = get_data_ch11(10)
trainer = tf.keras.optimizers.SGD
train_concise_ch11(trainer, {'learning_rate': 0.05},
data_iter)
```

```
loss: 0.253, 0.101 sec/epoch
```



12.5.6. الملخص

- يجعل Vectorization الكود أكثر كفاءة نظرًا لتقليل overhead الناشئ عن إطار عمل التعلم العميق وبسبب موقع الذاكرة الأفضل والتخزين المخبئي على وحدات المعالجة المركزية ووحدات معالجة الرسومات.
- هناك مفاضلة بين الكفاءة الإحصائية الناشئة عن الانحدار الاشتقاقي العشوائي والكفاءة الحسابية الناشئة عن معالجة مجموعات كبيرة من البيانات في وقت واحد.
- يوفر الانحدار الاشتقاقي العشوائي المصغر أفضل ما في العالمين: الكفاءة الحسابية والإحصائية.
- في الانحدار الاشتقاقي العشوائي المصغر، نقوم بمعالجة مجموعات البيانات التي تم الحصول عليها من خلال التقليب العشوائي random permutation لبيانات التدريب (على سبيل المثال، تتم معالجة كل ملاحظة مرة واحدة فقط لكل فترة، وإن كان ذلك بترتيب عشوائي).
- من المستحسن أن تضمحل معدلات التعلم أثناء التدريب.
- بشكل عام، يكون الانحدار الاشتقاقي العشوائي المصغر أسرع من الانحدار الاشتقاقي العشوائي والانحدار الاشتقاقي للتقارب مع مخاطر أصغر، عند قياسه من حيث الوقت.

12.5.7. التمارين

1. قم بتعديل حجم الدُفعة ومعدل التعلم ولاحظ معدل الانخفاض لقيمة دالة الهدف والوقت المستهلك في كل فترة.
2. اقرأ وثائق MXNet واستخدم دالة `set_learning_rate` لفئة `Trainer` لتقليل معدل التعلم للانحدار الاشتقاقي العشوائي المصغر إلى 10/1 من قيمته السابقة بعد كل فترة.

3. قارن بين الانحدار الاشتقاقي العشوائي المصغر مع النوع الذي يعيّنفي الواقع مع الاستبدال replacement من مجموعة التدريب. ماذا يحدث؟
4. يقوم الجني الشرير بتكرار مجموعة البيانات الخاصة بك دون إخبارك (على سبيل المثال، تحدث كل ملاحظة مرتين وتنمو مجموعة البيانات الخاصة بك إلى ضعف حجمها الأصلي، ولكن لم يخبرك أحد بذلك). كيف يتغير سلوك الانحدار الاشتقاقي العشوائي والانحدار الاشتقاقي العشوائي المصغر والانحدار الاشتقاقي؟

12.6. الزخم Momentum

في القسم 12.4، راجعنا ما يحدث عند إجراء الانحدار الاشتقاقي العشوائي، أي عند إجراء التحسين حيث يتوفر فقط نوع صاخب noisy variant من الانحدار. على وجه الخصوص، لاحظنا أنه بالنسبة للتدرجات الصاخبة noisy gradients، نحتاج إلى توخي مزيد من الحذر عندما يتعلق الأمر باختيار معدل التعلمفي مواجهة الضوضاء. إذا قللناها بسرعة كبيرة، فإن التقارب يتوقف. إذا كنا متساهلين للغاية، فإننا نفشلفي الوصول إلى حل جيد بما يكفي لأن الضوضاء تستمرفي إبعادنا عن المثالية.

12.6.1. الأساسيات Basics

في هذا القسم، سوف نستكشف خوارزميات تحسين أكثر فاعلية، خاصة لأنواع معينة من مشاكل التحسين الشائعةفي الممارسة.

12.6.1.1. المتوسطات المتسربة Leaky Averages

شاهدنا القسم السابق نناقش الانحدار الاشتقاقي العشوائي المصغر minibatch SGD كوسيلة لتسريع الحساب. كان له أيضًا تأثير جانبي لطيف يتمثلفي أن متوسط التدرجات (الانحدارات) يقلل من مقدار التباين. يمكن حساب minibatch SGD من خلال:

$$\mathbf{g}_{t,t-1} = \partial_{\mathbf{w}} \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} f(\mathbf{x}_i, \mathbf{w}_{t-1}) = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \mathbf{h}_{i,t-1}.$$

للإبقاء على التدوين بسيطًا، استخدمنا هنا $\mathbf{h}_{i,t-1} = \partial_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}_{t-1})$ الانحدار الاشتقاقي العشوائي للعينة i باستخدام الأوزان المحدثةفي الوقت $t-1$. سيكون من الرائع لو تمكنا من الاستفادة من تأثير تقليل التباين حتى بعد متوسط التدرجات على minibatch. أحد الخيارات لإنجاز هذه المهمة هو استبدال حساب التدرج بـ "متوسط متسرب leaky average ":

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1}$$

بالنسبة لبعض $\beta \in (0,1)$. يؤدي هذا بشكل فعال إلى استبدال الانحدار الاشتقاقي بالانحدار الاشتقاقي الذي تم حساب متوسطه على عدة انحدارات سابقة. $\mathbf{v}$ يسمى السرعة velocity. إنها تتراكم الانحدارات السابقة بشكل مشابه لكيفية قيام كرة ثقيلة بالتدحرج لأسفل في مشهد الدالة الموضوعية وتتكامل مع القوى السابقة. لمعرفة ما يحدث بمزيد من التفصيل، دعنا نوسع $\mathbf{v}_t$ بشكل متكرر في

$$\mathbf{v}_t = \beta^2\mathbf{v}_{t-2} + \beta\mathbf{g}_{t-1,t-2} + \mathbf{g}_{t,t-1} = \cdots, = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}.$$

تمثل كميات β الكبيرة متوسطًا طويل المدى، بينما تمثل كميات β الصغيرة تصحيحًا طفيفًا فقط بالنسبة إلى طريقة الانحدار الاشتقاقي. لم يعد استبدال الانحدار الاشتقاقي الجديد يشير إلى اتجاه أكثر انحدارًا في حالة معينة بعد الآن، بل في اتجاه متوسط الوزن للانحدارات السابقة. هذا يسمح لنا بإدراك معظم فوائد حساب المتوسط على دفعة دون تكلفة حساب الانحدارات عليها فعليًا. سنعيد النظر في إجراء حساب المتوسط بمزيد من التفصيل لاحقًا.

شكل المنطق أعلاه الأساس لما يعرف الآن باسم طرق الانحدار المتسارع accelerated gradient، مثل الانحدار مع الزخم gradients with momentum. إنهم يتمتعون بفائدة إضافية تتمثل في كونهم أكثر فاعلية في الحالات التي تكون فيها مشكلة التحسين سيئة (على سبيل المثال، عندما تكون هناك بعض الاتجاهات حيث يكون التقدم أبطأ بكثير من الآخرين، يشبه الوادي الضيق). علاوة على ذلك، فهي تسمح لنا بالمتوسط على التدرجات اللاحقة للحصول على اتجاهات نزول أكثر ثباتًا. في الواقع، يعتبر جانب التسارع حتى بالنسبة للمشكلات المحددة الخالية من الضوضاء أحد الأسباب الرئيسية لعمل الزخم ولماذا يعمل بشكل جيد.

كما قد يتوقع المرء، نظرًا لفعاليته، يعد موضوعًا مدروسًا جيدًا في تحسين التعلم العميق وما بعده. انظر على سبيل المثال، المقال التعريفي الجميل لـ Goh (2017) للحصول على تحليل متعمق ورسوم متحركة تفاعلية. تم اقتراحه من قبل Polyak (1964). Nesterov (2018) لديه مناقشة نظرية مفصلة في سياق التحسين المحدب. من المعروف أن الزخم في التعلم العميق مفيد لفترة طويلة. انظر على سبيل المثال، مناقشة Sutskever et al. (2013) لمزيد من التفاصيل.

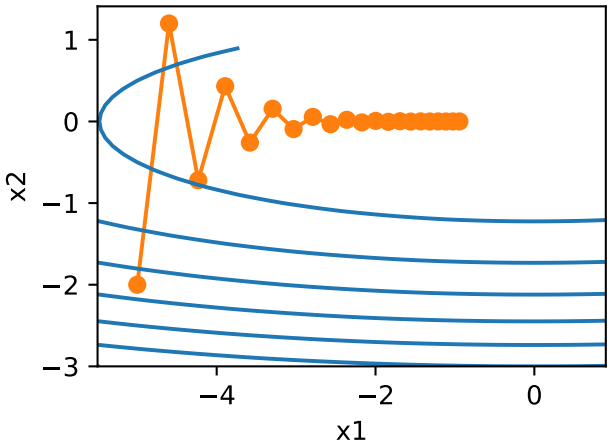
12.6.1.2. مشكلة غير مشروطة An Ill-conditioned Problem

للحصول على فهم أفضل للخصائص الهندسية لطريقة الزخم momentum، نعيد النظر في الانحدار الاشتقاقي، وإن كان ذلك بدالة هدف أقل إمتاعًا. تذكر أنه في القسم 12.3 استخدمنا $f(\mathbf{x}) = x_1^2 + 2x_2^2$، أي moderately distorted ellipsoid objective. نقوم بتشويه هذه الدالة بشكل أكبر عن طريق مدها في الاتجاه عبر

$$f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2.$$

كما كان f من قبل لديه الحد الأدنى في $(0,0)$. هذه الدالة مسطحة للغاية very flat في اتجاه x_1. دعنا نرى ما يحدث عندما نقوم بتنفيذ انحدار اشتقاقي كما كان من قبل في هذه الدالة الجديدة. نختار 0.4 كمعدل التعلم.

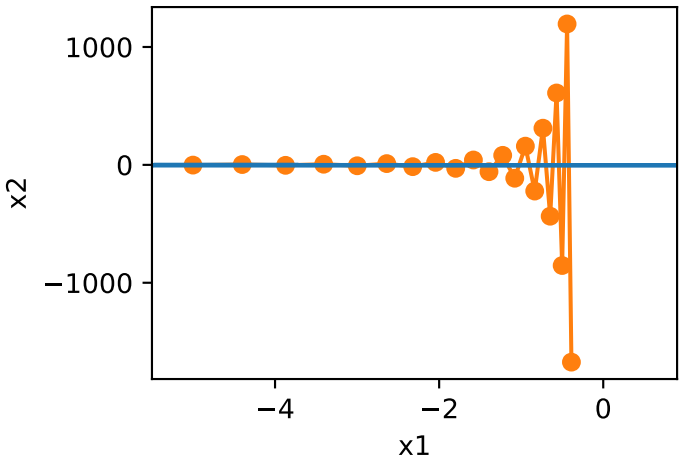
```
%matplotlib inline
import tensorflow as tf
from d2l import tensorflow as d2l

eta = 0.4
def f_2d(x1, x2):
    return 0.1 * x1 ** 2 + 2 * x2 ** 2
def gd_2d(x1, x2, s1, s2):
    return (x1 - eta * 0.2 * x1, x2 - eta * 4 * x2, 0,
0)

d2l.show_trace_2d(f_2d, d2l.train_2d(gd_2d))
epoch 20, x1: -0.943467, x2: -0.000073
```



من خلال البناء، يكون الانحدار في اتجاه x_2 أعلى بكثير ويتغير بسرعة أكبر بكثير من اتجاه x_1 الأفقي. وبالتالي نحن عالقون بين خيارين غير مرغوب فيهما: إذا اخترنا معدل تعلم صغيرًا، فإننا نضمن أن الحل لا يتباعد في الاتجاه ولكننا مثقلون بالتقارب البطيء في الاتجاه. على العكس من ذلك، مع معدل التعلم الكبير نتقدم بسرعة في اتجاه x_1 ولكننا نتباعد في x_2. يوضح المثال أدناه ما يحدث حتى بعد زيادة طفيفة في معدل التعلم من 0.4 إلى 0.6. يتحسن التقارب في الاتجاه ولكن جودة الحل بشكل عام أسوأ بكثير.

```
eta = 0.6
d2l.show_trace_2d(f_2d, d2l.train_2d(gd_2d))
```

```
epoch 20, x1: -0.387814, x2: -1673.365109
```



12.6.1.3. طريقة الزخم The Momentum Method

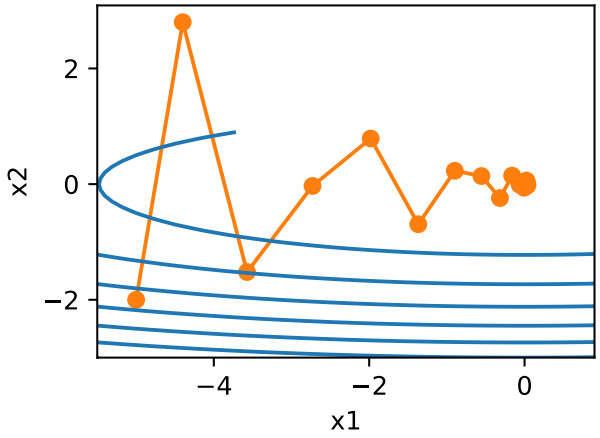
تسمح لنا طريقة الزخم بحل مشكلة الانحدار الاشتقاقي الموصوفة أعلاه. بالنظر إلى تتبع التحسين أعلاه، قد نستشعر أن متوسط الانحدارات على الماضي سيعمل بشكل جيد. بعد كل شيء،في اتجاه x_1 الذي سيجمع هذا الانحدارات المحاذاة جيداً، وبالتالي زيادة المسافة التي نغطيها مع كل خطوة. على العكس من ذلك،في اتجاه x_2 الذي تتأرجح فيه الانحدارات، سيقلل الانحدار الاشتقاقي الكلي من حجم الخطوة بسبب التذبذبات التي تلغي بعضها البعض. يؤدي استخدام $\mathbf{v}_t$ بدلا من الانحدار $\mathbf{g}_t$ إلى الحصول على معادلات التحديث التالية:

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1}, \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{v}_t. \end{aligned}$$

لاحظ ان $\beta = 0$ نستعيد الانحدار الاشتقاقي المنتظم. قبل الخوضفي الخصائص الرياضية، دعونا نلقي نظرة سريعة على كيفية تصرف الخوارزميةفي الممارسة العملية.

```
def momentum_2d(x1, x2, v1, v2):
    v1 = beta * v1 + 0.2 * x1
    v2 = beta * v2 + 4 * x2
    return x1 - eta * v1, x2 - eta * v2, v1, v2

eta, beta = 0.6, 0.5
d2l.show_trace_2d(f_2d, d2l.train_2d(momentum_2d))
```

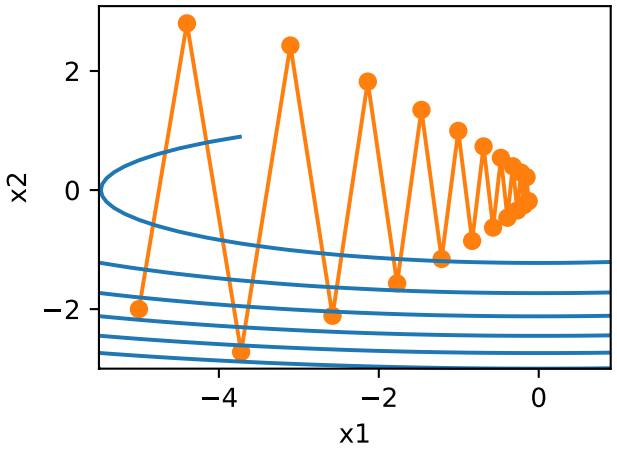
```
epoch 20, x1: 0.007188, x2: 0.002553
```



كما نرى، حتى مع نفس معدل التعلم الذي استخدمناه من قبل، لا يزال الزخم يتقارب جيداً. دعونا نرى ما يحدث عندما نخفض معامل الزخم. تقسيم $\beta = 0.25$ إلى النصف يؤدي إلى مسار يتقارب بالكاد على الإطلاق. ومع ذلك، فهو أفضل بكثير من عدم وجود زخم (عندما يتباعد الحل).

```
eta, beta = 0.6, 0.25
d2l.show_trace_2d(f_2d, d2l.train_2d(momentum_2d))
```

```
epoch 20, x1: -0.126340, x2: -0.186632
```

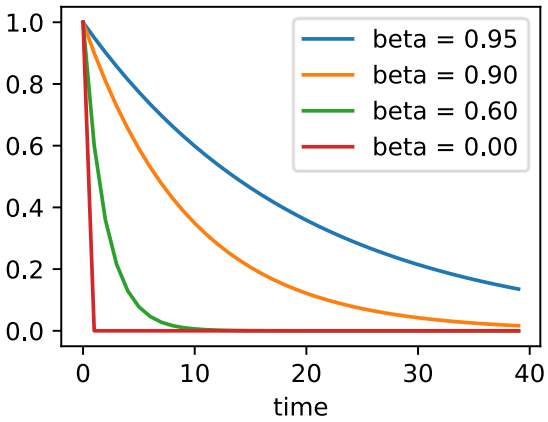


لاحظ أنه يمكننا دمج الزخم مع الانحدار الاشتقاقي العشوائي وعلى وجه الخصوص، الانحدار الاشتقاقي العشوائي المصغر. التغيير الوحيد هو أنهفي هذه الحالة نستبدل الانحدارات $\mathbf{g}_{t,t-1}$ بـ $\mathbf{g}_t$. أخيرًا، للسهولة، نقوم بتهيئة $\mathbf{v}_0 = 0$ في الوقت $t = 0$. دعونا نلقي نظرة على ما يفعله متوسط التسربفي الواقع للتحديثات.

12.6.1.4. وزن العينة الفعال Effective Sample Weight

تذكر ذلك $\mathbf{v}_t = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}$. في الحد تضاف المصطلحات إلى $\sum_{\tau=0}^{\infty} \beta^\tau = \frac{1}{1-\beta}$. بعبارة أخرى، بدلاً من اتخاذ خطوةفي الحجم ηفي الانحدار الاشتقاقي أو الانحدار الاشتقاقي العشوائي، فإننا نتخذ خطوةفي الحجم $\frac{\eta}{1-\beta}$ بينمافي نفس الوقت، التعامل مع اتجاه نزول من المحتمل أن يكون أفضل بكثير. هاتان فائدتانفي واحد. لتوضيح كيف يتصرف الترجيح weighting مع الخيارات المختلفة β ، ضعفي اعتبارك الرسم التخطيطي أدناه.

```
d2l.set_figsize()
betas = [0.95, 0.9, 0.6, 0]
for beta in betas:
    x = tf.range(40).numpy()
    d2l.plt.plot(x, beta ** x, label=f'beta =
{beta:.2f}')
d2l.plt.xlabel('time')
d2l.plt.legend();
```



12.6.2. تجارب عملية Practical Experiments

دعونا نرى كيف يعمل الزخم من الناحية العملية، أي عند استخدامهفي سياق مُحسِّن مناسب. لهذا نحن بحاجة إلى تنفيذ أكثر قابلية للتوسع إلى حد ما.

12.6.2.1. التنفيذ من البداية Implementation from Scratch

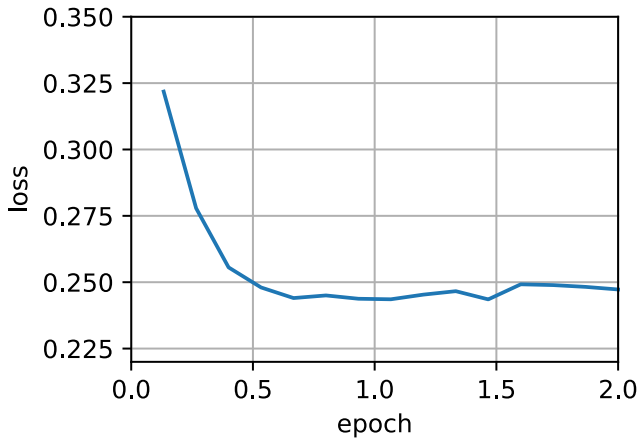
بالمقارنة مع الانحدار الاشتقاقي العشوائي (المصغر)، تحتاج طريقة الزخم إلى الحفاظ على مجموعة من المتغيرات المساعدة، أي السرعة velocity. لها نفس شكل الانحدارات (ومتغيرات مشكلة التحسين).في التنفيذ أدناه نسمي هذه المتغيرات `states`.

```
def init_momentum_states(features_dim):
```

```
    v_w = tf.Variable(tf.zeros((features_dim, 1)))
    v_b = tf.Variable(tf.zeros(1))
    return (v_w, v_b)

def sgd_momentum(params, grads, states, hyperparams):
    for p, v, g in zip(params, states, grads):
            v[:].assign(hyperparams['momentum'] * v + g)
            p[:].assign(p - hyperparams['lr'] * v)
```

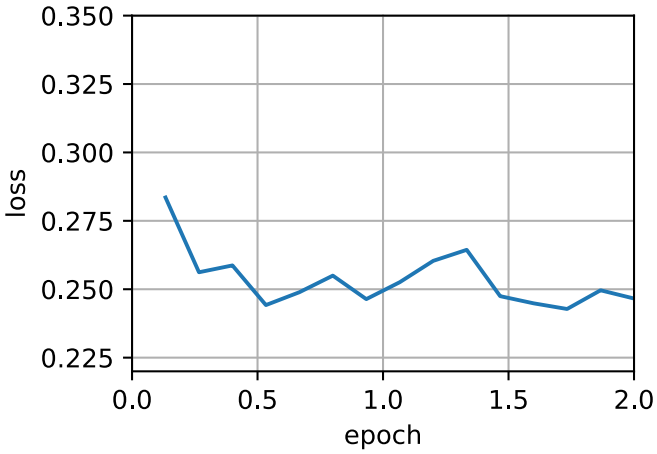
دعونا نرى كيف يعمل هذافي الممارسة.

```
def train_momentum(lr, momentum, num_epochs=2):
    d2l.train_ch11(sgd_momentum,
init_momentum_states(feature_dim),
                   {'lr': lr, 'momentum': momentum},
data_iter,
                   feature_dim, num_epochs)

data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
train_momentum(0.02, 0.5)
loss: 0.247, 0.101 sec/epoch
```



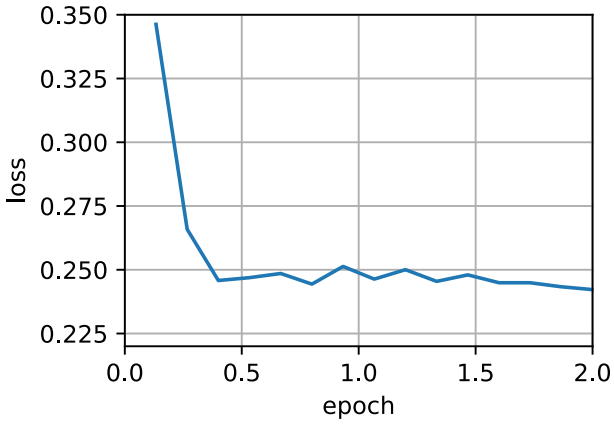
عندما نزيد زخم المعلمة الفائقة الزخم `momentum` إلى 0.9 ، فإنه يرقى إلى حجم عينة فعال أكبر بكثير من 10. نقوم بتقليل معدل التعلم قليلاً الى 0.01 لإبقاء الأمور تحت السيطرة.

```
train_momentum(0.01, 0.9)
loss: 0.247, 0.103 sec/epoch
```



يؤدي خفض معدل التعلم إلى معالجة أي مشكلة تتعلق بمشاكل التحسين غير السلس. ويؤدي تعيينه إلى 0.005 الى الحصول على خصائص تقارب جيدة.

```
train_momentum(0.005, 0.9)
loss: 0.242, 0.098 sec/epoch
```

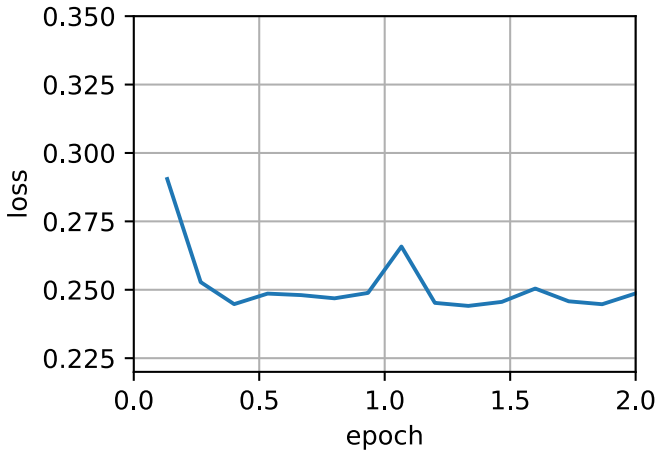


12.6.2.2. التنفيذ المختصر Concise Implementation

لا يوجد الكثير مما يجب فعلهفي Gluon نظرًا لأن محلل `sgd` القياسي يحتوي بالفعل على زخم مضمن. يؤدي تعيين معلمات المطابقة إلى الحصول على مسار مشابه جداً.

```
trainer = tf.keras.optimizers.SGD
d2l.train_concise_ch11(trainer, {'learning_rate': 0.005,
'momentum': 0.9},
                       data_iter)
```

```
loss: 0.249, 0.099 sec/epoch
```



12.6.3. التحليل النظري Theoretical Analysis

حتى الآن المثال ثنائي الأبعاد لـ $f(x) = 0.1x_1^2 + 2x_2^2$ بدا مفتعلًا إلى حد ما. سنرى الآن أن هذا في الواقع يمثل تمامًا أنواع المشكلات التي قد يواجهها المرء، على الأقل في حالة تقليل دوال الهدف التربيعية المحدبة convex quadratic objective functions.

12.6.3.1. دوال محدبة من الدرجة الثانية Quadratic Convex Functions

ضع في اعتبارك الدالة

$$h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{x}^\top \mathbf{c} + b.$$

هذه دالة تربيعية عامة. بالنسبة لمصفوفات التعريف الموجبة $\mathbf{Q} \succ 0$، أي بالنسبة للمصفوفات ذات القيم الذاتية الإيجابية، فإن هذا يحتوي على مقلل $\mathbf{x}^* = -\mathbf{Q}^{-1}\mathbf{c}$ مع اقل قيمة $b - \frac{1}{2}\mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c}$. ومن ثم يمكننا إعادة كتابة h كـ

$$h(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{Q}^{-1}\mathbf{c})^\top \mathbf{Q}(\mathbf{x} - \mathbf{Q}^{-1}\mathbf{c}) + b - \frac{1}{2}\mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c}.$$

يتم إعطاء الانحدار بواسطة $\partial_{\mathbf{x}} h(\mathbf{x}) = \mathbf{Q}(\mathbf{x} - \mathbf{Q}^{-1}\mathbf{c})$. أي أنها تُعطى بالمسافة بين $\mathbf{x}$ والمصغر مضروبة في $\mathbf{Q}$. وبالتالي، فإن السرعة أيضًا عبارة عن مجموعة خطية من المصطلحات $\mathbf{Q}(\mathbf{x}_t - \mathbf{Q}^{-1}\mathbf{c})$.

نظرًا لأن $\mathbf{Q}$ محدد إيجابي، يمكن أن يتحلل إلى نظام الخاص eigensystem به عبر $\mathbf{Q} = \mathbf{O}^\top\mathbf{\Lambda}\mathbf{O}$ لمصفوفة متعامدة (دوران) $\mathbf{O}$ ومصفوفة قطرية $\mathbf{\Lambda}$ لقيم ذاتية موجبة. يتيح لنا ذلك إجراء تغييرفي المتغيرات من $\mathbf{x}$ إلى $\mathbf{z} := \mathbf{O}(\mathbf{x} - \mathbf{Q}^{-1}\mathbf{c})$ للحصول على تعبير مبسط للغاية:

$$h(\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top\mathbf{\Lambda}\mathbf{z} + b'.$$

هنا $b' = b - \frac{1}{2}\mathbf{c}^\top\mathbf{Q}^{-1}\mathbf{c}$. نظرًا لأن $\mathbf{O}$ المصفوفة المتعامدة فقط، فإن هذا لا يزعج الانحدارات بطريقة ذات مغزى. معبرا عنها من حيث $\mathbf{z}$ الانحدار الاشتقاقي يصبح

$$\mathbf{z}_t = \mathbf{z}_{t-1} - \mathbf{\Lambda}\mathbf{z}_{t-1} = (\mathbf{I} - \mathbf{\Lambda})\mathbf{z}_{t-1}.$$

الحقيقة المهمةفي هذا التعبير هي أن الانحدار الاشتقاقي لا يختلط mix بين الفضاءات الذاتية eigenspaces المختلفة. أي، عندما يتم التعبير عنها من حيث النظام الذاتي الخاص بـ $\mathbf{Q}$ تستمر مشكلة التحسين بطريقة منسقة. هذا ينطبق أيضا على

$$\begin{aligned}\mathbf{v}_t &= \beta\mathbf{v}_{t-1} + \mathbf{\Lambda}\mathbf{z}_{t-1}\\ \mathbf{z}_t &= \mathbf{z}_{t-1} - \eta(\beta\mathbf{v}_{t-1} + \mathbf{\Lambda}\mathbf{z}_{t-1})\\ &= (\mathbf{I} - \eta\mathbf{\Lambda})\mathbf{z}_{t-1} - \eta\beta\mathbf{v}_{t-1}.\end{aligned}$$

عند القيام بذلك، أثبتنا النظرية التالية: يتحلل الانحدار الاشتقاقي مع أو بدون زخم لدالة تربيعية محدبة إلى تحسين تنسيقي في اتجاه المتجهات الذاتية eigenvectors للمصفوفة التربيعية.

12.6.3.2. دوال القيم القياسية Scalar Functions

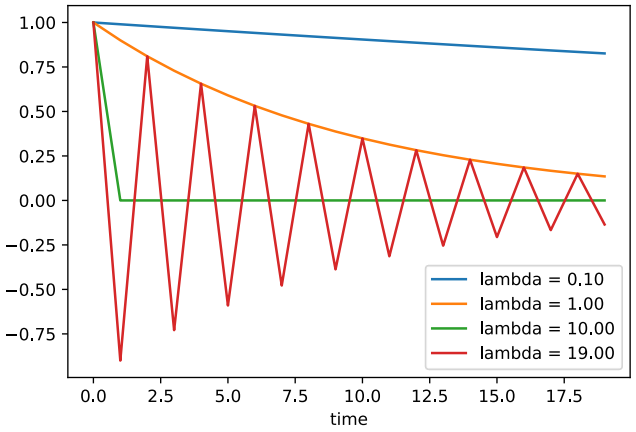
بالنظر إلى النتيجة أعلاه، دعونا نرى ما يحدث عندما نقوم بتقليل الدالة $f(x) = \frac{\lambda}{2}x^2$. بالنسبة للانحدار الاشتقاقي لدينا

$$x_{t+1} = x_t - \eta\lambda x_t = (1 - \eta\lambda)x_t.$$

عندما $|1 - \eta\lambda| < 1$ يتقارب هذا التحسين بمعدل أسي بعد t خطوات لدينا $x_t = (1 - \eta\lambda)^t x_0$. يوضح هذاكيف يتحسن معدل التقاربفي البداية مع زيادة معدل التعلم η حتى $\eta\lambda = 1$. أبعد من ذلك تتباعد الأشياء ولـ $\eta\lambda > 2$ تتباعد مشكلة التحسين.

```
lambdas = [0.1, 1, 10, 19]
eta = 0.1
d2l.set_figsize((6, 4))
for lam in lambdas:
    t = tf.range(20).numpy()
    d2l.plt.plot(t, (1 - eta * lam) ** t, label=f'lambda
= {lam:.2f}')
d2l.plt.xlabel('time')
```

```
d2l.plt.legend();
```



لتحليل التقارب في حالة الزخم، نبدأ بإعادة كتابة معادلات التحديث من حيث عددين: واحد من أجل x والآخر للسرعة v. هذا ينتج:

$$\begin{bmatrix} v_{t+1} \\ x_{t+1} \end{bmatrix} = \begin{bmatrix} \beta & \lambda \\ -\eta\beta & (1-\eta\lambda) \end{bmatrix} \begin{bmatrix} v_t \\ x_t \end{bmatrix} = \mathbf{R}(\beta, \eta, \lambda) \begin{bmatrix} v_t \\ x_t \end{bmatrix}.$$

استخدمنا $\mathbf{R}$ على الإشارة إلى 2×2 سلوك التقارب الحاكم governing convergence behavior. بعد t خطوات الاختيار الأولي $[v_0, x_0]$ يصبح $\mathbf{R}(\beta, \eta, \lambda)^t [v_0, x_0]$. ومن ثم، فإن الأمر متروك للقيم الذاتية لـ $\mathbf{R}$ لتحديد سرعة التقارب. راجع منشور Distill الخاص بـ Goh (2017) للحصول على رسوم متحركة رائعة وFlammarion and Bach (2015) للحصول على تحليل مفصل. يمكن يظهر أن $0 < \eta\lambda < 2 + 2\beta$ السرعة تتقارب. هذا هو نطاق أكبر من المعلمات الممكنة بالمقارنة مع $0 < \eta\lambda < 2$ للانحدار الاشتقاقي. كما يشير أيضًا إلى أن القيم الكبيرة بشكل عام مرغوبة. تتطلب المزيد من التفاصيل قدرًا لا بأس به من التفاصيل الفنية ونقترح أن يراجع القارئ المهتم المنشورات الأصلية.

12.6.4. الملخص

- يستبدل الزخم الانحدارات بمتوسط متسرب على الانحدارات السابقة. هذا يسرع التقارب بشكل كبير.
- من المستحسن لكل من الانحدار الاشتقاقي الخالي من الضوضاء والانحدار الاشتقاقي العشوائي (الصاخب).
- يمنع الزخم توقف عملية التحسين التي من المرجح أن تحدث للانحدار الاشتقاقي العشوائي.

- يتم إعطاء العدد الفعال للانحدارات التي تعطى بواسطة $\frac{1}{1-\beta}$ بسبب تناقص الاسي الأسي exponentiated downweighting للبيانات السابقة.
- في حالة المشاكل التربيعية المحدبة يمكن تحليل ذلك صراحة بالتفصيل.
- التنفيذ بسيط للغاية ولكنه يتطلب منا تخزين متجه حالة إضافي additional state vector (السرعة $\mathbf{v}$).

12.6.5. التمارين

1. استخدم مجموعات أخرى من المعلمات الفائقة للزخم ومعدلات التعلم وراقب وتحليل النتائج التجريبية المختلفة.
2. جرب الانحدار الاشتقاقي والزخم لمشكلة تربيعية حيث يكون لديك قيم ذاتية متعددة، أي، $f(x) = \frac{1}{2}\sum_i \lambda_i x_i^2$، على سبيل المثال، $\lambda_i = 2^{-i}$. ارسم كيف قيم x تتناقص لتهيئة $x_i = 1$.
3. اشتقق الحد الأدنى من القيمة والمقلل لـ $h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} + \mathbf{x}^\top\mathbf{c} + b$.
4. ما الذي يتغير عندما نؤدي انحدارا اشتقاقيا عشوائيًا مع الزخم؟ ماذا يحدث عندما نستخدم الانحدار الاشتقاقي العشوائي المصغر مع الزخم؟ التجربة مع المعلمات؟

12.7. Adagrad

لنبدأ بالتفكيرفي مشكلات التعلم مع الميزات التي تحدث بشكل غير متكرر.

12.7.1. ميزات متفرقة ومعدلات تعلم Sparse Features and Learning Rates

تخيل أننا نقوم بتدريب نموذج لغوي. للحصول على دقة جيدة، نريد عادةً تقليل معدل التعلم بينما نستمرفي التدريب، عادةً بمعدل $O(t^{-\frac{1}{2}})$ أو أبطأ. الآن ضعفي اعتبارك تدريبًا نموذجيًا على ميزات متفرقة، أي الميزات التي تحدث بشكل غير متكرر. هذا أمر شائع بالنسبة للغة الطبيعية، على سبيل المثال، من غير المرجح أن نرى كلمة تكيف مسبق preconditioning أكثر من التعلم learning. ومع ذلك، فهو شائع أيضًافي مجالات أخرى مثل الإعلانات الحسابية computational advertising والتصفية التعاونية المخصصة personalized collaborative filtering. بعد كل شيء، هناك العديد من الأشياء التي تهم فقط عدد قليل من الناس.

لا تتلقى المعلمات المرتبطة بالميزات غير المتكررة infrequent features تحديثات ذات مغزى إلا عند حدوث هذه الميزات. بالنظر إلى معدل التعلم المتناقص، قد ينتهي بنا الأمرفي موقف تتلاقى فيه معلمات السمات المشتركة بسرعة إلى حد ما مع قيمها المثلى، بينما بالنسبة

إلى الميزات غير المتكررة، ما زلنا نفتقر إلى مراقبتها بشكل متكرر قبل تحديد قيمها المثلى. وبعبارة أخرى، فإن معدل التعلم إما ينخفض ببطء شديد بالنسبة للميزات المتكررة أو بسرعة كبيرة للغاية بالنسبة للميزات غير المتكررة.

قد يكون الاختراق المحتمل لتصحيح هذه المشكلة هو حساب عدد المرات التي نرى فيها ميزة معينة واستخدامها كساعة لضبط معدلات التعلم. هذا هو، بدلاً من اختيار معدل التعلم للشكل $\eta = \frac{\eta_0}{\sqrt{t+c}}$ يمكننا استخدام $\eta_i = \frac{\eta_0}{\sqrt{s(i,t)+c}}$. تحسب هنا عدد العناصر التي ليست صفرية للميزة التي لاحظناها حتى الوقت. هذافي الواقع سهل التنفيذ دون أي تكاليف إضافية ذات مغزى. ومع ذلك، فإنه يفشل عندما لا يكون لدينا تباين تام، بل مجرد بيانات حيث تكون الانحدارات غالبًا صغيرة جدًا ونادرًا ما تكون كبيرة. بعد كل شيء، ليس من الواضح أين يمكن للمرء أن يرسم الخط الفاصل بين شيء مؤهل كميزة ملحوظة أم لا.

Adagrad بواسطة Duchi et al. (2011) يعالج هذا عن طريق استبدال العداد الخام $s(i,t)$ بمجموع مربعات الانحدارات الملحوظة سابقًا. على وجه الخصوص، يستخدم $s(i,t+1) = s(i,t) + (\partial_i f(\mathbf{x}))^2$ كوسيلة لضبط معدل التعلم. هذا له فائدتان: أولاً، لم نعد بحاجة إلى تحديد متى يكون الانحدار الاشتقاقي كبيرًا بدرجة كافية. ثانيًا، يتم قياسه تلقائيًا مع حجم الانحدارات. يتم تقليص الإحداثيات التي تتوافق بشكل روتيني مع الانحدارات الكبيرة بشكل كبير، بينما يتلقى الآخرون ذوو الانحدارات الصغيرة علاجًا أكثر لطفًا.في الممارسة العملية، يؤدي هذا إلى إجراء تحسين فعال للغاية للإعلان الحسابي والمشاكل ذات الصلة. لكن هذا يخفي بعض الفوائد الإضافية الكامنةفي Adagrad والتي يمكن فهمها بشكل أفضلفي سياق التكيف المسبق preconditioning.

12.7.2. التكيف المسبق Preconditioning

تعتبر مشاكل التحسين المحدب جيدة لتحليل خصائص الخوارزميات. بعد كل شيء، من الصعب الحصول على ضمانات نظرية ذات مغزى بالنسبة لمعظم المشكلات غير المحدبة، ولكن غالبًا ما يستخدم الحدس والبصيرة. دعونا نلقي نظرة على مشكلة تقليل $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + b$

كما رأينافي القسم 12.6، من الممكن إعادة كتابة هذه المشكلة من حيث التكوين الذاتي $\mathbf{Q} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U}$(eigendecomposition) للوصول إلى مشكلة مبسطة كثيرًا حيث يمكن حل كل إحداثي على حدة:

$$f(\mathbf{x}) = \overline{f}(\overline{\mathbf{x}}) = \frac{1}{2}\overline{\mathbf{x}}^\top \mathbf{\Lambda}\overline{\mathbf{x}} + \overline{\mathbf{c}}^\top \overline{\mathbf{x}} + b.$$

استخدمنا هنا $\overline{\mathbf{x}} = \mathbf{U}\mathbf{x}$ وبالتالي $\overline{\mathbf{c}} = \mathbf{U}\mathbf{c}$. المشكلة المعدلة لها المصغر $\overline{\mathbf{x}} = -\boldsymbol{\Lambda}^{-1}\overline{\mathbf{c}}$ والحد الأدنى $b - \frac{1}{2}\overline{\mathbf{c}}^{\mathsf{T}}\boldsymbol{\Lambda}^{-1}\overline{\mathbf{c}}$. هذا أسهل في الحساب لأنه $\boldsymbol{\Lambda}$ مصفوفة قطرية تحتوي على القيم الذاتية لـ $\mathbf{Q}$.

إذا أزعجنا $\mathbf{c}$ قليلاً، فإننا نأمل في العثور على تغييرات طفيفة فقط في المصغر f. للأسف، ليست هذه هي القضية. بينما تؤدي التغييرات الطفيفة في $\mathbf{c}$ الى حدوث تغييرات طفيفة بنفس القدر في $\overline{\mathbf{c}}$، ليس هذا هو الحال بالنسبة لمصغر f (و$\overline{f}$ على التوالي). عندما تكون القيم الذاتية $\boldsymbol{\Lambda}_i$ كبيرة، سنرى فقط تغييرات صغيرة في $\overline{x}_i$ وفي الحد الأدنى لـ $\overline{f}$. على العكس من ذلك، لإجراء تغييرات صغيرة في $\boldsymbol{\Lambda}_i$ يمكن أن تكون دراماتيكية. تسمى النسبة بين أكبر وأصغر قيمة ذاتية رقم الشرط condition number لمشكلة التحسين.

$$\kappa = \frac{\boldsymbol{\Lambda}_1}{\boldsymbol{\Lambda}_d}.$$

إذا كان رقم الشرط κ كبيرًا، فمن الصعب حل مشكلة التحسين بدقة. نحن بحاجة إلى التأكد من أننا حريصون على الحصول على نطاق ديناميكي كبير من القيم بشكل صحيح. يؤدي تحليلنا إلى سؤال واضح، وإن كان ساذجًا إلى حد ما: ألا يمكننا ببساطة "حل" المشكلة عن طريق تشويه المساحة بحيث تكون جميع القيم الذاتية 1. من الناحية النظرية، هذا سهل للغاية: نحتاج فقط إلى القيم الذاتية والمتجهات الذاتية لـ $\mathbf{Q}$ لإعادة قياس المشكلة من $\mathbf{x}$ واحدة في $\mathbf{z} := \boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}\mathbf{x}$. في نظام الإحداثيات الجديد $\mathbf{x}^{\mathsf{T}}\mathbf{Q}\mathbf{x}$ يمكن تبسيطه إلى $\|\mathbf{z}\|^2$. للأسف، هذا اقتراح غير عملي إلى حد ما. تعد حساب القيم الذاتية والمتجهات الذاتية بشكل عام أكثر تكلفة من حل المشكلة الفعلية.

في حين أن حساب القيم الذاتية قد يكون مكلفًا تمامًا، إلا أن تخمينها وحسابها حتى إلى حد ما تقريبًا قد يكون بالفعل أفضل بكثير من عدم القيام بأي شيء على الإطلاق. على وجه الخصوص، يمكننا استخدام الإدخالات القطرية لـ $\mathbf{Q}$ وإعادة قياسها وفقًا لذلك. هذا أرخص بكثير من حساب القيم الذاتية.

$$\tilde{\mathbf{Q}} = \operatorname{diag}^{-\frac{1}{2}}(\mathbf{Q})\mathbf{Q}\operatorname{diag}^{-\frac{1}{2}}(\mathbf{Q}).$$

في هذه الحالة لدينا $\tilde{\mathbf{Q}}_{ij} = \mathbf{Q}_{ij}/\sqrt{\mathbf{Q}_{ii}\mathbf{Q}_{jj}}$ وعلى وجه التحديد $\tilde{\mathbf{Q}}_{ii} = 1$ لكل i.في معظم الحالات، هذا يبسط رقم الشرط إلى حد كبير. على سبيل المثال، الحالات التي ناقشناها سابقًا، سيؤدي ذلك إلى القضاء تمامًا على المشكلة المطروحة نظرًا لأن المشكلة محاذية للمحور axis aligned.

لسوء الحظ، نواجه مشكلة أخرى:في التعلم العميق، لا نتمكن عادةً حتى من الوصول إلى المشتق الثاني من دالة الهدف: بالنسبة $\mathbf{x} \in \mathbb{R}^d$ المشتق الثاني حتى على الدفعات الصغيرة، قد يتطلب $\mathcal{O}(d^2)$ مساحة والعمل للحساب، مما يجعله عمليًا غير عملي. تتمثل فكرة Adagrad البارعة في استخدام وكيل proxy لذلك القطر المراوغ elusive diagonal من Hessian الذي يعتبر رخيصًا نسبيًا للحساب وفعال ـ حجم الانحدار نفسه.

من أجل معرفة سبب نجاح ذلك، دعنا نلقي نظرة على $\overline{f}(\overline{\mathbf{x}})$. لدينا هذا

$$\partial_{\overline{\mathbf{x}}}\overline{f}(\overline{\mathbf{x}}) = \boldsymbol{\Lambda}\overline{\mathbf{x}} + \overline{\mathbf{c}} = \boldsymbol{\Lambda}(\overline{\mathbf{x}} - \overline{\mathbf{x}}_0),$$

حيث $\overline{\mathbf{x}}_0$ هو المصغر لـ $\overline{f}$. ومن ثم فإن حجم الانحدار يعتمد على كل من $\boldsymbol{\Lambda}$ والمسافة من الأمثلية. إذا $\overline{\mathbf{x}} - \overline{\mathbf{x}}_0$ لم يتغير، سيكون هذا كل ما هو مطلوب. بعد كل شيء،في هذه الحالة حجم الانحدار $\partial_{\overline{\mathbf{x}}}\overline{f}(\overline{\mathbf{x}})$ يكفي. نظرًا لأن AdaGrad عبارة عن خوارزمية انحدار اشتقاقي عشوائي، فسنرى انحدارات ذات تباين غير صفري حتىفي الوضع الأمثل. نتيجة لذلك، يمكننا استخدام تباين الانحدارات بأمان كبديل رخيص لمقياس Hessian. التحليل الشامل خارج نطاق هذا القسم (سيكون من عدة صفحات). نحيل القارئ إلى (Duchi et al.، 2011) للحصول على التفاصيل.

12.7.3. الخوارزمية The Algorithm

دعونا نضفي الطابع الرسمي على المناقشة من أعلى. نستخدم المتغير $\mathbf{s}_t$ لتجميع تباين الانحدار السابق على النحو التالي.

$$\begin{aligned}
\mathbf{g}_t &= \partial_{\mathbf{w}} l(y_t, f(\mathbf{x}_t, \mathbf{w})),\\
\mathbf{s}_t &= \mathbf{s}_{t-1} + \mathbf{g}_t^2,\\
\mathbf{w}_t &= \mathbf{w}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t + \epsilon}} \cdot \mathbf{g}_t.
\end{aligned}$$

هنا يتم تطبيق عملية coordinate wise. هذا هو $\mathbf{v}^2$ ، لديه مدخلات v_i^2. على نفس المنوال، $\frac{1}{\sqrt{v}}$ لديه مدخلات $\frac{1}{\sqrt{v_i}}$ و $\mathbf{u} \cdot \mathbf{v}$ له مدخلات $u_i v_i$. كما كان من قبل هو معدل التعلم وϵ هو ثابت مضاف يضمن أننا لا نقسم على 0. أخيرًا، نقوم بتهيئة $\mathbf{s}_0 = \mathbf{0}$.

تمامًا كمافي حالة الزخم momentum، نحتاج إلى تتبع متغير إضافي،في هذه الحالة للسماح بمعدل تعلم فردي لكل إحداثي. لا يؤدي هذا إلى زيادة تكلفة Adagrad بشكل كبير بالنسبة إلى SGD، وذلك ببساطة لأن التكلفة الرئيسية هي عادةً حساب $l(y_t, f(\mathbf{x}_t, \mathbf{w}))$ ومشتقاته.

لاحظ أن تراكم الانحدارات التربيعيةفي $\mathbf{s}_t$ يعني أن $\mathbf{s}_t$ ينمو أساسًا بمعدل خطي (أبطأ إلى حد ما من الخطيةفي الممارسة العملية، لأن الانحدارات تتضاءلفي البداية). هذا يؤدي إلى $\mathcal{O}(t^{-\frac{1}{2}})$ معدل التعلم، وإن تم تعديله على أساس كل تنسيق. بالنسبة للمشاكل المحدبة، هذا مناسب تمامًا. في التعلم العميق، على الرغم من ذلك، قد نرغبفي تقليل معدل التعلم بشكل أبطأ. أدى هذا إلى عدد من متغيرات Adagrad التي سنناقشهافي الفصول اللاحقة. دعونا الآن نرى كيف يتصرف في مشكلة تربيعية محدبة. نستخدم نفس المشكلة كمافي السابق:

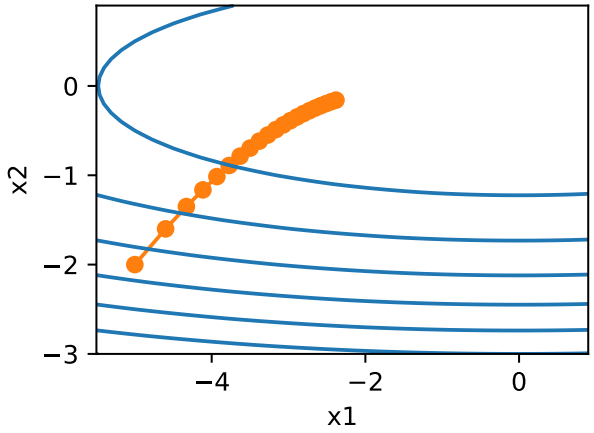
$$f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2.$$

سنقوم بتنفيذ Adagrad باستخدام نفس معدل التعلم سابقًا، أي $\eta = 0.4$. كما نرى، يكون المسار التكراري للمتغير المستقل أكثر سلاسة. ومع ذلك، نظرًا للتأثير التراكمي لـ $\boldsymbol{s}_t$، فإن معدل التعلم يتحلل باستمرار، لذلك لا يتحرك المتغير المستقل كثيرًا خلال مراحل لاحقة من التكرار.

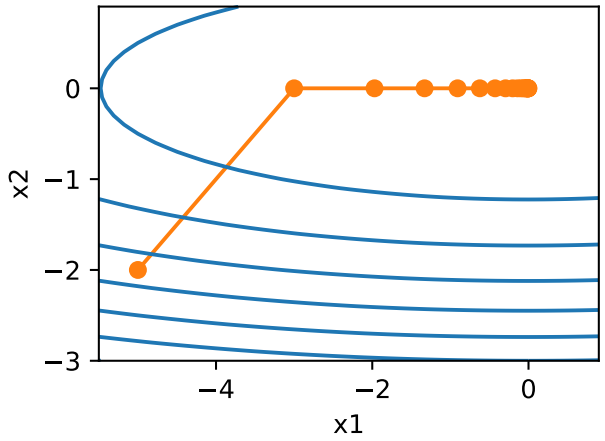
```
%matplotlib inline
import math
import tensorflow as tf
from d2l import tensorflow as d2l

def adagrad_2d(x1, x2, s1, s2):
    eps = 1e-6
    g1, g2 = 0.2 * x1, 4 * x2
    s1 += g1 ** 2
    s2 += g2 ** 2
    x1 -= eta / math.sqrt(s1 + eps) * g1
    x2 -= eta / math.sqrt(s2 + eps) * g2
    return x1, x2, s1, s2

def f_2d(x1, x2):
    return 0.1 * x1 ** 2 + 2 * x2 ** 2

eta = 0.4
d2l.show_trace_2d(f_2d, d2l.train_2d(adagrad_2d))
```

```
epoch 20, x1: -2.382563, x2: -0.158591
```

مع زيادة معدل التعلم الى 2 نرى سلوكًا أفضل بكثير. يشير هذا بالفعل إلى أن الانخفاضفي معدل التعلم قد يكون عدوانيًا إلى حد ما، حتىفي حالة عدم وجود ضوضاء ونحتاج إلى التأكد من أن المعلمات تتقارب بشكل مناسب.




```
eta = 2
d2l.show_trace_2d(f_2d, d2l.train_2d(adagrad_2d))
epoch 20, x1: -0.002295, x2: -0.000000
```



12.7.4. التنفيذ من البداية Implementation from Scratch

تمامًا مثل طريقة الزخم، يحتاج Adagrad إلى الحفاظ على متغير حالة من نفس شكل المعلمات.

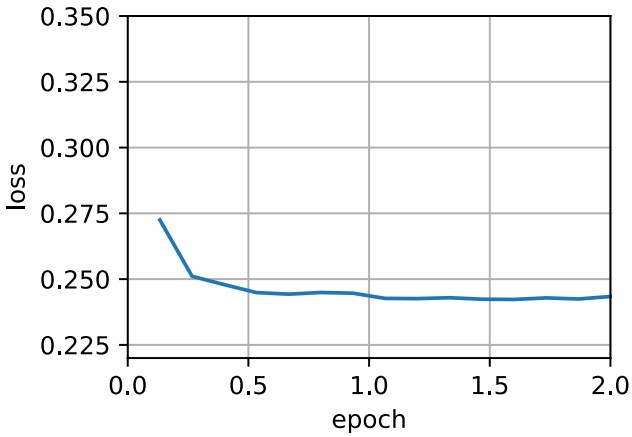
```
def init_adagrad_states(feature_dim):
    s_w = tf.Variable(tf.zeros((feature_dim, 1)))
    s_b = tf.Variable(tf.zeros(1))
    return (s_w, s_b)

def adagrad(params, grads, states, hyperparams):
    eps = 1e-6
```

```
    for p, s, g in zip(params, states, grads):
        s[:].assign(s + tf.math.square(g))
        p[:].assign(p - hyperparams['lr'] * g /
tf.math.sqrt(s + eps))
```

مقارنة بالتجربةفي القسم 12.5، نستخدم معدل تعلم أكبر لتدريب النموذج.

```
data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
d2l.train_ch11(adagrad,
init_adagrad_states(feature_dim),
               {'lr': 0.1}, data_iter, feature_dim);
```

```
loss: 0.243, 0.105 sec/epoch
```

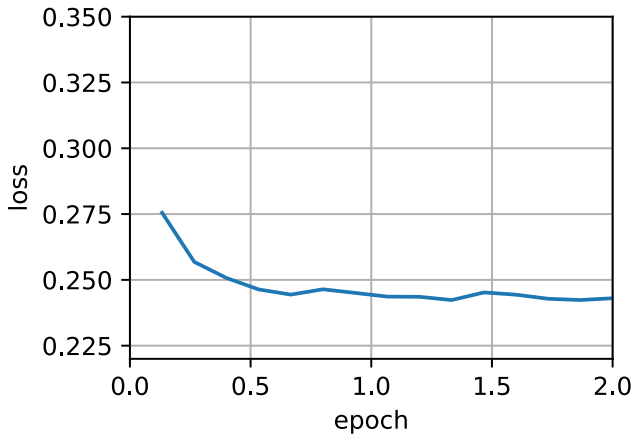


12.7.5. التنفيذ المختصر Concise Implementation

باستخدام مثيل المدرب لخوارزمية adagrad، يمكننا استدعاء خوارزمية Adagradفي Gluon.

```
trainer = tf.keras.optimizers.Adagrad
d2l.train_concise_ch11(trainer, {'learning_rate' : 0.1},
data_iter)
```

```
loss: 0.243, 0.102 sec/epoch
```



12.7.6. الملخص

- يقلل Adagrad من معدل التعلم ديناميكيًا على أساس كل تنسيق.
- يستخدم حجم الانحدار magnitude of the gradient كوسيلة لضبط مدى سرعة تحقيق التقدم ـ يتم تعويض الإحداثيات مع الانحدارات الكبيرة بمعدل تعلم أصغر.
- عادةً ما يكون حساب المشتق الثاني غير ممكنفي مشاكل التعلم العميقة بسبب قيود الذاكرة والحساب. يمكن أن يكون الانحدار وكيلًا proxy مفيدًا.
- إذا كانت مشكلة التحسين ذات بُنية غير متساوية إلى حد ما، يمكن أن يساعد Adagradفي تخفيف التشوه distortion.
- Adagrad فعال بشكل خاص للميزات المتفرقة sparse features حيث يحتاج معدل التعلم إلى الانخفاض بشكل أبطأ للمصطلحات التي لا تحدث بشكل متكرر.
- فيما يتعلق بمشاكل التعلم العميقة، قد يكون Adagrad أحيانًا عدوانيًا aggressive جدًافي تقليل معدلات التعلم. سنناقش استراتيجيات التخفيف mitigating من ذلك في سياق القسم 12.10.

12.7.7. التمارين

1. إثبت ما يلي بالنسبة للمصفوفة المتعامدة $\mathbf{U}$ والمتجه $\mathbf{c}$: $\|\mathbf{c} - \delta\|_2 = \|\mathbf{U}\mathbf{c} - \mathbf{U}\delta\|_2$. لماذا يعني هذا أن حجم الاضطرابات perturbations لا يتغير بعد تغيير متعامد للمتغيرات؟
2. جرب Adagrad من أجل $f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2$ وأيضًا بالنسبة لدالة الهدف تم تدويرها بمقدار 45 درجة، أي $f(\mathbf{x}) = 0.1(x_1 + x_2)^2 + 2(x_1 - x_2)^2$. هل تتصرف بشكل مختلف؟

3. إثبت نظرية الدائرة غيرشغورين Gerschgorin's circle theorem التي تنص على أن القيم الذاتية λ_i للمصفوفة $\mathbf{M}$ تحقق $|\lambda_i - \mathbf{M}_{jj}| \leq \sum_{k \neq j} |\mathbf{M}_{jk}|$ خيارًا واحدًا على الأقل من j.
4. ماذا تخبرنا نظرية غيرشغورين عن القيم الذاتية للمصفوفة المكيفة مسبقًا $\text{diag}^{-\frac{1}{2}}(\mathbf{M})\mathbf{M}\text{diag}^{-\frac{1}{2}}(\mathbf{M})$؟
5. جرب Adagrad للحصول على شبكة عميقة مناسبة، مثل القسم 7.6 عند تطبيقه على Fashion-MNIST.
6. كيف ستحتاج إلى تعديل Adagrad لتحقيق تدهور أقل حدة less aggressive decay في معدل التعلم؟

12.8 RMSProp

إحدى القضايا الرئيسيةفي القسم 12.7 هي أن معدل التعلم ينخفض وفقًا لجدول زمني محدد مسبقًا بشكل فعال $\mathcal{O}(t^{-\frac{1}{2}})$. في حين أن هذا مناسب بشكل عام للمشكلات المحدبة، فقد لا يكون مثاليًا للمشكلات غير المحدبة، مثل تلك التي نواجههافي التعلم العميق. ومع ذلك، فإن coordinate-wiseفي Adagrad مرغوب فيه للغاية كمكيف مسبق preconditioner.

اقترح خوارزمية RMSProp (Tieleman and Hinton، 2012) كإصلاح بسيط لفصل جدولة المعدل عن معدلات التعلم المنسق والتكيف coordinate-adaptive. تكمن المشكلة في أن Adagrad يقوم بتجميع مربعات الانحدارات في متجه الحالة $\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{g}_t^2$. نتيجة لذلك، $\mathbf{s}_t$ تستمر في النمو دون قيود بسبب الافتقار إلى التسوية، بشكل خطي وأساسي مع تقارب الخوارزمية.

طريقة واحدة لإصلاح هذه المشكلة ستكون باستخدام $\mathbf{s}_t/t$. لتوزيعات معقولة من $\mathbf{g}_t$ هذا سوف تتلاقى. لسوء الحظ، قد يستغرق الأمر وقتًا طويلاً جدًا حتى يبدأ السلوك المحدودفي الأهمية لأن الإجراء يتذكر المسار الكامل للقيم. البديل هو استخدام المتوسط المتسرب leaky average بنفس الطريقة التي استخدمناهافي طريقة الزخم، أي $\mathbf{s}_t \leftarrow \gamma \mathbf{s}_{t-1} + (1-\gamma)\mathbf{g}_t^2$ ، لبعض المعلمات $\gamma > 0$. يؤدي الحفاظ على جميع الأجزاء الأخرى دون تغيير إلى إنتاج RMSProp.

12.8.1 الخوارزمية The Algorithm

دعونا نكتب المعادلات بالتفصيل.

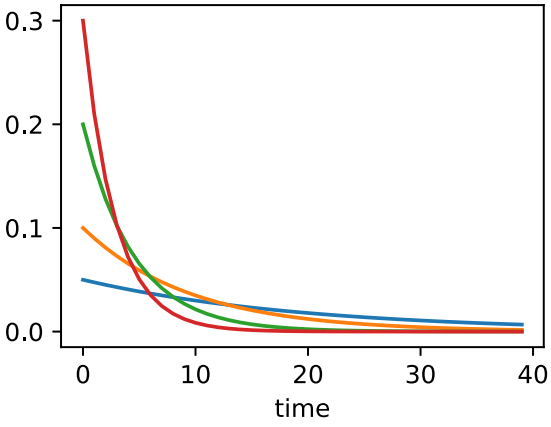
$$\begin{aligned} \mathbf{s}_t &\leftarrow \gamma \mathbf{s}_{t-1} + (1-\gamma)\mathbf{g}_t^2, \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t. \end{aligned}$$

يتم تعيين الثابت $\epsilon > 0$ الى 10^{-6} عادةً لضمان عدم تعرضنا للقسمة على صفر أو أحجام الخطوات الكبيرة جدًا. بالنظر إلى هذا التوسع، أصبحنا الآن أحرارًافي التحكمفي معدل التعلم η بشكل مستقل عن القياس المطبق على أساس كل إحداثيات. فيما يتعلق بالمتوسطات المتسربة، يمكننا تطبيق نفس المنطق كما تم تطبيقه مسبقًافي حالة طريقة الزخم. توسيع تعريف $\mathbf{s}_t$ ينتج

$$\begin{aligned}\mathbf{s}_t &= (1-\gamma)\mathbf{g}_t^2 + \gamma\mathbf{s}_{t-1} \\ &= (1-\gamma)(\mathbf{g}_t^2 + \gamma\mathbf{g}_{t-1}^2 + \gamma^2\mathbf{g}_{t-2} + \cdots,).\end{aligned}$$

كما سبقفي القسم 12.6 نستخدم $1 + \gamma + \gamma^2 + \cdots, = \frac{1}{1-\gamma}$. ومن ثم يتم تسوية مجموع الأوزان الى 1 مع فترة عمر النصف للمشاهدة γ^{-1}. دعونا نتخيل الأوزان للخطوات الزمنية الأربعين الماضية لاختيارات متنوعة من γ.

```
import math
import tensorflow as tf
from d2l import tensorflow as d2l

d2l.set_figsize()
gammas = [0.95, 0.9, 0.8, 0.7]
for gamma in gammas:
    x = tf.range(40).numpy()
    d2l.plt.plot(x, (1-gamma) * gamma ** x,
label=f'gamma = {gamma:.2f}')
d2l.plt.xlabel('time');
```

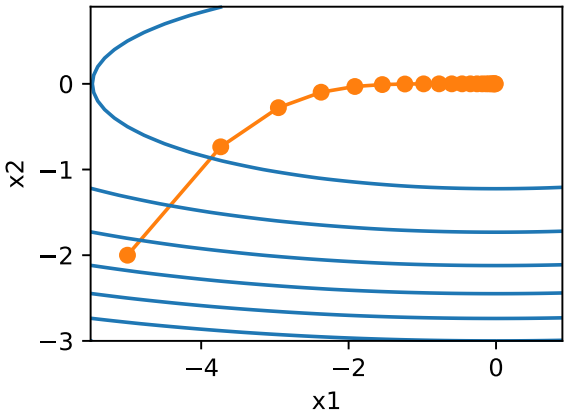


12.8.2. التنفيذ من البداية Implementation from Scratch

كما كان من قبل، نستخدم الدالة التربيعية $f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2$ لمراقبة مسار RMSProp. تذكر أنهفي القسم 12.7، عندما استخدمنا Adagrad بمعدل تعلم 0.4، تحركت المتغيرات ببطء

شديدفي المراحل اللاحقة من الخوارزمية حيث انخفض معدل التعلم بسرعة كبيرة. نظرًا لأن η يتم التحكم فيه بشكل منفصل، فإن هذا لا يحدث مع RMSProp.

```
def rmsprop_2d(x1, x2, s1, s2):
    g1, g2, eps = 0.2 * x1, 4 * x2, 1e-6
    s1 = gamma * s1 + (1 - gamma) * g1 ** 2
    s2 = gamma * s2 + (1 - gamma) * g2 ** 2
    x1 -= eta / math.sqrt(s1 + eps) * g1
    x2 -= eta / math.sqrt(s2 + eps) * g2
    return x1, x2, s1, s2

def f_2d(x1, x2):
    return 0.1 * x1 ** 2 + 2 * x2 ** 2

eta, gamma = 0.4, 0.9
d2l.show_trace_2d(f_2d, d2l.train_2d(rmsprop_2d))
epoch 20, x1: -0.010599, x2: 0.000000
```



بعد ذلك، نقوم بتنفيذ RMSProp لاستخدامهفي شبكة عميقة. هذا واضح بنفس القدر.

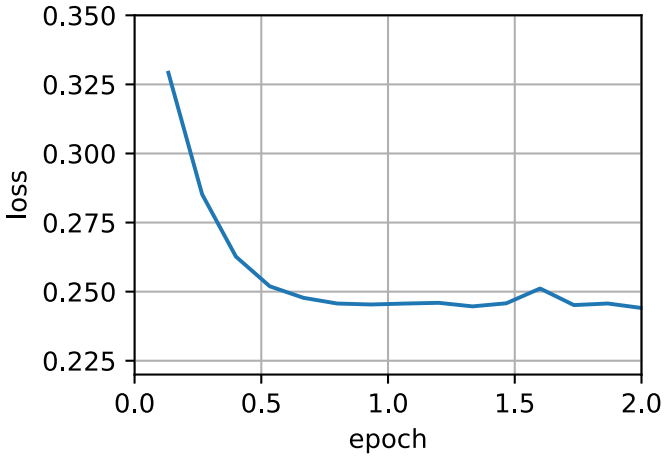
```
def init_rmsprop_states(feature_dim):
    s_w = tf.Variable(tf.zeros((feature_dim, 1)))
    s_b = tf.Variable(tf.zeros(1))
    return (s_w, s_b)

def rmsprop(params, grads, states, hyperparams):
    gamma, eps = hyperparams['gamma'], 1e-6
    for p, s, g in zip(params, states, grads):
```

```
        s[:].assign(gamma * s + (1 - gamma) *
tf.math.square(g))
        p[:].assign(p - hyperparams['lr'] * g /
tf.math.sqrt(s + eps))
```

قمنا بتعيين معدل التعلم الأولي على 0.01 ومصطلح الترجيح γ على 0.9. وهذا يعني ان **s**، تتجمع في المتوسط على $1/(1-\gamma) = 10$ مشاهدات السابقة للانحدار المربع.

```
data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
d2l.train_ch11(rmsprop,
init_rmsprop_states(feature_dim),
               {'lr': 0.01, 'gamma': 0.9}, data_iter,
feature_dim);
```

```
loss: 0.244, 0.114 sec/epoch
```

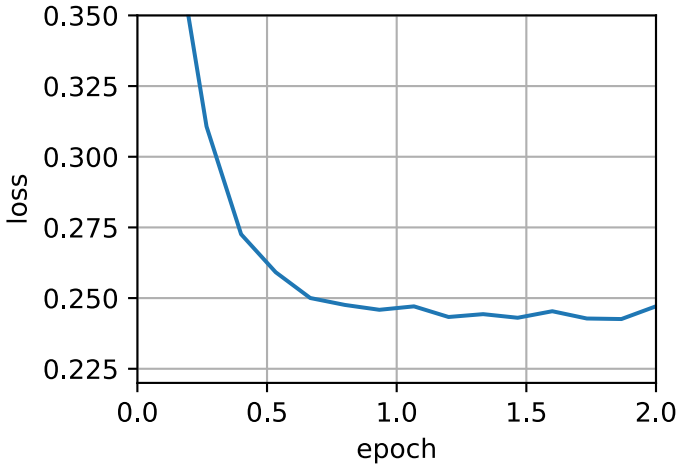


12.8.3. التنفيذ المختصر Concise Implementation

نظرًا لأن RMSProp هي خوارزمية شائعة إلى حد ما، فهي متوفرة أيضًا في نسخة `Trainer`. كل ما نحتاج إلى القيام به هو إنشاء مثيل له باستخدام خوارزمية تسمى `rmsprop`، مع تخصيص γ الى المعلمة `gamma1`.

```
trainer = tf.keras.optimizers.RMSprop
d2l.train_concise_ch11(trainer, {'learning_rate': 0.01,
'rho': 0.9},
                       data_iter)
```

```
loss: 0.247, 0.135 sec/epoch
```



12.8.4. الملخص

- يشبه RMSProp إلى حد بعيد Adagrad بقدر ما يستخدم كلاهما مربع الانحدار لقياس المعاملات.
- يشارك RMSProp مع الزخم المتوسط المتسرب momentum the leaky averaging. ومع ذلك، يستخدم RMSProp هذه التقنية لضبط عامل المكيف المسبق من حيث المعامل coefficient-wise preconditioner.
- معدل التعلم يحتاج إلى جدولة من قبل المجرب في الممارسة.
- يحدد المعامل γ مدة السجل (التاريخ) عند ضبط مقياس كل إحداثيات.

12.8.5. التمارين

1. ماذا يحدث تجريبيا إذا وضعنا $\gamma = 1$؟ لماذا؟
2. قم بتدوير مشكلة التحسين لتقليل $f(\mathbf{x}) = 0.1(x_1 + x_2)^2 + 2(x_1 - x_2)^2$. ماذا يحدث للتقارب convergence؟
3. جرب ما يحدث لـ RMSProp بشأن مشكلة حقيقية في التعلم الآلي، مثل التدريب على Fashion-MNIST. جرب خيارات مختلفة لتعديل معدل التعلم.
4. هل تريد تعديل γ مع تقدم التحسين؟ ما مدى حساسية RMSProp لهذا؟

12.9 Adadelta

Adadelta هو نوع آخر من AdaGrad (القسم 12.7). يكمن الاختلاف الرئيسي في حقيقة أنه يقلل من المقدار الذي يتكيف به معدل التعلم مع الإحداثيات. علاوة على ذلك، يُشار إليه تقليديًا على أنه ليس لديه معدل تعلم لأنه يستخدم مقدار التغيير نفسه كمعايرة للتغيير المستقبلي. تم

اقتراح الخوارزميةفي Zeiler (2012). إنه واضح ومباشر إلى حد ما، بالنظر إلى مناقشة الخوارزميات السابقة حتى الآن.

12.9.1. الخوارزمية The Algorithm

باختصار، يستخدم Adadelta متغيرين للحالة،$\mathbf{s}_t$ لتخزين متوسط متسرب للزخم الثاني من الانحدار و$\Delta\mathbf{x}_t$ لتخزين متوسط متسرب للحظة الثانية لتغيير المعلماتفي النموذج نفسه. لاحظ أننا نستخدم التدوين الأصلي وتسمية المؤلفين للتوافق مع المنشورات والتطبيقات الأخرى (لا يوجد سبب حقيقي آخر لاستخدام متغيرات يونانية مختلفة للإشارة إلى معلمة تخدم نفس الغرضفي الزخم، Adagrad، RMSProp ، و Adadelta).

فيما يلي التفاصيل الفنية لـ Adadelta. بالنظر إلى المعلمة du jour هي ρ ، نحصل على التحديثات التالية المتسربة بشكل مشابه للقسم 12.8:

$$\mathbf{s}_t = \rho\mathbf{s}_{t-1} + (1-\rho)\mathbf{g}_t^2.$$

يتمثل الاختلاففي القسم 12.8في أننا نجري تحديثات باستخدام الانحدار المعاد قياسه $\mathbf{g}_t'$، بمعنى آخر،

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \mathbf{g}_t'.$$

إذن ما هو الانحدار المعاد قياسه $\mathbf{g}_t'$؟ يمكننا حسابه على النحو التالي:

$$\mathbf{g}_t' = \frac{\sqrt{\Delta\mathbf{x}_{t-1} + \epsilon}}{\sqrt{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t,$$

حيث $\Delta\mathbf{x}_{t-1}$ هو المتوسط المتسرب من الانحدارات المعاد قياسها التربيعية $\mathbf{g}_t'$. نقوم بتهيئة $\Delta\mathbf{x}_0$ لتكون 0 وتحديثهافي كل خطوة مع $\mathbf{g}_t'$، بمعنى آخر،

$$\Delta\mathbf{x}_t = \rho\Delta\mathbf{x}_{t-1} + (1-\rho)\mathbf{g}_t'^2,$$

وϵ يتم إضافة (قيمة صغيرة مثل 10^{-5}) للحفاظ على الاستقرار العددي.

12.9.2. التنفيذ Implementation

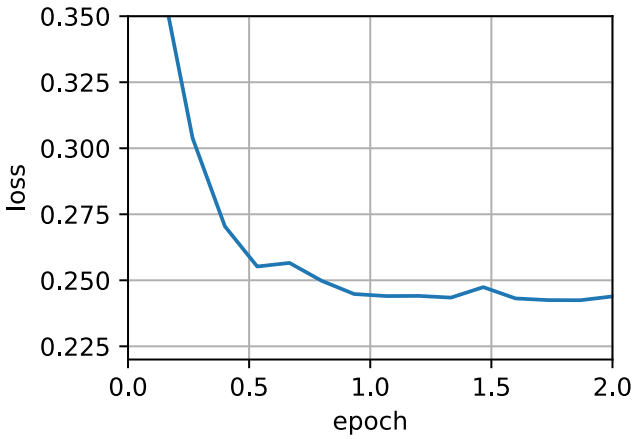
يحتاج Adadelta إلى الحفاظ على متغيرين للحالة لكل متغير،$\mathbf{s}_t$ و$\Delta\mathbf{x}_t$. هذا يؤدي إلى التنفيذ التالي.

```
%matplotlib inline
import tensorflow as tf
from d2l import tensorflow as d2l
```

```
def init_adadelta_states(feature_dim):
    s_w = tf.Variable(tf.zeros((feature_dim, 1)))
    s_b = tf.Variable(tf.zeros(1))
    delta_w = tf.Variable(tf.zeros((feature_dim, 1)))
    delta_b = tf.Variable(tf.zeros(1))
    return ((s_w, delta_w), (s_b, delta_b))

def adadelta(params, grads, states, hyperparams):
    rho, eps = hyperparams['rho'], 1e-5
    for p, (s, delta), grad in zip(params, states,
grads):
        s[:].assign(rho * s + (1 - rho) *
tf.math.square(grad))
        g = (tf.math.sqrt(delta + eps) / tf.math.sqrt(s
+ eps)) * grad
        p[:].assign(p - g)
        delta[:].assign(rho * delta + (1 - rho) * g * g)
```

اختيار كميات $\rho = 0.9$ نصف عمر قدرها 10 لكل تحديث للمعلمة. هذا يميل إلى العمل بشكل جيد. نحصل على السلوك التالي.

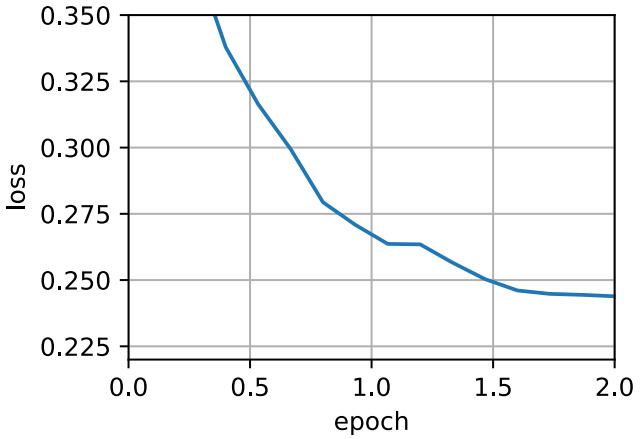
```
data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
d2l.train_ch11(adadelta,
init_adadelta_states(feature_dim),
               {'rho': 0.9}, data_iter, feature_dim);
loss: 0.244, 0.142 sec/epoch
```



لتنفيذ موجز، نستخدم ببساطة خوارزمية `adadelta` من فئة `Trainer`. ينتج عن هذا الخط الواحد التالي لاستدعاء أكثر إحكاما.

```
# adadelta is not converging at default learning rate
# but it's converging at lr = 5.0
trainer = tf.keras.optimizers.Adadelta
d2l.train_concise_ch11(trainer, {'learning_rate':5.0,
'rho': 0.9}, data_iter)
```

```
loss: 0.244, 0.099 sec/epoch
```



12.9.3. الملخص

- لا يوجد لدى Adadelta معلمة معدل التعلم. بدلاً من ذلك، يستخدم معدل التغييرفي المعلمات نفسها لتكييف معدل التعلم.
- يتطلب Adadelta متغيرين للحالة لتخزين اللحظات الثانية من الانحدار والتغييرفي المعلمات.
- يستخدم Adadelta متوسطات متسربة للاحتفاظ بتقدير مستمر للإحصاءات المناسبة.

12.9.4. التمارين

1. اضبط قيمة ρ. ماذا يحدث؟
2. أظهر كيفية تنفيذ الخوارزمية دون استخدام $\mathbf{g}_t'$. لماذا قد تكون هذه فكرة جيدة؟
3. هل Adadelta حقًا مجاني؟ هل يمكن أن تجد مشاكل التحسين التي تكسر Adadelta؟
4. قارن Adadelta مع Adagrad وRMSprop لمناقشة سلوك التقارب بينهما.

12.10. Adam

في المناقشات التي سبقت هذا القسم، واجهنا عددًا من التقنيات لتحسين الكفاءة. دعونا نلخصها بالتفصيل هنا:

- لقد رأينا أن القسم 12.4 أكثر فاعلية من الانحدار عند حل مشكلات التحسين، على سبيل المثال، نظرًا لمرونته المتأصلةفي التعامل مع البيانات الزائدة عن الحاجة redundant data.
- لقد رأينا أن القسم 12.5 يوفر كفاءة إضافية كبيرة ناشئة عن الفيكتورازيشن vectorization، باستخدام مجموعات أكبر من الملاحظاتفي دفعة مصغرة واحدة. هذا هو مفتاح المعالجة المتوازية الشاملة للماكينات المتعددة والمعالجات الرسومية المتعددة.
- أضاف القسم 12.6 آلية لتجميع تاريخ الانحدارات السابقة لتسريع التقارب.
- القسم 12.7 يستخدم القياس لكل إحداثيات per-coordinate scaling للسماح لمكيف مسبق preconditioner فعال حسابيًا.
- القسم 12.8 فصل القياس لكل إحداثيات عن تعديل معدل التعلم.

يجمع Adam، (2014,Kingma and Ba) كل هذه التقنياتفي خوارزمية تعليمية واحدة فعالة. كما هو متوقع، هذه خوارزمية أصبحت شائعة كواحدة من خوارزميات التحسين الأكثر قوة وفعالية لاستخدامهافي التعلم العميق. لا يخلو من المشاكل، على الرغم من ذلك. على وجه الخصوص، يوضح (Reddi et al.، 2019) أن هناك مواقف يمكن أن يتباعد diverge فيها Adam بسبب ضعف التحكمفي التباين.في عمل متابعة (Zaheer et al.، 2018) اقترح إصلاحًا عاجلاً لـ Adam يسمى Yogi يعالج هذه المشكلات. المزيد عن هذا لاحقًا. الآن دعونا نراجع خوارزمية Adam.

12.10.1. الخوارزمية The Algorithm

أحد المكونات الرئيسية لـ Adam هو أنه يستخدم المتوسطات المتحركة المرجحة الأسية (المعروفة أيضًا باسم المتوسط المتسرب leaky averaging) للحصول على تقدير لكل من الزخم واللحظة الثانية للانحدار. أي أنه يستخدم متغيرات الحالة

$$\begin{aligned}\mathbf{v}_t &\leftarrow \beta_1 \mathbf{v}_{t-1} + (1-\beta_1)\mathbf{g}_t,\\ \mathbf{s}_t &\leftarrow \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)\mathbf{g}_t^2.\end{aligned}$$

هنا β_1 و β_2 معلمات ترجيح غير سالبة nonnegative weighting parameters. الخيارات المشتركة بالنسبة لهم هي $\beta_1 = 0.9$ و$\beta_2 = 0.999$. أي أن تقدير التباين يتحرك بشكل أبطأ بكثير من مصطلح الزخم. لاحظ أنه إذا قمنا بتهيئة $\mathbf{v}_0 = \mathbf{s}_0 = 0$ ، فسيكون لدينا قدر كبير من

التحيزفي البداية تجاه القيم الأصغر. يمكن معالجة هذا باستخدام حقيقة أن $\sum_{i=0}^{t} \beta^i = \frac{1-\beta^t}{1-\beta}$ لإعادة تسوية الشروط.في المقابل، يتم إعطاء متغيرات الحالة الطبيعية بواسطة

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_2^t}.$$

مسلحين بالتقديرات الصحيحة يمكننا الآن كتابة معادلات التحديث. أولاً، نقوم بإعادة قياس الانحدار بطريقة تشبه إلى حد كبير طريقة RMSProp للحصول عليها

$$\mathbf{g}_t' = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}.$$

على عكس RMSProp، يستخدم التحديث الخاص بنا الزخم $\hat{\mathbf{v}}_t$ بدلا من الانحدار نفسه. علاوة على ذلك، هناك اختلاف تجميلي طفيف حيث يحدث إعادة القياس باستخدام $\frac{1}{\sqrt{\hat{\mathbf{s}}_t}+\epsilon}$ بدلاً من $\frac{1}{\sqrt{\hat{\mathbf{s}}_t+\epsilon}}$. يمكن القول إن السابق يعمل بشكل أفضل قليلاًفي الممارسة، ومن هنا جاء الانحراف عن RMSProp. عادةً ما نختار $\epsilon = 10^{-6}$ للمقايضة الجيدة بين الاستقرار العددي numerical stability والصحة fidelity.

الآن لدينا كل القطعفي مكانها لحساب التحديثات. يعد هذا من العوامل المضادة للتاثر إلى حد ما ولدينا تحديث بسيط للنموذج

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{g}_t'.$$

مراجعة تصميم Adam هو إلهام واضح. يظهر الزخم والحجم بوضوحفي متغيرات الحالة. يجبرنا تعريفهم الغريب إلى حد ما على المصطلحات إزالة التحيز debias terms (يمكن إصلاح ذلك من خلال شرط تهيئة وتحديث مختلف قليلاً). ثانيًا، يكون الجمع بين كلا المصطلحين واضحًا جدًا، بالنظر إلى RMSProp. أخيرًا، يسمح لنا معدل التعلم η الصريح بالتحكمفي طول الخطوة لمعالجة مشكلات التقارب.

12.10.2. التنفيذ Implementation

إن تنفيذ Adam من الصفر ليس أمرًا شاقًا للغاية. للسهولة، نقوم بتخزين عداد خطوات الوقت في قاموس `hyperparams`. أبعد من ذلك كل شيء واضح ومباشر.

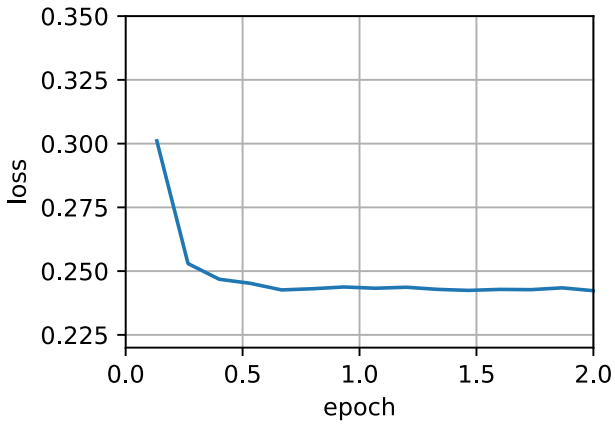
```
%matplotlib inline
import tensorflow as tf
```

```
from d2l import tensorflow as d2l


def init_adam_states(feature_dim):
    v_w = tf.Variable(tf.zeros((feature_dim, 1)))
    v_b = tf.Variable(tf.zeros(1))
    s_w = tf.Variable(tf.zeros((feature_dim, 1)))
    s_b = tf.Variable(tf.zeros(1))
    return ((v_w, s_w), (v_b, s_b))

def adam(params, grads, states, hyperparams):
    beta1, beta2, eps = 0.9, 0.999, 1e-6
    for p, (v, s), grad in zip(params, states, grads):
        v[:].assign(beta1 * v  + (1 - beta1) * grad)
        s[:].assign(beta2 * s + (1 - beta2) *
tf.math.square(grad))
        v_bias_corr = v / (1 - beta1 **
hyperparams['t'])
        s_bias_corr = s / (1 - beta2 **
hyperparams['t'])
        p[:].assign(p - hyperparams['lr'] * v_bias_corr
                    / tf.math.sqrt(s_bias_corr) + eps)
```

نحن على استعداد لاستخدام Adam لتدريب النموذج. نحن نستخدم معدل التعلم $\eta = 0.01$.

```
data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
d2l.train_ch11(adam, init_adam_states(feature_dim),
               {'lr': 0.01, 't': 1}, data_iter,
feature_dim);
```

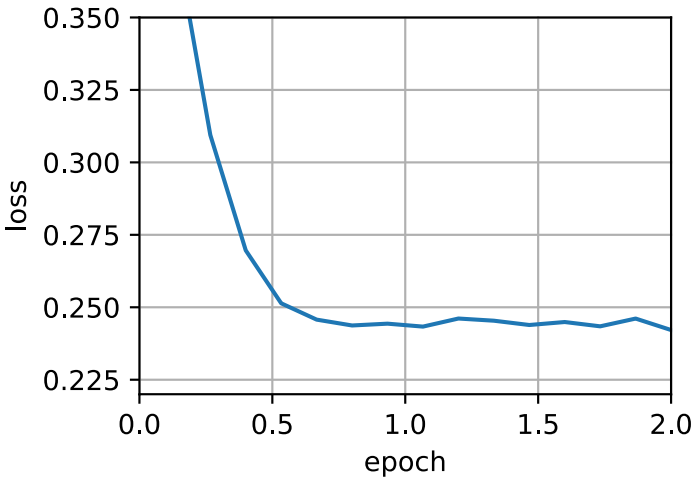
```
loss: 0.242, 0.153 sec/epoch
```



يعد التنفيذ الأكثر إيجازًا أمرًا مباشرًا نظرًا لأن `adam` هو أحد الخوارزميات المتوفرة كجزء من مكتبة تحسين Gluon `trainer`. ومن ثم نحتاج فقط إلى تمرير معلمات التكوين للتنفيذ في Gluon.

```
trainer = tf.keras.optimizers.Adam
d2l.train_concise_ch11(trainer, {'learning_rate': 0.01},
data_iter)
```

```
loss: 0.242, 0.118 sec/epoch
```



Yogi .12.10.3

إحدى مشاكل Adam هي أنه يمكن أن يفشل في التقارب حتى في البيئات المحدبة عند تقدير اللحظة الثانية في $\mathbf{s}_t$ الانفجارات. كإصلاح (Zaheer et al.، 2018) اقترح تحديثًا دقيقًا (وتهيئة) لـ $\mathbf{s}_t$. لفهم ما يجري، دعنا نعيد كتابة تحديث Adam على النحو التالي:

$$\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} + (1 - \beta_2)(\mathbf{g}_t^2 - \mathbf{s}_{t-1}).$$

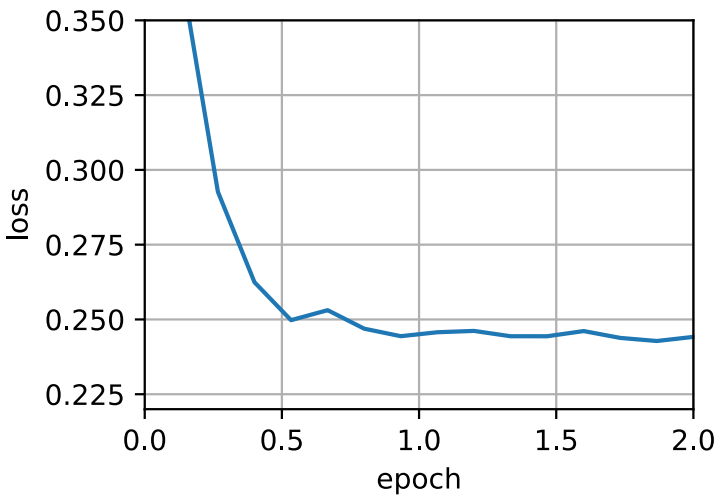
كلما كان $\mathbf{g}_t^2$ لديه تباين كبير أو تحديثات قليلة، $\mathbf{s}_t$ قد تنسى القيم السابقة بسرعة كبيرة. إصلاح محتمل لهذا هو استبدال $\mathbf{g}_t^2 - \mathbf{s}_{t-1}$ بواسطة $\mathbf{g}_t^2 \odot \operatorname{sgn}(\mathbf{g}_t^2 - \mathbf{s}_{t-1})$. الآن لم يعد حجم التحديث يعتمد على مقدار الانحراف. ينتج عن هذا تحديثات Yogi

$$\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2 \odot \operatorname{sgn}(\mathbf{g}_t^2 - \mathbf{s}_{t-1}).$$

علاوة على ذلك، ينصح المؤلفون بتهيئة الزخم على دفعة أولية أكبر بدلاً من مجرد تقدير أولي نقطي. نحذف التفاصيل لأنها ليست جوهرية للمناقشة وبما أنه حتى بدون هذا التقارب يظل جيداً.

```
def yogi(params, grads, states, hyperparams):
    beta1, beta2, eps = 0.9, 0.999, 1e-6
    for p, (v, s), grad in zip(params, states, grads):
        v[:].assign(beta1 * v  + (1 - beta1) * grad)
        s[:].assign(s + (1 - beta2) * tf.math.sign(
                    tf.math.square(grad) - s) *
tf.math.square(grad))
        v_bias_corr = v / (1 - beta1 **
hyperparams['t'])
        s_bias_corr = s / (1 - beta2 **
hyperparams['t'])
        p[:].assign(p - hyperparams['lr'] * v_bias_corr
                    / tf.math.sqrt(s_bias_corr) + eps)
    hyperparams['t'] += 1

data_iter, feature_dim =
d2l.get_data_ch11(batch_size=10)
d2l.train_ch11(yogi, init_adam_states(feature_dim),
               {'lr': 0.01, 't': 1}, data_iter,
feature_dim);
```

```
loss: 0.244, 0.153 sec/epoch
```



12.10.4. الملخص

- يجمع Adam ميزات العديد من خوارزميات التحسينفي قاعدة تحديث قوية إلى حد ما.

- تم إنشاؤه على أساس RMSProp، ويستخدم Adam أيضًا EWMA على الانحدار الاشتقاقي العشوائي المصغر.
- يستخدم Adam تصحيح التحيز لضبط بدء التشغيل البطيء عند تقدير الزخم واللحظة الثانية.
- بالنسبة للانحدارات ذات التباين الكبير، قد نواجه مشكلاتفي التقارب. يمكن تعديلها باستخدام الدفعات الصغيرة أكبر أو عن طريق التبديل إلى تقدير محسّن لـ $\mathbf{s}_t$. يقدم Yogi مثل هذا البديل.

12.10.5. التمارين

1. اضبط معدل التعلم وراقب وحلل النتائج التجريبية.
2. هل يمكنك إعادة كتابة تحديثات الزخم واللحظة الثانية بحيث لا تتطلب تصحيح التحيز؟
3. لماذا تحتاج إلى تقليل معدل التعلم η ونحن نتقارب؟
4. حاول بناء قضية يتباعد فيها Adam ويتقارب Yogi؟

12.11. جدولة معدل التعلم Learning Rate Scheduling

ركزنا حتى الآن بشكل أساسي على خوارزميات التحسين optimization algorithms لكيفية تحديث متجهات الوزن بدلاً من المعدل rate الذي يتم تحديثها به. ومع ذلك، فإن تعديل معدل التعلم غالبًا ما يكون بنفس أهمية الخوارزمية الفعلية. هناك عدد من الجوانب التي يجب مراعاتها:

- من الواضح أن حجم معدل التعلم مهم. إذا كانت كبيرة جدًا، فإن التحسين يتباعد، وإذا كان صغيرًا جدًا، فسيستغرق التدريب وقتًا طويلاً أو ينتهي بنا الأمر بنتيجة دون المستوى الأمثل. رأينا سابقًا أن رقم حالة المشكلة مهم (انظر على سبيل المثال، القسم 12.6 للحصول على التفاصيل). الحدس هو نسبة مقدار التغييرفي الاتجاه الأقل حساسية مقابل الاتجاه الأكثر حساسية.
- ثانيًا، معدل الاضمحلال rate of decay لا يقل أهمية. إذا ظل معدل التعلم كبيرًا، فقد ينتهي بنا الأمر ببساطة إلى الارتداد حول الحد الأدنى وبالتالي لا نصل إلى المستوى الأمثل. ناقش القسم 12.5 هذا ببعض التفاصيل وقمنا بتحليل ضمانات الأداءفي القسم 12.4. باختصار، نريد أن يتحلل المعدل، لكن ربما يكون أبطأ من $\mathcal{O}(t^{-\frac{1}{2}})$ والذي سيكون اختيارًا جيدًا لمشاكل محدبة.
- جانب آخر لا يقل أهمية هو التهيئة initialization. يتعلق هذا بكيفية تعيين المعلمات في البداية (راجع القسم 5.4 للحصول على التفاصيل) وكذلك كيفية تطورهافي البداية. هذا يندرج تحت لقب الإحماء warmup، أي مدى السرعة التي نبدأ بهافي التحرك

نحو الحل في البداية. قد لا تكون الخطوات الكبيرة في البداية مفيدة، خاصة وأن المجموعة الأولية من المعلمات عشوائية. قد تكون اتجاهات التحديث الأولية بلا معنى أيضًا.

- أخيرًا، هناك عدد من متغيرات التحسين التي تقوم بتعديل معدل التعلم الدوري cyclical learning rate. هذا خارج نطاق الفصل الحالي. نوصي القارئ بمراجعة التفاصيل في Izmailov et al. (2018)، على سبيل المثال، كيفية الحصول على حلول أفضل من خلال حساب المتوسط على مسار كامل entire path من المعلمات.

12.11.1. مشكلة لعبة Toy Problem

نبدأ بمشكلة لعبة رخيصة بما يكفي للحساب بسهولة، لكنها غير بديهية بما يكفي لتوضيح بعض الجوانب الرئيسية. لذلك نختار إصدارًا حديثًا قليلاً من LeNet (relu بدلاً من sigmoid، MaxPooling بدلاً من AveragePooling)، كما هو مطبق على Fashion-MNIST. علاوة على ذلك، نقوم بتهجين الشبكة من أجل الأداء. نظرًا لأن معظم الكود قياسي، فإننا نقدم الأساسيات فقط دون مزيد من المناقشة التفصيلية. انظر القسم 7 لاعادة التنشيط حسب الحاجة.

```
%matplotlib inline
import math
import tensorflow as tf
from tensorflow.keras.callbacks import
LearningRateScheduler
from d2l import tensorflow as d2l


def net():
    return tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(filters=6, kernel_size=5,
activation='relu',
                               padding='same'),
        tf.keras.layers.AvgPool2D(pool_size=2,
strides=2),
        tf.keras.layers.Conv2D(filters=16,
kernel_size=5,
                               activation='relu'),
        tf.keras.layers.AvgPool2D(pool_size=2,
strides=2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(120, activation='relu'),
        tf.keras.layers.Dense(84, activation='sigmoid'),
```

```
        tf.keras.layers.Dense(10)])


batch_size = 256
train_iter, test_iter =
d2l.load_data_fashion_mnist(batch_size=batch_size)

# The code is almost identical to `d2l.train_ch6`
defined in the
# lenet section of chapter convolutional neural networks
def train(net_fn, train_iter, test_iter, num_epochs, lr,
              device=d2l.try_gpu(), custom_callback =
False):
    device_name = device._device_name
    strategy =
tf.distribute.OneDeviceStrategy(device_name)
    with strategy.scope():
        optimizer =
tf.keras.optimizers.SGD(learning_rate=lr)
        loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logit
s=True)
        net = net_fn()
        net.compile(optimizer=optimizer, loss=loss,
metrics=['accuracy'])
    callback = d2l.TrainCallback(net, train_iter,
test_iter, num_epochs,
                                 device_name)
    if custom_callback is False:
        net.fit(train_iter, epochs=num_epochs,
verbose=0,
                callbacks=[callback])
    else:
         net.fit(train_iter, epochs=num_epochs,
verbose=0,
                 callbacks=[callback, custom_callback])
    return net
```

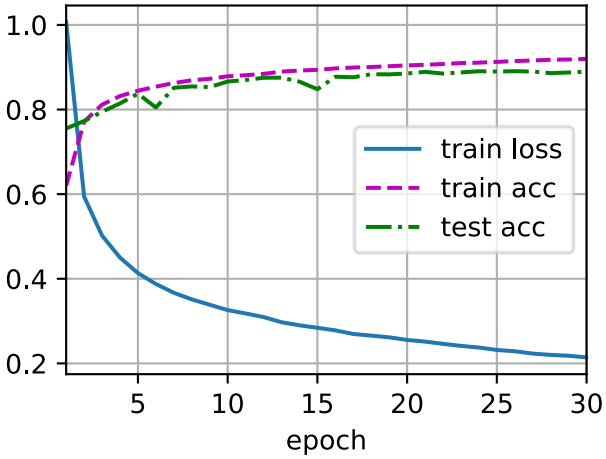
دعونا نلقي نظرة على ما يحدث إذا استدعينا هذه الخوارزمية بإعدادات افتراضية، مثل معدل التعلم 0.3 والتدريب على التكرارات 30. لاحظ كيف تستمر دقة التدريبفي الزيادة بينما التقدم

من حيث دقة الاختبار يتوقف عن نقطة. تشير الفجوة بين كلا المنحنيين إلى فرط التجهيز overfitting.

```
lr, num_epochs = 0.3, 30
train(net, train_iter, test_iter, num_epochs, lr)
```

```
loss 0.214, train acc 0.920, test acc 0.890
61082.9 examples/sec on /GPU:0
```

```
<keras.engine.sequential.Sequential at 0x7f760059e820>
```



12.11.2. المجدولات Schedulers

تتمثل إحدى طرق ضبط معدل التعلم في تعيينه بوضوح في كل خطوة. يتم تحقيق ذلك بسهولة من خلال طريقة `set_learning_rate`. يمكننا تعديله لأسفل بعد كل فترة (أو حتى بعد كل دفعة صغيرة)، على سبيل المثال، بطريقة ديناميكية استجابة لكيفية تقدم التحسين.

```
lr = 0.1
dummy_model =
tf.keras.models.Sequential([tf.keras.layers.Dense(10)])
dummy_model.compile(tf.keras.optimizers.SGD(learning_rat
e=lr), loss='mse')
print(f'learning rate is now ,',
dummy_model.optimizer.lr.numpy())
```

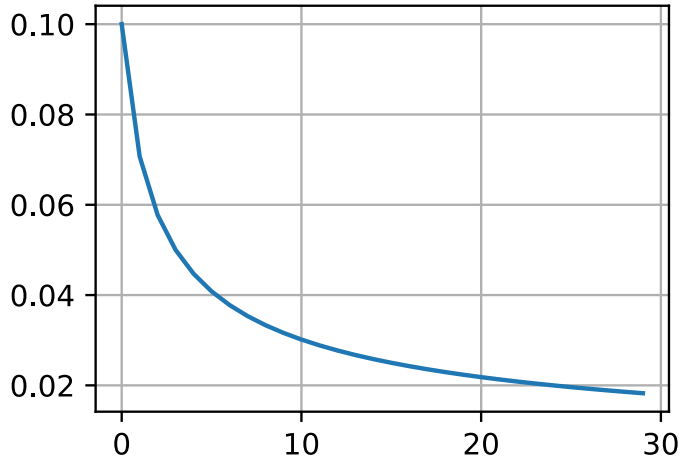
```
learning rate is now , 0.1
```

بشكل عام نريد تحديد scheduler. عندما يتم استدعاؤه مع عدد التحديثات، فإنه يُرجع القيمة المناسبة لمعدل التعلم. دعنا نحدد واحدًا بسيطًا يحدد معدل التعلم لـ $\eta = \eta_0(t+1)^{-\frac{1}{2}}$.

```
class SquareRootScheduler:
    def __init__(self, lr=0.1):
        self.lr = lr

    def __call__(self, num_update):
        return self.lr * pow(num_update + 1.0, -0.5)
```

دعونا نرسم سلوكها على مدى مجموعة من القيم.

```
scheduler = SquareRootScheduler(lr=0.1)
d2l.plot(tf.range(num_epochs), [scheduler(t) for t in
range(num_epochs)])
```

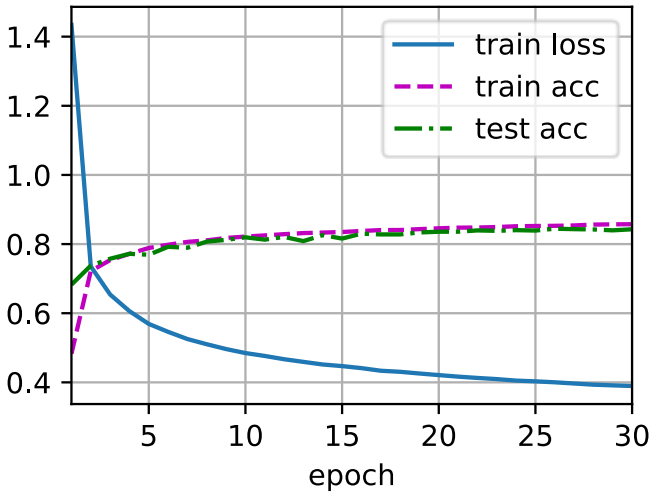


الآن دعونا نرى كيف يتم تنفيذ هذافي التدريب على Fashion-MNIST. نحن ببساطة نوفر scheduler كوسيلة إضافية لخوارزمية التدريب.

```
train(net, train_iter, test_iter, num_epochs, lr,
      custom_callback=LearningRateScheduler(scheduler))
```

```
loss 0.390, train acc 0.857, test acc 0.843
60020.8 examples/sec on /GPU:0
```

```
<keras.engine.sequential.Sequential at 0x7f74d85dc130>
```



هذا عمل أفضل قليلاً من السابق. يبرز شيئين: كان المنحنى أكثر سلاسة من ذي قبل. ثانياً، كان هناك فرط تجهيز أقل. لسوء الحظ، لم يتم حل هذا السؤال بشكل جيد حول السببفي أن بعض الاستراتيجيات تؤدي إلى أقل فرط تجهيز من الناحية النظرية. هناك بعض الحجة القائلة بأن حجم الخطوات الأصغر سيؤدي إلى معلمات أقرب إلى الصفر وبالتالي أبسط. ومع ذلك، فإن هذا لا يفسر الظاهرة تمامًا لأننا لا نتوقف مبكرًا حقًا ولكن ببساطة نخفض معدل التعلم بلطف.

12.11.3. سياسات Policies

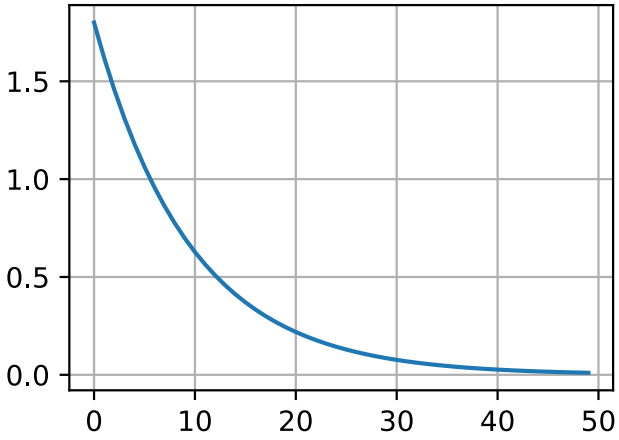
بينما لا يمكننا تغطية المجموعة الكاملة لجدولة معدل التعلم، فإننا نحاول تقديم نظرة عامة موجزة عن السياسات الشائعة أدناه. الخيارات الشائعة هي الاضمحلال متعدد الحدود polynomial decay والجدولات الثابتة متعددة التعريفات piecewise constant schedules. أبعد من ذلك، تم العثور على جدولات معدل التعلم لجيب التمام cosine learning rate schedules للعمل بشكل جيد تجريبيافي بعض المشاكل. أخيرًا،في بعض المشكلات، من المفيد تسخين المحسن قبل استخدام معدلات التعلم الكبيرة.

12.11.3.1. مجدول عامل Factor Scheduler

سيكون أحد البدائل للانحلال متعدد الحدود هو الضرب multiplicative، وهذا هو $\eta_{t+1} \leftarrow \eta_t \cdot \alpha$. لمنع معدل التعلم من الانحلال decaying إلى ما بعد الحد الأدنى المعقول، غالبًا ما يتم تعديل معادلة التحديث إلى $\eta_{t+1} \leftarrow \max(\eta_{\min}, \eta_t \cdot \alpha)$.

```
class FactorScheduler:
    def __init__(self, factor=1, stop_factor_lr=1e-7,
base_lr=0.1):
```

```
        self.factor = factor
        self.stop_factor_lr = stop_factor_lr
        self.base_lr = base_lr

    def __call__(self, num_update):
        self.base_lr = max(self.stop_factor_lr,
self.base_lr * self.factor)
        return self.base_lr

scheduler = FactorScheduler(factor=0.9,
stop_factor_lr=1e-2, base_lr=2.0)
d2l.plot(tf.range(50), [scheduler(t) for t in
range(50)])
```

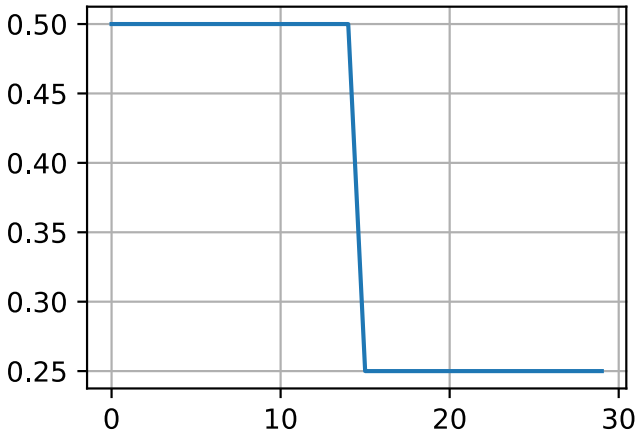


يمكن تحقيق ذلك أيضًا عن طريق برنامج جدولة مدمجفي MXNet عبر الكائن `lr_scheduler.FactorScheduler`. يستغرق الأمر عددًا قليلاً من المعلمات، مثل فترة الإحماء warmup period، ووضع الإحماء warmup mode (خطي أو ثابت)، والحد الأقصى لعدد التحديثات المطلوبة، وما إلى ذلك؛ من الآن فصاعدًا، سنستخدم المجدولين المدمجين حسب الاقتضاء وسنشرح وظائفهم فقط هنا. كما هو موضح، من السهل جدًا إنشاء برنامج الجدولة الخاص بك إذا لزم الأمر.

12.11.3.2. مجدول متعدد العوامل Multi Factor Scheduler

تتمثل الإستراتيجية الشائعة لتدريب الشبكات العميقةفي الحفاظ على ثبات معدل التعلم الجزئي وتقليله بمقدار معين بين الحين والآخر. بمعنى، بالنظر إلى مجموعة الأوقات التي يتم فيها تقليل المعدل، مثل $s = \{5,10,20\}$ يقلل $\eta_{t+1} \leftarrow \eta_t \cdot \alpha$ عندما $t \in s$. بافتراض أن القيم تنقسم إلى النصففي كل خطوة، يمكننا تنفيذ ذلك على النحو التالي.

```
class MultiFactorScheduler:
    def __init__(self, step, factor, base_lr):
        self.step = step
        self.factor = factor
        self.base_lr = base_lr

    def __call__(self, epoch):
        if epoch in self.step:
            self.base_lr = self.base_lr * self.factor
            return self.base_lr
        else:
            return self.base_lr

scheduler = MultiFactorScheduler(step=[15, 30],
factor=0.5, base_lr=0.5)
d2l.plot(tf.range(num_epochs), [scheduler(t) for t in
range(num_epochs)])
```

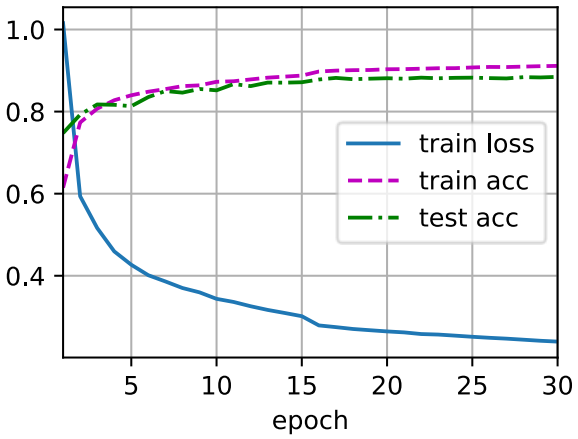


الحدس الكامن وراء جدولة معدل التعلم الثابت متعدد التعريف هذا هو أن المرء يتيح المضي قدمًا في التحسين حتى يتم الوصول إلى نقطة ثابتة من حيث توزيع متجهات الوزن. ثم (وبعد ذلك فقط) نقوم بتخفيض المعدل مثل الحصول على وكيل proxy عالي الجودة إلى حد أدنى محلي جيد. يوضح المثال أدناه كيف يمكن أن ينتج عن ذلك حلول أفضل قليلاً.

```
train(net, train_iter, test_iter, num_epochs, lr,
      custom_callback=LearningRateScheduler(scheduler))
```

```
loss 0.240, train acc 0.911, test acc 0.885
61052.7 examples/sec on /GPU:0
```

```
<keras.engine.sequential.Sequential at 0x7f754a2f4760>
```



12.11.3.3. مجدول جيب التمام Cosine Scheduler

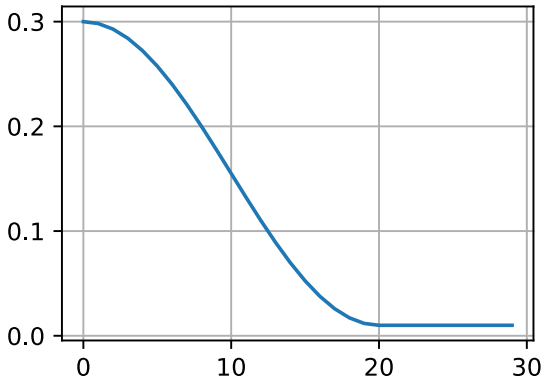
اقترح Loshchilov وHutter (2016) أسلوبًا محيرًا إلى حد ما. إنه يعتمد على المشاهدة التي مفادها أننا قد لا نرغب في تقليل معدل التعلم بشكل كبير جدًا في البداية، وعلاوة على ذلك، قد نرغب في "تنقية refine " الحل في النهاية باستخدام معدل تعلم صغير جدًا. ينتج عن هذا جدول يشبه جيب التمام مع الشكل الدالي التالي لمعدلات التعلم في النطاق $t \in [0, T]$.

$$\eta_t = \eta_T + \frac{\eta_0 - \eta_T}{2}(1 + \cos(\pi t/T))$$

هنا η_0 معدل التعلم الأولي، η_T هو المعدل المستهدف في الوقت T. علاوة على ذلك، لـ $t > T$ نحن ببساطة نثبِّت القيمة الى η_T بدون زيادتها مرة أخرى. في المثال التالي، قمنا بتعيين الحد الأقصى لخطوة التحديث $T = 20$.

```
class CosineScheduler:
    def __init__(self, max_update, base_lr=0.01,
final_lr=0,
                 warmup_steps=0, warmup_begin_lr=0):
        self.base_lr_orig = base_lr
        self.max_update = max_update
        self.final_lr = final_lr
        self.warmup_steps = warmup_steps
        self.warmup_begin_lr = warmup_begin_lr
        self.max_steps = self.max_update -
self.warmup_steps

    def get_warmup_lr(self, epoch):
```

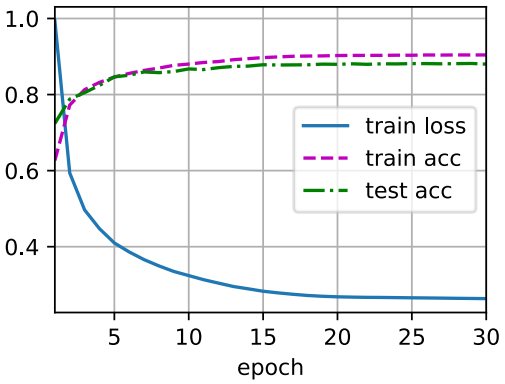
```
        increase = (self.base_lr_orig -
self.warmup_begin_lr) \
                        * float(epoch) /
float(self.warmup_steps)
        return self.warmup_begin_lr + increase

    def __call__(self, epoch):
        if epoch < self.warmup_steps:
            return self.get_warmup_lr(epoch)
        if epoch <= self.max_update:
            self.base_lr = self.final_lr + (
                self.base_lr_orig - self.final_lr) * (1
+ math.cos(
                math.pi * (epoch - self.warmup_steps) /
self.max_steps)) / 2
        return self.base_lr

scheduler = CosineScheduler(max_update=20, base_lr=0.3,
final_lr=0.01)
d2l.plot(tf.range(num_epochs), [scheduler(t) for t in
range(num_epochs)])
```



في سياق الرؤية الحاسوبية، يمكن أن يؤدي هذه الجدولة إلى نتائج محسنة. لاحظ، مع ذلك، أن هذه التحسينات ليست مضمونة (كما يمكن رؤيته أدناه).

```
train(net, train_iter, test_iter, num_epochs, lr,
      custom_callback=LearningRateScheduler(scheduler))
```

```
loss 0.263, train acc 0.904, test acc 0.880
59382.4 examples/sec on /GPU:0
<keras.engine.sequential.Sequential at 0x7f754a3e1040>
```

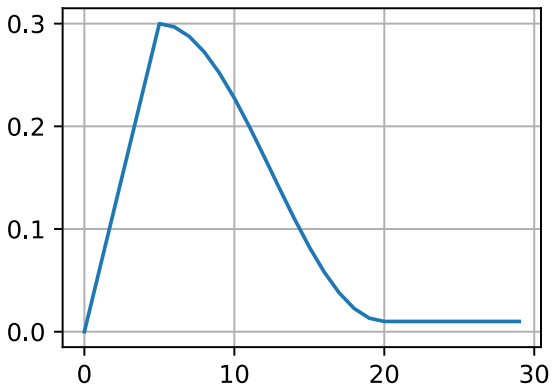


12.11.3.4. الاحماء Warmup

في بعض الحالات، لا تكون تهيئة المعلمات كافية لضمان حل جيد. هذه مشكلة خاصة لبعض تصميمات الشبكات المتقدمة التي قد تؤدي إلى مشاكل تحسين غير مستقرة. يمكننا معالجة هذا عن طريق اختيار معدل تعليمي صغير بما يكفي لمنع الاختلاففي البداية. لسوء الحظ، هذا يعني أن التقدم بطيء. على العكس من ذلك، يؤدي معدل التعلم الكبيرفي البداية إلى التباعد divergence.

يتمثل أحد الحلول البسيطة لهذه المعضلةفي استخدام فترة إحماء warmup period يرتفع خلالها معدل التعلم إلى الحد الأقصى الأولي وتهدئة المعدل حتى نهاية عملية التحسين. للتبسيط يستخدم المرء عادة زيادة خطية لهذا الغرض. هذا يؤدي إلى جدولة للنموذج الموضح أدناه.

```
scheduler = CosineScheduler(20, warmup_steps=5,
base_lr=0.3, final_lr=0.01)
d2l.plot(tf.range(num_epochs), [scheduler(t) for t in
range(num_epochs)])
```

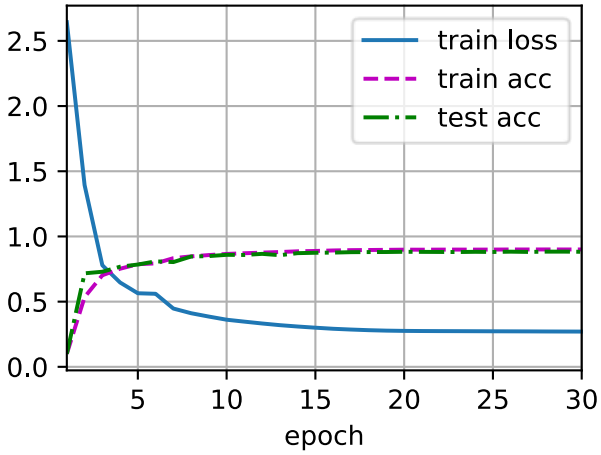


لاحظ أن الشبكة تتقارب بشكل أفضل في البداية (ولا سيما مراقبة الأداء خلال الفترات الخمس الأولى).

```
train(net, train_iter, test_iter, num_epochs, lr,
      custom_callback=LearningRateScheduler(scheduler))
```

```
loss 0.270, train acc 0.900, test acc 0.882
60590.3 examples/sec on /GPU:0
```

```
<keras.engine.sequential.Sequential at 0x7f7549e17250>
```



يمكن تطبيق الإحماء على أي مجدول (وليس فقط جيب التمام). لمزيد من المناقشة التفصيلية لجدولات معدل التعلم والعديد من التجارب، انظر أيضًا (Gotmare et al.، 2018). على وجه الخصوص، وجدوا أن مرحلة الاحماء تحد من مقدار اختلاف المعلمات في الشبكات العميقة جدًا. هذا منطقي بشكل بديهي لأننا نتوقع تباعدًا كبيرًا بسبب التهيئة العشوائية في تلك الأجزاء من الشبكة التي تستغرق معظم الوقت لإحراز تقدم في البداية.

12.11.4. الملخص

- يمكن أن يؤدي خفض معدل التعلم أثناء التدريب إلى تحسين الدقة و (الأكثر حيرة most perplexingly) تقليل فرط تجهيز النموذج.
- يعد التخفيض التدريجي لمعدل التعلم كلما توقف التقدم أمرًا فعالًا في الممارسة العملية. يضمن هذا بشكل أساسي أننا نتقارب بكفاءة إلى حل مناسب وبعد ذلك فقط نخفض التباين المتأصل في المعلمات عن طريق تقليل معدل التعلم.
- جدولة جيب التمام شائعة لبعض مشاكل الرؤية الحاسوبية. انظر على سبيل المثال، GluonCV للحصول على تفاصيل مثل هذا المجدول.

- فترة الإحماء warmup period قبل التحسين يمكن أن تمنع التباين divergence.
- يخدم التحسين أغراضًا متعددةفي التعلم العميق. إلى جانب تقليل هدف التدريب، يمكن أن تؤدي الاختيارات المختلفة لخوارزميات التحسين وجدولة معدل التعلم إلى كميات مختلفة إلى حد ما من التعميم وفرط التجهيز على مجموعة الاختبار (لنفس مقدار خطأ التدريب).

12.11.5. تمارين

1. جرب سلوك التحسين لمعدل تعلم ثابت معين. ما هو أفضل نموذج يمكنك الحصول عليه بهذه الطريقة؟
2. كيف يتغير التقارب إذا قمت بتغيير أس الانخفاضفي معدل التعلم؟ استخدم `PolyScheduler` من اجل راحتكفي التجارب.
3. قم بتطبيق جدولة جيب التمام على مشاكل الرؤية الحاسوبية الكبيرة، على سبيل المثال، تدريب ImageNet. كيف تؤثر على الأداء بالنسبة المجدولين الآخرين؟
4. كم من الوقت يجب أن تستمر عملية الإحماء؟
5. هل يمكنك ربط التحسين optimization وأخذ العينات sampling؟ ابدأ باستخدام النتائج من Welling and Teh (2011)في Stochastic Gradient Langevin Dynamics.

13

الأداء الحـــابي

13. الأداء الحسابي Computational Performance

في التعلم العميق، عادةً ما تكون مجموعات البيانات والنماذج كبيرة، مما يتضمن عمليات حسابية ثقيلة. لذلك، فإن الأداء الحسابي computational performance مهم كثيرًا. سيركز هذا الفصل على العوامل الرئيسية التي تؤثر على الأداء الحسابي: البرمجة الإلزامية imperative programming، والبرمجة الرمزية symbolic programming، والحوسبة غير المتزامنة asynchronous computing، والتوازي التلقائي automatic parallelism، والحساب متعدد وحدات معالجة الرسومات multi-GPU computation. من خلال دراسة هذا الفصل، يمكنك تحسين الأداء الحسابي لتلك النماذج المطبقةفي الفصول السابقة، على سبيل المثال، عن طريق تقليل وقت التدريب دون التأثير على الدقة.

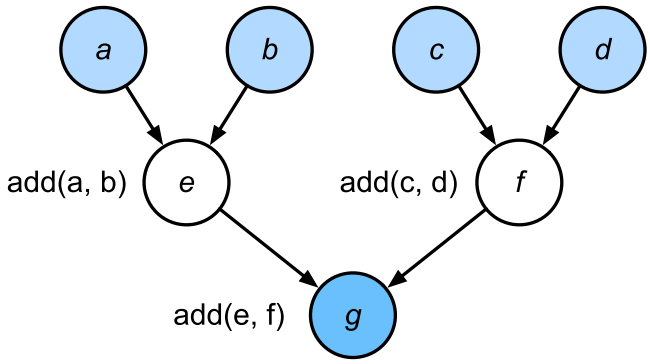
13.1. المترجمات والمفسرات Compilers and Interpreters

حتى الآن، ركز هذا الكتاب على البرمجة الإلزامية (الامرية) imperative programming، والتي تستخدم عبارات مثل `print`و `+` و `if` لتغيير حالة البرنامج. ضعفي اعتبارك المثال التالي لبرنامج أمر بسيط.

```
def add(a, b):
    return a + b

def fancy_func(a, b, c, d):
    e = add(a, b)
    f = add(c, d)
    g = add(e, f)
    return g

print(fancy_func(1, 2, 3, 4))
```

```
10
```

بايثون هي لغة مفسرة interpreted language. عند تقييم دالة `fancy_func` أعلاه، فإنها تقوم بتنفيذ العمليات التي يتكون منها جسم الدالة بالتسلسل. أي أنه سيقيم `e = add(a, b)` ويخزن النتائج كمتغير `e` ، وبالتالي يغير حالة البرنامج. سيتم تنفيذ العبارتين التاليتين `f = add(c, d)` و `g = add(e, f)` بشكل مشابه ، مع إجراء الإضافات وتخزين النتائج كمتغيرات. يوضح الشكل 13.1.1 تدفق البيانات.



الشكل 13.1.1 تدفق البياناتفي برنامج الزامي.

على الرغم من أن البرمجة الإلزامية ملائمة، إلا أنها قد تكون غير فعالة. من ناحية أخرى، حتى إذا تم استدعاء دالة الإضافة بشكل متكرر خلال `fancy_func`، فإن بايثون ستنفذ استدعاءات الدوال الثلاثة بشكل فردي. إذا تم تنفيذ هذه، على سبيل المثال، على وحدة معالجة الرسومات (أو حتى على وحدات معالجة رسومات متعددة)، فقد يصبح الحمل الزائد overhead الناتج عن مفسر بايثون ساحقًا. علاوة على ذلك، سيحتاج إلى حفظ القيم المتغيرة لـ `e` و `f` حتى يتم تنفيذ جميع العباراتفي `fancy_func`. هذا لأننا لا نعرف ما إذا كان سيتم استخدام المتغيرين `e` و `f` بواسطة أجزاء أخرى من البرنامج بعد تنفيذ العبارات `e = add(a, b)` و `f = add(c, d)`.

13.1.1. البرمجة الرمزية Symbolic Programming

ضعفي اعتبارك البرمجة الرمزية symbolic programming البديلة، حيث يتم إجراء الحساب عادةً بمجرد تحديد العملية بالكامل. يتم استخدام هذه الاستراتيجية من قبل العديد من أطر التعلم العميق، بمافي ذلك Theano وTensorFlow (هذا الأخير قد اكتسب امتدادات ضرورية). عادة ما يتضمن الخطوات التالية:

1. حدد العمليات التي سيتم تنفيذها.
2. جمع العمليات في برنامج قابل للتنفيذ.
3. وفر المدخلات المطلوبة واستدعي البرنامج المترجم للتنفيذ.

هذا يسمح بقدر كبير من التحسين. أولاً، يمكننا تخطي مفسر بايثونفي كثير من الحالات، وبالتالي إزالة الاختناقفي الأداء الذي يمكن أن يصبح مهمًافي العديد من وحدات معالجة الرسومات السريعة المقترنة بخيط thread بايثون واحد على وحدة المعالجة المركزية. ثانيًا، قد يقوم المترجم بتحسين الكود أعلاه وإعادة كتابته إلى `print((1 + 2) + (3 + 4))` أو حتى

`print(10)`. هذا ممكن لأن المترجم يمكنه رؤية الكود الكامل قبل تحويله إلى تعليمات الآلة. على سبيل المثال، يمكنه تحرير الذاكرة (أو عدم تخصيصها مطلقًا) متى لم تعد هناك حاجة إلى متغير. أو يمكنها تحويل الشفرة بالكامل إلى قطعة مكافئة. للحصول على فكرة أفضل، ضع في اعتبارك المحاكاة التالية للبرمجة الإلزامية (إنها بايثون بعد كل شيء) أدناه.

```
def add_():
    return '''
def add(a, b):
    return a + b
'''

def fancy_func_():
    return '''
def fancy_func(a, b, c, d):
    e = add(a, b)
    f = add(c, d)
    g = add(e, f)
    return g
'''

def evoke_():
    return add_() + fancy_func_() + 'print(fancy_func(1,
2, 3, 4))'

prog = evoke_()
print(prog)
y = compile(prog, '', 'exec')
exec(y)
```

```
def add(a, b):
    return a + b

def fancy_func(a, b, c, d):
    e = add(a, b)
    f = add(c, d)
    g = add(e, f)
    return g
print(fancy_func(1, 2, 3, 4))
10
```

الاختلافات بين البرمجة الالزامية (المفسرة) والبرمجة الرمزية هي كما يلي:

- البرمجة الالزامية أسهل. عند استخدام البرمجة الإلزامية في بايثون ، فإن غالبية التعليمات البرمجية مباشرة وسهلة الكتابة. من الأسهل أيضًا تصحيح أخطاء رمز البرمجة الإلزامي. هذا لأنه من الأسهل الحصول على جميع قيم المتغيرات الوسيطة ذات الصلة وطباعتها ، أو استخدام أدوات تصحيح الأخطاء المضمنة في بايثون.
- البرمجة الرمزية أكثر كفاءة وأسهل في النقل. تجعل البرمجة الرمزية من السهل تحسين الكود أثناء التجميع ، مع القدرة أيضًا على نقل البرنامج إلى تنسيق مستقل عن بايثون. يتيح ذلك تشغيل البرنامج في بيئة غير بايثون ، وبالتالي تجنب أي مشكلات محتملة تتعلق بالأداء تتعلق بمترجم بايثون.

13.1.2. البرمجة الهجينة Hybrid Programming

تاريخيًا، تختار معظم أطر التعلم العميقة بين نهج إلزامي أو رمزي. على سبيل المثال، تقوم Theano وTensorFlow (المستوحاة من السابق) و Keras و CNTK بصياغة النماذج بشكل رمزي. على العكس من ذلك، يتخذ Chainer و PyTorch نهجًا إلزاميًا. تمت إضافة وضع إلزامي إلى TensorFlow 2.0 و Kerasفي المراجعات اللاحقة.

نموذج البرمجة الإلزامية هو الآن الافتراضيفي Tensorflow 2، وهو تغيير ترحيبي لأولئك الجدد على اللغة. ومع ذلك، فإن نفس تقنيات البرمجة الرمزية والرسوم البيانية الحسابية اللاحقة لا تزال موجودةفي TensorFlow، ويمكن الوصول إليها بواسطة مصمم دالة `tf` سهل الاستخدام. أدى ذلك إلى جلب نموذج البرمجة الإلزامية إلى TensorFlow، مما سمح للمستخدمين بتحديد المزيد من الدوال البديهية، ثم لفها وتجميعهافي رسوم بيانية حسابية تلقائيًا باستخدام ميزة يشير إليها فريق TensorFlow بـ autograph.

13.1.3. تهجين الفئة المتسلسلة Hybridizing the Sequential Class

أسهل طريقة للتعرف على كيفية عمل التهجين هي التفكيرفي الشبكات العميقة ذات الطبقات المتعددة. تقليديًا، سيحتاج مترجم بايثون إلى تنفيذ التعليمات البرمجية لجميع الطبقات لإنشاء تعليمات يمكن إعادة توجيهها بعد ذلك إلى وحدة المعالجة المركزية أو وحدة معالجة الرسومات. بالنسبة لجهاز حوسبة واحد (سريع)، لا يسبب هذا أية مشكلات كبيرة. من ناحية أخرى، إذا استخدمنا خادم GPU_8 متقدمًا مثل AWS P3dn.24xlarge، فستواجه بايثون صعوبةفي إبقاء جميع وحدات معالجة الرسومات مشغولة. يصبح مترجم بايثون المفرد هو عنق الزجاجة هنا. دعونا نرى كيف يمكننا معالجة هذا لأجزاء مهمة من التعليمات البرمجية عن طريق استبدال التسلسل مع `HybridSequential`. نبدأ بتعريف MLP بسيط.

```
import tensorflow as tf
from tensorflow.keras.layers import Dense
from d2l import tensorflow as d2l


# Factory for networks
def get_net():
    net = tf.keras.Sequential()
    net.add(Dense(256, input_shape = (512,), activation
= "relu"))
    net.add(Dense(128, activation = "relu"))
    net.add(Dense(2, activation = "linear"))
    return net

x = tf.random.normal([1,512])
net = get_net()
net(x)
```

```
<tf.Tensor: shape=(1, 2), dtype=float32, numpy=array([[-
2.1462104, -0.9503298]], dtype=float32)>
```

في السابق، تم إنشاء جميع الدوال التي تم إنشاؤهافي TensorFlow كرسم بياني حسابي، وبالتالي تم تجميع JIT افتراضيًا. ومع ذلك، مع إصدار TensorFlow 2.X وEagerTensor، لم يعد هذا هو السلوك الافتراضي. سنقوم بإعادة تمكين هذه الدالة مع دالة tf. تُستخدم دالة tf بشكل أكثر شيوعًا كديكور دالي، ولكن من الممكن تسميتها مباشرة كدالة بايثون عادية، كما هو موضح أدناه. تظل نتيجة حساب النموذج دون تغيير.

```
net = tf.function(net)
net(x)
```

```
<tf.Tensor: shape=(1, 2), dtype=float32, numpy=array([[-
2.1462104, -0.9503298]], dtype=float32)>
```

يبدو هذا جيدًا جدًا لدرجة يصعب تصديقها: اكتب نفس الرمز كما كان من قبل وقم ببساطة بتحويل النموذج باستخدام دالة tf. بمجرد حدوث ذلك، يتم إنشاء الشبكة كرسم بياني حسابيفي التمثيل الوسيط MLIR الخاص بـ TensorFlow ويتم تحسينها بشكل كبير على مستوى المترجم للتنفيذ السريع (سنقوم بقياس الأداء أدناه). تؤدي إضافة `jit_compile = True` إلى استدعاء `tf.function()` إلى تمكين دالة XLA (الجبر الخطي المتسارعAccelerated Linear Algebra)في TensorFlow. يمكن لـ XLA تحسين كود JIT المترجمفي حالات معينة. يتم تمكين تنفيذ وضع الرسم البياني بدون هذا التعريف الواضح،

ومع ذلك يمكن لـ XLA إجراء عمليات جبر خطية كبيرة معينة (في سياق تلك التي نراهافي تطبيقات التعلم العميق) بشكل أسرع، لا سيمافي بيئة وحدة معالجة الرسومات.

13.1.3.1. التسريع عن طريق التهجين Acceleration by Hybridization

لإثبات تحسن الأداء المكتسب من خلال الترجمة compilation، نقارن الوقت اللازم لتقييم `net(x)` قبل وبعد التهجين. دعونا نحدد فصلًا لقياس هذه المرة أولاً. سيكون مفيدًا طوال الفصل حيث شرعنافي قياس (وتحسين) الأداء.

```
#@save
class Benchmark:
    """For measuring running time."""
    def __init__(self, description='Done'):
        self.description = description

    def __enter__(self):
        self.timer = d2l.Timer()
        return self

    def __exit__(self, *args):
        print(f'{self.description}:
{self.timer.stop():.4f} sec')
```

يمكننا الآن استدعاء الشبكة ثلاث مرات، مرة واحدة يتم تنفيذها بفارغ الصبر، ومرة بتنفيذ وضع الرسم البياني، ومرة أخرى باستخدام JIT المترجم XLA.

```
net = get_net()
with Benchmark('Eager Mode'):
    for i in range(1000): net(x)

net = tf.function(net)
with Benchmark('Graph Mode'):
    for i in range(1000): net(x)
```

```
Eager Mode: 1.2446 sec
Graph Mode: 0.7084 sec
```

كما لوحظفي النتائج المذكورة أعلاه، بعد كتابة `tf.keras.Sequential` مثيل باستخدام دالة `tf.function` ، يتم تحسين أداء الحوسبة من خلال استخدام البرمجة الرمزية عبر تنفيذ وضع الرسم البيانيفي tensorflow.

13.1.3.2. التسلسل Serialization

تتمثل إحدى فوائد ترجمة النماذجفي أنه يمكننا إجراء تسلسل serialize (حفظ) للنموذج ومعلماته على القرص. هذا يسمح لنا بتخزين نموذج بطريقة مستقلة عن لغة الواجهة الأمامية المختارة. يتيح لنا ذلك نشر النماذج المدربة على الأجهزة الأخرى واستخدام لغات البرمجة الأمامية الأخرى بسهولة أو تنفيذ نموذج مدرب على الخادم.في الوقت نفسه، غالبًا ما يكون الكود أسرع مما يمكن تحقيقهفي البرمجة الإلزامية. واجهة برمجة التطبيقات ذات المستوى المنخفض التي تسمح لنا بالحفظفي tensorflow هي `tf.saved_model`. دعونا نرى نسخة `saved_model` قيد التشغيل.

```
net = get_net()
tf.saved_model.save(net, 'my_mlp')
!ls -lh my_mlp*
```

```
INFO:tensorflow:Assets written to: my_mlp/assets
total 72K
drwxr-xr-x 2 d2l-worker d2l-worker 4.0K Sep  7 23:37
assets
-rw-rw-r-- 1 d2l-worker d2l-worker  64K Sep  7 23:37
saved_model.pb
drwxr-xr-x 2 d2l-worker d2l-worker 4.0K Sep  7 23:37
variables
```

13.1.4. الملخص

- تجعل البرمجة الالزامية Imperative programming من السهل تصميم نماذج جديدة حيث أنه من الممكن كتابة التعليمات البرمجية مع التحكم في التدفق والقدرة على استخدام قدر كبير من الايكوسيستم لبرمجيات بايثون.
- تتطلب البرمجة الرمزية Symbolic programming تحديد البرنامج وتجميعه قبل تنفيذه. الفائدة هي تحسين الأداء.

13.1.5. التمارين

1. راجع النماذج التي تهمك في الفصول السابقة. هل يمكنك تحسين أدائهم الحسابي من خلال إعادة تنفيذها؟

13.2. الحساب غير المتزامن Asynchronous Computation

أجهزة الكمبيوتر اليوم هي أنظمة متوازية للغاية، وتتألف من أنوية متعددة لوحدة المعالجة المركزية (غالبًا ما تكون خيوط متعددة لكل نواة)، وعناصر معالجة متعددة لكل وحدة معالجة رسومات، وغالبًا ما تكون وحدات معالجة رسومات متعددة لكل جهاز. باختصار، يمكننا معالجة العديد

من الأشياء المختلفةفي نفس الوقت، غالبًا على أجهزة مختلفة. لسوء الحظ، فإن بايثون ليست طريقة رائعة لكتابة كود متوازي وغير متزامن، على الأقل ليس بدون بعض المساعدة الإضافية. بعد كل شيء، بايثون هي خيط واحد ومن غير المرجح أن يتغير هذافي المستقبل. تتبنى أطر التعلم العميق مثل MXNet وTensorFlow نموذج برمجة غير متزامن لتحسين الأداء، بينما تستخدم PyTorch جدولة بايثون الخاصة مما يؤدي إلى مقايضة أداء مختلفة. بالنسبة إلى PyTorch، تكون عمليات وحدة معالجة الرسومات غير متزامنة بشكل افتراضي. عند استدعاء دالة تستخدم وحدة معالجة الرسومات، يتم وضع العملياتفي قائمة الانتظار لجهاز معين، ولكن لا يتم تنفيذها بالضرورة حتى وقت لاحق. يتيح لنا ذلك تنفيذ المزيد من العمليات الحسابية بالتوازي، بمافي ذلك العمليات على وحدة المعالجة المركزية أو وحدات معالجة الرسومات الأخرى.

```
import os
import subprocess
import numpy
from mxnet import autograd, gluon, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

13.2.1. عدم التزامن عبر الواجهة الخلفية Asynchrony via Backend

بالنسبة للإحماء warmup، ضعفي اعتبارك مشكلة اللعبة التالية: نريد إنشاء مصفوفة عشوائية وضربها. دعونا نفعل ذلكفي كل من NumPy و `mxnet.np` لمعرفة الفرق.

```
with d2l.Benchmark('numpy'):
    for _ in range(10):
        a = numpy.random.normal(size=(1000, 1000))
        b = numpy.dot(a, a)

with d2l.Benchmark('mxnet.np'):
    for _ in range(10):
        a = np.random.normal(size=(1000, 1000))
        b = np.dot(a, a)
```

```
numpy: 1.3209 sec
mxnet.np: 0.0261 sec
```

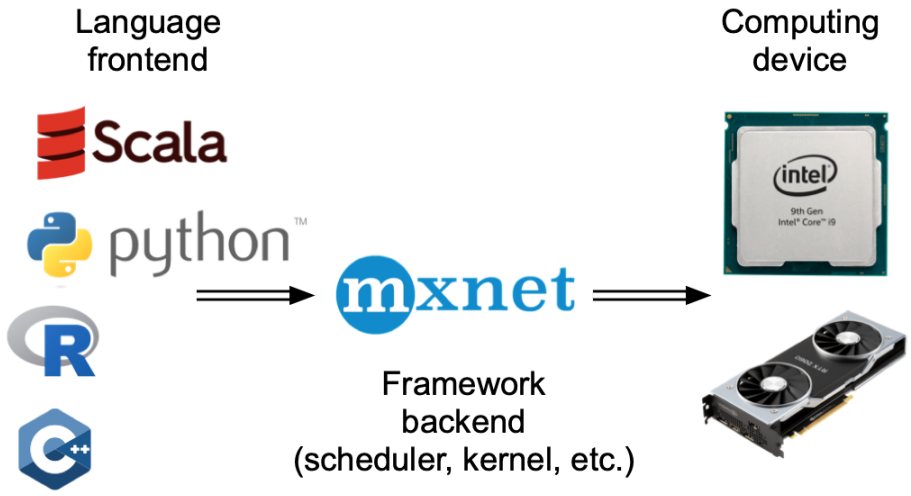
الناتج المعياري عبر MXNet هو أوامر من حيث الحجم أسرع. نظرًا لأن كليهما يتم تنفيذهما على نفس المعالج، فيجب أن يحدث شيء آخر. يُظهر إجبار MXNet على إنهاء جميع حسابات

الواجهة الخلفية backend قبل العودة ما حدث سابقًا: يتم تنفيذ الحساب بواسطة الواجهة الخلفية بينما تعيد الواجهة الأمامية frontend التحكم إلى بايثون.

```
with d2l.Benchmark():
    for _ in range(10):
        a = np.random.normal(size=(1000, 1000))
        b = np.dot(a, a)
    npx.waitall()
```

```
Done: 0.7954 sec
```

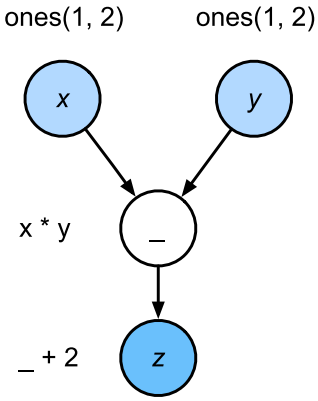
بشكل عام، لدى MXNet واجهة أمامية للتفاعلات المباشرة مع المستخدمين، على سبيل المثال، عبر بايثون، بالإضافة إلى الواجهة الخلفية التي يستخدمها النظام لإجراء الحساب. كما هو موضح في الشكل 13.2.1، يمكن للمستخدمين كتابة برامج MXNet بلغات أمامية مختلفة، مثل Python وR و Scala و C++. بغض النظر عن لغة برمجة الواجهة الأمامية المستخدمة، يتم تنفيذ برامج MXNet بشكل أساسي في الواجهة الخلفية لتطبيقات C++. يتم تمرير العمليات الصادرة عن لغة الواجهة الأمامية إلى الواجهة الخلفية للتنفيذ. تدير الواجهة الخلفية مؤشرات الترابط الخاصة بها التي تجمع باستمرار المهام الموضوعة في قائمة الانتظار وتنفذها. لاحظ أنه لكي يعمل هذا، يجب أن تكون الواجهة الخلفية قادرة على تتبع التبعيات بين الخطوات المختلفة في الرسم البياني الحسابي. وبالتالي، لا يمكن إجراء موازاة العمليات التي تعتمد على بعضها البعض.



الشكل 13.2.1 الواجهات الأمامية للغة البرمجة والخلفيات الخلفية لإطار التعلم العميق.

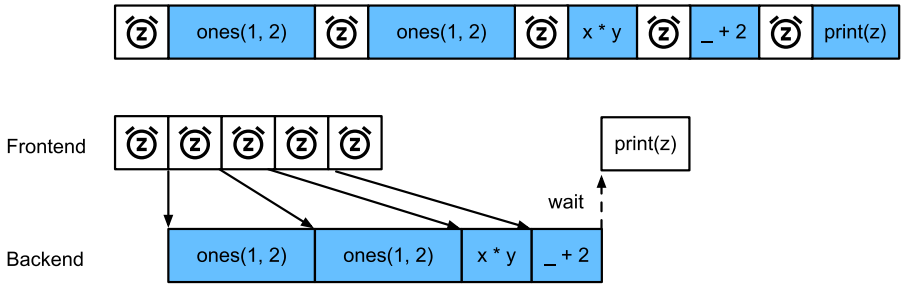
دعونا نلقي نظرة على مثال آخر للعبة لفهم الرسم البياني للتبعية بشكل أفضل قليلاً.

```
x = np.ones((1, 2))
y = np.ones((1, 2))
z = x * y + 2
z
array([[3., 3.]])
```



الشكل 13.2.2 تتعقب الواجهة الخلفية التبعيات بين الخطوات المختلفةفي الرسم البياني الحسابي.

مقتطف الشفرة أعلاه موضح أيضًافي الشكل 13.2.2. عندما ينفذ مؤشر ترابط الواجهة الأمامية لبايثون إحدى العبارات الثلاثة الأولى، فإنه يعيد المهمة ببساطة إلى قائمة انتظار الواجهة الخلفية. عندما تحتاج إلى طباعة نتائج العبارة الأخيرة، سينتظر مؤشر ترابط الواجهة الأمامية لبايثون حتى ينتهي مؤشر الترابط الخلفي C++ من حساب نتيجة المتغير z. تتمثل إحدى مزايا هذا التصميم في أن مؤشر ترابط الواجهة الأمامية لبايثون لا يحتاج إلى إجراء عمليات حسابية فعلية. وبالتالي، هناك تأثير ضئيل على الأداء العام للبرنامج، بغض النظر عن أداء بايثون. يوضح الشكل 13.2.3 كيفية تفاعل الواجهة الأمامية والخلفية.



الشكل 13.2.3 تفاعلات الواجهة الأمامية والخلفية.

13.2.4. الملخص

- قد تفصل أطر التعلم العميق واجهة بايثون الأمامية عن الواجهة الخلفية للتنفيذ. هذا يسمح بإدخال أوامر غير متزامن سريع في الواجهة الخلفية والتوازي المرتبط بها.
- يؤدي عدم التزامن إلى واجهة أمامية سريعة الاستجابة إلى حد ما. ومع ذلك ، احذر من ملء قائمة انتظار المهام بشكل زائد حيث قد يؤدي ذلك إلى استهلاك مفرط للذاكرة. يوصى بالمزامنة لكل دفعة صغيرة للحفاظ على تزامن الواجهة الأمامية والخلفية تقريبًا.
- يقدم بائعو الرقائق أدوات تحليل أداء متطورة للحصول على رؤية أكثر دقة حول كفاءة التعلم العميق.

13.2.5. التمارين

2. ذكرنا أعلاه أن استخدام الحساب غير المتزامن يمكن أن يقلل من إجمالي الوقت اللازم لإجراء 10000 عملية حسابية الى $t_1 + 10000t_2 + t_3$. لماذا علينا أن نفترض $10000t_2 > 9999t_1$ هنا؟
3. قم بقياس الفرق بين `waitall` و `wait_to_read`. تلميح: قم بتنفيذ عدد من التعليمات وقم بالمزامنة للحصول على نتيجة وسيطة.

13.3. التوازي التلقائي Automatic Parallelism

تُنشئ أطر التعلم العميق (مثل MXNet و PyTorch) الرسوم البيانية الحسابية تلقائيًافي الواجهة الخلفية. باستخدام الرسم البياني الحسابي، يكون النظام على دراية بجميع التبعيات، ويمكنه بشكل انتقائي تنفيذ مهام متعددة غير مترابطة بشكل متوازٍ لتحسين السرعة. على سبيل المثال، الشكل 13.2.2في القسم 13.2 يهيئ متغيرين بشكل مستقل. وبالتالي يمكن للنظام أن يختار تنفيذها بالتوازي.

عادةً ما يستخدم مشغل واحد جميع الموارد الحسابية على جميع وحدات المعالجة المركزية أو على وحدة معالجة رسومات واحدة. على سبيل المثال، سيستخدم مشغل `dot` جميع النوى (والخيوط) على جميع وحدات المعالجة المركزية ، حتى لو كانت هناك معالجات متعددة لوحدة المعالجة المركزية على جهاز واحد. الأمر نفسه ينطبق على وحدة معالجة رسومات واحدة. ومن ثم فإن الموازاة ليست مفيدة جدًا لأجهزة الكمبيوتر ذات الجهاز الواحد. مع وجود أجهزة متعددة، تكون الأشياء أكثر أهمية.في حين أن الموازاة عادة ما تكون ذات صلة بين العديد من وحدات معالجة الرسومات، فإن إضافة وحدة المعالجة المركزية المحلية سيزيد من الأداء قليلاً. على سبيل المثال، انظر Hadjis et al. (2016) الذي يركز على تدريب نماذج الرؤية الحاسوبية التي تجمع بين وحدة معالجة الرسومات ووحدة المعالجة المركزية. من خلال راحة إطار العمل الموازي تلقائيًا، يمكننا تحقيق نفس الهدفي بضعة أسطر من كود بايثون. على نطاق أوسع،

تركز مناقشتنا للحساب المتوازي التلقائي على الحساب الموازي باستخدام كل من وحدات المعالجة المركزية ووحدات معالجة الرسومات، بالإضافة إلى موازاة الحساب والتواصل.

لاحظ أننا بحاجة إلى وحدتي GPU على الأقل لإجراء التجارب في هذا القسم.

```
from mxnet import np, npx
from d2l import mxnet as d2l

npx.set_np()
```

13.3.1. الحساب المتوازي على وحدات معالجة الرسومات Parallel Computation on GPUs

لنبدأ بتعريف عبء العمل المرجعي للاختبار: تؤدي دالة `run` أدناه 10 ضرب مصفوفة مصفوفة على الجهاز الذي نختاره باستخدام البيانات المخصصة في متغيرين: `x_gpu1` و `x_gpu2`.

```
devices = d2l.try_all_gpus()
def run(x):
    return [x.dot(x) for _ in range(50)]

x_gpu1 = np.random.uniform(size=(4000, 4000),
ctx=devices[0])
x_gpu2 = np.random.uniform(size=(4000, 4000),
ctx=devices[1])
```

الآن نطبق الدالة على البيانات. للتأكد من أن التخزين المؤقت لا يلعب دورًا في النتائج، نقوم باحماء الأجهزة عن طريق إجراء تمريرة واحدة على أي منهما قبل القياس.

```
run(x_gpu1)  # Warm-up both devices
run(x_gpu2)
npx.waitall()

with d2l.Benchmark('GPU1 time'):
    run(x_gpu1)
    npx.waitall()

with d2l.Benchmark('GPU2 time'):
    run(x_gpu2)
    npx.waitall()
```

```
GPU1 time: 0.5095 sec
GPU2 time: 0.5061 sec
```

إذا أزلنا جملة `waitall` بين كلتا المهمتين ، فسيكون النظام حرًا في موازنة الحساب على كلا الجهازين تلقائيًا.

```
with d2l.Benchmark('GPU1 & GPU2'):
    run(x_gpu1)
    run(x_gpu2)
    npx.waitall()
```

```
GPU1 & GPU2: 0.5131 sec
```

في الحالة المذكورة أعلاه، يكون إجمالي وقت التنفيذ أقل من مجموع أجزائه، نظرًا لأن إطار عمل التعلم العميق يقوم تلقائيًا بجدولة الحساب على كلا جهازي GPU دون الحاجة إلى كود متطور نيابة عن المستخدم.

13.3.2. الحساب والاتصال المتوازي Parallel Computation and Communication

في كثير من الحالات، نحتاج إلى نقل البيانات بين الأجهزة المختلفة، على سبيل المثال بين وحدة المعالجة المركزية ووحدة معالجة الرسومات، أو بين وحدات معالجة الرسومات المختلفة. على سبيل المثال، يحدث هذا عندما نريد إجراء تحسين موزع حيث نحتاج إلى تجميع الانحدارات gradients على بطاقات تسريع متعددة. دعنا نحاكي هذا عن طريق الحوسبة على وحدة معالجة الرسومات ثم نسخ النتائج مرة أخرى إلى وحدة المعالجة المركزية.

```
def copy_to_cpu(x):
    return [y.copyto(npx.cpu()) for y in x]

with d2l.Benchmark('Run on GPU1'):
    y = run(x_gpu1)
    npx.waitall()

with d2l.Benchmark('Copy to CPU'):
    y_cpu = copy_to_cpu(y)
    npx.waitall()
```

```
Run on GPU1: 0.5355 sec
Copy to CPU: 2.4212 sec
```

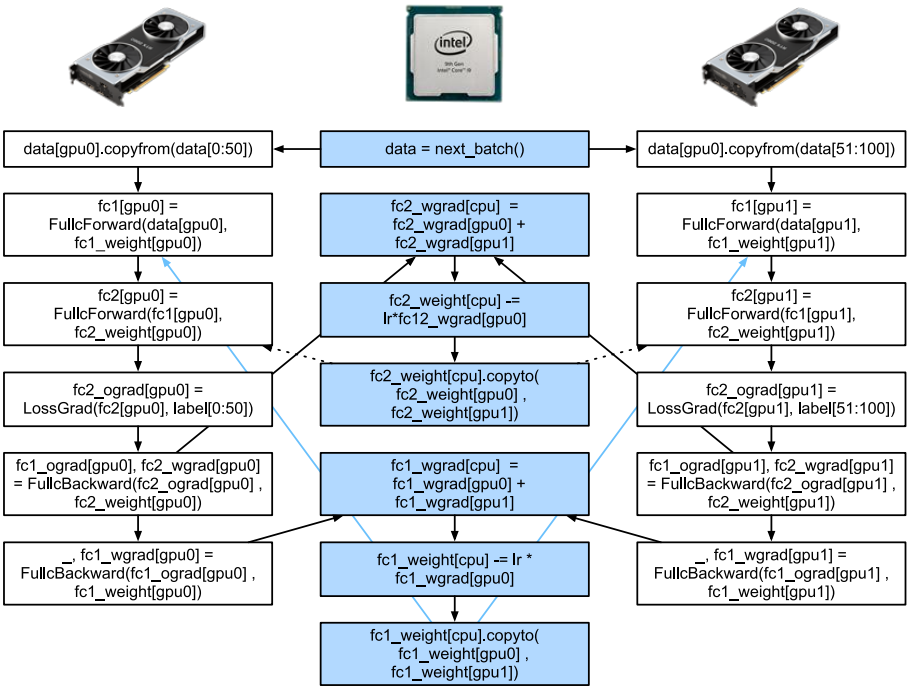
هذا غير فعال إلى حد ما. لاحظ أنه يمكننا بالفعل بدء نسخ أجزاء من `y` إلى وحدة المعالجة المركزية بينما لا يزال يتم حساب باقي القائمة. يحدث هذا الموقف، على سبيل المثال، عندما نحسب الانحدار على الدفعات الصغيرة minibatch. ستكون انحدارات بعض المعلمات متاحة في وقت أبكر من غيرها. ومن ثم فإنه يعمل لصالحنا لبدء استخدام النطاق الترددي لناقل PCI-Express أثناء استمرار تشغيل وحدة معالجة الرسومات. تسمح لنا إزالة الانتظار بين كلا الجزأين بمحاكاة هذا السيناريو.

```
with d2l.Benchmark('Run on GPU1 and copy to CPU'):
```

```
    y = run(x_gpu1)
    y_cpu = copy_to_cpu(y)
    npx.waitall()
Run on GPU1 and copy to CPU: 2.4604 sec
```

إجمالي الوقت المطلوب لكلتا العمليتين (كما هو متوقع) أقل من مجموع أجزائها. لاحظ أن هذه المهمة تختلف عن الحساب المتوازي لأنها تستخدم موردًا مختلفًا: الناقل bus بين وحدة المعالجة المركزية ووحدات معالجة الرسومات.في الواقع، يمكننا إجراء الحساب على كلا الجهازين والتواصل، كل ذلكفي نفس الوقت. كما هو مذكور أعلاه، هناك تبعية بين الحساب والاتصال: يجب حساب `y[i]` قبل نسخها إلى وحدة المعالجة المركزية. لحسن الحظ، يمكن للنظام نسخ `y[i-1]` أثناء حساب `y[i]` لتقليل إجمالي وقت التشغيل.

نختتم مع رسم توضيحي للرسم البياني الحسابي وتبعياته لـ MLP بسيط من طبقتين عند التدريب على وحدة المعالجة المركزية واثنين من وحدات معالجة الرسومات، كما هو موضحفي الشكل 13.3.1. سيكون من المؤلم جدولة البرنامج المتوازي الناتج عن ذلك يدويًا. هذا هو المكان الذي يكون فيه من المفيد أن يكون لديك خلفية حوسبة قائمة على الرسم البياني للتحسين.



الشكل 13.3.1 الرسم البياني الحسابي وتبعياتهفي MLP من طبقتين على وحدة المعالجة المركزية واثنين من وحدات معالجة الرسومات.

13.3.3. الملخص

- تحتوي الأنظمة الحديثة على مجموعة متنوعة من الأجهزة ، مثل العديد من وحدات معالجة الرسومات ووحدات المعالجة المركزية. يمكن استخدامها بشكل متوازٍ وغير متزامن.
- تحتوي الأنظمة الحديثة أيضًا على مجموعة متنوعة من الموارد للاتصال ، مثل PCI Express والتخزين (عادةً محركات الأقراص ذات الحالة الصلبة أو عبر الشبكات) وعرض النطاق الترددي للشبكة. يمكن استخدامها بالتوازي لتحقيق أقصى قدر من الكفاءة.
- يمكن للواجهة الخلفية تحسين الأداء من خلال الحساب والاتصال الموازي تلقائيًا.

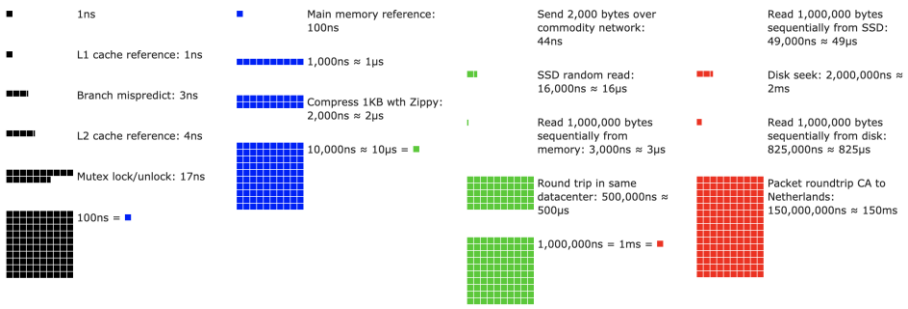
13.3.4. التمارين

1. تم إجراء ثماني عمليات في دالة `run` المحددة في هذا القسم. لا توجد تبعيات بينهما. صمم تجربة لمعرفة ما إذا كان إطار التعلم العميق سينفذها تلقائيًا بالتوازي.
2. عندما يكون عبء عمل المشغل الفردي صغيرًا بدرجة كافية ، يمكن أن تساعد الموازاة حتى في وحدة المعالجة المركزية أو وحدة معالجة الرسومات. صمم تجربة للتحقق من ذلك.
3. صمم تجربة تستخدم الحساب المتوازي على وحدات المعالجة المركزية ووحدات معالجة الرسومات والتواصل بين كلا الجهازين.
4. استخدم مصحح أخطاء مثل Nsight من NVIDIA للتحقق من كفاءة الكود الخاص بك.
5. تصميم مهام حسابية تتضمن تبعيات بيانات أكثر تعقيدًا ، وإجراء تجارب لمعرفة ما إذا كان يمكنك الحصول على النتائج الصحيحة أثناء تحسين الأداء.

13.4. المكونات المادية Hardware

يتطلب بناء الأنظمة ذات الأداء الرائع فهمًا جيدًا للخوارزميات والنماذج لالتقاط الجوانب الإحصائية للمشكلة.في الوقت نفسه، من الضروري أيضًا أن يكون لديك على الأقل قدر ضئيل من المعرفة بالمكونات المادية الأساسية. لا يعد القسم الحالي بديلاً عن الدورة التدريبية المناسبة حول تصميم الأجهزة والنظام. بدلاً من ذلك، قد يكون بمثابة نقطة انطلاق لفهم سبب كون بعض الخوارزميات أكثر كفاءة من غيرها وكيفية تحقيق معدل نقل جيد. يمكن للتصميم الجيد أن يحدث فرقًا بسهولةفي ترتيب الحجم، وهذا بدوره يمكن أن يحدث فرقًا بين القدرة على تدريب شبكة (على سبيل المثال،في غضون أسبوع) وليس على الإطلاق (في 3 أشهر، وبالتالي تفويت الموعد النهائي). سنبدأ بالنظر إلى أجهزة الكمبيوتر. ثم سنقوم بالتكبير للنظر بعناية أكبرفي وحدات

المعالجة المركزية ووحدات معالجة الرسومات. أخيرًا، نقوم بالتصغير لمراجعة كيفية اتصال أجهزة كمبيوتر متعددةفي مركز الخادم أوفي السحابة.



شكل 13.4.1 أرقام التأخير Latency Numbers التي يجب أن يعرفها كل مبرمج.

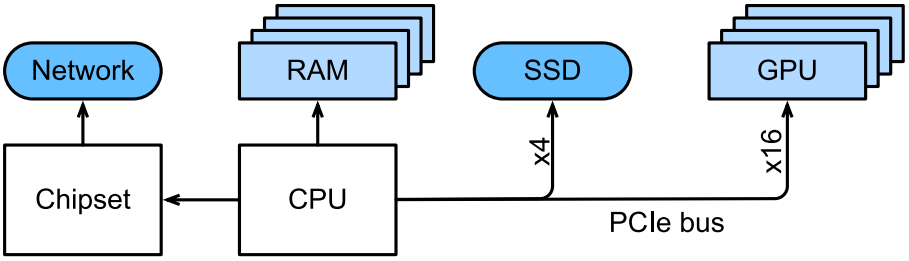
قد يتمكن القراء الذين نفد صبرهم من التغلب على الشكل 13.4.1. إنه مأخوذ من المشاركة التفاعلية interactive post لكولين سكوت والتي تقدم نظرة عامة جيدة على التقدم المحرز خلال العقد الماضي. ترجع الأرقام الأصلية إلى حديث جيف دينفي ستانفورد من عام 2010. توضح المناقشة أدناه بعض الأسباب المنطقية لهذه الأرقام وكيف يمكن أن ترشدنافي تصميم الخوارزميات. المناقشة أدناه عالية المستوى وسريعة. من الواضح أنه ليس بديلاً عن الدورة التدريبية المناسبة، ولكنه يهدف فقط إلى توفير معلومات كافية لمصمم إحصائي لاتخاذ قرارات التصميم المناسبة. للحصول على نظرة عامة متعمقة على هندسة الكمبيوتر، نحيل القارئ إلى (Hennessy and Patterson، 2011) أو دورة تدريبية حديثة حول هذا الموضوع، مثل تلك التي كتبها Arste Asanovic.

13.4.1. أجهزة الكمبيوتر Computers

يتمتع معظم الباحثين والممارسينفي مجال التعلم العميق بإمكانية الوصول إلى جهاز كمبيوتر به قدر لا بأس به من الذاكرة أو الحساب أو بعض أشكال المسرعات مثل وحدة معالجة الرسومات (GPU) أو مضاعفاتها. يتكون الكمبيوتر من المكونات الرئيسية التالية:

- معالج processor (يُشار إليه أيضًا باسم وحدة المعالجة المركزية CPU) قادر على تنفيذ البرامج التي نقدمها له (بالإضافة إلى تشغيل نظام التشغيل والعديد من الأشياء الأخرى) ، ويتكون عادةً من 8 نوى أو أكثر.
- ذاكرة Memory (RAM) لتخزين واسترداد النتائج من الحساب ، مثل متجهات الوزن والتنشيطات وبيانات التدريب.

- اتصال شبكة إيثرنت Ethernet network connection (متعدد أحيانًا) بسرعات تتراوح من 1 جيجابايت / ثانية إلى 100 جيجابايت / ثانية. يمكن العثور على المزيد من الوصلات البينية المتقدمة على الخوادم المتطورة.
- ناقل توسعة عالي السرعة high speed expansion bus (PCIe) لتوصيل النظام بواحدة أو أكثر من وحدات معالجة الرسومات. تحتوي الخوادم على ما يصل إلى 8 مسرعات ، وغالبًا ما تكون متصلة بطوبولوجيا متقدمة ، بينما تحتوي أنظمة سطح المكتب على 1 أو 2 ، اعتمادًا على ميزانية المستخدم وحجم مصدر الطاقة.
- تخزين متين Durable storage، مثل محرك الأقراص الثابتة الممغنط HDD، ومحرك الأقراص ذي الحالة الصلبة SSD، في كثير من الحالات متصل باستخدام ناقل PCIe. يوفر النقل الفعال لبيانات التدريب إلى النظام وتخزين نقاط التفتيش الوسيطة حسب الحاجة.



الشكل 13.4.2 توصيل مكونات الكمبيوتر.

كما يشير الشكل 13.4.2، فإن معظم المكونات (الشبكة ووحدة معالجة الرسومات والتخزين) متصلة بوحدة المعالجة المركزية عبر ناقل PCIe. يتكون من عدة ممرات متصلة مباشرة بوحدة المعالجة المركزية. على سبيل المثال، يحتوي Threadripper 3 من AMD على 64 فتحة PCIe 4.0، كل منها قادر على نقل البيانات بسرعة 16 جيجابت / ثانيةفي كلا الاتجاهين. يتم توصيل الذاكرة مباشرة بوحدة المعالجة المركزية بنطاق ترددي (bandwidth) إجمالي يصل إلى 100 جيجابايت / ثانية.

عندما نقوم بتشغيل الكود على جهاز كمبيوتر، نحتاج إلى تبديل البيانات إلى المعالجات (وحدات المعالجة المركزية أو وحدات معالجة الرسومات)، وإجراء العمليات الحسابية، ثم نقل النتائج من المعالج إلى ذاكرة الوصول العشوائي والتخزين الدائم. ومن ثم، من أجل الحصول على أداء جيد، نحتاج إلى التأكد من أن هذا يعمل بسلاسة دون أن يصبح أي من الأنظمة عقبة كبيرة. على سبيل المثال، إذا لم نتمكن من تحميل الصور بسرعة كافية، فلن يكون لدى المعالج أي عمل للقيام به. وبالمثل، إذا لم نتمكن من نقل المصفوفات بسرعة كافية إلى وحدة المعالجة المركزية (أو وحدة معالجة الرسومات)، فإن عناصر المعالجة الخاصة بها سوف تتضور جوعًا.

أخيرًا، إذا أردنا مزامنة أجهزة كمبيوتر متعددة عبر الشبكة، فلا ينبغي أن يؤدي هذا الأخير إلى إبطاء الحساب. أحد الخيارات هو تداخل الاتصال والحساب. دعونا نلقي نظرة على المكونات المختلفة بمزيد من التفصيل.

13.4.2. الذاكرة Memory

تُستخدم الذاكرة الأساسية لتخزين البيانات التي يجب الوصول إليها بسهولة.في الوقت الحالي، عادةً ما تكون ذاكرة الوصول العشوائي لوحدة المعالجة المركزية من مجموعة DDR4، وتقدم نطاقًا ترددياً يتراوح من 20 إلى 25 جيجابايت / ثانية لكل وحدة. تحتوي كل وحدة على ناقل بعرض 64 بت. عادةً ما يتم استخدام أزواج من وحدات الذاكرة للسماح بقنوات متعددة. تحتوي وحدات المعالجة المركزية (CPU) على ما بين 2 و4 قنوات ذاكرة، أي أن لديها عرض نطاق ترددي للذاكرة يتراوح بين 4 و0 جيجابايت / ثانية و100 جيجابايت / ثانية. غالبًا ما يوجد بنكان banks لكل قناة. على سبيل المثال، يحتوي Zen 3 Threadripper من AMD على 8 منافذ slots.

في حين أن هذه الأرقام مثيرة للإعجاب، إلا أنهافي الواقع تحكي جزءًا فقط من القصة. عندما نريد قراءة جزء من الذاكرة، نحتاج أولاً إلى إخبار وحدة الذاكرة بمكان العثور على المعلومات. وهذا يعني أننا نحتاج أولاً إلى إرسال العنوان إلى RAM. بمجرد الانتهاء من ذلك، يمكننا اختيار قراءة سجل (record) 64 بت واحد فقط أو سلسلة طويلة من السجلات. هذا الأخير يسمى قراءة الانفجار burst read. باختصار، يستغرق إرسال عنوان إلى الذاكرة وإعداد النقل حوالي 100 نانوثانية (تعتمد التفاصيل على معاملات التوقيت المحددة لرقائق الذاكرة المستخدمة)، ويستغرق كل نقل لاحق 0.2 نانوثانية فقط. باختصار، القراءة الأولى أغلى 500 مرة من القراءة اللاحقة! لاحظ أنه يمكننا إجراء ما يصل إلى 10000000 قراءة عشوائيةفي الثانية. يشير هذا إلى أننا نتجنب الوصول العشوائي للذاكرة قدر الإمكان ونستخدم عمليات القراءة المنفجرة (والكتابة) بدلاً من ذلك.

تصبح الأمور أكثر تعقيداً بعض الشيء عندما نأخذفي الاعتبار أن لدينا بنوكًا متعددة. يمكن لكل بنك قراءة الذاكرة إلى حد كبير بشكل مستقل. هذا يعني شيئين. من ناحية أخرى، يكون العدد الفعال للقراءات العشوائية أعلى بأربع مرات، بشرط أن يتم توزيعها بالتساوي عبر الذاكرة. هذا يعني أيضًا أنه لا يزال إجراء قراءات عشوائية فكرة سيئة نظرًا لأن القراءات المتلاحقة أسرع 4 مرات أيضًا. من ناحية أخرى، نظرًا لمحاذاة الذاكرة مع حدود 64 بت، فمن الجيد محاذاة أي هياكل بيانات مع نفس الحدود. يقوم المترجمون بذلك تلقائيًا إلى حد كبير عند تعيين العلامات المناسبة. يتم تشجيع القراء الفضوليين على مراجعة محاضرة حول DRAM مثل تلك التي كتبها Zeshan Chishti.

تخضع ذاكرة وحدة معالجة الرسومات GPU memory لمتطلبات نطاق ترددي (bandwidth) أعلى نظرًا لاحتوائها على العديد من عناصر المعالجة أكثر من وحدات المعالجة المركزية. بشكل عام، هناك خياران لمعالجتها. الأول هو جعل ناقل الذاكرة memory bus أوسع بشكل ملحوظ. على سبيل المثال، يحتوي RTX 2080 Ti من NVIDIA على ناقل بعرض 352 بت. هذا يسمح بنقل المزيد من المعلوماتفي نفس الوقت. ثانيًا، تستخدم وحدات معالجة الرسومات ذاكرة محددة عالية الأداء. عادةً ما تستخدم الأجهزة من فئة المستهلكين، مثل سلسلة NVIDIA's RTX وTitan شرائح GDDR6 مع عرض نطاق ترددي إجمالي يزيد عن 500 جيجابايت / ثانية. البديل هو استخدام وحدات HBM (ذاكرة النطاق الترددي العالي high bandwidth memory). يستخدمون واجهة مختلفة تمامًا ويتصلون مباشرةً بوحدات معالجة الرسومات على رقاقة سيليكون مخصصة. هذا يجعلها باهظة الثمن ويقتصر استخدامها عادةً على شرائح الخوادم المتطورة، مثل سلسلة مسرعات NVIDIA Volta V100. ليس من المستغرب أن تكون ذاكرة وحدة معالجة الرسومات بشكل عام أصغر بكثير من ذاكرة وحدة المعالجة المركزية نظرًا لارتفاع تكلفة الأولى. لأغراضنا، تتشابه خصائص أدائها إلى حد كبير، ولكن بشكل أسرع كثيرًا. يمكننا تجاهل التفاصيل بأمان لغرض هذا الكتاب. إنها مهمة فقط عند ضبط نواة GPU لإنتاجية عالية.

13.4.3. التخزين Storage

لقد رأينا أن بعض الخصائص الرئيسية لذاكرة الوصول العشوائي هي النطاق الترددي bandwidth ووقت الاستجابة latency. وينطبق الشيء نفسه على أجهزة التخزين storage devices، لكن الاختلافات يمكن أن تكون أكثر تطرفًا.

13.4.3.1. محركات الأقراص الصلبة Hard Disk Drives

تم استخدام محركات الأقراص الثابتة (HDDs) لأكثر من نصف قرن. باختصار، تحتوي على عدد من الأطباق الدوارة ذات الرؤوس التي يمكن وضعها للقراءة أو الكتابةفي أي مسار معين. تستوعب الأقراص المتطورة ما يصل إلى 16 تيرابايت على 9 أطباق. تتمثل إحدى الفوائد الرئيسية لمحركات الأقراص الصلبةفي أنها غير مكلفة نسبيًا. واحدة من عيوبها العديدة هي أوضاع فشلها الكارثية ووقت استجابة القراءة المرتفع نسبيًا.

لفهم هذا الأخير، ضعفي اعتبارك حقيقة أن محركات الأقراص الثابتة تدور بسرعة حوالي 7200 دورةفي الدقيقة (دوراتفي الدقيقة). إذاكانت أسرع بكثير فإنها ستتحطم بسبب قوة الطرد المركزي التي تمارس على الأطباق. هذا له جانب سلبي كبير عندما يتعلق الأمر بالوصول إلى قطاع معين على القرص: نحتاج إلى الانتظار حتى يدور الطبقفي موضعه (يمكننا تحريك الرؤوس ولكن لا يمكننا تسريع الأقراص الفعلية). ومن ثم يمكن أن يستغرق الأمر أكثر من 8 مللي ثانية حتى تتوفر

البيانات المطلوبة. الطريقة الشائعة للتعبير عن ذلك هي القول بأن محركات الأقراص الصلبة يمكن أن تعمل في حوالي 100 IOPs (عمليات الإدخال / الإخراج في الثانية). ظل هذا الرقم بشكل أساسي دون تغيير خلال العقدين الماضيين. والأسوأ من ذلك، أنه من الصعب بنفس القدر زيادة عرض النطاق الترددي (في حدود 100 ـ 200 ميغا بايت / ثانية). بعد كل شيء، يقرأ كل رأس مسارًا من البتات، ومن ثم فإن معدل البت يتساوى فقط مع الجذر التربيعي لكثافة المعلومات. نتيجة لذلك، سرعان ما أصبحت محركات الأقراص الصلبة محجوزة للتخزين الأرشيفي والتخزين منخفض الدرجة لمجموعات البيانات الكبيرة جدًا.

13.4.3.2. محركات الأقراص ذات الحالة الثابتة Solid State Drives

تستخدم محركات الأقراص ذات الحالة الثابتة (SSD) ذاكرة فلاش لتخزين المعلومات باستمرار. هذا يسمح لوصول أسرع بكثير إلى السجلات المخزنة. يمكن أن تعمل محركات الأقراص ذات الحالة الثابتة الحديثة من 100،000 إلى 500،000 IOPs، أي ما يصل إلى 3 ترتيب من حيث الحجم أسرع من محركات الأقراص الثابتة. علاوة على ذلك، يمكن أن يصل عرض النطاق الترددي الخاص بهم إلى 1 ـ 3 جيجابايت / ثانية، أي ترتيب واحد من حيث الحجم أسرع من محركات الأقراص الثابتة. تبدو هذه التحسينات جيدة جدًا لدرجة يصعب تصديقها.في الواقع، تأتي مع المحاذير التالية، نظرًا للطريقة التي تم بها تصميم محركات أقراص الحالة الثابتة.

- تخزن محركات أقراص الحالة الثابتة المعلومات في مجموعات (256 كيلوبايت أو أكبر). لا يمكن كتابتها إلا ككل ، الأمر الذي يستغرق وقتًا طويلاً. وبالتالي ، فإن عمليات الكتابة العشوائية على SSD لها أداء ضعيف للغاية. وبالمثل ، فإن كتابة البيانات بشكل عام تستغرق وقتًا طويلاً حيث يجب قراءة الكتلة ومسحها ثم إعادة كتابتها بمعلومات جديدة. حتى الآن طورت وحدات تحكم SSD والبرامج الثابتة خوارزميات للتخفيف من ذلك. ومع ذلك ، يمكن أن تكون عمليات الكتابة أبطأ بكثير، لا سيما بالنسبة لمحركات أقراص الحالة الصلبة QLC (ذات الخلايا الرباعية المستوى quad level cell). المفتاح لتحسين الأداء هو الاحتفاظ بقائمة انتظار من العمليات ، وتفضيل القراءة والكتابة في كتل كبيرة إن أمكن.
- تُبلى خلايا الذاكرة في محركات الأقراص ذات الحالة الثابتة بسرعة نسبيًا (غالبًا بعد بضعة آلاف من عمليات الكتابة). خوارزميات الحماية على مستوى التآكل قادرة على نشر التدهور على العديد من الخلايا. ومع ذلك ، لا يوصى باستخدام محركات أقراص الحالة الثابتة (SSD) لمبادلة الملفات أو لتجمعات كبيرة من ملفات السجل.
- أخيرًا ، أدت الزيادة الهائلة في النطاق الترددي إلى إجبار مصممي الكمبيوتر على إرفاق محركات أقراص الحالة الثابتة (SSD) مباشرة بحافلة PCIe. يمكن لمحركات

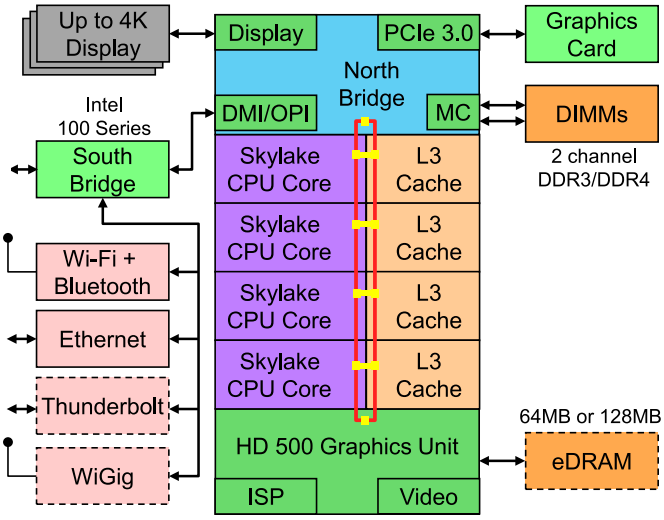
الأقراص القادرة على التعامل مع هذا ، والمشار إليها باسم NVMe (ذاكرة غير متطايرة محسنة Non Volatile Memory enhanced) ، استخدام ما يصل إلى 4 ممرات PCIe. يصل هذا إلى 8 جيجابايت / ثانية على PCIe 4.0.

13.4.3.3. التخزين السحابي Cloud Storage

يوفر التخزين السحابي Cloud storage مجموعة من الأداء القابل للتكوين configurable. أي أن تخصيص التخزين للأجهزة الافتراضية ديناميكي، سواء من حيث الكمية أو من حيث السرعة، على النحو الذي يختاره المستخدمون. نوصي بأن يزيد المستخدمون من العدد المخصص لعمليات الإدخال والإخراج كلما كان وقت الاستجابة مرتفعًا جدًا، على سبيل المثال، أثناء التدريب مع العديد من السجلات الصغيرة.

13.4.4. وحدات المعالجة المركزية CPUs

وحدات المعالجة المركزية (CPUs) هي حجر الزاويةفي أي جهاز كمبيوتر. وهي تتكون من عدد من المكونات الرئيسية: نوى المعالج processor cores القادرة على تنفيذ كود الجهاز، ناقل bus يربط بينها (يختلف الهيكل المحدد اختلافًا كبيرًا بين طرازات المعالج والأجيال والموردين)، وذاكرة التخزين المخبئية caches للسماح بنطاق ترددي أعلى وذاكرة زمن وصول أقل الوصول إلى ما هو ممكن بقراءات من الذاكرة الرئيسية. أخيرًا، تحتوي جميع وحدات المعالجة المركزية الحديثة تقريبًا على وحدات معالجة متجهية vector processing units للمساعدةفي الجبر الخطي والتلافيفات عالية الأداء، حيث إنها شائعةفي معالجة الوسائط والتعلم الآلي.

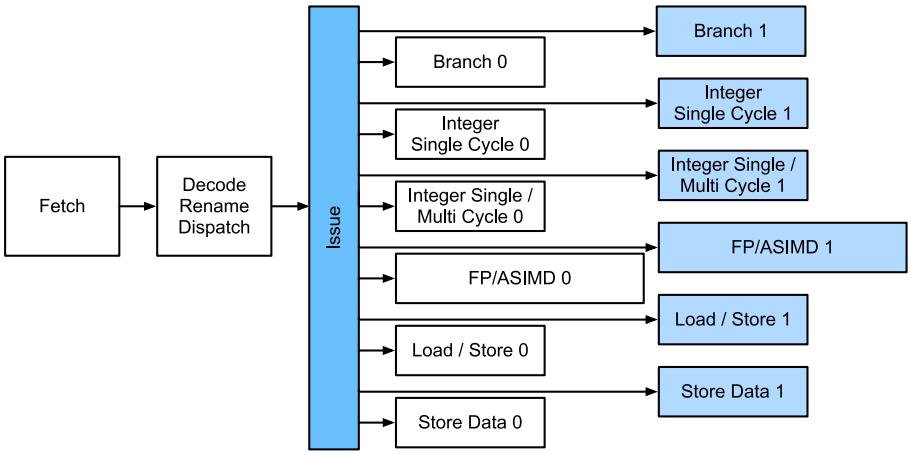


الشكل. 13.4.3 وحدة المعالجة المركزية Intel Skylake للمستهلك رباعية النوى.

الشكل 13.4.3 يصور وحدة المعالجة المركزية Intel Skylake رباعية النوى للمستهلكين. يحتوي على وحدة معالجة الرسومات المدمجة، وذاكرة التخزين المؤقت، وRingbus الذي يربط بين النوى الأربعة. تعد الأجهزة الطرفية Peripherals، مثل Ethernet و WiFi و Bluetooth ووحدة تحكم SSD و USB، إما جزءًا من مجموعة الشرائح أو متصلة مباشرة (PCIe) بوحدة المعالجة المركزية.

13.4.4.1. المعمارية الدقيقة Microarchitecture

يتكون كل من نوى المعالج من مجموعة معقدة من المكونات. بينما تختلف التفاصيل بين الأجيال والموردين، فإن الوظيفة الأساسية قياسية إلى حد كبير. تقوم الواجهة الأمامية بتحميل التعليمات وتحاول التنبؤ بالمسار الذي سيتم اتخاذه (على سبيل المثال، للتحكمفي التدفق control flow). ثم يتم فك التعليمات من كود التجميع assembly code إلى التعليمات الدقيقة. غالبًا ما لا يكون رمز التجميع هو أقل مستوى رمز ينفذه المعالج. بدلاً من ذلك، يمكن فك تشفير التعليمات المعقدة إلى مجموعة من العمليات ذات المستوى الأدنى. ثم يتم معالجتها من قبل جوهر التنفيذ الفعلي. غالبًا ما يكون الأخير قادرًا على إجراء العديد من العملياتفي وقت واحد. على سبيل المثال، يمكن لنواة ARM Cortex A77في الشكل 13.4.4 إجراء ما يصل إلى 8 عملياتفي وقت واحد.



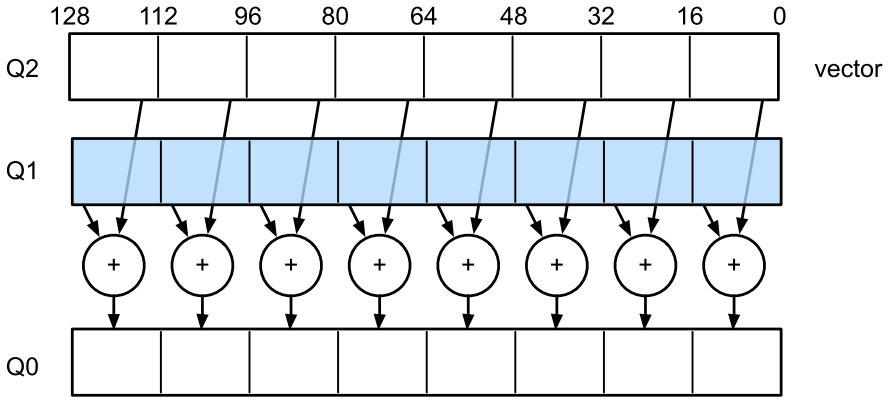
الشكل. 13.4.4 الهندسة المعمارية الدقيقة ARM Cortex A77.

هذا يعني أن البرامج الفعالة قد تكون قادرة على أداء أكثر من تعليمة واحدة لكل دورة ساعة، بشرط أن يتم تنفيذها بشكل مستقل. لم يتم إنشاء جميع الوحدات على قدم المساواة. يتخصص البعضفي تعليمات الأعداد الصحيحة integer بينما يتم تحسين البعض الآخر لأداء النقطة العائمة floating point. لزيادة الإنتاجية، قد يتبع المعالج أيضًا مسارات رمز متعددةفي وقت

واحدفي تعليمات متفرعة ثم يتجاهل نتائج الفروع التي لم يتم أخذها. هذا هو سبب أهمية وحدات التنبؤ بالفروع (في الواجهة الأمامية) بحيث يتم متابعة المسارات الواعدة فقط.

13.4.4.2. الفيكتورازيشن Vectorization

التعلم العميق متعطش للغاية للحوسبة compute-hungry. ومن ثم، لجعل وحدات المعالجة المركزية مناسبة للتعلم الآلي، يحتاج المرء إلى إجراء العديد من العملياتفي دورة ساعة واحدة. يتم تحقيق ذلك عن طريق وحدات المتجه vector units. لديهم أسماء مختلفة:في ARM يطلق عليهم NEON،في x86 يشار إليهم (جيل حديث) بوحدات AVX2. الجانب الشائع هو أنهم قادرون على إجراء عمليات SIMD (تعليمات واحدة متعددة البيانات). يوضح الشكل 13.4.5 كيف يمكن إضافة 8 أعداد صحيحة قصيرةفي دورة ساعة واحدة على ARM.



الشكل. 13.4.5 فيكتورازيشن نيون 128 بت.

اعتمادًا على خيارات البُنية، يصل طول هذه السجلات إلى 512 بت، مما يسمح بدمج ما يصل إلى 64 زوجًا من الأرقام. على سبيل المثال، قد نضرب رقمين ونضيفهما إلى رقم ثالث، وهو ما يُعرف أيضًا باسم الجمع المضاعف المدمج fused multiply-add. تستخدم OpenVino من Intel هذه لتحقيق إنتاجية محترمة للتعلم العميق على وحدات المعالجة المركزية على مستوى الخادم. لاحظ، مع ذلك، أن هذا الرقم يتضاءل تمامًا مع ما تستطيع وحدات معالجة الرسومات تحقيقه. على سبيل المثال، يحتوي RTX 2080 Ti من NVIDIA على 4352 نواة CUDA، كل منها قادر على معالجة مثل هذه العمليةفي أي وقت.

13.4.4.3. الذاكرة المخبئية Cache

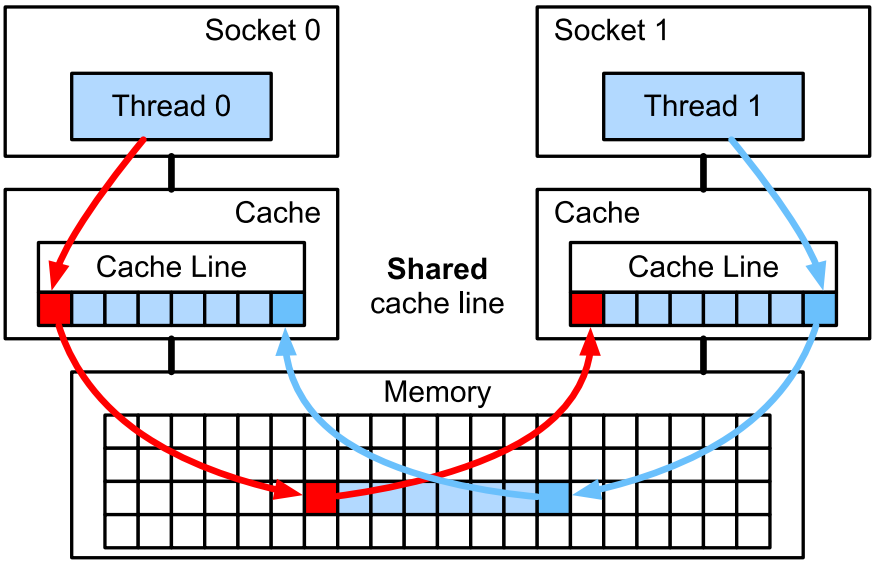
ضعفي اعتبارك الموقف التالي: لدينا نواة متواضعة لوحدة المعالجة المركزية ذات 4 نوى كما هو موضحفي الشكل 13.4.3 أعلاه، تعمل بتردد 2 جيجاهرتز. علاوة على ذلك، لنفترض أن لدينا

عدد IPC (ايعازات لكل ساعة instructions per clock) يبلغ 1 وأن الوحدات بها AVX2 مع تمكين عرض 256 بت. دعنا نفترض علاوة على ذلك أنه يجب استرداد واحد على الأقل من السجلات المستخدمة لعمليات AVX2 من الذاكرة. هذا يعني أن وحدة المعالجة المركزية تستهلك $4 \times 256 \text{ bit} = 128 \text{ bytes}$ من البيانات لكل دورة على مدار الساعة. ما لم نتمكن من نقل $2 \times 10^9 \times 128 = 256 \times 10^9$ بايت إلى المعالجفي الثانية، فإن عناصر المعالجة سوف تتضور جوعًا. لسوء الحظ، لا تدعم واجهة الذاكرة الخاصة بهذه الشريحة سوى نقل البيانات من 20 إلى 40 جيجابايت / ثانية، أي أقل بمقدار واحد. يتمثل الإصلاحفي تجنب تحميل بيانات جديدة من الذاكرة قدر الإمكان وبدلاً من ذلك تخزينها مؤقتًا محليًا على وحدة المعالجة المركزية. هذا هو المكان الذي تصبح فيه الذاكرة المخبئيةفي متناول اليد. يتم استخدام الأسماء أو المفاهيم التالية بشكل شائع:

- السجلات Registers بالمعنى الدقيق للكلمة ليست جزءًا من ذاكرة التخزين المخبئية. يساعدون في تنظيم التعليمات (الايعازات). ومع ذلك ، فإن سجلات وحدة المعالجة المركزية هي مواقع ذاكرة يمكن لوحدة المعالجة المركزية الوصول إليها بسرعة الساعة دون أي عقوبة تأخير. تحتوي وحدات المعالجة المركزية (CPU) على عشرات من السجلات. الأمر متروك للمترجم (أو المبرمج) لاستخدام السجلات بكفاءة. على سبيل المثال ، تحتوي لغة البرمجة C على كلمة أساسية `register`.
- L1 caches هي خط الدفاع الأول ضد متطلبات عرض النطاق الترددي للذاكرة العالية. L1 caches صغيرة (قد تكون الأحجام النموذجية 32ـ64 كيلوبايت) وغالبًا ما تنقسم إلى ذواكر مخبئية للبيانات والتعليمات. عند العثور على البيانات في ذاكرة L1 caches، يكون الوصول سريعًا جدًا. إذا تعذر العثور عليها هناك ، يتقدم البحث أسفل التسلسل الهرمي لذاكرة التخزين المخبئية.
- L2 caches هي المحطة التالية. اعتمادًا على التصميم المعماري وحجم المعالج ، قد تكون حصرية. قد لا يمكن الوصول إليها إلا من خلال نواة معينة أو مشتركة بين نوى متعددة. تكون L2 caches أكبر (عادةً 256ـ512 كيلوبايت لكل نواة) وأبطأ من L1. علاوة على ذلك ، للوصول إلى شيء ما في L2 ، نحتاج أولاً إلى التحقق من أن البيانات ليست في L1 ، مما يضيف قدرًا صغيرًا من زمن الانتقال الإضافي.
- يتم مشاركة L3 caches بين نوى متعددة ويمكن أن تكون كبيرة جدًا. تحتوي وحدات المعالجة المركزية لخادم Epyc 3 من AMD على 256 ميجا بايت من ذاكرة التخزين المخبئي موزعة على عدة شرائح. تقع الأرقام الأكثر شيوعًا في النطاق من 4 إلى 8 ميغا بايت.

يعد توقع عناصر الذاكرة المطلوبة بعد ذلك أحد معايير التحسين الرئيسية في تصميم الرقاقة. على سبيل المثال ، يُنصح باجتياز الذاكرة في اتجاه أمامي لأن معظم خوارزميات التخزين المخبئية ستحاول القراءة للأمام بدلاً من الرجوع للخلف. وبالمثل ، يعد الحفاظ على أنماط الوصول إلى الذاكرة محلية طريقة جيدة لتحسين الأداء.

إضافة ذاكرة مخبئية سيف ذو حدين. من ناحية ، فإنهم يضمنون أن نوى المعالج لا تتضور جوعًا في البيانات. في نفس الوقت يزيدون من حجم الرقاقة ، مستخدمين المساحة التي كان من الممكن أن يتم إنفاقها على زيادة قوة المعالجة. علاوة على ذلك ، يمكن أن تكون أخطاء ذاكرة التخزين المخبئي باهظة الثمن. فكر في أسوأ سيناريو ، المشاركة الخاطئة false sharing، كما هو موضح في الشكل 13.4.6. يتم تخزين موقع الذاكرة مؤقتًا على المعالج 0 عندما يطلب مؤشر ترابط على المعالج 1 البيانات. للحصول عليها ، يحتاج المعالج 0 إلى إيقاف ما يقوم به ، وإعادة كتابة المعلومات إلى الذاكرة الرئيسية ثم السماح للمعالج 1 بقراءتها من الذاكرة. أثناء هذه العملية ، ينتظر كلا المعالجين. من المحتمل جدًا أن تعمل هذه التعليمات البرمجية بشكل أبطأ على معالجات متعددة عند مقارنتها بالتنفيذ الفعال للمعالج الواحد. هذا سبب آخر لوجود حد عملي لأحجام ذاكرة التخزين المخبئية (إلى جانب حجمها المادي).



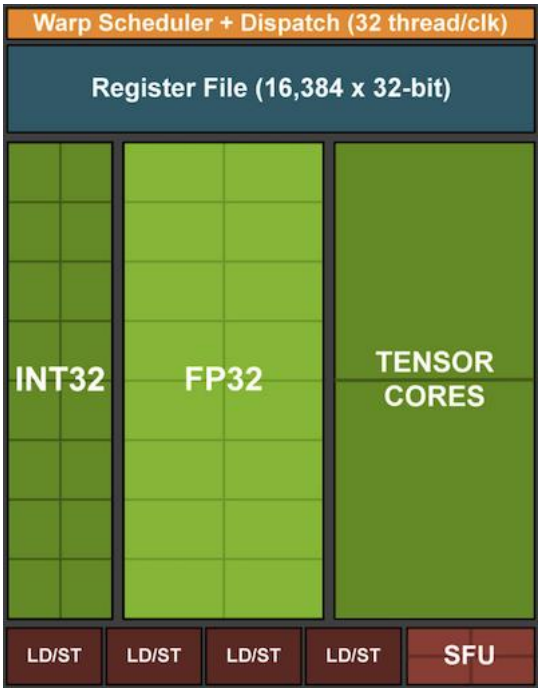
الشكل 13.4.6 المشاركة الخاطئة (الصورة مقدمة من Intel).

13.4.5. وحدات معالجة الرسومات والمسرعات الأخرى GPUs and other Accelerators

ليس من المبالغة الادعاء بأن التعلم العميق لم يكن لينجح بدون وحدات معالجة الرسومات GPU. على نفس المنوال ، من المنطقي القول أن ثروات مصنعي GPU قد زادت بشكل كبير بسبب التعلم العميق. أدى هذا التطور المشترك للأجهزة والخوارزميات إلى وضع يكون فيه التعلم العميق الأفضل أو الأسوأ هو نموذج النمذجة الإحصائية المفضل. ومن ثم فإنه من المفيد فهم الفوائد المحددة لوحدات معالجة الرسومات GPUs والمسرعات accelerators ذات الصلة مثل TPU (Jouppi et al.,2017).

من الجدير بالملاحظة التمييز الذي يتم إجراؤه غالبًا في الممارسة: يتم تحسين المسرعات إما للتدريب training أو الاستدلال inference. بالنسبة للأخير ، نحتاج فقط إلى حساب الانتشار الأمامي forward propagation في الشبكة. لا حاجة لتخزين البيانات الوسيطة من أجل الانتشار الخلفي backpropagation. علاوة على ذلك ، قد لا نحتاج إلى حساب دقيق للغاية (عادةً ما يكفي FP16 أو INT8). من ناحية أخرى ، أثناء التدريب ، تحتاج جميع النتائج الوسيطة إلى تخزين لحساب التدرجات (الانحدارات) gradients. علاوة على ذلك ، يتطلب تراكم التدرجات دقة أعلى لتجنب التدفق underflow الرقمي (أو الفائض overflow). هذا يعني أن FP16 (أو الدقة المختلطة مع FP32) هي الحد الأدنى من المتطلبات. كل هذا يتطلب ذاكرة أسرع وأكبر (HBM2 مقابل GDDR6) وقوة معالجة أكبر. على سبيل المثال ، تم تحسين وحدات معالجة الرسومات Turing T4 من NVIDIA للاستدلال بينما تُفضل وحدات معالجة الرسومات V100 للتدريب.

استدعي الفيكتورايزيشن كما هو موضح في الشكل 13.4.5. سمحت لنا إضافة وحدات المتجه إلى قلب المعالج بزيادة الإنتاجية بشكل كبير. على سبيل المثال ، في المثال الوارد في الشكل 13.4.5 ، تمكنا من إجراء 16 عملية في وقت واحد. أولاً ، ماذا لو أضفنا عمليات لم تُحسِّن العمليات بين المتجهات فحسب ، بل أيضًا بين المصفوفات؟ أدت هذه الإستراتيجية إلى نوى موتر tensor cores (سيتم تغطيتها قريبًا). ثانيًا ، ماذا لو أضفنا المزيد من النوى؟ باختصار ، تلخص هاتان الإستراتيجيتان قرارات التصميم في وحدات معالجة الرسومات. يعطي الشكل 13.4.7 نظرة عامة على كتلة معالجة أساسية. يحتوي على 16 عددًا صحيحًا و 16 وحدة فاصلة عائمة. بالإضافة إلى ذلك ، تعمل نواتان موتران على تسريع مجموعة فرعية ضيقة من العمليات الإضافية ذات الصلة بالتعلم العميق. يتكون كل متدفق متعدد المعالجات streaming multiprocessor من أربع كتل من هذا القبيل.



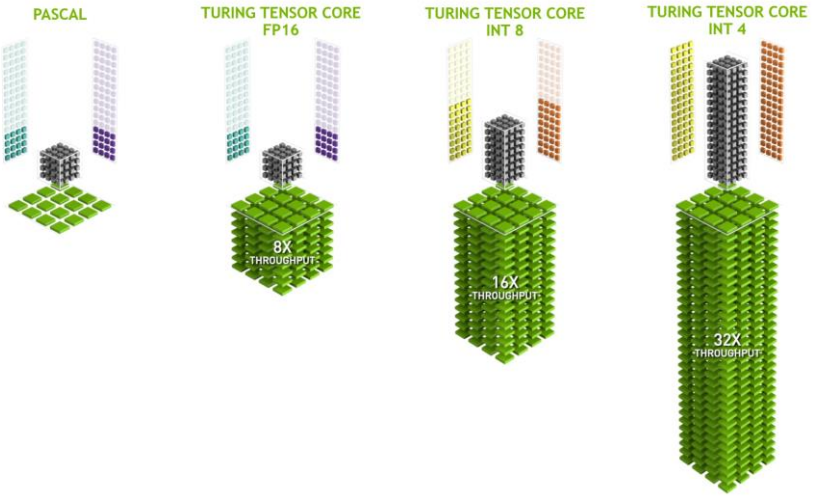
الشكل. 13.4.7 كتلة معالجة NVIDIA Turing (الصورة مقدمة من NVIDIA).

بعد ذلك ، تم تجميع 12 معالجات متدفقة متعددة في مجموعات معالجة الرسومات التي تشكل معالجات TU102 المتطورة. تكمل قنوات الذاكرة الكبيرة وذاكرة التخزين المخبئية L2 الإعداد. يحتوي الشكل 13.4.8 على التفاصيل ذات الصلة. أحد أسباب تصميم مثل هذا الجهاز هو أنه يمكن إضافة أو إزالة الكتل الفردية حسب الحاجة للسماح بمزيد من الرقائق المدمجة وللتعامل مع مشكلات الإنتاجية (قد لا يتم تنشيط الوحدات المعيبة faulty modules). لحسن الحظ ، فإن برمجة مثل هذه الأجهزة مخفية جيدًا عن باحث التعلم العميق غير الرسمي أسفل طبقات CUDA ورمز اطار العمل. على وجه الخصوص ، قد يتم تنفيذ أكثر من برنامج في وقت واحد على وحدة معالجة الرسومات ، بشرط توفر الموارد. ومع ذلك ، من المفيد أن تكون على دراية بقيود الأجهزة لتجنب اختيار النماذج التي لا تتناسب مع ذاكرة الجهاز.

الجانب الأخير الجدير بالذكر بمزيد من التفصيل هو نوى الموتر tensor cores. إنها مثال على الاتجاه الحديث لإضافة المزيد من الدوائر المحسّنة التي تكون فعالة بشكل خاص للتعلم العميق. على سبيل المثال ، أضاف TPU مصفوفة انقباضية systolic array (Kung، 1988) لضرب المصفوفة السريع. كان هناك تصميم لدعم عدد صغير جدًا (واحد للجيل الأول من TPU) من العمليات الكبيرة. النوى الموتر في الطرف الآخر. تم تحسينها للعمليات الصغيرة التي تتضمن

بين المصفوفات 4 × 4 و16 × 16 ، اعتمادًا على دقتها العددية. يعطي الشكل 13.4.9 نظرة عامة على التحسينات.

الشكل 13.4.8 معمارية NVIDIA Turing (الصورة مقدمة من NVIDIA)



الشكل 13.4.9 نوى موتر NVIDIA في Turing (الصورة مقدمة من NVIDIA).

من الواضح أنه عند التحسين من أجل الحساب ، ينتهي بنا الأمر إلى تقديم بعض التنازلات. أحدها هو أن وحدات معالجة الرسومات ليست جيدة جدًا في التعامل مع المقاطعات interrupts والبيانات المتفرقة sparse data. في حين أن هناك استثناءات ملحوظة ، مثل Gunrock، (Wang et al., 2016) ، فإن نمط الوصول للمصفوفات المتفرقة والمتجهات لا تسير على ما يرام مع عمليات قراءة الاندفاع burst read ذات النطاق الترددي العالي حيث تتفوق وحدات معالجة الرسومات. مطابقة كلا الهدفين هو مجال البحث النشط. انظر على سبيل المثال ، DGL ، مكتبة تم ضبطها للتعلم العميق على الرسوم البيانية.

13.4.6. الشبكات والنواقل Networks and Buses

عندما يكون جهاز واحد غير كافٍ للتحسين ، نحتاج إلى نقل البيانات منه وإليه لمزامنة المعالجة. هذا هو المكان الذي تصبح فيه الشبكات networks والنواقل buses في متناول اليد. لدينا عدد من معايير التصميم: النطاق الترددي والتكلفة والمسافة والمرونة. من ناحية ، لدينا شبكة WiFi ذات نطاق جيد جدًا ، ومن السهل جدًا استخدامها (لا توجد أسلاك ، بعد كل شيء)، وهي رخيصة ولكنها توفر نطاقًا ترددياً ووقت استجابة متواضعين نسبيًا. لن يستخدمه أي باحث في التعلم الآلي في عقله الصحيح لبناء مجموعة من الخوادم. فيما يلي نركز على الترابط المناسب للتعلم العميق.

- **PCIe** عبارة عن ناقل مخصص للاتصالات ذات النطاق الترددي العالي جدًا من نقطة إلى نقطة (حتى 32 جيجابايت / ثانية على PCIe 4.0 في منفذ slot ذات 16 مسارًا) لكل حارة. معدل الاستجابة (التاخير) Latency بترتيب ميكروثانية من رقم واحد (5 ميكروثانية). روابط PCIe ثمينة. تحتوي المعالجات على عدد محدود منها فقط: تحتوي AMD's EPYC 3 على 128 مسارًا lanes ، بينما تمتلك Intel's Xeon ما يصل إلى 48 مسارًا لكل شريحة ؛ على وحدات المعالجة المركزية لسطح المكتب، الأرقام هي 20 (Ryzen 9) و 16 (Core i9) على التوالي. نظرًا لأن وحدات معالجة الرسومات تحتوي عادةً على 16 مسارًا ، فإن هذا يحد من عدد وحدات معالجة الرسومات التي يمكنها الاتصال بوحدة المعالجة المركزية بنطاق ترددي كامل. بعد كل شيء ، يحتاجون إلى مشاركة الروابط مع الأجهزة الطرفية الأخرى ذات النطاق الترددي العالي مثل التخزين والإيثرنت. تمامًا كما هو الحال مع الوصول إلى ذاكرة الوصول العشوائي (RAM) ، يُفضل إجراء عمليات نقل كبيرة الحجم نظرًا لانخفاض overhead الحزمة.
- الإيثرنت **Ethernet** هي الطريقة الأكثر استخدامًا لتوصيل أجهزة الكمبيوتر. في حين أنه أبطأ بكثير من PCIe ، إلا أنه رخيص جدًا ومرن في التثبيت ويغطي مسافات أطول بكثير. النطاق الترددي النموذجي للخوادم منخفضة الدرجة هو 1 جيجابت / ثانية.

تقدم الأجهزة المتطورة (مثل مثيلات C5 في السحابة) عرض نطاق ترددي يتراوح بين 10 و 100 جيجابت / ثانية. كما هو الحال في جميع الحالات السابقة ، فإن نقل البيانات له تكاليف عامة كبيرة. لاحظ أننا لا نستخدم شبكة إيثرنت خام بشكل مباشر تقريبًا ، بل نستخدم بروتوكولًا يتم تنفيذه أعلى الاتصال المادي (مثل UDP أو TCP / IP). هذا يضيف المزيد من النفقات العامة. مثل PCIe ، تم تصميم Ethernet لتوصيل جهازين ، على سبيل المثال ، جهاز كمبيوتر ومحول switch.

- تسمح لنا المحولات **Switches** بتوصيل أجهزة متعددة بطريقة يمكن لأي زوج منها إجراء اتصال من نقطة إلى نقطة (عادةً بعرض نطاق كامل) في وقت واحد. على سبيل المثال ، قد تقوم محولات Ethernet بتوصيل 40 خادمًا بنطاق ترددي مقطعي عالٍ. لاحظ أن المحولات ليست فريدة بالنسبة لشبكات الكمبيوتر التقليدية. حتى ممرات PCIe يمكن تبديلها. يحدث هذا ، على سبيل المثال ، لتوصيل عدد كبير من وحدات معالجة الرسومات بمعالج مضيف ، كما هو الحال بالنسبة لمثيلات P2.
- **NVLink** هو بديل لـ PCIe عندما يتعلق الأمر بوصلات النطاق الترددي العالية جدًا. يوفر معدل نقل بيانات يصل إلى 300 جيجابت / ثانية لكل ارتباط. تحتوي وحدات معالجة الرسومات الخاصة بالخادم (Volta V100) على ستة روابط بينما تحتوي وحدات معالجة الرسومات الخاصة بالمستهلك (RTX 2080 Ti) على رابط واحد فقط، يعمل بمعدل 100 جيجابت / ثانية مخفض. نوصي باستخدام NCCL لتحقيق نقل بيانات عالي بين وحدات معالجة الرسومات.

13.4.7. المزيد من أرقام زمن الاستجابة (التأخير) More Latency Numbers

الملخص في القسم 13.4.7 والقسم 13.4.7 مأخوذ من Eliot Eshelman الذي يحتفظ بنسخة محدثة من الأرقام كـ GitHub gist.

Action	Time	Notes
L1 cache reference/hit	1.5 ns	4 cycles
Floating-point add/mult/FMA	1.5 ns	4 cycles
L2 cache reference/hit	5 ns	12 ~ 17 cycles

Action	Time	Notes
Branch mispredict	6 ns	15 ~ 20 cycles
L3 cache hit (unshared cache)	16 ns	42 cycles
L3 cache hit (shared in another core)	25 ns	65 cycles
Mutex lock/unlock	25 ns	
L3 cache hit (modified in another core)	29 ns	75 cycles
L3 cache hit (on a remote CPU socket)	40 ns	100 ~ 300 cycles (40 ~ 116 ns)
QPI hop to a another CPU (per hop)	40 ns	
64MB memory ref. (local CPU)	46 ns	TinyMemBench on Broadwell E5-2690v4
64MB memory ref. (remote CPU)	70 ns	TinyMemBench on Broadwell E5-2690v4
256MB memory ref. (local CPU)	75 ns	TinyMemBench on Broadwell E5-2690v4
Intel Optane random write	94 ns	UCSD Non-Volatile Systems Lab
256MB memory ref. (remote CPU)	120 ns	TinyMemBench on Broadwell E5-2690v4
Intel Optane random read	305 ns	UCSD Non-Volatile Systems Lab
Send 4KB over 100 Gbps HPC fabric	1 µs	MVAPICH2 over Intel Omni-Path

Action	Time	Notes
Compress 1KB with Google Snappy	3 µs	
Send 4KB over 10 Gbps ethernet	10 µs	
Write 4KB randomly to NVMe SSD	30 µs	DC P3608 NVMe SSD (QOS 99% is 500µs)
Transfer 1MB to/from NVLink GPU	30 µs	~33GB/s on NVIDIA 40GB NVLink
Transfer 1MB to/from PCI-E GPU	80 µs	~12GB/s on PCIe 3.0 x16 link
Read 4KB randomly from NVMe SSD	120 µs	DC P3608 NVMe SSD (QOS 99%)
Read 1MB sequentially from NVMe SSD	208 µs	~4.8GB/s DC P3608 NVMe SSD
Write 4KB randomly to SATA SSD	500 µs	DC S3510 SATA SSD (QOS 99.9%)
Read 4KB randomly from SATA SSD	500 µs	DC S3510 SATA SSD (QOS 99.9%)
Round trip within same data center	500 µs	One-way ping is ~250µs
Read 1MB sequentially from SATA SSD	2 ms	~550MB/s DC S3510 SATA SSD
Read 1MB sequentially from disk	5 ms	~200MB/s server HDD
Random Disk Access (seek+rotation)	10 ms	
Send packet CA->Netherlands->CA	150 ms	

Table: Common Latency Numbers.

Action	Time	Notes
GPU Shared Memory access	30 ns	30~90 cycles (bank conflicts add latency)
GPU Global Memory access	200 ns	200~800 cycles
Launch CUDA kernel on GPU	10 µs	Host CPU instructs GPU to start kernel
Transfer 1MB to/from NVLink GPU	30 µs	~33GB/s on NVIDIA 40GB NVLink
Transfer 1MB to/from PCI-E GPU	80 µs	~12GB/s on PCI-Express x16 link

الجدول: أرقام زمن الاستجابة Latency الشائعة.

Action	Time	Notes
GPU Shared Memory access	30 ns	30~90 cycles (bank conflicts add latency)
GPU Global Memory access	200 ns	200~800 cycles
Launch CUDA kernel on GPU	10 µs	Host CPU instructs GPU to start kernel
Transfer 1MB to/from NVLink GPU	30 µs	~33GB/s on NVIDIA 40GB NVLink
Transfer 1MB to/from PCI-E GPU	80 µs	~12GB/s on PCI-Express x16 link

الجدول: أرقام زمن الاستجابة لوحدات معالجة الرسومات NVIDIA Tesla.

13.4.8. الملخص

- الأجهزة لها overheads للعمليات. ومن ثم فمن المهم استهداف عدد صغير من التحويلات الكبيرة بدلاً من العديد من التحويلات الصغيرة. ينطبق هذا على ذاكرة الوصول العشوائي ومحركات الأقراص الصلبة والشبكات ووحدات معالجة الرسومات.
- الفيكتروزايشن هو مفتاح الأداء. تأكد من أنك على دراية بالقدرات المحددة للمسرع الخاص بك. على سبيل المثال ، بعض وحدات المعالجة المركزية Intel Xeon جيدة بشكل خاص لعمليات INT8 ، وتتفوق NVIDIA Volta GPUs في عمليات مصفوفة FP16 وتتألق NVIDIA Turing في عمليات FP16 و INT8 و INT4.
- يمكن أن يكون الفائض العددي Numerical overflow بسبب أنواع البيانات الصغيرة مشكلة أثناء التدريب (وبدرجة أقل أثناء الاستدلال).
- يمكن أن يؤدي التعرج Aliasing إلى تدهور الأداء بشكل كبير. على سبيل المثال ، يجب أن تتم محاذاة الذاكرة على وحدات المعالجة المركزية 64 بت فيما يتعلق بحدود 64 بت. في وحدات معالجة الرسومات، من الجيد الحفاظ على محاذاة أحجام الالتفاف ، على سبيل المثال، مع نوى الموتر.
- قم بمطابقة الخوارزميات الخاصة بك مع الأجهزة (على سبيل المثال ، بصمة الذاكرة وعرض النطاق الترددي). يمكن تحقيق تسريع كبير (الترتيب من حيث الحجم) عند تركيب المعلمات في ذواكر مخبئية.
- نوصي برسم أداء خوارزمية جديدة على الورق قبل التحقق من النتائج التجريبية. التناقضات Discrepancies في الترتيب من حيث الحجم أو أكثر هي أسباب للقلق.
- استخدم ملفات التعريف لتصحيح اختناقات الأداء performance bottlenecks.
- أجهزة التدريب والاستدلال لها نقاط حلوة مختلفة من حيث السعر والأداء.

13.4.9. التمارين

1. اكتب كود C لاختبار ما إذا كان هناك أي اختلاف في السرعة بين الوصول إلى الذاكرة المحاذية aligned أو المحاذاة بشكل غير صحيح misaligned بالنسبة لواجهة الذاكرة الخارجية. تلميح: احذر من تأثيرات التخزين المؤقت caching.
2. اختبر الفرق في السرعة بين الوصول إلى الذاكرة بالتسلسل أو بخطوة معينة.
3. كيف يمكنك قياس أحجام ذاكرة التخزين المؤقت (الذاكرة المخبئية) على وحدة المعالجة المركزية؟

4. كيف يمكنك تخطيط البيانات عبر قنوات ذاكرة متعددة للحصول على أقصى عرض نطاق ترددي؟ كيف يمكنك وضعه إذا كان لديك العديد من الخيوط الصغيرة small threads؟
5. يدور محرك الأقراص الثابتة من فئة المؤسسات بسرعة 10000 دورة في الدقيقة. ما هو الحد الأدنى المطلق من الوقت الذي يحتاجه محرك الأقراص الثابتة لقضاء أسوأ حالة قبل أن يتمكن من قراءة البيانات (يمكنك افتراض أن الرؤوس تتحرك بشكل فوري تقريبًا)؟ لماذا أصبحت محركات الأقراص الثابتة مقاس 2.5 بوصة شائعة بالنسبة للخوادم التجارية (مقارنة بمحركات الأقراص مقاس 3.5 بوصة و 5.25 بوصة)؟
6. افترض أن الشركة المصنعة للأقراص الصلبة تزيد كثافة التخزين من 1 تيرابايت لكل بوصة مربعة إلى 5 تيرابايت لكل بوصة مربعة. ما مقدار المعلومات التي يمكنك تخزينها على حلقة على محرك أقراص ثابتة مقاس 2.5 بوصة؟ هل هناك فرق بين المسارين الداخلي والخارجي inner and outer tracks ؟
7. يؤدي الانتقال من أنواع بيانات 8 بت إلى 16 بت إلى زيادة كمية السيليكون بمقدار أربعة أضعاف تقريبًا. لماذا ا؟ لماذا قد تضيف NVIDIA عمليات INT4 إلى وحدات معالجة الرسومات Turing الخاصة بها؟
8. ما مدى سرعة القراءة إلى الأمام من خلال الذاكرة مقابل القراءة العكسية؟ هل يختلف هذا الرقم بين أجهزة الكمبيوتر المختلفة وبائعي وحدة المعالجة المركزية؟ لماذا؟ اكتب كود C وجربه.
9. هل يمكنك قياس حجم ذاكرة التخزين المخبئية cache للقرص الخاص بك؟ ما هو لمحرك الأقراص الصلبة HDD النموذجي؟ هل تحتاج محركات الأقراص ذات الحالة الثابتة SSD إلى ذاكرة تخزين مؤقت؟
10. قم بقياس overhead الحزمة عند إرسال الرسائل عبر Ethernet. ابحث عن الفرق بين اتصالات UDP و TCP / IP.
11. يسمح الوصول المباشر للذاكرة للأجهزة بخلاف وحدة المعالجة المركزية بالكتابة (والقراءة) مباشرة إلى (من) الذاكرة. لماذا هذه الفكرة جيدة؟
12. انظر إلى أرقام الأداء لوحدة معالجة الرسومات Turing T4. لماذا يتضاعف الأداء "فقط" كلما انتقلت من FP16 إلى INT8 و INT4؟
13. ما هو أقصر وقت تستغرقه حزمة في رحلة ذهابًا وإيابًا بين سان فرانسيسكو وأمستردام؟ تلميح: يمكنك أن تفترض أن المسافة 10000 كيلومتر.

13.5. التدريب على وحدات معالجة الرسوميات المتعددة Training on Multiple GPUs

ناقشنا حتى الآن كيفية تدريب النماذج بكفاءة على وحدات المعالجة المركزية ووحدات معالجة الرسومات. لقد أظهرنا حتى كيف تسمح أطر التعلم العميقة للفرد بموازنة الحساب والتواصل تلقائيًا بينهمافي القسم 13.3. أظهرنا أيضًافي القسم 6.7 كيفية سرد جميع وحدات معالجة الرسومات المتاحة على جهاز كمبيوتر باستخدام الأمر `nvidia-smi`. ما لم نناقشه هو كيفية موازاة تدريب التعلم العميق. بدلاً من ذلك، ضمنافي تمرير أن المرء سيقسم البيانات بطريقة ما عبر أجهزة متعددة ويجعلها تعمل. يملأ القسم الحالي التفاصيل ويوضح كيفية تدريب شبكة بالتوازي عند البدء من نقطة الصفر. تم نقل التفاصيل المتعلقة بكيفية الاستفادة من الوظائففي واجهات برمجة التطبيقات عالية المستوى إلى القسم 13.6. نفترض أنك على دراية بخوارزميات الانحدار الاشتقاقي العشوائي المصغر mini SGD مثل تلك الموضحةفي القسم 12.5.

13.5.1. تقسيم المشكلة Splitting the Problem

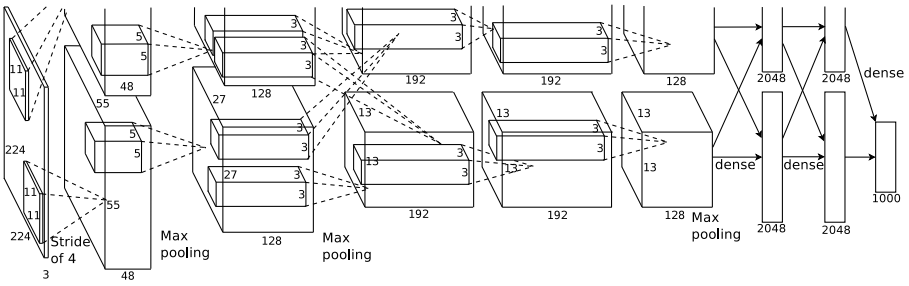
لنبدأ بمشكلة بسيطة في الرؤية الحاسوبية وشبكة قديمة بعض الشيء ، على سبيل المثال ، بطبقات متعددة من التلافيف والتجميع وربما بضع طبقات متصلة بالكامل في النهاية. بمعنى ، لنبدأ بشبكة تشبه إلى حد بعيد LeNet (LeCun et al.,1998) أو AlexNet، (Krizhevsky et al.,2012). بالنظر إلى وحدات معالجة الرسومات المتعددة (2 إذاكان خادمًا مكتبيًا ، 4 على مثيل AWS g4dn.12xlarge، 8 على p3.16xlarge ، أو 16 على p2.16xlarge) ، نريد تقسيم التدريب بطريقة تحقق تسريعًا جيداً مع الاستفادة في الوقت نفسه من خيارات التصميم البسيطة والقابلة للتكرار. بعد كل شيء ، تعمل وحدات معالجة الرسومات المتعددة على زيادة الذاكرة memory والقدرة على الحساب computation. باختصار ، لدينا الخيارات التالية ، بالنظر إلى مجموعة صغيرة من بيانات التدريب التي نريد تصنيفها.

أولاً ، يمكننا تقسيم الشبكة عبر وحدات معالجة رسومات متعددة multiple GPUs. أي أن كل وحدة معالجة رسومات (GPU) تأخذ البيانات المتدفقة إلى طبقة معينة كمدخلات ، وتعالج البيانات عبر عدد من الطبقات اللاحقة ثم ترسل البيانات إلى وحدة معالجة الرسومات التالية. يتيح لنا ذلك معالجة البيانات بشبكات أكبر عند مقارنتها بما يمكن أن تتعامل معه وحدة معالجة رسومات واحدة. إلى جانب ذلك ، يمكن التحكم في بصمة الذاكرة memory footprint لكل وحدة معالجة رسومات (GPU) بشكل جيد (إنها جزء صغير من إجمالي بصمة الشبكة network footprint).

ومع ذلك ، تتطلب الواجهة بين الطبقات (وبالتالي وحدات معالجة الرسومات) تزامنًا دقيقًا. قد يكون هذا أمرًا صعبًا ، خاصةً إذا لم تتم مطابقة أعباء العمل الحسابية بشكل صحيح بين الطبقات.

تتفاقم المشكلة بالنسبة لعدد كبير من وحدات معالجة الرسومات. تتطلب الواجهة بين الطبقات أيضًا كميات كبيرة من نقل البيانات ، مثل عمليات التنشيط والانحدارات. هذا قد يطغى على عرض النطاق الترددي لناقلات GPU. علاوة على ذلك ، العمليات الحسابية المكثفة ، ولحد الان المتسلسلة ليست بديهية للتقسيم. انظر على سبيل المثال Mirhoseini et al. (2017) لأفضل جهد في هذا الصدد. تظل مشكلة صعبة وليس من الواضح ما إذا كان من الممكن تحقيق مقياس جيد (خطي) للمشاكل غير البسيطة. لا نوصي بذلك ما لم يكن هناك إطار عمل أو دعم نظام تشغيل ممتاز لربط وحدات معالجة الرسومات المتعددة معًا.

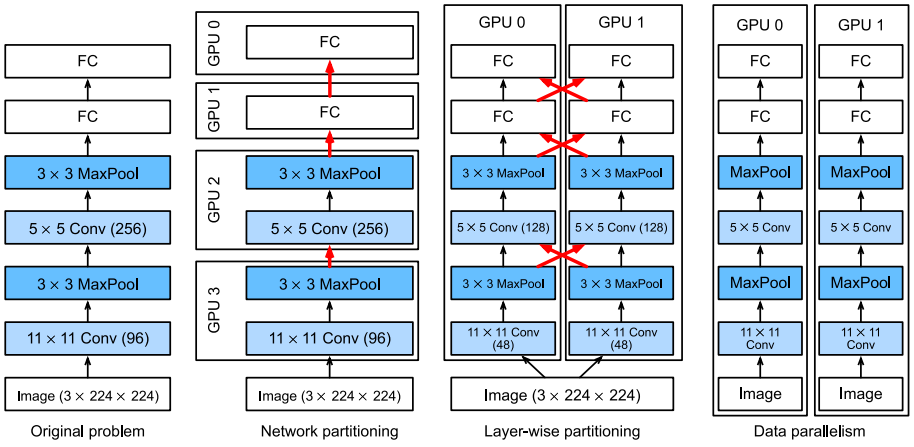
ثانيًا ، يمكننا تقسيم العمل إلى تقسيم طبقي layerwise. على سبيل المثال ، بدلاً من حساب 64 قناة على وحدة معالجة رسومات واحدة ، يمكننا تقسيم المشكلة عبر 4 وحدات معالجة رسومات ، كل منها يولد بيانات لـ 16 قناة. وبالمثل ، بالنسبة للطبقة المتصلة بالكامل، يمكننا تقسيم عدد وحدات الإخراج. يوضح الشكل 13.5.1 (مأخوذ من Krizhevsky et al. (2012) هذا التصميم ، حيث تم استخدام هذه الإستراتيجية للتعامل مع وحدات معالجة الرسومات التي لها مساحة ذاكرة صغيرة جدًا (2 جيجابايت في ذلك الوقت). يسمح هذا بقياس جيد من حيث الحساب ، بشرط ألا يكون عدد القنوات (أو الوحدات) صغيرًا جدًا. إلى جانب ذلك ، يمكن لوحدات معالجة الرسومات المتعددة معالجة شبكات أكبر بشكل متزايد نظرًا لأن حجم الذاكرة المتاحة يتسع خطيًا.



الشكل 13.5.1 نموذج التوازي في تصميم AlexNet الأصلي بسبب ذاكرة وحدة معالجة الرسومات المحدودة.

ومع ذلك ، نحتاج إلى عدد كبير جدًا من عمليات المزامنة synchronization أو الحاجز barrier لأن كل طبقة تعتمد على النتائج من جميع الطبقات الأخرى. علاوة على ذلك، من المحتمل أن تكون كمية البيانات التي يجب نقلها أكبر مما هي عليه عند توزيع الطبقات عبر وحدات معالجة الرسومات. وبالتالي ، لا نوصي بهذا الأسلوب نظرًا لتكلفة النطاق الترددي وتعقيده.

أخيرًا ، يمكننا تقسيم البيانات عبر وحدات معالجة رسومات متعددة. بهذه الطريقة ، تؤدي جميع وحدات معالجة الرسومات نفس نوع العمل ، وإن كان ذلك بناءً على مشاهدات مختلفة. يتم تجميع الانحدارات عبر وحدات معالجة الرسومات بعد كل دفعة صغيرة من بيانات التدريب. هذا هو أبسط نهج ويمكن تطبيقه في أي موقف. نحتاج فقط إلى المزامنة بعد كل دفعة صغيرة. ومع ذلك ، من المستحسن للغاية البدء في تبادل معلمات الانحدارات بالفعل بينما لا يزال يتم حساب معايير أخرى. علاوة على ذلك ، تؤدي الأعداد الكبيرة من وحدات معالجة الرسومات إلى أحجام أكبر من الدفعات الصغيرة ، وبالتالي زيادة كفاءة التدريب. ومع ذلك ، فإن إضافة المزيد من وحدات معالجة الرسومات لا يسمح لنا بتدريب نماذج أكبر.



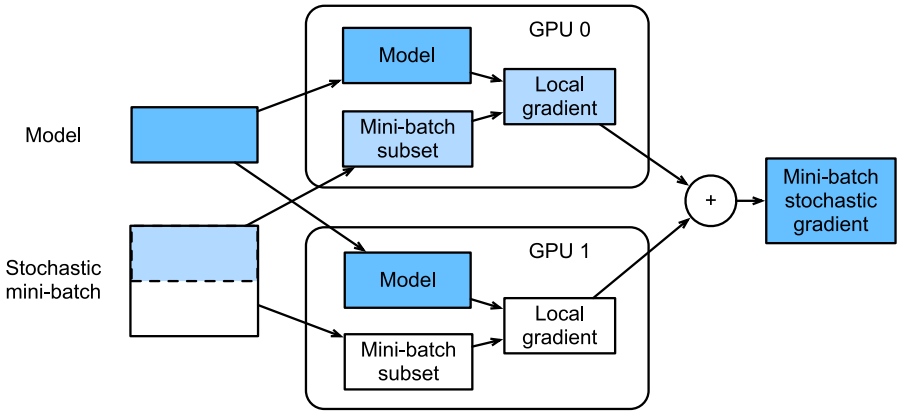
الشكل. 13.5.2 الموازاة Parallelization على وحدات معالجة رسومات متعددة. من اليسار إلى اليمين: المشكلة الأصلية ، تقسيم الشبكة network partitioning، التقسيم الطبقي layerwise partitioning، توازي البيانات data parallelism.

يوضح الشكل 13.5.2 مقارنة بين الطرق المختلفة للتوازي parallelization على وحدات معالجة رسومات متعددة. بشكل عام ، يعد توازي البيانات data parallelism هو الطريقة الأكثر ملاءمة للمتابعة ، بشرط أن نتمكن من الوصول إلى وحدات معالجة الرسومات ذات الذاكرة الكبيرة بدرجة كافية. انظر أيضًا (Li et al.، 2014) للحصول على وصف تفصيلي للتقسيم للتدريب الموزع. اعتادت ذاكرة وحدة معالجة الرسومات أن تكون مشكلة في الأيام الأولى للتعلم العميق. حتى الآن تم حل هذه المشكلة في جميع الحالات باستثناء الحالات الأكثر غرابة. نحن نركز على توازي البيانات فيما يلي.

13.5.2. توازي البيانات Data Parallelism

افترض أن هناك وحدات معالجة رسومات على الجهاز. بالنظر إلى النموذج الذي سيتم تدريبه ، ستحتفظ كل وحدة معالجة رسومات بمجموعة كاملة من معلمات النموذج بشكل مستقل على

الرغم من أن قيم المعلمات عبر وحدات معالجة الرسومات متطابقة ومتزامنة. كمثال ، يوضح الشكل 13.5.3 التدريب مع توازي البيانات عندما $k = 2$.



الشكل 13.5.3 حساب الانحدار الاشتقاقي العشوائي المصغر باستخدام توازي البيانات على اثنين من وحدات معالجة الرسومات.

بشكل عام ، يتم التدريب على النحو التالي:

- في أي تكرار للتدريب ، مع إعطاء دفعة صغيرة عشوائية ، قمنا بتقسيم الأمثلة في الدفعة إلى أجزاء وتوزيعها بالتساوي عبر وحدات معالجة الرسومات.
- تقوم كل وحدة معالجة رسومات (GPU) بحساب الخطأ والانحدار لمعلمات النموذج بناءً على مجموعة فرعية صغيرة تم تعيينها لها.
- يتم تجميع الانحدارات المحلية لكل وحدة من وحدات معالجة الرسومات للحصول على الانحدار الاشتقاقي العشوائي المصغر الحالي.
- يُعاد توزيع الانحدار الكلي على كل وحدة معالجة رسومات.
- تستخدم كل وحدة معالجة رسومات (GPU) هذا الانحدار الاشتقاقي العشوائي المصغر لتحديث مجموعة كاملة من معلمات النموذج التي تحتفظ بها.

لاحظ أنه من الناحية العملية نقوم بزيادة حجم الدفعات الصغيرة $k-fold$ أضعاف عند التدريب على وحدات معالجة الرسومات بحيث يكون لكل وحدة معالجة رسومات نفس القدر من العمل الذي يجب القيام به كما لو كنا نتدرب على وحدة معالجة رسومات واحدة فقط. على خادم 16 ـ GPU، يمكن أن يؤدي ذلك إلى زيادة حجم الدفعات الصغيرة بشكل كبير وقد نضطر إلى زيادة معدل التعلم وفقًا لذلك. لاحظ أيضًا أن تسوية الدُفعاتفي القسم 8.5 تحتاج إلى تعديل، على سبيل المثال، عن طريق الاحتفاظ بمعامل تسوية منفصل للدفعة لكل وحدة معالجة

رسومات. فيما يلي سوف نستخدم شبكة ألعاب Toy Network لتوضيح التدريب على وحدات معالجة الرسوميات المتعددة multi-GPU training.

```
%matplotlib inline
from mxnet import autograd, gluon, np, npx
from d2l import mxnet as d2l

npx.set_np()
```

13.5.3. شبكة ألعاب A Toy Network

نستخدم LeNet كما وردفي القسم 7.6 (مع تعديلات طفيفة). نحدده من البداية لتوضيح تبادل المعلمات ومزامنتها بالتفصيل.

```
# Initialize model parameters
scale = 0.01
W1 = np.random.normal(scale=scale, size=(20, 1, 3, 3))
b1 = np.zeros(20)
W2 = np.random.normal(scale=scale, size=(50, 20, 5, 5))
b2 = np.zeros(50)
W3 = np.random.normal(scale=scale, size=(800, 128))
b3 = np.zeros(128)
W4 = np.random.normal(scale=scale, size=(128, 10))
b4 = np.zeros(10)
params = [W1, b1, W2, b2, W3, b3, W4, b4]

# Define the model
def lenet(X, params):
    h1_conv = npx.convolution(data=X, weight=params[0],
bias=params[1],
                              kernel=(3, 3),
num_filter=20)
    h1_activation = npx.relu(h1_conv)
    h1 = npx.pooling(data=h1_activation,
pool_type='avg', kernel=(2, 2),
                     stride=(2, 2))
    h2_conv = npx.convolution(data=h1, weight=params[2],
bias=params[3],
                              kernel=(5, 5),
num_filter=50)
    h2_activation = npx.relu(h2_conv)
    h2 = npx.pooling(data=h2_activation,
pool_type='avg', kernel=(2, 2),
```

```
                        stride=(2, 2))
    h2 = h2.reshape(h2.shape[0], -1)
    h3_linear = np.dot(h2, params[4]) + params[5]
    h3 = npx.relu(h3_linear)
    y_hat = np.dot(h3, params[6]) + params[7]
    return y_hat

# Cross-entropy loss function
loss = gluon.loss.SoftmaxCrossEntropyLoss()
```

13.5.4. تزامن البيانات Data Synchronization

للتدريب الفعال على وحدات معالجة الرسومات المتعددة، نحتاج إلى عمليتين أساسيتين. نحتاج أولاً إلى القدرة على توزيع قائمة المعلمات على أجهزة متعددة وإرفاق الانحدارات (`get_params`). بدون معلمات، من المستحيل تقييم الشبكة على وحدة معالجة الرسومات. ثانيًا، نحتاج إلى القدرة على جمع المعلمات عبر أجهزة متعددة، أي أننا بحاجة إلى دالة `allreduce`.

```
def get_params(params, device):
    new_params = [p.copyto(device) for p in params]
    for p in new_params:
        p.attach_grad()
    return new_params
```

دعنا نجرب ذلك عن طريق نسخ معلمات النموذج إلى وحدة معالجة رسومات واحدة.

```
new_params = get_params(params, d2l.try_gpu(0))
print('b1 weight:', new_params[1])
print('b1 grad:', new_params[1].grad)
```

```
b1 weight: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] @gpu(0)
b1 grad: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] @gpu(0)
```

نظرًا لأننا لم نقم بإجراء أي حساب حتى الآن، فإن الانحدار فيما يتعلق بمعامل التحيز لا يزال صفراً. لنفترض الآن أن لدينا متجهًا موزعًا عبر وحدات معالجة رسومات متعددة. تضيف دالة `allreduce` التالية جميع المتجهات وتبث النتيجة مرة أخرى إلى جميع وحدات معالجة الرسومات. لاحظ أنه لكي يعمل هذا، نحتاج إلى نسخ البيانات إلى الجهاز الذي يقوم بتجميع النتائج.

```
def allreduce(data):
    for i in range(1, len(data)):
        data[0][:] += data[i].copyto(data[0].ctx)
```

```
    for i in range(1, len(data)):
        data[0].copyto(data[i])
```

دعنا نختبر هذا عن طريق إنشاء متجهات بقيم مختلفة على أجهزة مختلفة وجمعها.

```
data = [np.ones((1, 2), ctx=d2l.try_gpu(i)) * (i + 1)
for i in range(2)]
print('before allreduce:\n', data[0], '\n', data[1])
allreduce(data)
print('after allreduce:\n', data[0], '\n', data[1])
```

```
before allreduce:
 [[1. 1.]] @gpu(0)
 [[2. 2.]] @gpu(1)
after allreduce:
 [[3. 3.]] @gpu(0)
 [[3. 3.]] @gpu(1)
```

13.5.5. توزيع البيانات Distributing Data

نحتاج إلى دالة أداة مساعدة بسيطة لتوزيع الدفعات الصغيرة minibatch بالتساوي عبر وحدات معالجة رسومات متعددة. على سبيل المثال،في اثنين من وحدات معالجة الرسومات، نود أن يكون لدينا نصف البيانات المراد نسخها إلى أي من وحدات معالجة الرسومات. نظرًا لأنها أكثر ملاءمة وإيجازًا، فإننا نستخدم الدالة المضمنة من إطار عمل التعلم العميق لتجربتها على مصفوفة 4×5.

```
data = np.arange(20).reshape(4, 5)
devices = [npx.gpu(0), npx.gpu(1)]
split = gluon.utils.split_and_load(data, devices)
print('input :', data)
print('load into', devices)
print('output:', split)
```

```
input : [[ 0.  1.  2.  3.  4.]
 [ 5.  6.  7.  8.  9.]
 [10. 11. 12. 13. 14.]
 [15. 16. 17. 18. 19.]]
load into [gpu(0), gpu(1)]
output: [array([[0., 1., 2., 3., 4.],
       [5., 6., 7., 8., 9.]], ctx=gpu(0)), array([[10.,
11., 12., 13., 14.],
       [15., 16., 17., 18., 19.]], ctx=gpu(1))]
```

لإعادة الاستخدام لاحقًا، نحدد دالة `split_batch` التي تقسم البيانات والتسميات labels.

```
#@save
```

```
def split_batch(X, y, devices):
    """Split `X` and `y` into multiple devices."""
    assert X.shape[0] == y.shape[0]
    return (gluon.utils.split_and_load(X, devices),
            gluon.utils.split_and_load(y, devices))
```

13.5.6. التدريب Training

الآن يمكننا تنفيذ تدريب GPU متعدد على دفعة صغيرة واحدة. يعتمد تنفيذه بشكل أساسي على نهج توازي البيانات الموصوففي هذا القسم. سنستخدم الدوال المساعدة التي ناقشناها للتو، `allreduce` و `split_and_load` ، لمزامنة البيانات بين وحدات معالجة الرسومات المتعددة. لاحظ أننا لا نحتاج إلى كتابة أي كود محدد لتحقيق التوازي. نظرًا لأن الرسم البياني الحسابي لا يحتوي على أي تبعيات عبر الأجهزة داخل الدفعات الصغيرة، يتم تنفيذه بالتوازي تلقائيًا automatically.

```
def train_batch(X, y, device_params, devices, lr):
    X_shards, y_shards = split_batch(X, y, devices)
    with autograd.record():  # Loss is calculated
separately on each GPU
        ls = [loss(lenet(X_shard, device_W), y_shard)
              for X_shard, y_shard, device_W in zip(
                  X_shards, y_shards, device_params)]
    for l in ls:  # Backpropagation is performed
separately on each GPU
        l.backward()
    # Sum all gradients from each GPU and broadcast them
to all GPUs
    for i in range(len(device_params[0])):
        allreduce([device_params[c][i].grad for c in
range(len(devices))])
    # The model parameters are updated separately on
each GPU
    for param in device_params:
        d2l.sgd(param, lr, X.shape[0])  # Here, we use a
full-size batch
```

الآن، يمكننا تحديد دالة التدريب. يختلف قليلاً عن تلك المستخدمةفي الفصول السابقة: نحتاج إلى تخصيص وحدات معالجة الرسومات ونسخ جميع معلمات النموذج إلى جميع الأجهزة. من الواضح أن كل دفعة تتم معالجتها باستخدام دالة `train_batch` للتعامل مع وحدات معالجة رسومات متعددة. للسهولة (وإيجاز الكود) نحسب الدقة على وحدة معالجة رسومات واحدة،

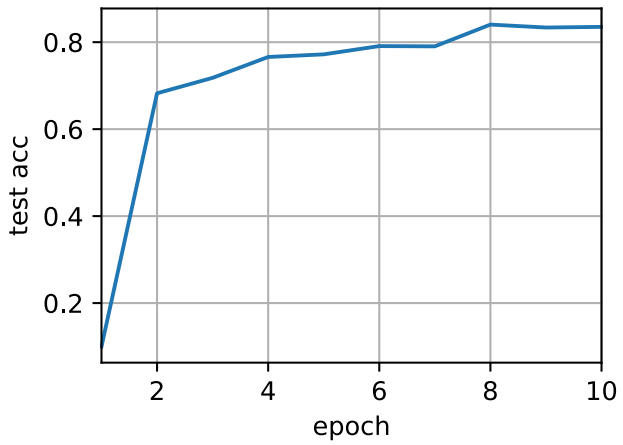
على الرغم من أن هذا غير كفوء inefficient لأن وحدات معالجة الرسومات الأخرى خاملة idle.

```
def train(num_gpus, batch_size, lr):
    train_iter, test_iter =
d2l.load_data_fashion_mnist(batch_size)
    devices = [d2l.try_gpu(i) for i in range(num_gpus)]
    # Copy model parameters to `num_gpus` GPUs
    device_params = [get_params(params, d) for d in
devices]
    num_epochs = 10
    animator = d2l.Animator('epoch', 'test acc',
xlim=[1, num_epochs])
    timer = d2l.Timer()
    for epoch in range(num_epochs):
        timer.start()
        for X, y in train_iter:
            # Perform multi-GPU training for a single
minibatch
            train_batch(X, y, device_params, devices,
lr)
            npx.waitall()
        timer.stop()
        # Evaluate the model on GPU 0
        animator.add(epoch + 1,
(d2l.evaluate_accuracy_gpu(
            lambda x: lenet(x, device_params[0]),
test_iter, devices[0]),))
    print(f'test acc: {animator.Y[0][-1]:.2f},
{timer.avg():.1f} sec/epoch '
          f'on {str(devices)}')
```

دعونا نرى كيف يعمل هذا بشكل جيد على وحدة معالجة رسومات واحدة. نستخدم أولاً حجم دفعة 256 ومعدل تعلم 0.2.

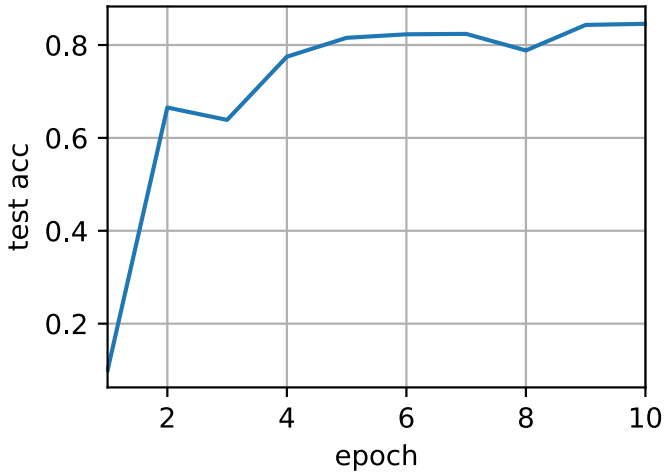
```
train(num_gpus=1, batch_size=256, lr=0.2)
```

```
test acc: 0.84, 2.3 sec/epoch on [gpu(0)]
```



من خلال الحفاظ على حجم الدُفعة ومعدل التعلم دون تغيير وزيادة عدد وحدات معالجة الرسومات إلى 2 ، يمكننا أن نرى أن دقة الاختبار تظل كما هي تقريبًا مقارنة بالتجربة السابقة. من حيث خوارزميات التحسين ، فهي متطابقة. لسوء الحظ ، ليس هناك تسريع ذي مغزى يمكن اكتسابه هنا: النموذج صغير جدًا ؛ علاوة على ذلك ، ليس لدينا سوى مجموعة بيانات صغيرة ، حيث عانى نهجنا البسيط إلى حد ما في تنفيذ التدريب على وحدات معالجة الرسومات المتعددة من زيادة كبيرة في بايثون. سنواجه نماذج أكثر تعقيدًا وطرقًا أكثر تعقيدًا للموازنة في المستقبل. دعونا نرى ما يحدث مع ذلك لـ Fashion-MNIST.

```
train(num_gpus=2, batch_size=256, lr=0.2)
test acc: 0.85, 4.7 sec/epoch on [gpu(0), gpu(1)]
```



13.5.7. الملخص

- هناك عدة طرق لتقسيم تدريب الشبكة العميقة على وحدات معالجة رسومات متعددة. يمكننا تقسيمها بين طبقات أو عبر طبقات أو عبر البيانات. يتطلب النوعان السابقان عمليات نقل بيانات منسقة بدقة. توازي البيانات Data parallelism هو أبسط استراتيجية.
- التدريب الموازي للبيانات Data parallel training واضح ومباشر. ومع ذلك ، فإنه يزيد من حجم الدفعات الصغيرة الفعال ليكون كفوء.
- في توازي البيانات ، يتم تقسيم البيانات عبر العديد من وحدات معالجة الرسومات، حيث تنفذ كل وحدة معالجة رسومات عملياتها الخاصة للأمام والخلف ، ثم يتم تجميع الانحدارات ويتم بث النتائج مرة أخرى إلى وحدات معالجة الرسومات.
- قد نستخدم معدلات تعلم متزايدة بشكل طفيف للدفعات الصغيرة الأكبر.

13.5.8. التمارين

1. عند التدريب على k وحدات معالجة الرسومات ، قم بتغيير حجم الدفعات الصغيرة من b إلى $k \cdot b$ ، أي ، قم بتوسيعه حسب عدد وحدات معالجة الرسومات.
2. قارن الدقة لمعدلات التعلم المختلفة. كيف يتناسب مع عدد وحدات معالجة الرسومات؟
3. نفذ دالة `allreduce` الاكثر كفاءة والتي تجمع معلمات مختلفة على وحدات معالجة رسومية مختلفة؟ لماذا هو أكثر كفاءة؟
4. نفذ حساب دقة الاختبار لـ GPU المتعدد.

13.6. التنفيذ الموجز لوحدات معالجة الرسومات المتعددة Concise Implementation for Multiple GPUs

إن تطبيق التوازي من الصفر لكل نموذج جديد ليس بالأمر الممتع. علاوة على ذلك ، هناك فائدة كبيرة في تحسين أدوات المزامنة للحصول على أداء عالٍ. في ما يلي سنوضح كيفية القيام بذلك باستخدام واجهات برمجة التطبيقات عالية المستوى لأطر التعلم العميق. الرياضيات والخوارزميات هي نفسها الواردة في القسم 13.5. ليس من المستغرب تمامًا أنك ستحتاج إلى وحدتي GPU على الأقل لتشغيل كود هذا القسم.

```
from mxnet import autograd, gluon, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

13.6.1. شبكة ألعاب A Toy Network

دعنا نستخدم شبكة أكثر جدوى من LeNet من القسم 13.5 والتي لا تزال سهلة وسريعة التدريب بدرجة كافية. نختار متغير ResNet-18، (2016,He et al.). نظرًا لأن صور الإدخال صغيرة ، فإننا نقوم بتعديلها قليلاً. على وجه الخصوص ، الاختلاف عن القسم 8.6 هو أننا نستخدم نواة التفاف convolution kernel أصغر ، وخطوة stride، وحشو padding في البداية. علاوة على ذلك ، نقوم بإزالة طبقة التجميع القصوى max-pooling layer.

```
#@save
def resnet18(num_classes):
    """A slightly modified ResNet-18 model."""
    def resnet_block(num_channels, num_residuals,
first_block=False):
        blk = nn.Sequential()
        for i in range(num_residuals):
            if i == 0 and not first_block:
                blk.add(d2l.Residual(
                    num_channels, use_1x1conv=True,
strides=2))
            else:
                blk.add(d2l.Residual(num_channels))
        return blk

    net = nn.Sequential()
    # This model uses a smaller convolution kernel,
stride, and padding and
    # removes the max-pooling layer
    net.add(nn.Conv2D(64, kernel_size=3, strides=1,
padding=1),
            nn.BatchNorm(), nn.Activation('relu'))
    net.add(resnet_block(64, 2, first_block=True),
            resnet_block(128, 2),
            resnet_block(256, 2),
            resnet_block(512, 2))
    net.add(nn.GlobalAvgPool2D(), nn.Dense(num_classes))
    return net
```

13.6.2. تهيئة الشبكة Network Initialization

تسمح لنا دالة `initialize` بتهيئة المعلمات على جهاز من اختيارنا. لتجديد معلومات حول طرق التهيئة ، انظر القسم 5.4. ما هو ملائم بشكل خاص هو أنه يسمح لنا أيضًا بتهيئة الشبكة على أجهزة متعددة multiple في وقت واحد. دعونا نجرب كيف يعمل هذا في الممارسة.

```
net = resnet18(10)
# Get a list of GPUs
devices = d2l.try_all_gpus()
# Initialize all the parameters of the network
net.initialize(init=init.Normal(sigma=0.01),
ctx=devices)
```

باستخدام دالة `split_and_load` المقدمة في القسم 13.5 ، يمكننا تقسيم دفعة صغيرة من البيانات ونسخ أجزاء إلى قائمة الأجهزة التي يوفرها متغير `devices`. يستخدم مثيل الشبكة تلقائيًا GPU المناسب لحساب قيمة الانتشار الأمامي. هنا نولد 4 مشاهدات ونقسمها على وحدات معالجة الرسومات.

```
x = np.random.uniform(size=(4, 1, 28, 28))
x_shards = gluon.utils.split_and_load(x, devices)
net(x_shards[0]), net(x_shards[1])
```

```
[23:01:13] src/operator/nn/./cudnn/./cudnn_algoreg-
inl.h:97: Running performance tests to find the best
convolution algorithm, this can take a while... (set the
environment variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0
to disable)
(array([[ 2.2610209e-06,  2.2046002e-06, -5.4046795e-06,
1.2869943e-06,
          5.1373145e-06, -3.8297976e-06,  1.4338764e-07,
5.4683437e-06,
         -2.8279201e-06, -3.9651104e-06],
        [ 2.0698662e-06,  2.0084678e-06, -5.6382491e-06,
1.0498463e-06,
          5.5506434e-06, -4.1065459e-06,  6.0830268e-07,
5.4521765e-06,
         -3.7365021e-06, -4.1891644e-06]], ctx=gpu(0)),
 array([[ 2.4629803e-06,  2.6015516e-06, -5.4362627e-06,
1.2938222e-06,
          5.6387917e-06, -4.1360108e-06,  3.5759149e-07,
5.5125265e-06,
         -3.1957311e-06, -4.2976321e-06],
        [ 1.9431664e-06,  2.2600414e-06, -5.2698206e-06,
1.4807424e-06,
          5.4830957e-06, -3.9678876e-06,  7.5752268e-08,
5.6764356e-06,
         -3.2530236e-06, -4.0943919e-06]], ctx=gpu(1)))
```

بمجرد مرور البيانات عبر الشبكة ، تتم تهيئة المعلمات المقابلة على الجهاز الذي تمر البيانات خلاله. هذا يعني أن التهيئة تحدث على أساس كل جهاز. نظرًا لأننا اخترنا GPU 0 و GPU 1 للتهيئة ، تتم تهيئة الشبكة هناك فقط، وليس على وحدة المعالجة المركزية. في الواقع ، لا توجد المعلمات حتى على وحدة المعالجة المركزية. يمكننا التحقق من ذلك من خلال طباعة المعلمات ومراقبة أي أخطاء قد تنشأ.

```
weight = net[0].params.get('weight')

try:
    weight.data()
except RuntimeError:
    print('not initialized on cpu')
weight.data(devices[0])[0], weight.data(devices[1])[0]
```

```
not initialized on cpu
(array([[[ 0.01382882, -0.01183044,  0.01417865],
         [-0.00319718,  0.00439528,  0.02562625],
         [-0.00835081,  0.01387452, -0.01035946]]],
ctx=gpu(0)),
 array([[[ 0.01382882, -0.01183044,  0.01417865],
         [-0.00319718,  0.00439528,  0.02562625],
         [-0.00835081,  0.01387452, -0.01035946]]],
ctx=gpu(1)))
```

بعد ذلك ، دعنا نستبدل الكود لتقييم الدقة بواحد يعمل بالتوازي عبر أجهزة متعددة. يعمل هذا كبديل لدالة `evaluate_accuracy_gpu` من القسم 7.6. الفرق الرئيسي هو أننا قمنا بتقسيم الدفعات الصغيرة minibatch قبل استدعاء الشبكة. كل شيء آخر متطابق بشكل أساسي.

```
#@save
def evaluate_accuracy_gpus(net, data_iter,
split_f=d2l.split_batch):
    """Compute the accuracy for a model on a dataset
using multiple GPUs."""
    # Query the list of devices
    devices =
list(net.collect_params().values())[0].list_ctx()
    # No. of correct predictions, no. of predictions
    metric = d2l.Accumulator(2)
    for features, labels in data_iter:
        X_shards, y_shards = split_f(features, labels,
devices)
```

```
        # Run in parallel
        pred_shards = [net(X_shard) for X_shard in
X_shards]
        metric.add(sum(float(d2l.accuracy(pred_shard,
y_shard)) for
                       pred_shard, y_shard in zip(
                           pred_shards, y_shards)),
labels.size)
    return metric[0] / metric[1]
```

13.6.3. التدريب

كما في السابق ، يحتاج كود التدريب إلى أداء عدة دوال أساسية للتوازي الفعال:

- يجب تهيئة معلمات الشبكة عبر جميع الأجهزة.
- أثناء التكرار على مجموعات بيانات الدفعات الصغيرة يجب تقسيمها عبر جميع الأجهزة.
- نحسب الخسارة وتدرجها بالتوازي عبر الأجهزة.
- يتم تجميع الانحدارات ويتم تحديث المعلمات وفقًا لذلك.

في النهاية نحسب الدقة (مرة أخرى بالتوازي) للإبلاغ عن الأداء النهائي للشبكة. روتين التدريب مشابه تمامًا لعمليات التنفيذ في الفصول السابقة ، باستثناء أننا بحاجة إلى تقسيم البيانات وتجميعها.

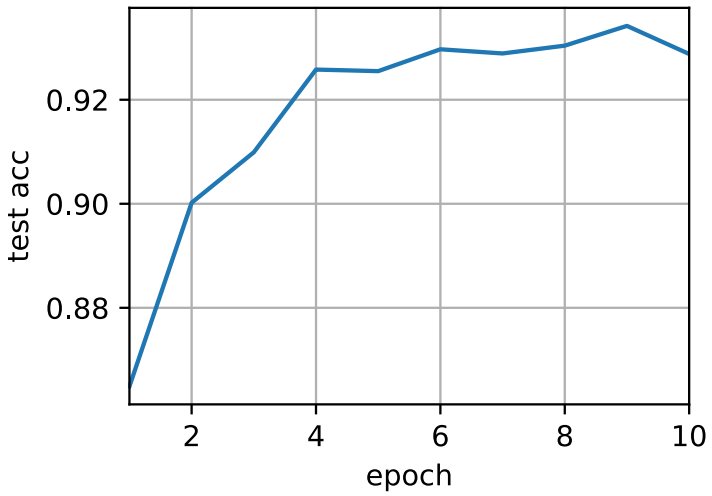
```
def train(num_gpus, batch_size, lr):
    train_iter, test_iter =
d2l.load_data_fashion_mnist(batch_size)
    ctx = [d2l.try_gpu(i) for i in range(num_gpus)]
    net.initialize(init=init.Normal(sigma=0.01),
ctx=ctx, force_reinit=True)
    trainer = gluon.Trainer(net.collect_params(), 'sgd',
                            {'learning_rate': lr})
    loss = gluon.loss.SoftmaxCrossEntropyLoss()
    timer, num_epochs = d2l.Timer(), 10
    animator = d2l.Animator('epoch', 'test acc',
xlim=[1, num_epochs])
    for epoch in range(num_epochs):
        timer.start()
        for features, labels in train_iter:
            X_shards, y_shards =
d2l.split_batch(features, labels, ctx)
            with autograd.record():
```

```
                ls = [loss(net(X_shard), y_shard) for
X_shard, y_shard
                      in zip(X_shards, y_shards)]
            for l in ls:
                l.backward()
            trainer.step(batch_size)
        npx.waitall()
        timer.stop()
        animator.add(epoch + 1,
(evaluate_accuracy_gpus(net, test_iter),))
    print(f'test acc: {animator.Y[0][-1]:.2f},
{timer.avg():.1f} sec/epoch '
          f'on {str(ctx)}')
```

دعونا نرى كيف يعمل هذا في الممارسة. كإحماء ، نقوم بتدريب الشبكة على وحدة معالجة رسومات واحدة.

```
train(num_gpus=1, batch_size=256, lr=0.1)
```

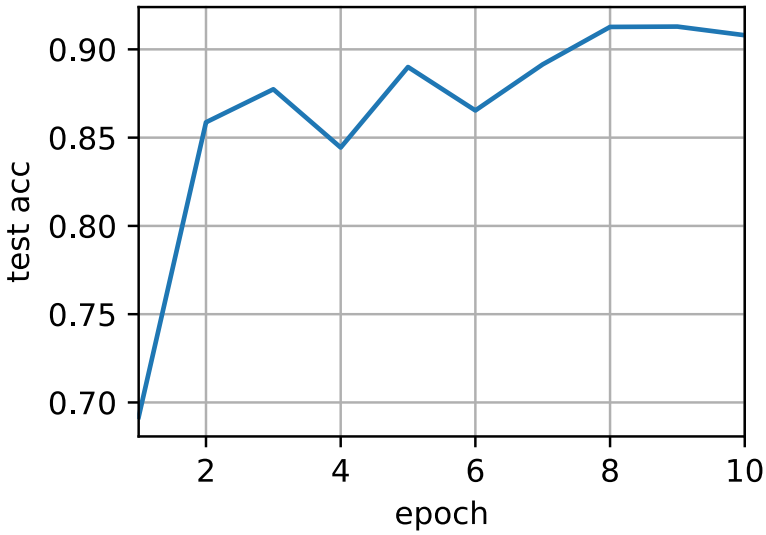
```
test acc: 0.93, 13.1 sec/epoch on [gpu(0)]
```



بعد ذلك ، نستخدم وحدتي معالجة رسومات (GPU) للتدريب. بالمقارنة مع LeNet الذي تم تقييمه في القسم 13.5 ، فإن نموذج ResNet-18 أكثر تعقيدًا إلى حد كبير. هذا هو المكان الذي يظهر فيه التوازي ميزته. وقت الحساب أكبر من وقت مزامنة المعلمات. هذا يحسن قابلية التوسع لأن الحمل overhead للتوازي أقل أهمية.

```
train(num_gpus=2, batch_size=512, lr=0.2)
```

```
test acc: 0.91, 6.8 sec/epoch on [gpu(0), gpu(1)]
```



13.6.4. الملخص

- يتم تقييم البيانات تلقائيًا على الأجهزة حيث يمكن العثور على البيانات.
- احرص على تهيئة الشبكات على كل جهاز قبل محاولة الوصول إلى المعلمات الموجودة على هذا الجهاز. وإلا سوف تواجه خطأ.
- تتجمع خوارزميات التحسين تلقائيًا عبر وحدات معالجة رسومات متعددة.

13.6.5. التمارين

1. يستخدم هذا القسم ResNet-18. جرب فترات مختلفة وأحجام دفعات ومعدلات تعلم مختلفة. استخدم المزيد من وحدات معالجة الرسومات للحساب. ماذا يحدث إذا جربت ذلك باستخدام 16 وحدة معالجة رسومات (على سبيل المثال، على مثيل AWS p2.16xlarge)؟
2. في بعض الأحيان ، توفر الأجهزة المختلفة قوة حوسبة مختلفة. يمكننا استخدام وحدات معالجة الرسومات ووحدة المعالجة المركزية في نفس الوقت. كيف نقسم العمل؟ هل يستحق كل هذا الجهد؟ لماذا؟ لما لا؟
3. ماذا يحدث إذا أسقطنا `npx.waitall()`؟ كيف يمكنك تعديل التدريب بحيث يكون لديك تداخل يصل إلى خطوتين للتوازي؟

13.7. خوادم المعلمات Parameter Servers

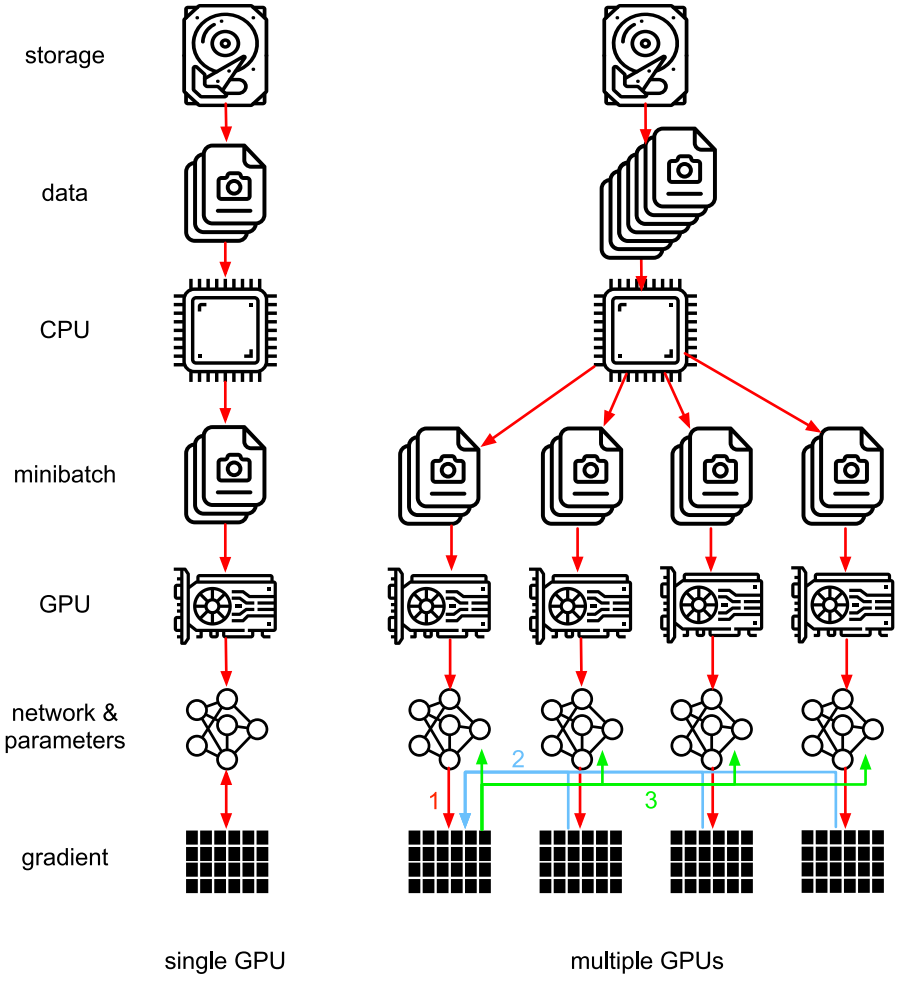
نظرًا لأننا ننتقل من وحدة معالجة رسومات واحدة إلى وحدات معالجة رسومات متعددة ثم إلى خوادم متعددة تحتوي على وحدات معالجة رسومات متعددة ، وربما تنتشر جميعها عبر رفوف متعددة multiple racks ومبدلات الشبكة network switches، يجب أن تصبح خوارزمياتنا الخاصة بالتدريب الموزع والمتوازي أكثر تعقيدًا. التفاصيل مهمة نظرًا لأن التوصيلات البينية المختلفة لها نطاق ترددي مختلف تمامًا (على سبيل المثال ، يمكن أن تقدم NVLink ما يصل إلى 100 جيجابايت / ثانية عبر 6 روابط في إعداد مناسب ، يوفر PCIe 4.0 (16 مسارًا) 32 جيجابايت / ثانية ، بينما حتى سرعة 100 جيجابت إيثرنت عالية السرعة فقط إلى 10 جيجابايت / ثانية). في الوقت نفسه، من غير المعقول توقع أن يكون المصمم الإحصائي خبيرًا في الشبكات والأنظمة.

تم تقديم الفكرة الأساسية لخادم المعلمات parameter server في Smola و Narayanamurthy (2010) في سياق النماذج المتغيرة الكامنة الموزعة. وصف لدلالات الدفع والجذب ثم تبع ذلك في Ahmed et al. (2012) ووصف للنظام ومكتبة مفتوحة المصدر في Li et al. (2014). في ما يلي سوف نقوم بتحفيز المكونات اللازمة لتحقيق الكفاءة.

13.7.1. التدريب الموازي للبيانات Data-Parallel Training

دعونا نراجع نهج التدريب الموازي للبيانات للتدريب الموزع. سنستخدم هذا لاستبعاد كل الآخرين في هذا القسم لأنه من الأسهل تنفيذه في الممارسة العملية. لا توجد حالات استخدام تقريبًا (إلى جانب التعلم العميق على الرسوم البيانية) حيث يُفضل استخدام أي استراتيجية أخرى للتوازي نظرًا لأن وحدات معالجة الرسومات لديها الكثير من الذاكرة في الوقت الحاضر. يصف الشكل 13.7.1 متغير توازي البيانات الذي طبقناه في القسم 13.5. الجانب الرئيسي في ذلك هو أن تجميع الانحدارات يحدث في GPU 0 قبل إعادة بث المعلمات المحدثة إلى جميع وحدات معالجة الرسومات.

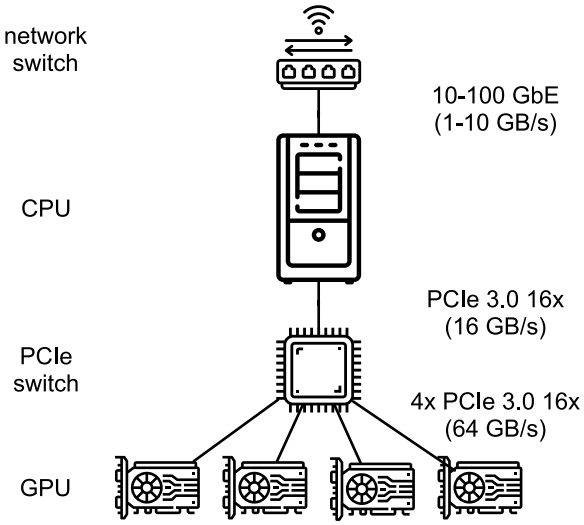
في الماضي ، يبدو قرار التجميع على GPU 0 مؤقتًا إلى حد ما. بعد كل شيء ، قد نقوم أيضًا بتجميع وحدة المعالجة المركزية. في الواقع ، يمكننا حتى أن نقرر تجميع بعض المعلمات في وحدة معالجة رسومات واحدة وبعضها الآخر على وحدة معالجة أخرى. شريطة أن تدعم خوارزمية التحسين ذلك ، فلا يوجد سبب حقيقي لعدم تمكننا من ذلك. على سبيل المثال ، إذا كان لدينا أربعة متجهات معلمات مع الانحدارات المرتبطة $\mathbf{g}_1, \ldots, \mathbf{g}_4$، فيمكننا تجميع الانحدارات في وحدة معالجة رسومات واحدة لكل $\mathbf{g}_i (i = 1, \ldots, 4)$.



الشكل. 13.7.1 يسار: تدريب واحد لوحدة معالجة الرسومات. اليمين: متغير من التدريب على وحدات معالجة الرسومات المتعددة: (1) نحسب الخسارة والانحدار، (2) يتم تجميع جميع الانحدارات في وحدة معالجة رسومات واحدة ، (3) يحدث تحديث للمعلمات ويتم إعادة توزيع المعلمات على جميع وحدات معالجة الرسومات.

يبدو هذا المنطق تعسفيًا وتافهًا. بعد كل شيء ، الرياضيات هي نفسها طوال الوقت. ومع ذلك ، فإننا نتعامل مع أجهزة مادية حقيقية حيث يكون للناقلات المختلفة عرض نطاق ترددي (bandwidth) مختلف كما تمت مناقشته في القسم 13.4. خذ بعين الاعتبار خادم GPU حقيقي رباعي الاتجاهات كما هو موضح في الشكل 13.7.2. إذا كان متصلاً جيدًا بشكل خاص، فقد يحتوي على بطاقة شبكة 100 جيجابت. تقع الأرقام الأكثر شيوعًا في النطاق من 1 إلى 10 جيجابت إيثرنت مع عرض نطاق ترددي فعال من 100 ميجابايت / ثانية إلى 1 جيجابايت /

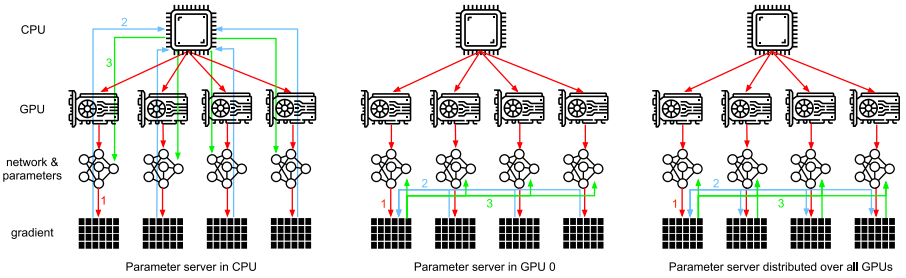
ثانية. نظرًا لأن وحدات المعالجة المركزية لديها عدد قليل جدًا من ممرات (مسارات) PCIe للاتصال بجميع وحدات معالجة الرسومات مباشرة (على سبيل المثال ، تحتوي وحدات المعالجة المركزية Intel المخصصة للمستهلكين على 24 مسارًا) ، فنحن بحاجة إلى مجمع multiplexer. عرض النطاق الترددي من وحدة المعالجة المركزية على ارتباط x Gen316 هو 16 جيجابايت / ثانية. هذه أيضًا السرعة التي يتم بها توصيل كل وحدة من وحدات معالجة الرسومات بالمبدل switch. هذا يعني أنه أكثر فعالية للتواصل بين الأجهزة.



الشكل 13.7.2 خادم GPU 4-way.

من أجل النقاش ، دعنا نفترض أن الانحدارات هي 160 ميغا بايت. في هذه الحالة ، يستغرق الأمر 30 مللي ثانية لإرسال الانحدارات من جميع وحدات معالجة الرسومات الثلاثة المتبقية إلى وحدة المعالجة الرسومية الرابعة (تستغرق كل عملية نقل 10 مللي ثانية = 160 ميجابايت / 16 جيجابايت / ثانية). بإضافة 30 مللي ثانية أخرى لنقل متجهات الوزن مرة أخرى ، نصل إلى إجمالي 60 مللي ثانية. إذا أرسلنا جميع البيانات إلى وحدة المعالجة المركزية ، فإننا نتحمل عقوبة قدرها 40 مللي ثانية نظرًا لأن كل وحدة من وحدات معالجة الرسومات الأربعة تحتاج إلى إرسال البيانات إلى وحدة المعالجة المركزية ، مما ينتج عنه إجمالي 80 مللي ثانية. افترض أخيرًا أننا قادرون على تقسيم الانحدارات إلى 4 أجزاء كل منها 40 ميجابايت. يمكننا الآن تجميع كل جزء من الأجزاء على وحدة معالجة رسومات مختلفة في وقت واحد لأن مفتاح PCIe يوفر عملية عرض نطاق ترددي كامل بين جميع الروابط. بدلاً من 30 مللي ثانية ، يستغرق هذا 7.5 مللي ثانية ، مما ينتج عنه إجمالي 15 مللي ثانية لعملية التزامن. باختصار ، بناءً على كيفية مزامنة

المعلمات ، يمكن أن تستغرق العملية نفسها من 15 مللي ثانية إلى 80 مللي ثانية. يوضح الشكل 13.7.3 الاستراتيجيات المختلفة لتبادل المعلمات.



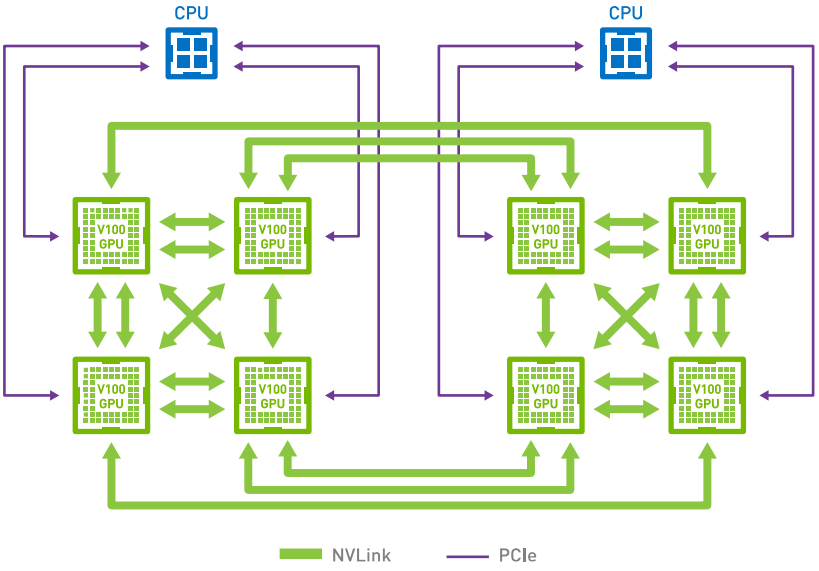
الشكل 13.7.3 استراتيجيات تزامن المعلمات.

لاحظ أن لدينا أداة أخرى تحت تصرفنا عندما يتعلق الأمر بتحسين الأداء: في شبكة عميقة يستغرق الأمر بعض الوقت لحساب جميع الانحدارات من الأعلى إلى الأسفل. يمكننا البدء في مزامنة الانحدارات لبعض مجموعات المعلمات حتى ونحن ما زلنا مشغولين بحسابها للآخرين. انظر على سبيل المثال ، Sergeev و Del Balso (2018) للحصول على تفاصيل حول كيفية القيام بذلك في Horovod.

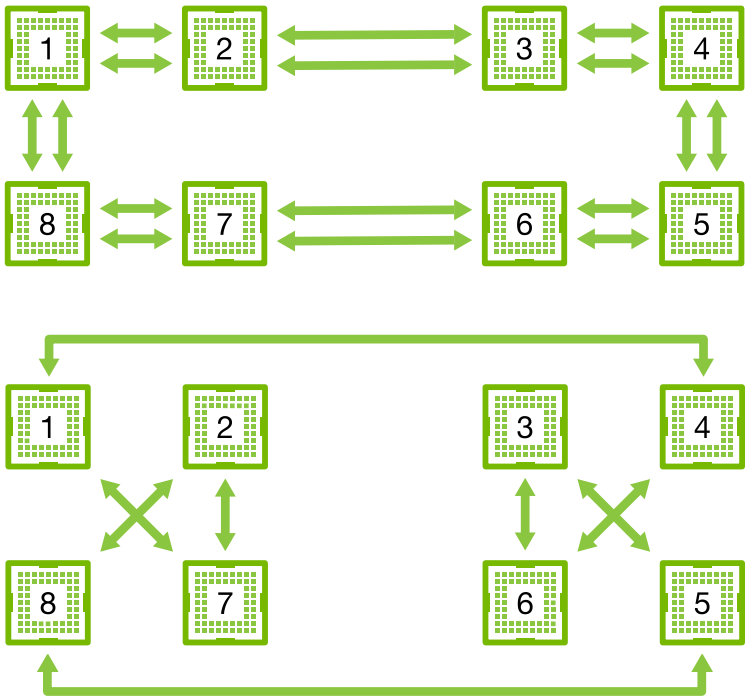
13.7.2. مزامنة الحلقة Ring Synchronization

عندما يتعلق الأمر بالمزامنة على أجهزة التعلم العميق الحديثة ، فإننا غالبًا ما نواجه اتصال شبكة مفصل بشكل كبير. على سبيل المثال ، تشترك مثيلات AWS p3.16xlarge و NVIDIA DGX-2 في بُنية الاتصال بالشكل 13.7.4. تتصل كل وحدة معالجة رسومات (GPU) بوحدة معالجة مركزية مضيفة عبر ارتباط PCIe الذي يعمل في أحسن الأحوال بسرعة 16 جيجابايت/ثانية. بالإضافة إلى ذلك ، تحتوي كل وحدة معالجة رسومات أيضًا على 6 اتصالات NVLink ، كل منها قادر على نقل 300 جيجابت/ثانية ثنائي الاتجاه. هذا يصل إلى حوالي 18 جيجا بايت/ثانية لكل رابط لكل اتجاه. باختصار ، عرض النطاق الترددي الكلي NVLink أعلى بكثير من عرض النطاق الترددي لـ PCIe. السؤال هو كيفية استخدامه بأكبر قدر من الكفاءة.

اتضح أن استراتيجية المزامنة المثلى هي تفكيك الشبكة إلى حلقتين two rings واستخدامهما لمزامنة البيانات مباشرة (Wang et al.، 2018). يوضح الشكل 13.7.5 أن الشبكة يمكن أن تتحلل إلى حلقة واحدة (1–2–3–4–5–6–7–8–1) بعرض نطاق ترددي مزدوج NVLink وفي حلقة واحدة (1–4–6–3– 5–8–2–7–1) مع عرض النطاق الترددي العادي. تصميم بروتوكول مزامنة فعال في هذه الحالة هو أمر غير بديهي.

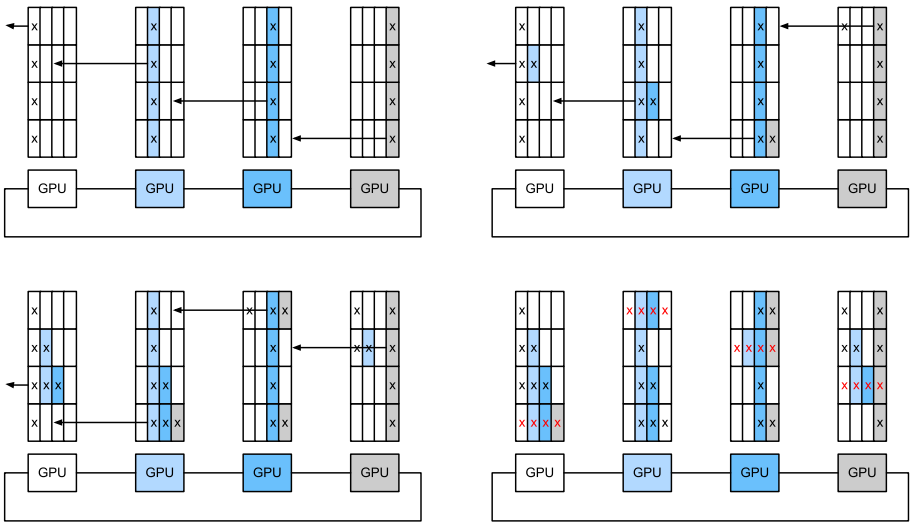


الشكل 13.7.4. اتصال NVLink على 8 خوادم GPU V100 (الصورة مقدمة من NVIDIA).



الشكل 13.7.5 تحلل شبكة NVLink إلى حلقتين.

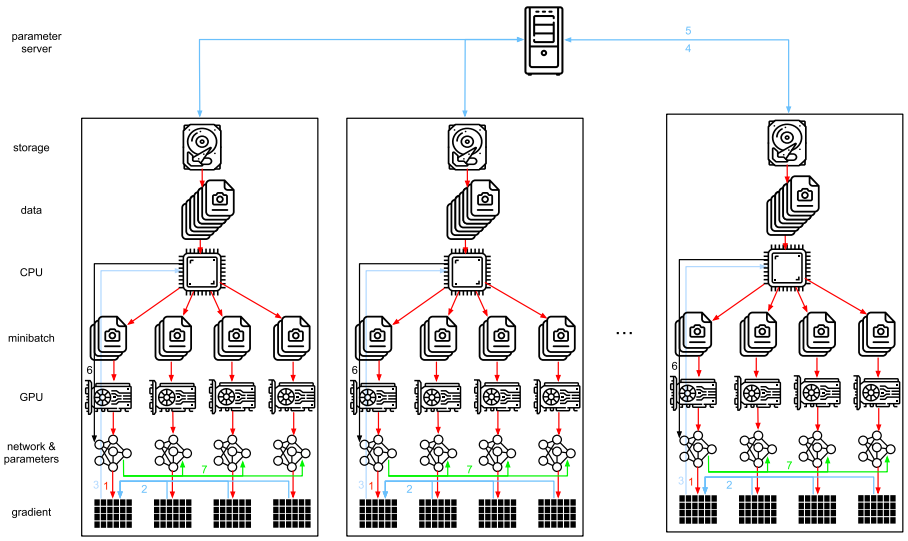
ضع في اعتبارك التجربة الفكرية التالية: بالنظر إلى حلقة من n عقد الحوسبة (أو وحدات معالجة الرسومات) ، يمكننا إرسال الانحدارات من العقدة الأولى إلى العقدة الثانية. هناك تتم إضافته إلى الانحدار المحلي وإرساله إلى العقدة الثالثة ، وهكذا. بعد $n - 1$ خطوات يمكن العثور على الانحدار الكلي في العقدة الأخيرة التي تمت زيارتها. أي أن وقت تجميع الانحدارات ينمو بشكل خطي مع عدد العقد. ولكن إذا فعلنا ذلك ، فإن الخوارزمية غير فعالة تمامًا. بعد كل شيء ، في أي وقت هناك واحدة فقط من العقد تتواصل. ماذا لو قسمنا الانحدارات إلى n قطع (chunks) وبدأنا في مزامنة القطعة i بدءًا من العقدة؟ نظرًا لأن كل قطعة لها حجم $1/n$ ، فإن الوقت الإجمالي هو الآن $(n - 1)/n \approx 1$. بمعنى آخر ، الوقت الذي يقضيه في تجميع الانحدارات لا ينمو does not grow كلما زاد حجم الحلقة. هذه نتيجة مذهلة للغاية. يوضح الشكل 13.7.6 تسلسل الخطوات على العقد $n = 4$.



الشكل. 13.7.6 تزامن الحلقة Ring synchronization عبر 4 عقد. تبدأ كل عقدة في نقل أجزاء من الانحدارات إلى جارتها اليسرى حتى يمكن العثور على الانحدار المجمع في جارتها اليمنى.

إذا استخدمنا نفس المثال لمزامنة 160 ميجابايت عبر 8 وحدات معالجة رسومات V100 ، فسنصل تقريبًا الى $2 \cdot 160\text{MB}/(3 \cdot 18\text{GB/s}) \approx 6\text{ms}$. هذا أفضل من استخدام ناقل PCIe ، على الرغم من أننا نستخدم الآن 8 وحدات معالجة رسومات. لاحظ أن هذه الأرقام أسوأ قليلاً من الناحية العملية ، نظرًا لأن أطر التعلم العميق غالبًا ما تفشل في تجميع الاتصالات في عمليات نقل متقطعة كبيرة.

لاحظ أن هناك فكرة خاطئة شائعة مفادها أن مزامنة الحلقة ring synchronization تختلف اختلافًا جوهريًا عن خوارزميات المزامنة الأخرى. الاختلاف الوحيد هو أن مسار المزامنة أكثر تفصيلاً إلى حد ما عند مقارنته بشجرة بسيطة.



الشكل. 13.7.7 تدريب متوازي موزع متعدد الآلات متعدد المعالجات الرسومية.

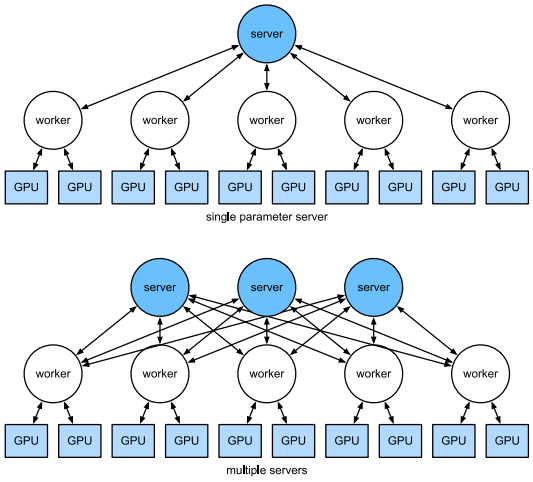
13.7.3. تدريب متعدد الآلات Multi-Machine Training

يضيف التدريب الموزع Distributed training على أجهزة متعددة تحديًا إضافيًا: نحتاج إلى التواصل مع الخوادم المتصلة فقط عبر نسيج نطاق ترددي أقل نسبيًا يمكن أن يكون أبطأ في بعض الحالات. المزامنة عبر الأجهزة صعبة. بعد كل شيء ، سيكون للآلات المختلفة التي تقوم بتشغيل كود التدريب سرعة مختلفة بمهارة. ومن ثم نحتاج إلى مزامنتها synchronize إذا أردنا استخدام التحسين الموزع المتزامن. يوضح الشكل 13.7.7 كيفية حدوث التدريب الموازي الموزع distributed parallel training.

1. تتم قراءة مجموعة (مختلفة) من البيانات على كل جهاز ، وتقسيمها عبر وحدات معالجة رسومات متعددة ونقلها إلى ذاكرة وحدة معالجة الرسومات. هناك تنبؤات وانحدارات يتم حسابها على كل دفعة GPU على حدة.
2. يتم تجميع الانحدارات من جميع وحدات معالجة الرسومات المحلية في وحدة معالجة رسومات واحدة (أو يتم تجميع أجزاء منها عبر وحدات معالجة رسومات مختلفة).
3. يتم إرسال الانحدارات إلى وحدات المعالجة المركزية (CPU).

4. ترسل وحدات المعالجة المركزية الانحدارات إلى خادم معلمات مركزي يجمع كل الانحدارات.
5. ثم يتم استخدام الانحدارات الكلية لتحديث المعلمات ويتم بث المعلمات المحدثة مرة أخرى إلى وحدات المعالجة المركزية الفردية.
6. يتم إرسال المعلومات إلى وحدة معالجة رسومات واحدة (أو متعددة).
7. تنتشر المعلمات المحدثة عبر جميع وحدات معالجة الرسومات.

تبدو كل من هذه العمليات مباشرة نوعًا ما. وبالفعل ، يمكن تنفيذها بكفاءة داخل آلة واحدة. بمجرد أن ننظر إلى أجهزة متعددة ، يمكننا أن نرى أن خادم المعلمات المركزية يصبح عنق الزجاجة. بعد كل شيء ، عرض النطاق الترددي لكل خادم محدود ، وبالتالي بالنسبة لـ m عاملين، فإن الوقت الذي يستغرقه إرسال جميع الانحدارات إلى الخادم هو $\mathcal{O}(m)$. يمكننا اختراق هذا الحاجز عن طريق زيادة عدد الخوادم إلى n. في هذه المرحلة ، يحتاج كل خادم فقط إلى تخزين $\mathcal{O}(1/n)$ من المعلمات ، ومن ثم يصبح الوقت الإجمالي للتحديثات والتحسين $\mathcal{O}(m/n)$. تؤدي مطابقة كلا الرقمين إلى قياس ثابت بغض النظر عن عدد العمال الذين نتعامل معهم. من الناحية العملية ، نستخدم نفس الأجهزة كعمال وخوادم. يوضح الشكل 13.7.8 التصميم (انظر أيضًا (Li et al.، 2014) للحصول على التفاصيل). على وجه الخصوص ، فإن ضمان عمل العديد من الأجهزة دون تأخيرات غير معقولة ليس بالأمر الهين. نحذف التفاصيل المتعلقة بالحواجز ولن نتطرق لفترة وجيزة إلا إلى التحديثات المتزامنة وغير المتزامنة أدناه.



الشكل 13.7.8 في الاعلى: الخادم ذو المعلمة الواحدة هو عنق الزجاجة لأن عرض النطاق الخاص به محدود. في الأسفل: تخزن الخوادم متعددة المعلمات أجزاءً من المعلمات بنطاق ترددي إجمالي.

13.7.4. مخازن المفتاح والقيمة Key—Value Stores

تنفيذ الخطوات المطلوبة للتدريب الموزع على وحدات معالجة الرسومات المتعددة في الممارسة العملية أمر غير بديهي. هذا هو السبب في أنه من المفيد استخدام تجريد مشترك ، أي مخزن القيمة المفتاح -key

value store مع إعادة تعريف دلالات التحديث.

عبر العديد من العاملين والعديد من وحدات معالجة الرسومات ، يمكن تعريف حساب الانحدار i على أنه

$$\mathbf{g}_i = \sum_{k \in \text{workers}} \sum_{j \in \text{GPUs}} \mathbf{g}_{ijk},$$

حيث $\mathbf{g}_{ijk}$ هو جزء من انقسام الانحدار i على GPU للعامل k. يتمثل الجانب الرئيسي في هذه العملية في أنها تخفيض تبادلي commutative reduction، أي أنها تحول العديد من المتجهات إلى متجه واحد ولا يهم ترتيب تطبيق العملية. يعد هذا أمرًا رائعًا لأغراضنا نظرًا لأننا لا (نحتاج) إلى التحكم الدقيق في وقت تلقي الانحدار. الى جانب ذلك ، لاحظ أن هذه العملية مستقلة بين مختلف i.

يتيح لنا ذلك تحديد العمليتين التاليتين: الدفع push ، الذي يراكم الانحدارات، والسحب pull، الذي يسترد الانحدارات الكلية. نظرًا لأن لدينا مجموعات مختلفة من الانحدارات (بعد كل شيء ، لدينا العديد من الطبقات) ، نحتاج إلى فهرسة الانحدارات باستخدام مفتاح. هذا التشابه مع مخازن القيمة_المفتاح ، مثل ذلك الذي تم تقديمه في Dynamo، (DeCandia et al.2007) ليس من قبيل الصدفة. كما أنها تلبي العديد من الخصائص المتشابهة ، لا سيما عندما يتعلق الأمر بتوزيع المعلمات عبر خوادم متعددة.

يتم وصف عمليات الدفع والسحب لمخازن القيمة_المفتاح key-value stores على النحو التالي:

- push(key, value) يرسل انحداراً معينًا (القيمة) من عامل إلى مخزن مشترك. هناك يتم تجميع القيمة ، على سبيل المثال ، من خلال تلخيصها.
- pull(key, value) يسترد قيمة إجمالية من التخزين المشترك ، على سبيل المثال ، بعد الجمع بين الانحدارات من جميع العمال.

من خلال إخفاء كل التعقيدات المتعلقة بالمزامنة وراء عملية الدفع والسحب البسيطة ، يمكننا فصل اهتمامات المصممين الإحصائيين الذين يريدون أن يكونوا قادرين على التعبير عن التحسين بعبارات بسيطة ومهندسي النظام الذين يحتاجون إلى التعامل مع التعقيد المتأصل في المزامنة الموزعة distributed synchronization.

13.7.5. الملخص

- يجب أن تكون المزامنة شديدة التكيف مع البُنية التحتية للشبكة المحددة والاتصال داخل الخادم. يمكن أن يحدث هذا فرقًا كبيرًا في الوقت الذي تستغرقه المزامنة.
- يمكن أن تكون مزامنة الحلقة Ring-synchronization هي الأمثل لخوادم p3 و DGX-2. للآخرين ربما ليس كثيرًا.
- تعمل استراتيجية المزامنة الهرمية hierarchical synchronization بشكل جيد عند إضافة خوادم متعددة المعلمات لزيادة عرض النطاق الترددي.

13.7.6. التمارين

1. هل يمكنك زيادة تزامن الحلقة أكثر من ذلك؟ تلميح: يمكنك إرسال رسائل في كلا الاتجاهين.
2. هل من الممكن السماح بالاتصال غير المتزامن (بينما الحساب لا يزال جاريًا)؟ كيف تؤثر على الأداء؟
3. ماذا لو فقدنا خادمًا أثناء عملية حسابية طويلة الأمد؟ كيف يمكننا تصميم آلية للتسامح مع الخطأ fault tolerance mechanism لتجنب إعادة تشغيل الحساب بالكامل؟

14

الرؤية الحاسوبية

14. الرؤية الحاسوبية Computer Vision

سواء كان التشخيص الطبي، أو المركبات ذاتية القيادة، أو مراقبة الكاميرا، أو المرشحات الذكية، فإن العديد من التطبيقاتفي مجال الرؤية الحاسوبية computer vision ترتبط ارتباطًا وثيقًا بحياتنا الحالية والمستقبلية.في السنوات الأخيرة، كان التعلم العميق هو القوة التحويلية للنهوض بأداء أنظمة الرؤية الحاسوبية. يمكن القول إن تطبيقات الرؤية الحاسوبية الأكثر تقدمًا تكاد لا تنفصل عن التعلم العميق.في ضوء ذلك، سيركز هذا الفصل على مجال الرؤية الحاسوبية، ويبحث في الأساليب والتطبيقات التي كان لها مؤخرًا تأثيرفي الأوساط الأكاديمية والصناعية.

في القسم 7 والقسم 8، درسنا العديد من الشبكات العصبية التلافيفية CNN التي يشيع استخدامهافي الرؤية الحاسوبية، وقمنا بتطبيقها على مهام بسيطة لتصنيف الصور.في بداية هذا الفصل، سنصف طريقتين من الممكن أن تحسن تعميم النموذج، وهما زيادة الصورة image augmentation والضبط الدقيق fine-tuning، وتطبيقها على تصنيف الصور. نظرًا لأن الشبكات العصبية العميقة يمكن أن تمثل الصور بشكل فعالفي مستويات متعددة، فقد تم استخدام هذه التمثيلات الطبقية layerwise representations بنجاحفي العديد من مهام الرؤية الحاسوبية مثل اكتشاف الكائنات object detection والتجزئة الدلالية semantic segmentation ونقل النمط style transfer. باتباع الفكرة الرئيسية المتمثلةفي الاستفادة من التمثيلات الطبقيةفي الرؤية الحاسوبية، سنبدأ بالمكونات والتقنيات الرئيسية لاكتشاف الكائنات. بعد ذلك، سوف نوضح كيفية استخدام الشبكات التلافيفية بالكامل للتجزئة الدلالية للصور. ثم نشرح كيفية استخدام تقنيات نقل الأنماط لإنشاء صور مثل غلاف هذا الكتاب.في النهاية، نختتم هذا الفصل بتطبيق مواد هذا الفصل والعديد من الفصول السابقة على مجموعتي بيانات رائعتين لقياس الرؤية الحاسوبية.

14.1. زيادة الصورة Image Augmentation

في القسم 8.1، ذكرنا أن مجموعات البيانات الكبيرة هي شرط أساسي لنجاح الشبكات العصبية العميقةفي التطبيقات المختلفة. يولد زيادة الصورة image augmentation أمثلة تدريبية متشابهة ولكنها مميزة بعد سلسلة من التغييرات العشوائية على صور التدريب، وبالتالي توسيع حجم مجموعة التدريب. بدلاً من ذلك، يمكن أن يكون الدافع وراء زيادة الصورة هو حقيقة أن التعديلات العشوائية لأمثلة التدريب تسمح للنماذج بتقليل الاعتماد على سمات معينة، وبالتالي تحسين قدرتها على التعميم. على سبيل المثال، يمكننا قص صورة بطرق مختلفة لجعل موضوع الاهتمام يظهرفي مواضع مختلفة، وبالتالي تقليل اعتماد النموذج على موضع الكائن. يمكننا أيضًا ضبط عوامل مثل السطوع واللون لتقليل حساسية النموذج للون. ربما يكون صحيحًا أن زيادة

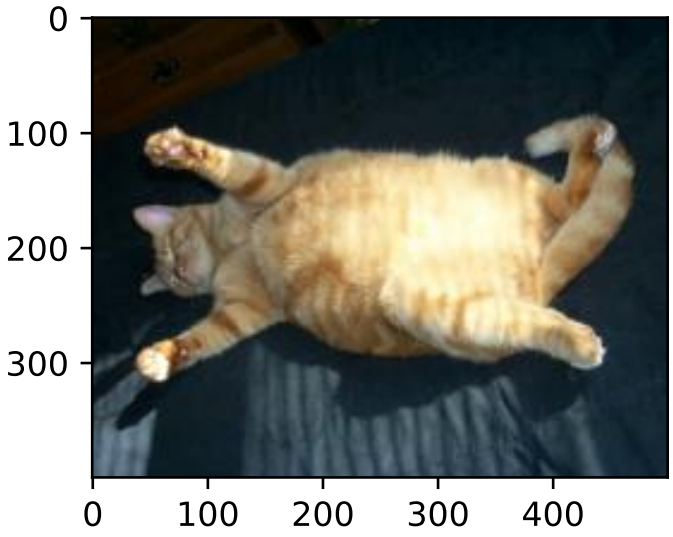
الصورة كان لا غنى عنه لنجاح AlexNet في ذلك الوقت. سنناقش في هذا القسم هذه التقنية المستخدمة على نطاق واسع في الرؤية الحاسوبية.

```
%matplotlib inline
from mxnet import autograd, gluon, image, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

14.1.1. طرق زيادة الصورة الشائعة Common Image Augmentation Methods

في بحثنا عن طرق زيادة الصورة الشائعة، سنستخدم الصورة التالية 500 × 400 كمثال.

```
d2l.set_figsize()
img = image.imread('../img/cat1.jpg')
d2l.plt.imshow(img.asnumpy());
```



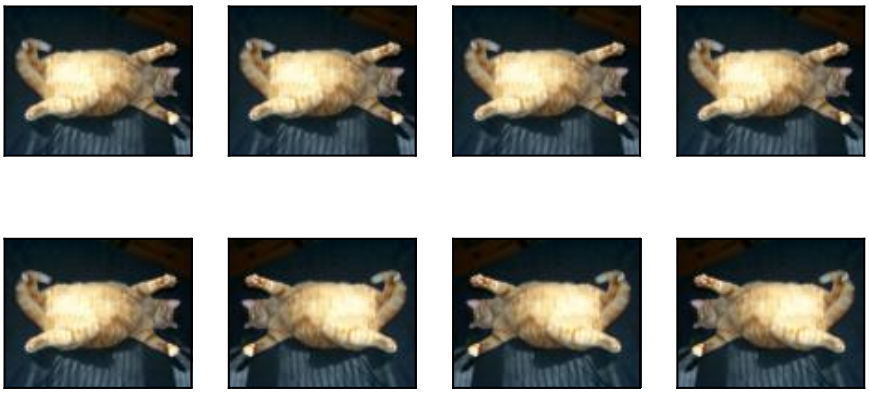
تتمتع معظم طرق زيادة الصور بدرجة معينة من العشوائية. لتسهيل ملاحظة تأثير زيادة الصورة، نقوم بعد ذلك بتعريف الدالة الإضافية التي يتم تطبيقها. تعمل هذه الدالة على تشغيل طريقة زيادة الصورة عدة مرات على صورة الإدخال `img` وتظهر جميع النتائج.

```
def apply(img, aug, num_rows=2, num_cols=4, scale=1.5):
    Y = [aug(img) for _ in range(num_rows * num_cols)]
    d2l.show_images(Y, num_rows, num_cols, scale=scale)
```

14.1.1.1. التقليب والاقتصاص Flipping and Cropping

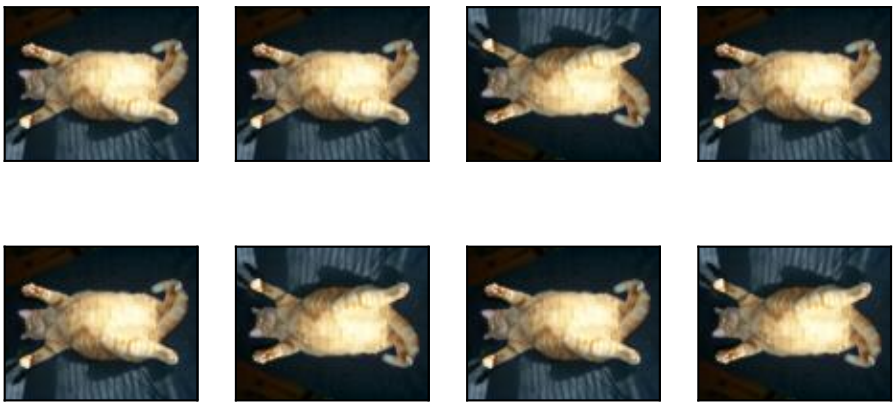
عادةً لا يؤدي قلب Flipping الصورة إلى اليسار واليمين إلى تغيير فئة الكائن. هذه واحدة من أقدم الطرق وأكثرها استخدامًا لزيادة الصورة. بعد ذلك، نستخدم وحدة `transforms` لإنشاء مثيل `RandomFlipLeftRight` ، والذي يقلب الصورة إلى اليسار واليمين مع فرصة بنسبة 50٪.

```
apply(img,
gluon.data.vision.transforms.RandomFlipLeftRight())
```



التقليب لأعلى ولأسفل ليس شائعًا مثل التقليب لليسار واليمين. ولكن على الأقل بالنسبة لهذه الصورة النموذجية، لا يؤدي التقليب لأعلى ولأسفل إلى إعاقة التعرف. بعد ذلك، نقوم بإنشاء مثيل `RandomFlipTopBottom` لقلب الصورة لأعلى ولأسفل مع فرصة بنسبة 50٪.

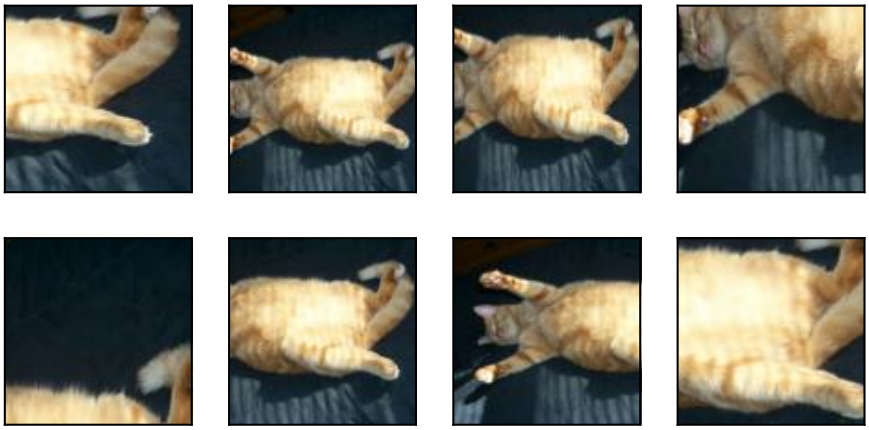
```
apply(img,
gluon.data.vision.transforms.RandomFlipTopBottom())
```



في مثال الصورة التي استخدمناها، يوجد القطفي منتصف الصورة، ولكن قد لا يكون هذا هو الحال بشكل عام.في القسم 7.5، أوضحنا أن طبقة التجميع pooling layer يمكن أن تقلل من حساسية الطبقة التلافيفية convolutional layer للموضع المستهدف. بالإضافة إلى ذلك، يمكننا أيضًا قص crop الصورة بشكل عشوائي لجعل الكائنات تظهرفي مواضع مختلفةفي الصورة بمقاييس مختلفة، مما قد يقلل أيضًا من حساسية النموذج إلى الموضع المستهدف.

في الكود أدناه، نقوم بشكل عشوائي بقص منطقة بمساحة 100% ~ 10% من المنطقة الأصلية في كل مرة، ويتم تحديد نسبة العرض إلى الارتفاع لهذه المنطقة بشكل عشوائي من 2 ~ 0.5. بعد ذلك، يتم قياس عرض المنطقة وارتفاعها إلى 200 بكسل. ما لم يتم تحديد خلاف ذلك، يشير الرقم العشوائي الموجود بين a وb في هذا القسم إلى قيمة مستمرة تم الحصول عليها عن طريق أخذ عينات عشوائية وموحدة من الفاصل $[a, b]$.

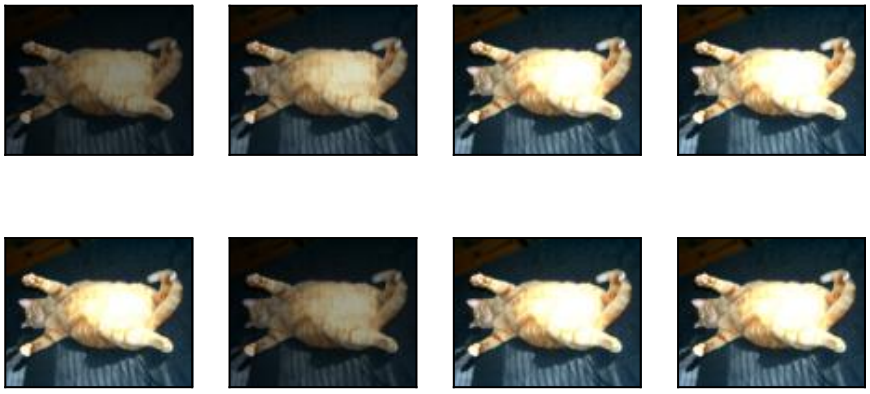
```
shape_aug =
gluon.data.vision.transforms.RandomResizedCrop(
    (200, 200), scale=(0.1, 1), ratio=(0.5, 2))
apply(img, shape_aug)
```



14.1.1.2. تغيير الألوان Changing Colors

طريقة زيادة أخرى هي تغيير الألوان changing colors. يمكننا تغيير أربعة جوانب من لون الصورة: السطوع brightness والتباين contrast والتشبع saturation وتدرج اللون hue.في المثال أدناه، قمنا بتغيير سطوع الصورة بشكل عشوائي إلى قيمة تتراوح بين 50٪ (0.5 − 1) و150٪ (0.5 + 1) من الصورة الأصلية.

```
apply(img,
gluon.data.vision.transforms.RandomBrightness(0.5))
```

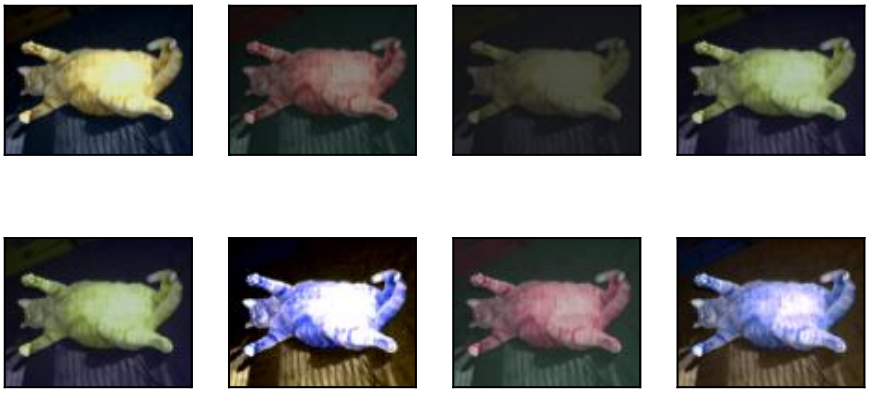


وبالمثل، يمكننا تغيير لون الصورة بشكل عشوائي.

```
apply(img, gluon.data.vision.transforms.RandomHue(0.5))
```

يمكننا أيضًا إنشاء مثيل `RandomColorJitter` وتعيين كيفية تغيير سطوع الصورة وتباينها وتشبعها وتدرجها بشكل عشوائيفي نفس الوقت.

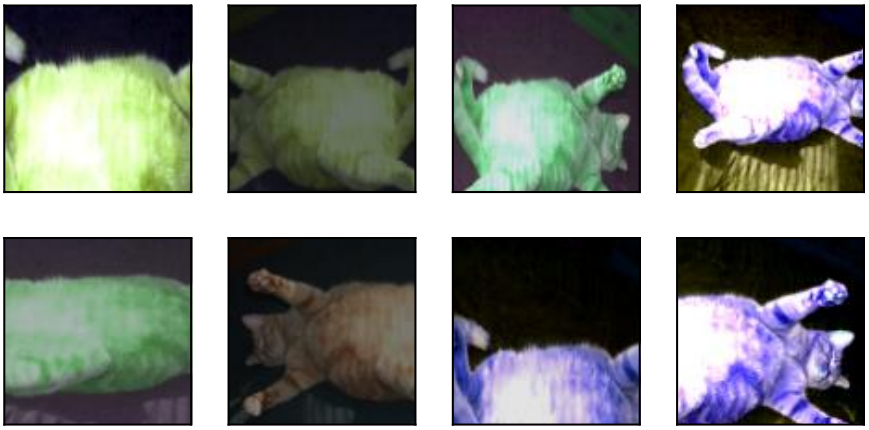
```
color_aug =
gluon.data.vision.transforms.RandomColorJitter(
    brightness=0.5, contrast=0.5, saturation=0.5,
hue=0.5)
apply(img, color_aug)
```



14.1.1.3. الجمع بين طرق زيادة الصور المتعددة Combining Multiple Image Augmentation Methods

في الممارسة العملية، سوف نقوم بدمج طرق متعددة لزيادة الصورة. على سبيل المثال، يمكننا دمج طرق زيادة الصور المختلفة المحددة أعلاه وتطبيقها على كل صورة عبر مثيل `Compose` .

```
augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.RandomFlipLeftRight(),
color_aug, shape_aug])
apply(img, augs)
```

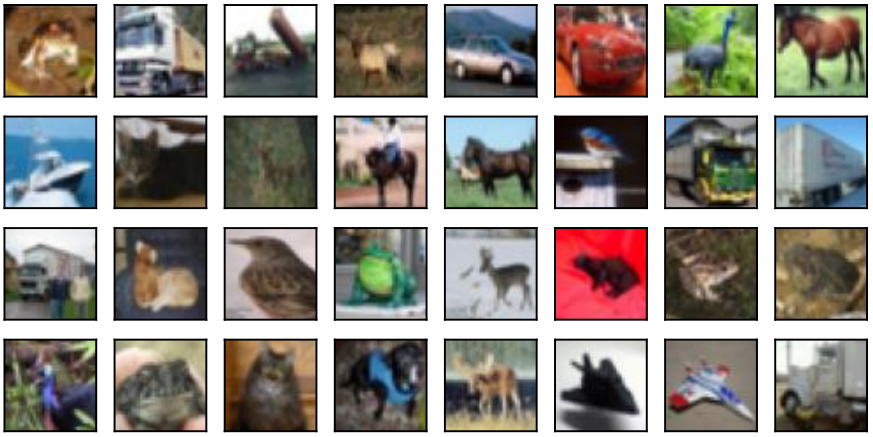


14.1.2. التدريب مع زيادة الصورة Training with Image Augmentation

دعنا ندرب نموذجًا مع زيادة الصورة. هنا نستخدم مجموعة بيانات CIFAR-10 بدلاً من مجموعة بيانات Fashion-MNIST التي استخدمناها من قبل. هذا لأن موضع وحجم

الكائناتفي مجموعة بيانات Fashion-MNIST قد تم تسويتها،في حين أن لون وحجم الكائناتفي مجموعة بيانات CIFAR-10 لهما اختلافات أكثر أهمية. يتم عرض أول 32 صورة تدريبيةفي مجموعة بيانات CIFAR-10 أدناه.

```
d2l.show_images(gluon.data.vision.CIFAR10(
    train=True)[:32][0], 4, 8, scale=0.8);
```



من أجل الحصول على نتائج نهائية أثناء التنبؤ، عادة ما نطبق فقط زيادة الصورة على أمثلة التدريب، ولا نستخدم زيادة الصورة مع العمليات العشوائية أثناء التنبؤ. هنا نستخدم فقط أبسط طريقة عشوائية للقلب من اليسار إلى اليمين. بالإضافة إلى ذلك، نستخدم مثيل `ToTensor` لتحويل مجموعة صغيرة من الصور إلى التنسيق الذي يتطلبه إطار عمل التعلم العميق ، أي أرقام الفاصلة العائمة 32 بت بين 0 و 1 مع شكل (حجم الدُفعة ، عدد القنوات ، الارتفاع ، العرض).

```
train_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.RandomFlipLeftRight(),
    gluon.data.vision.transforms.ToTensor()])

test_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.ToTensor()])
```

بعد ذلك، نحدد دالة مساعدة لتسهيل قراءة الصورة وتطبيق زيادة الصورة. تعمل دالة `transform_first` التي توفرها مجموعات بيانات Gluon على تطبيق زيادة الصورة على العنصر الأول من كل مثال تدريبي (الصورة والتسمية image and label) ، أي الصورة. للحصول على مقدمة مفصلة عن `DataLoader`، يرجى الرجوع إلى القسم 4.2.

```
def load_cifar10(is_train, augs, batch_size):
    return gluon.data.DataLoader(
```

```
gluon.data.vision.CIFAR10(train=is_train).transform_firs
t(augs),
        batch_size=batch_size, shuffle=is_train,
        num_workers=d2l.get_dataloader_workers())
```

14.1.2.1. تدريب GPU متعدد Multi-GPU Training

نقوم بتدريب نموذج ResNet-18 من القسم 8.6في مجموعة بيانات CIFAR-10. راجع مقدمة التدريب على وحدات معالجة الرسومات المتعددةفي القسم 13.6. فيما يلي، نحدد دالة لتدريب النموذج وتقييمه باستخدام وحدات معالجة رسومات متعددة.

```
#@save
def train_batch_ch13(net, features, labels, loss,
trainer, devices,
                     split_f=d2l.split_batch):
    """Train for a minibatch with multiple GPUs (defined
in Chapter 13)."""
    X_shards, y_shards = split_f(features, labels,
devices)
    with autograd.record():
        pred_shards = [net(X_shard) for X_shard in
X_shards]
        ls = [loss(pred_shard, y_shard) for pred_shard,
y_shard
              in zip(pred_shards, y_shards)]
    for l in ls:
        l.backward()
    # The `True` flag allows parameters with stale
gradients, which is useful
    # later (e.g., in fine-tuning BERT)
    trainer.step(labels.shape[0],
ignore_stale_grad=True)
    train_loss_sum = sum([float(l.sum()) for l in ls])
    train_acc_sum = sum(d2l.accuracy(pred_shard,
y_shard)
                        for pred_shard, y_shard in
zip(pred_shards, y_shards))
    return train_loss_sum, train_acc_sum

#@save
def train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs,
```

```
                devices=d2l.try_all_gpus(),
split_f=d2l.split_batch):
    """Train a model with multiple GPUs (defined in
Chapter 13)."""
    timer, num_batches = d2l.Timer(), len(train_iter)
    animator = d2l.Animator(xlabel='epoch', xlim=[1,
num_epochs], ylim=[0, 1],
                            legend=['train loss', 'train
acc', 'test acc'])
    for epoch in range(num_epochs):
        # Sum of training loss, sum of training
accuracy, no. of examples,
        # no. of predictions
        metric = d2l.Accumulator(4)
        for i, (features, labels) in
enumerate(train_iter):
            timer.start()
            l, acc = train_batch_ch13(
                net, features, labels, loss, trainer,
devices, split_f)
            metric.add(l, acc, labels.shape[0],
labels.size)
            timer.stop()
            if (i + 1) % (num_batches // 5) == 0 or i ==
num_batches - 1:
                animator.add(epoch + (i + 1) /
num_batches,
                             (metric[0] / metric[2],
metric[1] / metric[3],
                              None))
        test_acc = d2l.evaluate_accuracy_gpus(net,
test_iter, split_f)
        animator.add(epoch + 1, (None, None, test_acc))
    print(f'loss {metric[0] / metric[2]:.3f}, train acc
'
          f'{metric[1] / metric[3]:.3f}, test acc
{test_acc:.3f}')
    print(f'{metric[2] * num_epochs / timer.sum():.1f}
examples/sec on '
          f'{str(devices)}')
```

الآن يمكننا تحديد دالة `train_with_data_aug` لتدريب النموذج باستخدام زيادة الصورة. تحصل هذه الدالة على جميع وحدات معالجة الرسومات المتاحة، وتستخدم Adam

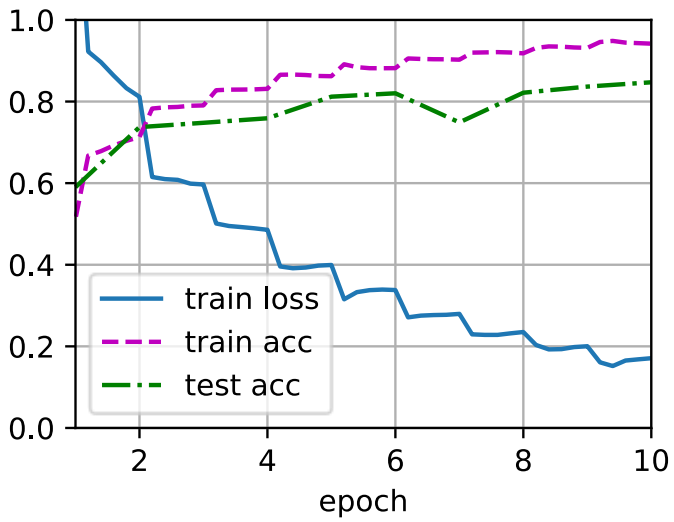
كخوارزمية تحسين، وتطبق زيادة الصورة على مجموعة بيانات التدريب، وتستدعي أخيرًا دالة `train_ch13` التي تم تحديدها للتو لتدريب النموذج وتقييمه.

```
batch_size, devices, net = 256, d2l.try_all_gpus(),
d2l.resnet18(10)
net.initialize(init=init.Xavier(), ctx=devices)

def train_with_data_aug(train_augs, test_augs, net,
lr=0.001):
    train_iter = load_cifar10(True, train_augs,
batch_size)
    test_iter = load_cifar10(False, test_augs,
batch_size)
    loss = gluon.loss.SoftmaxCrossEntropyLoss()
    trainer = gluon.Trainer(net.collect_params(),
'adam',
                            {'learning_rate': lr})
    train_ch13(net, train_iter, test_iter, loss,
trainer, 10, devices)
```

دعونا ندرب النموذج باستخدام زيادة الصورة على أساس التقليب العشوائي من اليسار إلى اليمين random left-right flipping.

```
train_with_data_aug(train_augs, test_augs, net)
```

```
loss 0.171, train acc 0.942, test acc 0.847
4050.4 examples/sec on [gpu(0), gpu(1)]
```



14.1.3. الملخص

- يولد زيادة الصورة Image augmentation صورًا عشوائية بناءً على بيانات التدريب الموجودة لتحسين قدرة التعميم للنماذج.
- من أجل الحصول على نتائج نهائية أثناء التنبؤ، عادة ما نطبق فقط زيادة الصورة على أمثلة التدريب، ولا نستخدم زيادة الصورة مع العمليات العشوائية أثناء التنبؤ.
- توفر أطر التعلم العميق العديد من طرق زيادة الصور المختلفة، والتي يمكن تطبيقهافي وقت واحد.

14.1.4. التمارين

1. درب النموذج دون استخدام زيادة الصورة: `train_with_data_aug(test_augs, test_augs)`. قارن بين دقة التدريب والاختبار عند استخدام زيادة الصورة وعند عدم استخدامها. هل يمكن لهذه التجربة المقارنة أن تدعم الحجة القائلة بأن زيادة الصورة يمكن أن يخفف من فرط التعلم overfitting؟ لماذا؟
2. اجمع بين عدة طرق مختلفة لزيادة الصورفي تدريب النموذج على مجموعة بيانات CIFAR-10. هل يحسن دقة الاختبار؟
3. ارجع إلى التوثيق عبر الإنترنت لإطار عمل التعلم العميق. ما هي طرق زيادة الصور الأخرى التي يوفرها أيضًا؟

14.2. الضبط الدقيق Fine-Tuning

في الفصول السابقة، ناقشنا كيفية تدريب النماذج على مجموعة بيانات Fashion-MNIST التدريبية باستخدام 60000 صورة فقط. وصفنا أيضًا ImageNet، مجموعة بيانات الصور واسعة النطاق الأكثر استخدامًافي الأوساط الأكاديمية، والتي تحتوي على أكثر من 10 ملايين صورة و1000 عنصر. ومع ذلك، فإن حجم مجموعة البيانات التي نواجهها عادة يكون بين حجم مجموعتي البيانات.

افترض أننا نريد التعرف على أنواع مختلفة من الكراسي من الصور، ثم نوصي المستخدمين بروابط الشراء. تتمثل إحدى الطرق الممكنةفي تحديد 100 كرسي مشترك أولاً، والتقاط 1000 صورة من زوايا مختلفة لكل كرسي، ثم تدريب نموذج تصنيف على مجموعة بيانات الصور المجمعة. على الرغم من أن مجموعة بيانات الكرسي هذه قد تكون أكبر من مجموعة بيانات Fashion-MNIST، إلا أن عدد الأمثلة لا يزال أقل من عُشر ذلكفي ImageNet. قد يؤدي ذلك إلى تجهيز نماذج معقدة مناسبة لـ ImageNetفي مجموعة بيانات الكرسي هذه. علاوة

على ذلك، نظرًا للكمية المحدودة من أمثلة التدريب، قد لا تفي دقة النموذج المدرب بالمتطلبات العملية.

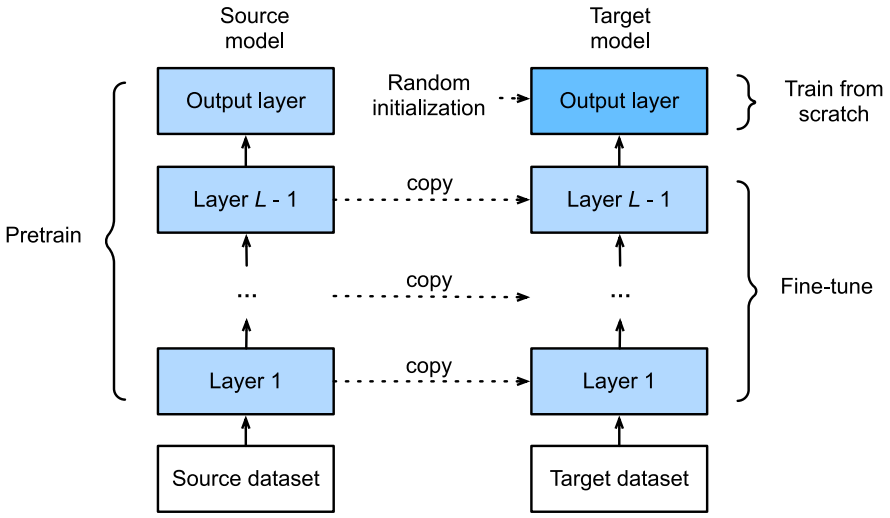
من أجل معالجة المشكلات المذكورة أعلاه، يتمثل أحد الحلول الواضحةفي جمع المزيد من البيانات. ومع ذلك، قد يستغرق جمع البيانات وتصنيفها الكثير من الوقت والمال. على سبيل المثال، من أجل جمع مجموعة بيانات ImageNet، أنفق الباحثون ملايين الدولارات من تمويل الأبحاث. على الرغم من انخفاض التكلفة الحالية لجمع البيانات بشكل كبير، إلا أنه لا يمكن تجاهل هذه التكلفة.

حل آخر هو تطبيق نقل التعلم transfer learning لنقل المعرفة المستفادة من مجموعة البيانات المصدر إلى مجموعة البيانات المستهدفة. على سبيل المثال، على الرغم من أن معظم الصورفي مجموعة بيانات ImageNet لا علاقة لها بالكراسي، فإن النموذج المدرب على مجموعة البيانات هذه قد يستخرج المزيد من ميزات الصورة العامة، والتي يمكن أن تساعدفي تحديد الحواف والأنسجة والأشكال وتكوين الكائن. قد تكون هذه الميزات المماثلة فعالة أيضًافي التعرف على الكراسي.

14.2.1. الخطوات Steps

في هذا القسم، سوف نقدم تقنية شائعةفي نقل التعلم: الضبط الدقيق fine-tuning. كما هو مبينفي الشكل 14.2.1، يتكون الضبط الدقيق من أربع خطوات:

1. قم بإجراء اختبار مسبق لنموذج الشبكة العصبية، أي النموذج المصدر، على مجموعة بيانات المصدر (على سبيل المثال، مجموعة بيانات ImageNet).
2. قم بإنشاء نموذج شبكة عصبية جديد، أي النموذج المستهدف. يؤدي هذا إلى نسخ جميع تصميمات النموذج ومعلماتهافي النموذج المصدر باستثناء طبقة الإخراج. نفترض أن معلمات النموذج هذه تحتوي على المعرفة المستفادة من مجموعة البيانات المصدر وستكون هذه المعرفة قابلة للتطبيق أيضًا على مجموعة البيانات المستهدفة. نفترض أيضًا أن طبقة الإخراج للنموذج المصدر مرتبطة ارتباطًا وثيقًا بتسميات مجموعة البيانات المصدر؛ وبالتالي لا يتم استخدامهفي النموذج المستهدف.
3. أضف طبقة مخرجات إلى النموذج المستهدف، الذي يكون عدد مخرجاته هو عدد الفئاتفي مجموعة البيانات الهدف. ثم قم بتهيئة معلمات نموذج هذه الطبقة بشكل عشوائي.
4. قم بتدريب النموذج المستهدف على مجموعة البيانات المستهدفة، مثل مجموعة بيانات الكرسي. سيتم تدريب طبقة الإخراج من البداية، بينما يتم ضبط معلمات جميع الطبقات الأخرى بدقة بناءً على معلمات النموذج المصدر.



الشكل 14.2.1 الضبط الدقيق Fine tuning.

عندما تكون مجموعات البيانات المستهدفة أصغر بكثير من مجموعات البيانات المصدر، يساعد الضبط الدقيق في تحسين قدرة التعميم generalization في النماذج.

14.2.2. التعرف على هوت دوغ Hot Dog Recognition

دعنا نظهر ضبطًا دقيقًا من خلال حالة ملموسة: التعرف على الهوت دوغ Hot Dog. سنقوم بضبط نموذج ResNet على مجموعة بيانات صغيرة، والتي تم اختبارها مسبقًا على مجموعة بيانات ImageNet. تتكون مجموعة البيانات الصغيرة هذه من آلاف الصور مع الهوت دوغ وبدونها. سوف نستخدم النموذج الدقيق للتعرف على الهوت دوغ من الصور.

```
%matplotlib inline
import os
from mxnet import gluon, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

14.2.2.1. قراءة مجموعة البيانات Reading the Dataset

تم أخذ مجموعة بيانات الهوت دوغ التي نستخدمها من الصور عبر الإنترنت. تتكون مجموعة البيانات هذه من 1400 صورة من الفئة الإيجابية تحتوي على الهوت دوغ والعديد من الصور ذات الفئة السلبية التي تحتوي على أطعمة أخرى. يتم استخدام 1000 صورة من كلا الفئتين للتدريب والباقي للاختبار.

بعد فك ضغط مجموعة البيانات التي تم تنزيلها، نحصل على مجلدين `hotdog/train` و `hotdog/test`. يحتوي كلا المجلدين على مجلدات فرعية `hotdog` و`not-hotdog`، يحتوي أي منهما على صور للفئة المقابلة.

```
#@save
d2l.DATA_HUB['hotdog'] = (d2l.DATA_URL + 'hotdog.zip',

'fba480ffa8aa7e0febbb511d181409f899b9baa5')

data_dir = d2l.download_extract('hotdog')
```

```
Downloading ../data/hotdog.zip from http://d2l-data.s3-
accelerate.amazonaws.com/hotdog.zip...
```

```
train_imgs = gluon.data.vision.ImageFolderDataset(
    os.path.join(data_dir, 'train'))
test_imgs = gluon.data.vision.ImageFolderDataset(
    os.path.join(data_dir, 'test'))
```

أول 8 أمثلة إيجابية وآخر 8 صور سلبية موضحة أدناه. كما ترى، تختلف الصورفي الحجم ونسبة العرض إلى الارتفاع.

```
hotdogs = [train_imgs[i][0] for i in range(8)]
not_hotdogs = [train_imgs[-i - 1][0] for i in range(8)]
d2l.show_images(hotdogs + not_hotdogs, 2, 8, scale=1.4);
```

أثناء التدريب، نقوم أولاً بقص منطقة عشوائية ذات حجم عشوائي ونسبة عرض إلى ارتفاع عشوائية من الصورة، ثم نقوم بتحجيم هذه المنطقة إلى صورة إدخال 224×224. أثناء الاختبار، نقوم بقياس ارتفاع وعرض الصورة إلى 256 بكسل، ثم نقوم باقتصاص منطقة مركزية 224×224 كمدخلات. بالإضافة إلى ذلك، بالنسبة لقنوات الألوان الثلاث RGB (الأحمر والأخضر والأزرق)، نقوم بتوحيد standardize قيمها قناة تلو الأخرى. بشكل ملموس، يتم طرح القيمة المتوسطة للقناة من كل قيمة لتلك القناة ثم يتم تقسيم النتيجة على الانحراف المعياري لتلك القناة.

```
# Specify the means and standard deviations of the three
RGB channels to
# standardize each channel
normalize = gluon.data.vision.transforms.Normalize(
    [0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

train_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.RandomResizedCrop(224),
    gluon.data.vision.transforms.RandomFlipLeftRight(),
    gluon.data.vision.transforms.ToTensor(),
    normalize])

test_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.Resize(256),
    gluon.data.vision.transforms.CenterCrop(224),
    gluon.data.vision.transforms.ToTensor(),
    normalize])
```

14.2.2.2. تعريف النموذج وتهيئته Defining and Initializing the Model

نستخدم ResNet-18، الذي تم اختباره مسبقًا على مجموعة بيانات ImageNet، كنموذج المصدر. هنا، نحدد `pretrained=True` لتنزيل معلمات النموذج المحددة مسبقًا تلقائيًا. إذا تم استخدام هذا النموذج لأول مرة، يلزم الاتصال بالإنترنت للتنزيل.

```
pretrained_net =
gluon.model_zoo.vision.resnet18_v2(pretrained=True)
```

يحتوي مثيل نموذج المصدر الذي تم اختباره مسبقًا على متغيرين من الأعضاء: الميزات `features` والإخراج `output`. الأول يحتوي على جميع طبقات النموذج باستثناء طبقة الإخراج، والأخيرة هي طبقة الإخراج للنموذج. الغرض الرئيسي من هذا التقسيم هو تسهيل الضبط الدقيق لمعلمات النموذج لجميع الطبقات باستثناء طبقة الإخراج. يظهر الناتج المتغير العضو لنموذج المصدر أدناه.

```
pretrained_net.output
```

```
Dense(512 -> 1000, linear)
```

وباعتبارها طبقة متصلة بالكامل، فإنها تحول نواتج متوسط التجميع العالمي global average pooling النهائية لـ ResNet إلى مخرجات فئة 1000 من مجموعة بيانات ImageNet. ثم نقوم ببناء شبكة عصبية جديدة كنموذج مستهدف. يتم تعريفه بنفس طريقة نموذج المصدر الذي تم اختباره مسبقًا فيما عدا أنه تم تعيين عدد مخرجاتهفي الطبقة النهائية على عدد الفئاتفي مجموعة البيانات المستهدفة (بدلاً من 1000).

في الكود أدناه، تتم تهيئة معلمات النموذج قبل طبقة الإخراج لمثيل النموذج المستهدف finetune_net لنمذجة معلمات الطبقات المقابلة من النموذج المصدر. نظرًا لأنه تم الحصول على معلمات النموذج هذه عن طريق التدريب المسبق على ImageNet، فهي فعالة. لذلك، لا يمكننا استخدام سوى معدل تعليمي صغير لضبط هذه المعلمات المحددة مسبقًا.في المقابل، تتم تهيئة معلمات النموذجفي طبقة المخرجات بشكل عشوائي وتتطلب عمومًا معدل تعلم أكبر يمكن تعلمه من نقطة الصفر. مع ترك معدل التعلم الأساسي يكون η ، سيتم استخدام معدل التعلم 10η لتكرار معلمات النموذجفي طبقة المخرجات.

```
finetune_net =
gluon.model_zoo.vision.resnet18_v2(classes=2)
finetune_net.features = pretrained_net.features
finetune_net.output.initialize(init.Xavier())
# The model parameters in the output layer will be
iterated using a learning
# rate ten times greater
finetune_net.output.collect_params().setattr('lr_mult',
10)
```

14.2.2.3. الضبط الدقيق للنموذج Fine-Tuning the Model

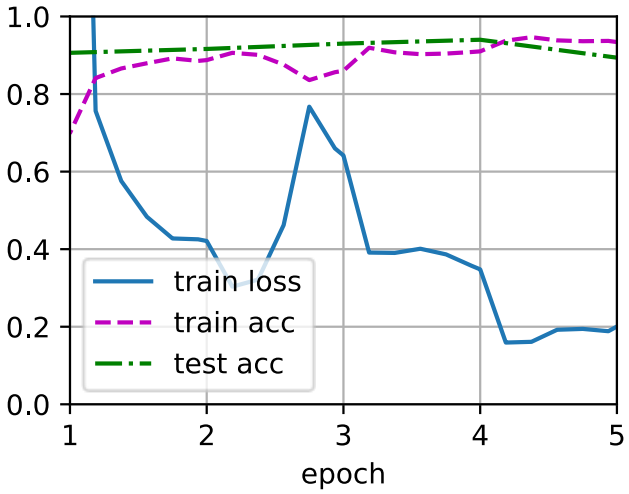
أولاً، نحدد دالة تدريب train_fine_tuning التي تستخدم الضبط الدقيق بحيث يمكن استدعاؤها عدة مرات.

```
def train_fine_tuning(net, learning_rate,
batch_size=128, num_epochs=5):
    train_iter = gluon.data.DataLoader(
        train_imgs.transform_first(train_augs),
batch_size, shuffle=True)
    test_iter = gluon.data.DataLoader(
        test_imgs.transform_first(test_augs),
batch_size)
    devices = d2l.try_all_gpus()
    net.collect_params().reset_ctx(devices)
    net.hybridize()
    loss = gluon.loss.SoftmaxCrossEntropyLoss()
    trainer = gluon.Trainer(net.collect_params(), 'sgd',
{
        'learning_rate': learning_rate, 'wd': 0.001})
    d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs,
                   devices)
```

قمنا بتعيين معدل التعلم الأساسي على قيمة صغيرة من أجل الضبط الدقيق fine-tune لمعلمات النموذج التي تم الحصول عليها عن طريق التدريب المسبق pretraining. استنادًا إلى الإعدادات السابقة، سنقوم بتدريب معلمات طبقة الإخراج للنموذج المستهدف من البداية باستخدام معدل تعلم أكبر بعشر مرات.

```
train_fine_tuning(finetune_net, 0.01)
```

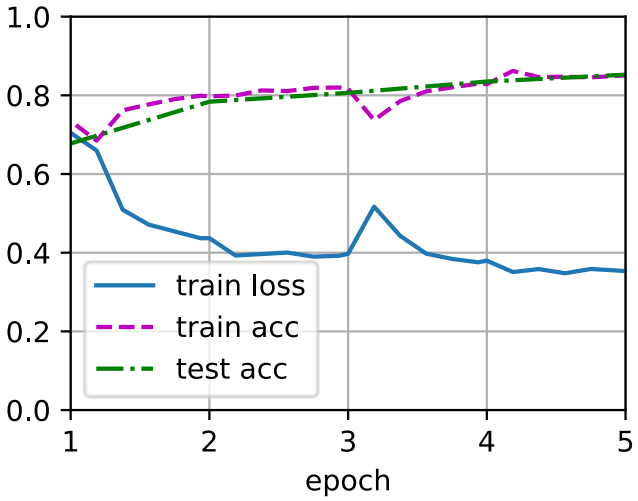
```
loss 0.200, train acc 0.934, test acc 0.894
196.9 examples/sec on [gpu(0), gpu(1)]
```



للمقارنة، نحدد نموذجًا متطابقًا، لكننا نهيئ جميع معلمات النموذج الخاصة به إلى قيم عشوائية. نظرًا لأن النموذج بأكمله يحتاج إلى التدريب من البداية، يمكننا استخدام معدل تعلم أكبر.

```
scratch_net =
gluon.model_zoo.vision.resnet18_v2(classes=2)
scratch_net.initialize(init=init.Xavier())
train_fine_tuning(scratch_net, 0.1)
```

```
loss 0.353, train acc 0.851, test acc 0.853
177.6 examples/sec on [gpu(0), gpu(1)]
```



كما نرى، يميل النموذج المضبوط بدقة fine-tuned model إلى الأداء بشكل أفضل لنفس الفترة لأن قيم معلماته الأولية أكثر فاعلية.

14.2.3. الملخص

- نقل التعلم Transfer learning ينقل المعرفة المكتسبة من مجموعة البيانات المصدر إلى مجموعة البيانات المستهدفة. الضبط الدقيق Fine-tuning هو أسلوب شائع لنقل التعلم.
- ينسخ النموذج الهدف جميع تصميمات النموذج مع معلماتها من النموذج المصدر باستثناء طبقة الإخراج، ويضبط هذه المعلمات استنادًا إلى مجموعة البيانات المستهدفة.في المقابل، يجب تدريب طبقة الإخراج للنموذج المستهدف من البداية.
- بشكل عام، يستخدم الضبط الدقيق للمعلمات معدل تعلم أصغر، بينما يمكن أن يستخدم تدريب طبقة الإخراج من البداية معدل تعلم أكبر.

14.2.4. التمارين

1. استمرفي زيادة معدل التعلم لـ `finetune_net`. كيف تتغير دقة النموذج؟
2. مزيد من ضبط hyperparameters من `finetune_net` و `scratch_net`في التجربة المقارنة. هل ما زالوا يختلفونفي الدقة؟
3. قم بتعيين المعلمات قبل طبقة الإخراج من `finetune_net` على تلك الخاصة بالنموذج المصدر ولا تقم بتحديثها أثناء التدريب. كيف تتغير دقة النموذج؟ يمكنك استخدام الكود التالي.

```
finetune_net.features.collect_params().setattr('grad_req
', 'null')
```

4. في الواقع، هناك فئة "hotdog"في مجموعة بيانات `ImageNet`. يمكن الحصول على معلمة الوزن المقابلةفي طبقة الإخراج عبر الكود التالي. كيف يمكننا الاستفادة من معلمة الوزن هذه؟

```
weight = pretrained_net.output.weight
hotdog_w = np.split(weight.data(), 1000, axis=0)[713]
hotdog_w.shape
```

```
(1, 512)
```

14.3. اكتشاف الكائنات والمربعات المحيطة Object Detection and Bounding Boxes

في الأقسام السابقة (على سبيل المثال، القسم 8.1 ـ القسم 8.4)، قدمنا نماذج مختلفة لتصنيف الصور.في مهام تصنيف الصور، نفترض أن هناك كائنًا رئيسيًا واحدًا فقطفي الصورة ونركز فقط على كيفية التعرف على فئتها. ومع ذلك، غالبًا ما توجد كائنات متعددةفي الصورة محل الاهتمام. لا نريد فقط معرفة فئاتهم، ولكن أيضًا مواقعهم المحددةفي الصورة.في الرؤية الحاسوبية، نشير إلى مثل هذه المهام باكتشاف الأشياء object detection (أو التعرف على الأشياء object recognition).

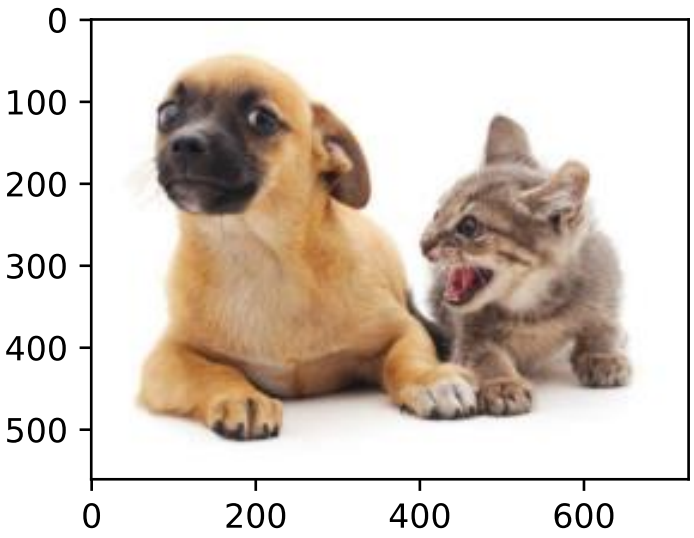
تم تطبيق اكتشاف الكائن Object detection على نطاق واسعفي العديد من المجالات. على سبيل المثال، تحتاج القيادة الذاتية إلى التخطيط لطرق السفر من خلال اكتشاف مواقع المركبات والمشاة والطرق والعقباتفي صور الفيديو الملتقطة. إلى جانب ذلك، قد تستخدم الروبوتات هذه التقنية لاكتشاف وتحديد العناصر محل الاهتمام خلال تنقلهافي بيئة ما. علاوة على ذلك، قد تحتاج أنظمة الأمان إلى اكتشاف أشياء غير طبيعية، مثل الدخلاء أو القنابل.

في الأقسام القليلة التالية، سوف نقدم العديد من طرق التعلم العميق لاكتشاف الكائنات. سنبدأ بمقدمة لمواضع positions (أو مواقع locations) الكائنات.

```
%matplotlib inline
import tensorflow as tf
from d2l import tensorflow as d2l
```

سنقوم بتحميل عينة الصورة لاستخدامهافي هذا القسم. يمكننا أن نرى أن هناك كلبًا على الجانب الأيسر من الصورة وقطة على اليمين. هما العنصران الرئيسيانفي هذه الصورة.

```
d2l.set_figsize()
img = d2l.plt.imread('../img/catdog.jpg')
d2l.plt.imshow(img);
```



14.3.1. المربعات المحيطة Bounding Boxes

في اكتشاف الكائن، نستخدم عادةً مربعًا محيطًا bounding box لوصف الموقع المكاني للكائن. الصندوق المحيط مستطيل الشكل، يتم تحديده بواسطة الإحداثيات x و y للزاوية اليسرى العلوية من المستطيل وإحداثياتها من الزاوية اليمنى السفلية. تمثيل مربع إحاطة آخر شائع الاستخدام هو إحداثيات المحور لمركز المربع المحيط وعرض الصندوق وارتفاعه.

نحدد هنا دوال للتحويل بين هذين التمثيلين: يحول `box_corner_to_center` من التمثيل ذي الزاويتين إلى عرض المركز والعرض والارتفاع ، و `box_center_to_corner` العكس بالعكس. يجب أن تكون مربعات وسيطة الإدخال موترًا ثنائي الأبعاد للشكل $(n, 4)$، حيث يوجد عدد المربعات المحيطة.

```
#@save
def box_corner_to_center(boxes):
    """Convert from (upper-left, lower-right) to
(center, width, height)."""
    x1, y1, x2, y2 = boxes[:, 0], boxes[:, 1], boxes[:,
2], boxes[:, 3]
    cx = (x1 + x2) / 2
    cy = (y1 + y2) / 2
    w = x2 - x1
    h = y2 - y1
    boxes = tf.stack((cx, cy, w, h), axis=-1)
    return boxes
```

```
#@save
def box_center_to_corner(boxes):
    """Convert from (center, width, height) to (upper-
left, lower-right)."""
    cx, cy, w, h = boxes[:, 0], boxes[:, 1], boxes[:,
2], boxes[:, 3]
    x1 = cx - 0.5 * w
    y1 = cy - 0.5 * h
    x2 = cx + 0.5 * w
    y2 = cy + 0.5 * h
    boxes = tf.stack((x1, y1, x2, y2), axis=-1)
    return boxes
```

سنحدد المربعات المحيطة بالكلب والقطفي الصورة بناءً على معلومات الإحداثيات. أصل الإحداثياتفي الصورة هو الزاوية العلوية اليسرى للصورة، وإلى اليمين والأسفل توجد الاتجاهات الإيجابية للمحاور والمحاور x وy ، على التوالي.

```
# Here `bbox` is the abbreviation for bounding box
dog_bbox, cat_bbox = [60.0, 45.0, 378.0, 516.0], [400.0,
112.0, 655.0, 493.0]
```

يمكننا التحقق من صحة دالتي تحويل الصندوق المحيط عن طريق التحويل converting مرتين.

```
boxes = tf.constant((dog_bbox, cat_bbox))
box_center_to_corner(box_corner_to_center(boxes)) ==
boxes
```

```
<tf.Tensor: shape=(2, 4), dtype=bool, numpy=
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True]])>
```

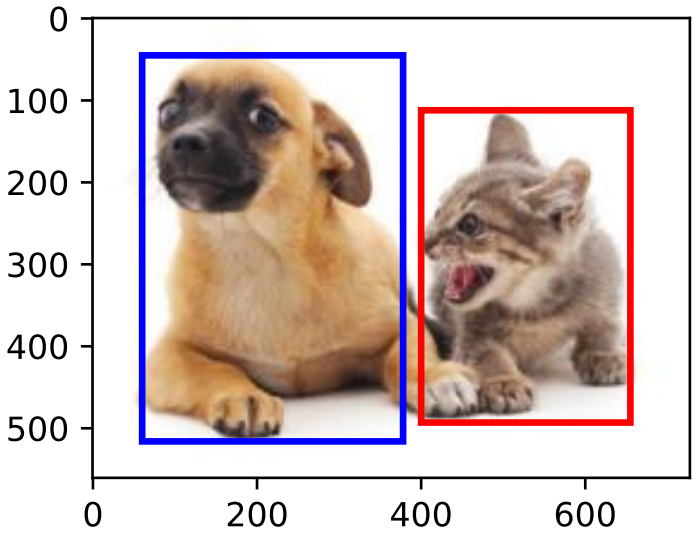
دعنا نرسم المربعات المحيطة بالصورة للتحقق مما إذا كانت دقيقة. قبل الرسم، سنقوم بتعريف الدالة المساعدة `bbox_to_rect`. يمثل المربع المحيط بتنسيق المربع المحيط لحزمة `matplotlib`.

```
#@save
def bbox_to_rect(bbox, color):
    """Convert bounding box to matplotlib format."""
    # Convert the bounding box (upper-left x, upper-left
y, lower-right x,
    # lower-right y) format to the matplotlib format:
((upper-left x,
    # upper-left y), width, height)
```

```
    return d2l.plt.Rectangle(
        xy=(bbox[0], bbox[1]), width=bbox[2]-bbox[0],
height=bbox[3]-bbox[1],
        fill=False, edgecolor=color, linewidth=2)
```

بعد إضافة المربعات المحيطة بالصورة، يمكننا أن نرى أن المخطط التفصيلي الرئيسي لكائنين موجودان أساسًا داخل المربعين.

```
fig = d2l.plt.imshow(img)
fig.axes.add_patch(bbox_to_rect(dog_bbox, 'blue'))
fig.axes.add_patch(bbox_to_rect(cat_bbox, 'red'));
```



14.3.2. الملخص

- لا يتعرف اكتشاف الكائن Object detection على جميع الكائنات ذات الأهميةفي الصورة فحسب، بل يتعرف أيضًا على مواقعها. يتم تمثيل الموقف بشكل عام من خلال مربع محيط مستطيل.
- يمكننا التحويل بين اثنين من تمثيلات الصندوق المحيط bounding box شائعة الاستخدام.

14.3.3. التمارين

5. ابحث عن صورة أخرى وحاول تسمية المربع المحيط الذي يحتوي على الكائن. قارن بين المربعات والفئات المحيطة بوضع العلامات: ما الذي يستغرق عادةً وقتًا أطول؟

6. لماذا يكون البعد الأعمق لمربعات وسيطة الإدخالفي `boxes` لـ `box_corner_to_center` و `box_center_to_corner always` دائمًا 4؟

14.4. مربعات التحديد Anchor Boxes

عادةً ما تقوم خوارزميات اكتشاف الكائنات بأخذ عينات من عدد كبير من المناطقفي صورة الإدخال، وتحديد ما إذا كانت هذه المناطق تحتوي على كائنات مهمة، وضبط حدود المناطق بحيث يمكن التنبؤ بمربعات إحاطة الحقيقة الاساسية للكائنات بشكل أكثر دقة. قد تعتمد النماذج المختلفة مخططات مختلفة لأخذ العيناتفي المنطقة. نقدم هنا إحدى هذه الطرق: فهي تنشئ مربعات إحاطة متعددة بمقاييس مختلفة ونسب عرض إلى ارتفاع تتمحور حول كل بكسل. تسمى هذه الصناديق المحيطة مربعات التحديد anchor boxes. سنقوم بتصميم نموذج لاكتشاف الكائن بناءً على مربعات التحديدفي القسم 14.7.

أولاً، دعنا نعدل دقة الطباعة لمخرجات أكثر إيجازًا.

```
%matplotlib inline
from mxnet import gluon, image, np, npx
from d2l import mxnet as d2l

np.set_printoptions(2)  # Simplify printing accuracy
npx.set_np()
```

14.4.1. إنشاء صناديق تحديد متعددة Generating Multiple Anchor Boxes

افترض أن صورة الإدخال يبلغ ارتفاعها وعرضها. نقوم بإنشاء مربعات تحديد بأشكال مختلفة تتمحور حول كل بكسل من الصورة. دع المقياس $s \in (0,1]$ يكون ونسبة العرض إلى الارتفاع (نسبة العرض إلى الارتفاع) هي $r > 0$. ثم يكون عرض وارتفاع صندوق التحديد $ws\sqrt{r}$ و $hs/\sqrt{r}$ ، على التوالي. لاحظ أنه عند تحديد موضع المركز، يتم تحديد مربع تحديد بعرض وارتفاع معروفين.

لإنشاء مربعات تحديد متعددة بأشكال مختلفة، فلنقم بتعيين سلسلة من المقاييس $s_1, \ldots, s_n$ وسلسلة من نسب العرض إلى الارتفاع $r_1, \ldots, r_m$. عند استخدام جميع مجموعات هذه المقاييس ونسب العرض إلى الارتفاع مع كل بكسل كمركز، ستحتوي صورة الإدخال على إجمالي مربعات التحديد $whnm$. على الرغم من أن مربعات التحديد هذه قد تغطي جميع المربعات المحيطة بالحقيقة الأساسية، إلا أن التعقيد الحسابي مرتفع للغاية بسهولة.في الممارسة العملية، لا يسعنا إلا أن نأخذفي الاعتبار هؤلاء s_1 او r_1 .

$$(s_1, r_1), (s_1, r_2), \ldots, (s_1, r_m), (s_2, r_1), (s_3, r_1), \ldots, (s_n, r_1).$$

وهذا يعني أن عدد مربعات التحديد المتمركزة على نفس البكسل هو $n + m - 1$. بالنسبة لصورة الإدخال بأكملها، سنقوم بإنشاء إجمالي $wh(n + m - 1)$ مربعات تحديد.

يتم تنفيذ الطريقة المذكورة أعلاه لتوليد مربعات التثبيتفي دالة `multibox_prior` التالية. نحدد صورة الإدخال وقائمة المقاييس وقائمة نسب العرض إلى الارتفاع، ثم ستعيد هذه الدالة جميع مربعات التحديد.

```
#@save
def multibox_prior(data, sizes, ratios):
    """Generate anchor boxes with different shapes
centered on each pixel."""
    in_height, in_width = data.shape[-2:]
    device, num_sizes, num_ratios = data.ctx,
len(sizes), len(ratios)
    boxes_per_pixel = (num_sizes + num_ratios - 1)
    size_tensor = np.array(sizes, ctx=device)
    ratio_tensor = np.array(ratios, ctx=device)
    # Offsets are required to move the anchor to the
center of a pixel. Since
    # a pixel has height=1 and width=1, we choose to
offset our centers by 0.5
    offset_h, offset_w = 0.5, 0.5
    steps_h = 1.0 / in_height  # Scaled steps in y-axis
    steps_w = 1.0 / in_width  # Scaled steps in x-axis

    # Generate all center points for the anchor boxes
    center_h = (np.arange(in_height, ctx=device) +
offset_h) * steps_h
    center_w = (np.arange(in_width, ctx=device) +
offset_w) * steps_w
    shift_x, shift_y = np.meshgrid(center_w, center_h)
    shift_x, shift_y = shift_x.reshape(-1),
shift_y.reshape(-1)

    # Generate `boxes_per_pixel` number of heights and
widths that are later
    # used to create anchor box corner coordinates
(xmin, xmax, ymin, ymax)
    w = np.concatenate((size_tensor *
np.sqrt(ratio_tensor[0]),
```

```
                            sizes[0] *
np.sqrt(ratio_tensor[1:]))) \
                            * in_height / in_width  # Handle
rectangular inputs
    h = np.concatenate((size_tensor /
np.sqrt(ratio_tensor[0]),
                            sizes[0] /
np.sqrt(ratio_tensor[1:])))
    # Divide by 2 to get half height and half width
    anchor_manipulations = np.tile(np.stack((-w, -h, w,
h)).T,
                                   (in_height *
in_width, 1)) / 2

    # Each center point will have `boxes_per_pixel`
number of anchor boxes, so
    # generate a grid of all anchor box centers with
`boxes_per_pixel` repeats
    out_grid = np.stack([shift_x, shift_y, shift_x,
shift_y],
                        axis=1).repeat(boxes_per_pixel,
axis=0)
    output = out_grid + anchor_manipulations
    return np.expand_dims(output, axis=0)
```

يمكننا أن نرى أن شكل متغير مربع التحديد المرتجع Y هو (حجم الدُفعة، عدد مربعات التحديد، 4).

```
img = image.imread('../img/catdog.jpg').asnumpy()
h, w = img.shape[:2]

print(h, w)
X = np.random.uniform(size=(1, 3, h, w))  # Construct
input data
Y = multibox_prior(X, sizes=[0.75, 0.5, 0.25],
ratios=[1, 2, 0.5])
Y.shape
```

```
561 728
(1, 2042040, 4)
```

بعد تغيير شكل متغير صندوق التحديد Y إلى (ارتفاع الصورة، عرض الصورة، عدد مربعات التحديد المتمركزة على نفس البكسل، 4)، يمكننا الحصول على جميع مربعات التحديد المتمركزةفي موضع بكسل محدد. فيما يلي، نصل إلى أول صندوق مرساة متمركز على (250،

250). يحتوي على أربعة عناصر:(x, y) إحداثيات المحورفي الزاوية العلوية اليسرى و(x, y) إحداثيات المحورفي الزاوية اليمنى السفلية من صندوق التحديد. قيم إحداثيات كلا المحورين مقسومة على عرض الصورة وارتفاعها، على التوالي؛ وبالتالي، يكون النطاق بين 0 و 1.

```
boxes = Y.reshape(h, w, 5, 4)
boxes[250, 250, 0, :]
```

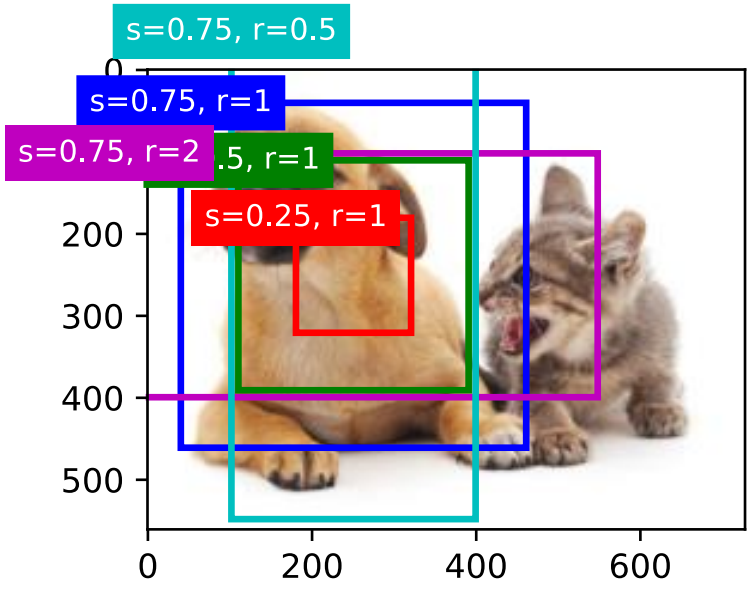
```
array([0.06, 0.07, 0.63, 0.82])
```

لإظهار جميع مربعات التحديد المتمركزة على بكسل واحدفي الصورة، نحدد دالة `show_bboxes` التالية لرسم مربعات إحاطة متعددة على الصورة.

```
#@save
def show_bboxes(axes, bboxes, labels=None, colors=None):
    """Show bounding boxes."""

    def make_list(obj, default_values=None):
        if obj is None:
            obj = default_values
        elif not isinstance(obj, (list, tuple)):
            obj = [obj]
        return obj

    labels = make_list(labels)
    colors = make_list(colors, ['b', 'g', 'r', 'm',
'c'])
    for i, bbox in enumerate(bboxes):
        color = colors[i % len(colors)]
        rect = d2l.bbox_to_rect(bbox.asnumpy(), color)
        axes.add_patch(rect)
        if labels and len(labels) > i:
            text_color = 'k' if color == 'w' else 'w'
            axes.text(rect.xy[0], rect.xy[1], labels[i],
                      va='center', ha='center',
fontsize=9, color=text_color,
                      bbox=dict(facecolor=color, lw=0))
```

كما رأينا للتو، تم تقسيم قيم إحداثيات المحاور x و yفي المتغير `boxes` على عرض الصورة وارتفاعها، على التوالي. عند رسم مربعات التحديد، نحتاج إلى استعادة قيم إحداثياتها الأصلية؛ وبالتالي، نحدد المتغير `bbox_scale` أدناه. الآن، يمكننا رسم كل مربعات التحديد المتمركزة على (250، 250)في الصورة. كما ترى، فإن المربع الأزرق ذو المقياس 0.75 ونسبة العرض إلى الارتفاع 1 جيدًا يحيط بالكلبفي الصورة.

```
d2l.set_figsize()
bbox_scale = np.array((w, h, w, h))
fig = d2l.plt.imshow(img)
show_bboxes(fig.axes, boxes[250, 250, :, :] *
bbox_scale,
            ['s=0.75, r=1', 's=0.5, r=1', 's=0.25, r=1',
's=0.75, r=2',
             's=0.75, r=0.5'])
```



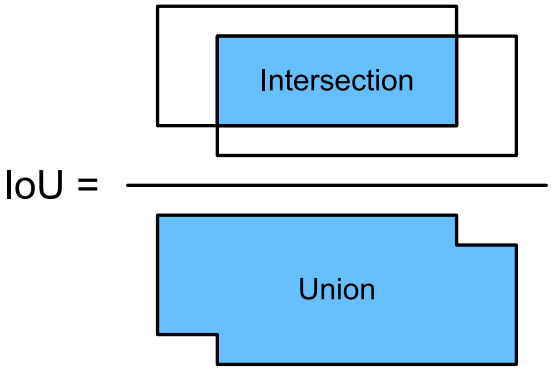
4.4.2. تقاطع على الاتحاد (IoU)

لقد ذكرنا للتو أن صندوق التحديد anchor box "جيدًا" يحيط بالكلبفي الصورة. إذا كان الصندوق المحيط بالحقيقة الأساسية ground-truth bounding box للكائن معروفًا، فكيف يمكن قياس "جيدًا" هنا؟ حدسيًا، يمكننا قياس التشابه بين صندوق التحديد ومربع إحاطة الحقيقة الاساسية. نعلم أن مؤشر Jaccard يمكنه قياس التشابه بين مجموعتين. المجموعات المعطاة $\mathcal{A}$ و $\mathcal{B}$ ومؤشر Jaccard الخاص بهم هو حجم تقاطعهم مقسومًا على حجم اتحادهم:

$$J(\mathcal{A},\mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}.$$

في الواقع، يمكننا اعتبار منطقة البكسل لأي مربع محيط كمجموعة من وحدات البكسل. بهذه الطريقة، يمكننا قياس التشابه بين المربعين المحيطين بواسطة فهرس Jaccard لمجموعات البكسل الخاصة بهم. بالنسبة إلى الصندوقين المحيطين، نشير عادةً إلى مؤشر Jaccard الخاص

بهم على أنه تقاطع على الاتحاد (IoU)، وهو نسبة منطقة التقاطع إلى منطقة الاتحاد الخاصة بهم، كما هو موضح في الشكل 14.4.1. نطاق IoU بين 0 و1: 0 يعني أن الصندوقين المحيطين لا يتداخلان على الإطلاق، بينما 1 يشير إلى أن الصندوقين المحيطين متساويان.



الشكل 14.4.1 IoU هو نسبة منطقة التقاطع إلى منطقة الاتحاد IoU لمربعين محيطين.

بالنسبة لبقية هذا القسم، سنستخدم *intersection over union* (*IoU*) لقياس التشابه بين مربعات التحديد والمربعات المحيطة بالحقيقة الأرضية، وبين مربعات التحديد المختلفة. بالنظر إلى قائمتين من مربعات التحديد أو المربعات المحيطة، فإن المربع التالي `box_iou` يحسب IoU الثنائي عبر هاتين القائمتين.

```
#@save
def box_iou(boxes1, boxes2):
    """Compute pairwise IoU across two lists of anchor
or bounding boxes."""
    box_area = lambda boxes: ((boxes[:, 2] - boxes[:,
0]) *
                              (boxes[:, 3] - boxes[:,
1]))
    # Shape of `boxes1`, `boxes2`, `areas1`, `areas2`:
(no. of boxes1, 4),
    # (no. of boxes2, 4), (no. of boxes1,), (no. of
boxes2,)
    areas1 = box_area(boxes1)
    areas2 = box_area(boxes2)
    # Shape of `inter_upperlefts`, `inter_lowerrights`,
`inters`: (no. of
    # boxes1, no. of boxes2, 2)
```

```
    inter_upperlefts = np.maximum(boxes1[:, None, :2],
boxes2[:, :2])
    inter_lowerrights = np.minimum(boxes1[:, None, 2:],
boxes2[:, 2:])
    inters = (inter_lowerrights -
inter_upperlefts).clip(min=0)
    # Shape of `inter_areas` and `union_areas`: (no. of
boxes1, no. of boxes2)
    inter_areas = inters[:, :, 0] * inters[:, :, 1]
    union_areas = areas1[:, None] + areas2 - inter_areas
    return inter_areas / union_areas
```

14.4.3. تسمية مربعات التحديد في بيانات التدريب Labeling Anchor Boxes in Training Data

في مجموعة بيانات التدريب، نعتبر كل مربع تحديد كمثال تدريب. من أجل تدريب نموذج اكتشاف الكائن، نحتاج إلى تسميات فئة وإزاحة لكل مربع تحديد، حيث يكون الأول هو فئة الكائن ذات الصلة بصندوق التحديد والأخير هو إزاحة المربع المحيط بالحقيقة الاساسية بالنسبة لصندوق التحديد. أثناء التوقع، نقوم بإنشاء مربعات تحديد متعددة لكل صورة، ونتنبأ بالفئات والإزاحات لجميع مربعات التحديد ، وضبط مواضعها وفقًا للإزاحات المتوقعة للحصول على مربعات الإحاطة المتوقعة ، وأخيراً فقط نخرج تلك المربعات المحيطة المتوقعة التي تفي بمعايير معينة .

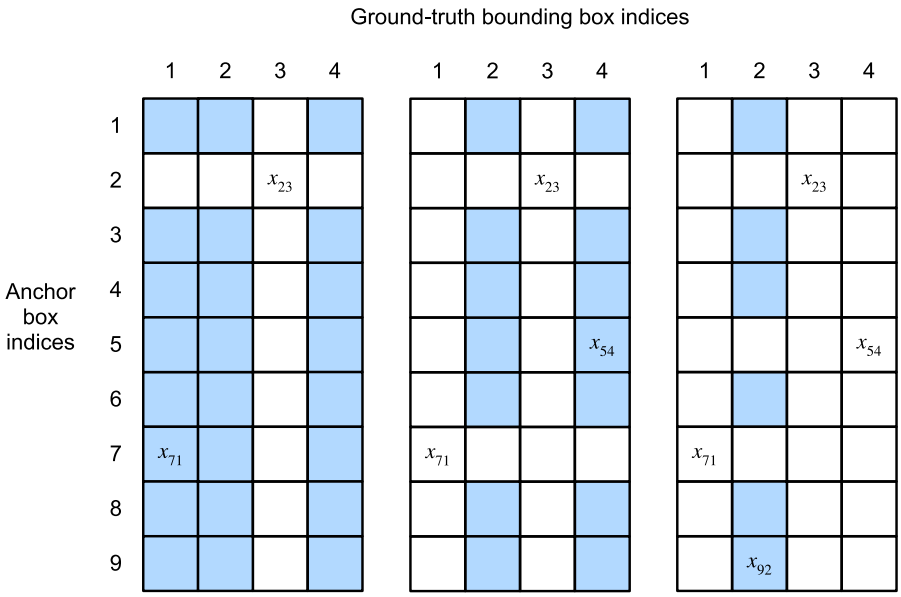
كما نعلم، تأتي مجموعة التدريب على اكتشاف الأشياء مصحوبة بتسميات لمواقع المربعات المحيطة بالحقيقة الأرضية وفئات الكائنات المحيطة بها. لتسمية أي مربع تحديد تم إنشاؤه، نشير إلى الموقع المسمى وفئة المربع المحيط بالحقيقة الأرضية المخصصة له والأقرب إلى مربع التحديد. فيما يلي، نصف خوارزمية لتعيين أقرب مربعات إحاطة للحقيقة الاساسية لمربعات التحديد.

14.4.3.1. تعيين مربعات الاحاطة للحقيقة الأرضية لمربعات التحديد Assigning Ground-Truth Bounding Boxes to Anchor Boxes

بالنظر إلى الصورة، افترض أن مربعات التحديد هي $A_1, A_2, \ldots, A_{n_a}$ موجودة وأن المربعات المحيطة بالحقيقة الاساسية هي $B_1, B_2, \ldots, B_{n_b}$ ، حيث $n_a \geq n_b$. دعنا نعرف مصفوفة $\mathbf{X} \in \mathbb{R}^{n_a \times n_b}$، عنصرها x_{ij} في الصف i^{th} والعمود j^{th} هو IoU لمربع التحديد A_i ومربع إحاطة الحقيقة الأرضية B_j. تتكون الخوارزمية من الخطوات التالية:

1. ابحث عن أكبر عنصرفي المصفوفة **X** وقم بالإشارة إلى فهارس الصفوف والعمود كـ i_1 و j_1 على التوالي. ثم يتم تخصيص مربع إحاطة الحقيقة الأرضية B_{j_1} لمربع التحديد A_{i_1}. هذا أمر بديهي تمامًا لأنه الأقرب بين جميع أزواج مربعات التحديد وصناديق ربط الحقيقة الأرضية. بعد المهمة الأولى، تجاهل كل العناصرفي الصف $i_1{}^{\text{th}}$ والعمود $j_1{}^{\text{th}}$في المصفوفة **X**.
2. ابحث عن أكبر العناصر المتبقيةفي المصفوفة **X** وقم بالإشارة إلى فهارس الصفوف والعمودكـ i_2 و j_2 على التوالي. نقوم بتعيين مربع إحاطة الحقيقة الاساسية B_{j_2} لمربع التحديد A_{i_2} وتجاهل جميع العناصر الموجودةفي الصف $i_2{}^{\text{th}}$ والعمود $j_2{}^{\text{th}}$في المصفوفة.
3. في هذه المرحلة، تم تجاهل العناصر الموجودةفي صفين وعمودينفي المصفوفة **X**. ننتقل حتى يتم التخلص من جميع العناصر الموجودةفي الأعمدة n_bفي المصفوفة **X**. في هذا الوقت، قمنا بتعيين مربع إحاطة للحقيقة الاساسية لكل من مربعات التحديد n_b.
4. اجتياز فقط من خلال مربعات التحديد المتبقية $n_a - n_b$. على سبيل المثال، بالنظر إلى أي مربع تحديد A_i، ابحث عن المربع المحيط بالحقيقة الاساسية B_j مع أكبر وحدة IoU مع A_i داخل صف i^{th} المصفوفة **X**، وقم بتعيين B_j الى A_i فقط إذاكان هذا الإدخال IoU أكبر من عتبة محددة مسبقًا.

دعونا نوضح الخوارزمية أعلاه باستخدام مثال ملموس. كما هو مبينفي الشكل 14.4.2 (يسار)، بافتراض أن القيمة القصوىفي المصفوفة **X** هي x_{23}، فإننا نخصص المربع المحيط بالحقيقة الأرضية B_3 لصندوق التحديد A_2. بعد ذلك، نتجاهل جميع العناصر الموجودةفي الصف 2 والعمود 3 من المصفوفة، ونجد أكبر x_{71}في العناصر المتبقية (المنطقة المظللة)، ونخصص المربع المحيط بالحقيقة الأرضية B_1 لمربع التحديد A_7. بعد ذلك، كما هو موضحفي الشكل 14.4.2 (الوسط)، تجاهل جميع العناصر الموجودةفي الصف 7 والعمود 1 من المصفوفة، وابحث عن أكبر x_{54}في العناصر المتبقية (المنطقة المظللة)، وقم بتعيين المربع المحيط بالحقيقة الاساسية B_4 إلى صندوق التحديد A_5. أخيرًا، كما هو موضحفي الشكل 14.4.2 (يمين)، تجاهل جميع العناصر الموجودةفي الصف 5 والعمود 4 من المصفوفة، وابحث عن أكبر x_{92}في العناصر في العناصر المتبقية (المنطقة المظللة)، وقم بتعيين المربع المحيط بالحقيقة الأرضية B_2 إلى صندوق التحديد A_9. بعد ذلك، نحتاج فقط إلى اجتياز مربعات التحديد المتبقية A_1, A_3, A_4, A_6, A_8 وتحديد ما إذا كنا سنخصص لها مربعات احاطة الحقيقة الاساسية وفقًا للعتبة.



الشكل 14.4.2 تعيين مربعات إحاطة الحقيقة الاساسية لمربعات التحديد.

يتم تنفيذ هذه الخوارزمية في دالة `assign_anchor_to_bbox` التالية.

```
#@save
def assign_anchor_to_bbox(ground_truth, anchors, device,
iou_threshold=0.5):
    """Assign closest ground-truth bounding boxes to
anchor boxes."""
    num_anchors, num_gt_boxes = anchors.shape[0],
ground_truth.shape[0]
    # Element x_ij in the i-th row and j-th column is
the IoU of the anchor
    # box i and the ground-truth bounding box j
    jaccard = box_iou(anchors, ground_truth)
    # Initialize the tensor to hold the assigned ground-
truth bounding box for
    # each anchor
    anchors_bbox_map = np.full((num_anchors,), -1,
dtype=np.int32, ctx=device)
    # Assign ground-truth bounding boxes according to
the threshold
    max_ious, indices = np.max(jaccard, axis=1),
np.argmax(jaccard, axis=1)
    anc_i = np.nonzero(max_ious >= iou_threshold)[0]
```

```
    box_j = indices[max_ious >= iou_threshold]
    anchors_bbox_map[anc_i] = box_j
    col_discard = np.full((num_anchors,), -1)
    row_discard = np.full((num_gt_boxes,), -1)
    for _ in range(num_gt_boxes):
        max_idx = np.argmax(jaccard)  # Find the largest
IoU
        box_idx = (max_idx %
num_gt_boxes).astype('int32')
        anc_idx = (max_idx /
num_gt_boxes).astype('int32')
        anchors_bbox_map[anc_idx] = box_idx
        jaccard[:, box_idx] = col_discard
        jaccard[anc_idx, :] = row_discard
    return anchors_bbox_map
```

14.4.3.2. فئات التسمية والإزاحة Labeling Classes and Offsets

الآن يمكننا تسمية الفئة والإزاحة لكل مربع تحديد. افترض أنه تم تخصيص مربع إحاطة الحقيقة الاساسية B لمربع التحديد A. من ناحية، سيتم تسمية فئة مربع التحديد A بأنها فئة B. من ناحية أخرى، سيتم تسمية إزاحة offset صندوق التحديد A وفقًا للموضع النسبي بين الإحداثيات المركزية B و Aومع الحجم النسبي بين هذين الصندوقين. بالنظر إلى المواضع والأحجام المختلفة للمربعات المختلفةفي مجموعة البيانات، يمكننا تطبيق تحويلات على تلك المواضع والأحجام النسبية التي قد تؤدي إلى إزاحات موزعة بشكل منتظم يسهل ملاءمتها. هنا نصف تحول مشترك. بالنظر إلى الإحداثيات المركزية لـ AوB كـ (x_a, y_a) و(x_b, y_b) ، عرضهم كـ w_a و w_b، وارتفاعاتهم h_a وh_b على التوالي. قد نقوم بتسمية الإزاحة لـ A كـ

$$\left(\frac{\frac{x_b - x_a}{w_a} - \mu_x}{\sigma_x}, \frac{\frac{y_b - y_a}{h_a} - \mu_y}{\sigma_y}, \frac{\log\frac{w_b}{w_a} - \mu_w}{\sigma_w}, \frac{\log\frac{h_b}{h_a} - \mu_h}{\sigma_h}\right),$$

حيث تكون القيم الافتراضية للثوابت هي $\mu_x = \mu_y = \mu_w = \mu_h = 0, \sigma_x = \sigma_y = 0.1$ ، و$\sigma_w = \sigma_h = 0.2$. يتم تنفيذ هذا التحويل أدناهفي دالة `offset_boxes`.

```
#@save
def offset_boxes(anchors, assigned_bb, eps=1e-6):
    """Transform for anchor box offsets."""
    c_anc = d2l.box_corner_to_center(anchors)
    c_assigned_bb =
d2l.box_corner_to_center(assigned_bb)
```

```
    offset_xy = 10 * (c_assigned_bb[:, :2] - c_anc[:,
:2]) / c_anc[:, 2:]
    offset_wh = 5 * np.log(eps + c_assigned_bb[:, 2:] /
c_anc[:, 2:])
    offset = np.concatenate([offset_xy, offset_wh],
axis=1)
    return offset
```

إذا لم يتم تخصيص مربع إحاطة الحقيقة الاساسية لمربع التحديد، فإننا نسمي فئة مربع التحديد على أنها "خلفية background". غالبًا ما يشار إلى مربعات التحديد التي تكون فصولهافي الخلفية باسم مربعات التحديد السلبية negative anchor boxes، ويطلق على الباقي صناديق التحديد الإيجابية positive anchor boxes. نقوم بتنفيذ دالة `multibox_target` التالية لتسمية الفئات والإزاحات لمربعات التحديد (وسيطة `anchors`) باستخدام مربعات إحاطة الحقيقة الاساسية (وسيطة `labels`). تحدد هذه الدالة فئة الخلفية إلى الصفر وتزيد من فهرس العدد الصحيح للفئة الجديدة بمقدار واحد.

```
#@save
def multibox_target(anchors, labels):
    """Label anchor boxes using ground-truth bounding
boxes."""
    batch_size, anchors = labels.shape[0],
anchors.squeeze(0)
    batch_offset, batch_mask, batch_class_labels = [],
[], []
    device, num_anchors = anchors.ctx, anchors.shape[0]
    for i in range(batch_size):
        label = labels[i, :, :]
        anchors_bbox_map = assign_anchor_to_bbox(
            label[:, 1:], anchors, device)
        bbox_mask =
np.tile((np.expand_dims((anchors_bbox_map >= 0),
                                                axis=-1)),
(1, 4)).astype('int32')
        # Initialize class labels and assigned bounding
box coordinates with
        # zeros
        class_labels = np.zeros(num_anchors,
dtype=np.int32, ctx=device)
        assigned_bb = np.zeros((num_anchors, 4),
dtype=np.float32,
                               ctx=device)
```

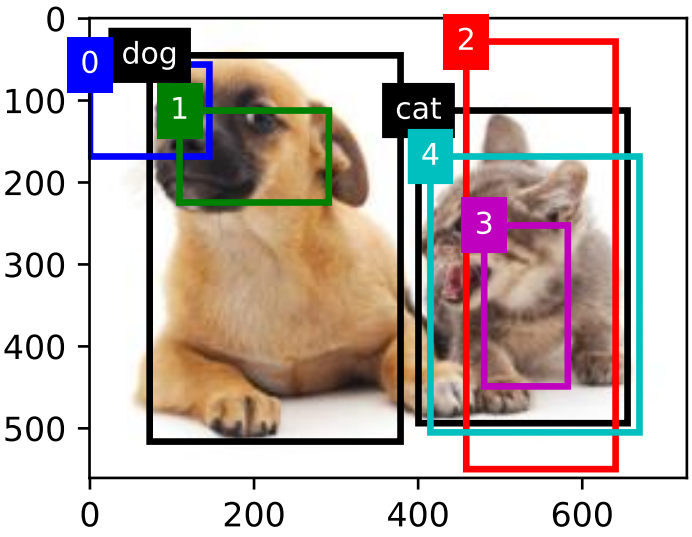
```
        # Label classes of anchor boxes using their
assigned ground-truth
        # bounding boxes. If an anchor box is not
assigned any, we label its
        # class as background (the value remains zero)
        indices_true = np.nonzero(anchors_bbox_map >=
0)[0]
        bb_idx = anchors_bbox_map[indices_true]
        class_labels[indices_true] = label[bb_idx,
0].astype('int32') + 1
        assigned_bb[indices_true] = label[bb_idx, 1:]
        # Offset transformation
        offset = offset_boxes(anchors, assigned_bb) *
bbox_mask
        batch_offset.append(offset.reshape(-1))
        batch_mask.append(bbox_mask.reshape(-1))
        batch_class_labels.append(class_labels)
    bbox_offset = np.stack(batch_offset)
    bbox_mask = np.stack(batch_mask)
    class_labels = np.stack(batch_class_labels)
    return (bbox_offset, bbox_mask, class_labels)
```

14.4.3.3. مثال An Example

دعنا نوضح تسمية صندوق التحديد عبر مثال ملموس. نحدد المربعات المحيطة بالحقيقة الاساسية للكلب والقطفي الصورة المحملة، حيث يكون العنصر الأول هو الفئة (0 للكلب و 1 للقط) والعناصر الأربعة المتبقية هي إحداثيات المحور(x, y) في الزاوية العلوية اليسرى و الزاوية اليمنى السفلية (النطاق بين 0 و 1). نقوم أيضًا ببناء خمس مربعات تحديد ليتم تسميتها باستخدام إحداثيات الزاوية العلوية اليسرى والزاوية اليمنى السفلية:$A_0, \ldots, A_4$ (يبدأ الفهرس من 0). ثم نرسم هذه المربعات المحيطة بالحقيقة الاساسية ومربعات التحديدفي الصورة.

```
ground_truth = np.array([[0, 0.1, 0.08, 0.52, 0.92],
                         [1, 0.55, 0.2, 0.9, 0.88]])
anchors = np.array([[0, 0.1, 0.2, 0.3], [0.15, 0.2, 0.4,
0.4],
                    [0.63, 0.05, 0.88, 0.98], [0.66,
0.45, 0.8, 0.8],
                    [0.57, 0.3, 0.92, 0.9]])

fig = d2l.plt.imshow(img)
show_bboxes(fig.axes, ground_truth[:, 1:] * bbox_scale,
['dog', 'cat'], 'k')
```

```
show_bboxes(fig.axes, anchors * bbox_scale, ['0', '1',
'2', '3', '4']);
```



باستخدام دالة `multibox_target` المحددة أعلاه، يمكننا تسمية فئات وإزاحات مربعات التحديد هذه بناءً على المربعات المحيطة بالحقيقة الاساسية للكلب والقط.في هذا المثال، مؤشرات فئات الخلفية والكلب والقط هي 0 و 1 و 2 على التوالي. أدناه نضيف بعدًا لأمثلة من مربعات التحديد وصناديق إحاطة الحقيقة الاساسية.

```
labels = multibox_target(np.expand_dims(anchors,
axis=0),
                         np.expand_dims(ground_truth,
axis=0))
```

هناك ثلاثة عناصرفي النتيجة التي تم إرجاعها، وكلها بتنسيق الموتر. يحتوي العنصر الثالث على الفئات المسماة لمربعات تحديد الإدخال.

دعنا نحلل تسميات الفئات التي تم إرجاعها أدناه بناءً على مربع التحديد ومواضع مربعات الاحاطة للحقيقة الاساسيةفي الصورة. أولاً، من بين جميع أزواج مربعات التحديد والصناديق المحيطة بالحقيقة الاساسية ، فإن IoU لصندوق التحديد والصندوق المحيط بالحقيقة الاساسية للقط هو الأكبر. وهكذا، فإن فئة A_4 من يسمى القط. أخذ أزواج تحتوي على A_4 او الصندوق المحيط بالحقيقة الاساسية للقط، من بين الباقي، يكون زوج صندوق التحديد A_1 والصندوق المحيط بالحقيقة الاساسية للكلب أكبر IoU. لذلك تم تصنيف فئة A_1 كـ "الكلب". بعد ذلك، نحتاج إلى المرور عبر مربعات الربط الثلاثة المتبقية غير المسماة: A_0وA_2 وA_3 و.

بالنسبة إلى A_0 فئة المربع المحيط بالحقيقة الاساسية مع أكبر IoU هو الكلب ، ولكن IoU أقل من العتبة المحددة مسبقًا (0.5) ، لذلك يتم تصنيف الفئة كخلفية ؛ بالنسبة إلى A_2، فئة الصندوق المحيط بالحقيقة الاساسية مع أكبر IoU هي القطة وتتجاوز IoU العتبة ، لذلك يُطلق على الفئة اسم القط ؛ بالنسبة إلى A_3، فإن فئة المربع المحيط بالحقيقة الاساسية مع أكبر وحدة IoU هي القطة ، ولكن القيمة أقل من الحد الأدنى ، لذلك يتم تصنيف الفئة كخلفية.

```
labels[2]
```

```
array([[0, 1, 2, 0, 2]], dtype=int32)
```

العنصر الثاني المرتجع هو متغير قناع mask variable للشكل (حجم الدُفعة، أربعة أضعاف عدد مربعات التحديد). تتوافق كل أربعة عناصرفي متغير القناع مع قيم الإزاحة الأربع لكل مربع ربط. نظرًا لأننا لا نهتم باكتشاف الخلفية، يجب ألا تؤثر إزاحات هذه الفئة السلبية على الدالة الهدف. من خلال عمليات الضرب الأولية، ستقوم الأصفار الموجودةفي متغير القناع بتصفية إزاحة الفئة السلبية قبل حساب الدلة الهدف.

```
labels[1]
```

```
array([[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
1, 1, 1, 1]],
      dtype=int32)
```

يحتوي العنصر الأول الذي تم إرجاعه على قيم الإزاحة الأربع المعنونة لكل مربع تحديد. لاحظ أن إزاحات مربعات التحديد من الفئة السالبة يتم تصنيفها على أنها أصفار.

```
labels[0]
```

```
array([[-0.00e+00, -0.00e+00, -0.00e+00, -0.00e+00,
1.40e+00,  1.00e+01,
         2.59e+00,  7.18e+00, -1.20e+00,  2.69e-01,
1.68e+00, -1.57e+00,
        -0.00e+00, -0.00e+00, -0.00e+00, -0.00e+00, -
5.71e-01, -1.00e+00,
         4.17e-06,  6.26e-01]])
```

14.4.4. توقع المربعات المحيطة مع عدم الحد الأقصى للكبت Predicting Bounding Boxes with Non-Maximum Suppression

أثناء التنبؤ، نقوم بإنشاء مربعات تحديد متعددة للصورة ونتوقع الفئات والإزاحات لكل منها. وهكذا يتم الحصول على الصندوق المحيط المتوقع وفقًا لصندوق التحديد بإزاحته المتوقعة. أدناه نقوم بتنفيذ دالة `offset_inverse` التي تأخذفي التحديد وتوقعات الإزاحة كمدخلات وتطبق تحويلات الإزاحة العكسية لإرجاع إحداثيات المربع المحيط المتوقع.

```
#@save
def offset_inverse(anchors, offset_preds):
    """Predict bounding boxes based on anchor boxes with
predicted offsets."""
    anc = d2l.box_corner_to_center(anchors)
    pred_bbox_xy = (offset_preds[:, :2] * anc[:, 2:] /
10) + anc[:, :2]
    pred_bbox_wh = np.exp(offset_preds[:, 2:] / 5) *
anc[:, 2:]
    pred_bbox = np.concatenate((pred_bbox_xy,
pred_bbox_wh), axis=1)
    predicted_bbox = d2l.box_center_to_corner(pred_bbox)
    return predicted_bbox
```

عندما يكون هناك العديد من مربعات التحديد، يمكن إخراج العديد من مربعات الإحاطة المتشابهة (مع تداخل كبير) لإحاطة نفس الكائن. لتبسيط الإخراج، يمكننا دمج مربعات إحاطة متوقعة مماثلة تنتمي إلى نفس الكائن باستخدام عدم الحد الأقصى للكبت (NMS) non-maximum suppression.

إليك كيفية NMS. بالنسبة لمربع الإحاطة المتوقع B ، يحسب نموذج اكتشاف الكائن الاحتمالية المتوقعة لكل فئة. بالدلالة على p الاحتمال الأكبر المتوقع، فإن الفئة المقابلة لهذا الاحتمال هي الفئة المتوقعة لـ B. على وجه التحديد، نشير إلى p كثقة confidence (درجة score) المربع المحيط المتوقع B.في نفس الصورة، يتم فرز جميع المربعات غير المتوقعة المحيطة بالخلفية حسب الثقة بترتيب تنازلي لإنشاء قائمة L. ثم نتعامل مع القائمة المصنفةفي الخطوات التالية:

1. حدد مربع الإحاطة المتوقع B_1 بأعلى مستوى من الثقة من L كأساس وقم بإزالة جميع المربعات المحيطة التي لا أساس لها والتي يتجاوز IoU مع B_1 عتبة محددًة مسبقًا ϵ من L.في هذه المرحلة،L يحتفظ بالمربع المحيط المتوقع بأعلى مستوى من الثقة، لكنه يتجاهل الآخرين الذين يشبهونه كثيرًا. باختصار، يتم كبت suppressed أولئك الذين لم يحصلوا على أقصى درجات ثقة.
2. حدد المربع المحيط المتوقع B_2 مع ثاني أعلى مستوى من الثقة من L كأساس آخر وقم بإزالة جميع المربعات المحيطة المتوقعة التي لا تعتمد على أساس والتي يوجد بها IoU مع B_2 تتجاوز ϵ من L.
3. كرر العملية المذكورة أعلاه حتى يتم استخدام جميع المربعات المحيطة المتوقعةفي L كأساس.في هذا الوقت، يكون IoU لأي زوج من المربعات المحيطة المتوقعةفي L أقل من العتبة ϵ ؛ وبالتالي، لا يوجد زوج متشابه للغاية مع بعضهما البعض.
4. قم بإخراج جميع المربعات المحيطة المتوقعةفي القائمة L.

تقوم دالة nms التالية بفرز درجات الثقة confidence scores بترتيب تنازلي وإرجاع مؤشراتها.

```
#@save
def nms(boxes, scores, iou_threshold):
    """Sort confidence scores of predicted bounding
boxes."""
    B = scores.argsort()[::-1]
    keep = []  # Indices of predicted bounding boxes
that will be kept
    while B.size > 0:
        i = B[0]
        keep.append(i)
        if B.size == 1: break
        iou = box_iou(boxes[i, :].reshape(-1, 4),
                      boxes[B[1:], :].reshape(-1,
4)).reshape(-1)
        inds = np.nonzero(iou <= iou_threshold)[0]
        B = B[inds + 1]
    return np.array(keep, dtype=np.int32, ctx=boxes.ctx)
```

نحدد التالي multibox_detection لتطبيق كبت غير أقصى NMS للتنبؤ بالمربعات المحيطة. لا تقلق إذا وجدت التنفيذ معقداً بعض الشيء: سوف نوضح كيف يعمل مع مثال ملموس بعد التنفيذ مباشرة.

```
#@save
def multibox_detection(cls_probs, offset_preds, anchors,
nms_threshold=0.5,
                       pos_threshold=0.009999999):
    """Predict bounding boxes using non-maximum
suppression."""
    device, batch_size = cls_probs.ctx,
cls_probs.shape[0]
    anchors = np.squeeze(anchors, axis=0)
    num_classes, num_anchors = cls_probs.shape[1],
cls_probs.shape[2]
    out = []
    for i in range(batch_size):
        cls_prob, offset_pred = cls_probs[i],
offset_preds[i].reshape(-1, 4)
        conf, class_id = np.max(cls_prob[1:], 0),
np.argmax(cls_prob[1:], 0)
```

```
        predicted_bb = offset_inverse(anchors,
offset_pred)
        keep = nms(predicted_bb, conf, nms_threshold)
        # Find all non-`keep` indices and set the class
to background
        all_idx = np.arange(num_anchors, dtype=np.int32,
ctx=device)
        combined = np.concatenate((keep, all_idx))
        unique, counts = np.unique(combined,
return_counts=True)
        non_keep = unique[counts == 1]
        all_id_sorted = np.concatenate((keep, non_keep))
        class_id[non_keep] = -1
        class_id =
class_id[all_id_sorted].astype('float32')
        conf, predicted_bb = conf[all_id_sorted],
predicted_bb[all_id_sorted]
        # Here `pos_threshold` is a threshold for
positive (non-background)
        # predictions
        below_min_idx = (conf < pos_threshold)
        class_id[below_min_idx] = -1
        conf[below_min_idx] = 1 - conf[below_min_idx]
        pred_info =
np.concatenate((np.expand_dims(class_id, axis=1),
                                    np.expand_dims(conf,
axis=1),
                                    predicted_bb), axis=1)
        out.append(pred_info)
    return np.stack(out)
```

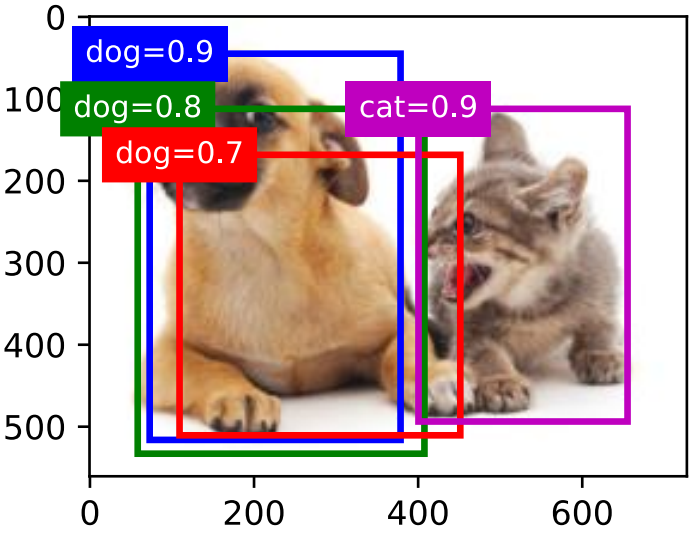
الآن دعنا نطبق التطبيقات المذكورة أعلاه على مثال ملموس بأربعة صناديق anchor boxes تحديد. للتبسيط، نفترض أن الإزاحات المتوقعة predicted offsets كلها أصفار. هذا يعني أن المربعات المحيطة المتوقعة هي مربعات تحديد. لكل فئة بين الخلفية والكلب والقط، نحدد أيضًا احتمالية توقعها.

```
anchors = np.array([[0.1, 0.08, 0.52, 0.92], [0.08, 0.2,
0.56, 0.95],
                    [0.15, 0.3, 0.62, 0.91], [0.55,
0.2, 0.9, 0.88]])
offset_preds = np.array([0] * d2l.size(anchors))
```

```
cls_probs = np.array([[0] * 4,  # Predicted background
likelihood
                      [0.9, 0.8, 0.7, 0.1],  # Predicted
dog likelihood
                      [0.1, 0.2, 0.3, 0.9]])  #
Predicted cat likelihood
```

يمكننا رسم هذه المربعات المحيطة المتوقعة مع ثقتهمفي الصورة.

```
fig = d2l.plt.imshow(img)
show_bboxes(fig.axes, anchors * bbox_scale,
            ['dog=0.9', 'dog=0.8', 'dog=0.7',
'cat=0.9'])
```



يمكننا الآن استدعاء دالة `multibox_detection` لإجراء كبت بدون حد اقصى، حيث يتم ضبط العتبة على 0.5. لاحظ أننا نضيف بُعدًا للأمثلةفي إدخال الموتر.

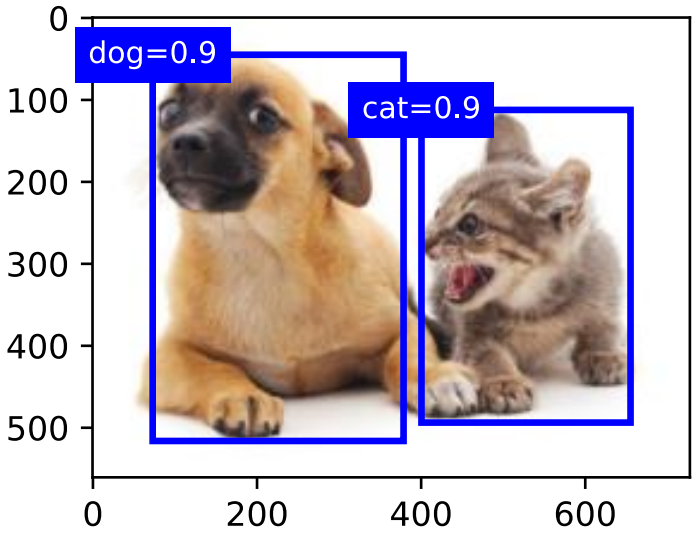
يمكننا أن نرى أن شكل النتيجة التي تم إرجاعها هو (حجم الدُفعة، عدد مربعات التحديد، 6). توفر العناصر الستةفي البعد الداخلي معلومات الإخراج لنفس المربع المحيط المتوقع. العنصر الأول هو فهرس الفئة المتنبأ به predicted class index، والذي يبدأ من 0 (0 هو كلب و1 قطة). تشير القيمة 1 ـ إلى الخلفية أو الإزالةفي حالة الكبت بدون حد اقصى. العنصر الثاني هو الثقة لمربع الاحاطة المتوقع. العناصر الأربعة المتبقية هي إحداثيات المحور (x, y)في الزاوية العلوية اليسرى والزاوية اليمنى السفلية لمربع الإحاطة المتوقع، على التوالي (النطاق بين 0 و 1).

```
output = multibox_detection(np.expand_dims(cls_probs, axis=0),
                            np.expand_dims(offset_preds, axis=0),
                            np.expand_dims(anchors, axis=0),
                            nms_threshold=0.5)
output
```

```
array([[[ 1.  ,  0.9 ,  0.55,  0.2 ,  0.9 ,  0.88],
        [ 0.  ,  0.9 ,  0.1 ,  0.08,  0.52,  0.92],
        [-1.  ,  0.8 ,  0.08,  0.2 ,  0.56,  0.95],
        [-1.  ,  0.7 ,  0.15,  0.3 ,  0.62,  0.91]]])
```

بعد إزالة تلك المربعات المحيطة المتوقعة للفئة 1 ـ ، يمكننا إخراج مربع الإحاطة النهائي المتوقع المحتفظ به من خلال عدم الحد الأقصى للكبت NMS.للكبت NMS

```
fig = d2l.plt.imshow(img)
for i in output[0].asnumpy():
    if i[0] == -1:
        continue
    label = ('dog=', 'cat=')[int(i[0])] + str(i[1])
    show_bboxes(fig.axes, [np.array(i[2:]) *
bbox_scale], label)
```



في الممارسة العملية، يمكننا إزالة مربعات الاحاطة المتوقعة بثقة أقل حتى قبل إجراء كبت غير أقصى، وبالتالي تقليل الحساب في هذه الخوارزمية. قد نقوم أيضًا بمعالجة ناتج عدم الحد الأقصى للكبت، على سبيل المثال، فقط من خلال الاحتفاظ بالنتائج بثقة أعلى في الناتج النهائي.

14.4.5. الملخص

- نقوم بإنشاء مربعات تحديد anchor boxes بأشكال مختلفة تتمحور حول كل بكسل من الصورة.
- يقيس التقاطع على الاتحاد (IoU)، المعروف أيضًا باسم مؤشر Jaccard، تشابه صندوقين احاطة. إنها نسبة منطقة تقاطعهم إلى منطقة اتحادهم.
- في مجموعة التدريب، نحتاج إلى نوعين من التسميات لكل صندوق تحديد. أحدهما هو فئة الكائن ذات الصلة بصندوق التحديد والآخر هو إزاحة المربع المحيط بالحقيقة الاساسية بالنسبة لمربع التحديد.
- أثناء التنبؤ، يمكننا استخدام عدم الحد الأقصى للكبت (NMS) لإزالة مربعات الإحاطة المتوقعة المماثلة، وبالتالي تبسيط الإخراج.

14.4.6. التمارين

1. قم بتغيير قيم `sizes` و `ratios` في دالة `multibox_prior`. ما هي التغييرات التي تم إجراؤها على مربعات التحديد التي تم إنشاؤها؟
2. قم ببناء وتصور صندوقين احاطة بوحدة IoU تبلغ 0.5. كيف تتداخل مع بعضها البعض؟
3. قم بتحديث المتغير `anchors` في القسم 14.4.3 والقسم 14.4.4. كيف تتغير النتائج؟
4. عدم الحد الأقصى للكبت هو خوارزمية جشعة greedy algorithm تمنع مربعات الاحاطة المتوقعة بإزالتها. هل من الممكن أن تكون بعض هذه الأشياء التي تمت إزالتها مفيدة بالفعل؟ كيف يمكن تعديل هذه الخوارزمية لكبتها suppress بنعومة softly؟ يمكنك الرجوع إلى Soft-NMS، (Bodla et al., 2017).
5. بدلاً من أن تكون مصنوعة يدويًا hand-crafted، هل يمكن تعلم عدم الحد الأقصى من الكبت NMS؟

14.5. كشف كائن متعدد القياسات Multiscale Object Detection

في القسم 14.4، أنشأنا عدة مربعات تحديد anchor boxes تتمحور حول كل بكسل من صورة الإدخال. تمثل مربعات التحديد بشكل أساسي عينات من مناطق مختلفة من الصورة. ومع ذلك، قد ينتهي بنا الأمر بوجود عدد كبير جدًا من مربعات التحديد التي لا يمكن حسابها إذا تم إنشاؤها لكل بكسل. فكرفي صورة إدخال 728×561. إذا تم إنشاء خمس مربعات تحديد بأشكال مختلفة لكل بكسل كمركز لها، فيجب تسمية أكثر من مليوني صندوق تحديد ($5 \times 728 \times 561$) والتنبؤ بها على الصورة.

14.5.1. مربعات التحديد متعددة القياسات Multiscale Anchor Boxes

قد تدرك أنه ليس من الصعب تقليل مربعات التحديد على الصورة. على سبيل المثال، يمكننا فقط أخذ عينات موحدة من جزء صغير من البكسل من صورة الإدخال لإنشاء مربعات تحديد تتمحور حولها. بالإضافة إلى ذلك، يمكننا على مستويات مختلفة إنشاء أعداد مختلفة من صناديق التحديد ذات الأحجام المختلفة. حدسيًا، من المرجح أن تظهر الكائنات الأصغر على صورة أكثر من الكائنات الأكبر حجمًا. على سبيل المثال، يمكن أن تظهر 1×1، 1×2 و2×1 و2×2 ، الكائنات على صورة 2×2في 4 و2 و1 بأربع طرق ممكنة على التوالي. لذلك، عند استخدام مربعات تحديد أصغر لاكتشاف الكائنات الأصغر، يمكننا أخذ عينات من مناطق أكثر، بينما بالنسبة للأجسام الأكبر حجمًا، يمكننا أخذ عينات من مناطق أقل.

لتوضيح كيفية إنشاء مربعات التحديد بمقاييس متعددة، دعنا نقرأ الصورة. يبلغ ارتفاعها وعرضها 561 و728 بكسل على التوالي.

```
%matplotlib inline
from mxnet import image, np, npx
from d2l import mxnet as d2l

npx.set_np()

img = image.imread('../img/catdog.jpg')
h, w = img.shape[:2]
h, w
```

```
(561, 728)
```

تذكر أنهفي القسم 7.2 نطلق على ناتج مصفوفة ثنائية الأبعاد لطبقة تلافيفية خريطة المعالم feature map. من خلال تحديد شكل خريطة المعالم، يمكننا تحديد مراكز مربعات التحديد التي تم أخذ عينات منها بشكل موحد على أي صورة.

تم تعريف دالة `display_anchors` أدناه. نقوم بإنشاء مربعات التحديد (`anchors`) على خريطة المعالم (`fmap`) مع كل وحدة (`pixel`) كمركز صندوق التحديد. نظرًا لأن قيم إحداثيات المحور (x, y)في مربعات التحديد (`anchors`) تم تقسيمها على عرض خريطة المعالم وارتفاعها (`fmap`)، فإن هذه القيم تقع بين 0 و1، والتي تشير إلى المواضع النسبية لمربعات التحديدفي خريطة المعالم.

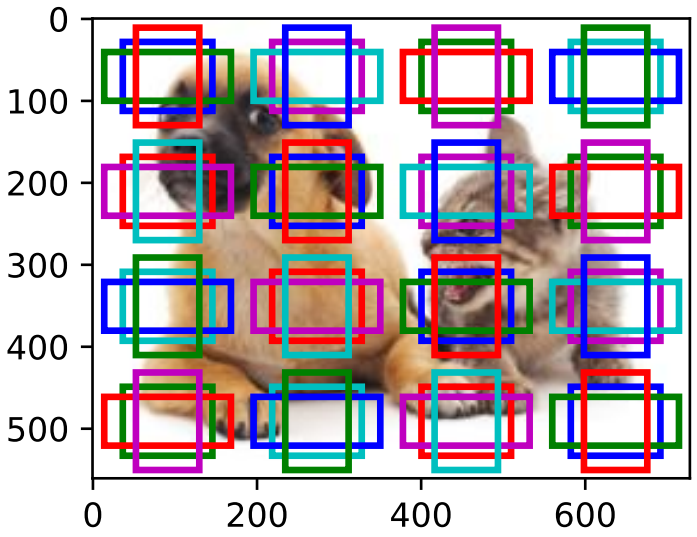
نظرًا لأن مراكز مربعات التحديد (`anchors`) منتشرة على جميع الوحدات على خريطة المعالم (`fmap`)، يجب توزيع هذه المراكز بشكل منتظم uniformly على أي صورة إدخال من حيث مواضعها المكانية النسبية. بشكل أكثر تحديداً، نظرًا لعرض وارتفاع خريطة المعالم `fmap_w`

و fmap_h ، على التوالي ، ستقوم الدالة التالية بأخذ عينات البكسل بشكل منتظمفي صفوف fmap_hوأعمدة fmap_w على أي صورة إدخال. بالتركيز على وحدات البكسل التي تم أخذ عينات منها بشكل منتظم، سيتم إنشاء مربعات تحديد ذات مقياس s (بافتراض أن طول القائمة s هو 1) ونسب أبعاد مختلفة (ratios).

```
def display_anchors(fmap_w, fmap_h, s):
    d2l.set_figsize()
    # Values on the first two dimensions do not affect
the output
    fmap = np.zeros((1, 10, fmap_h, fmap_w))
    anchors = npx.multibox_prior(fmap, sizes=s,
ratios=[1, 2, 0.5])
    bbox_scale = np.array((w, h, w, h))
    d2l.show_bboxes(d2l.plt.imshow(img.asnumpy()).axes,
                    anchors[0] * bbox_scale)
```

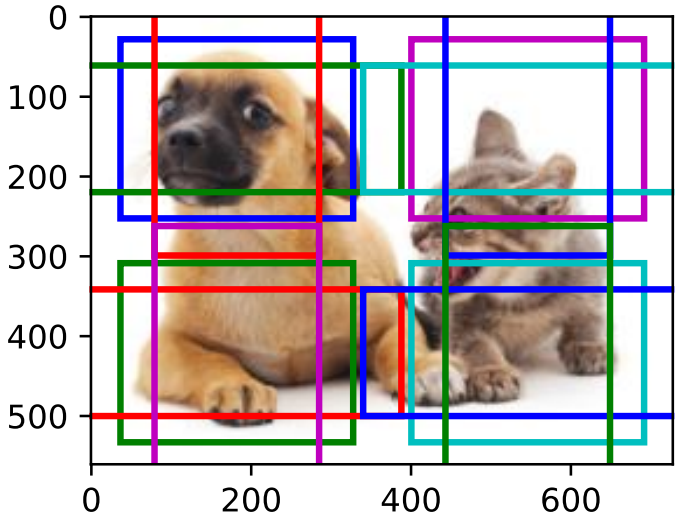
أولا، دعونا ننظرفي الكشف عن الأجسام الصغيرة. لتسهيل التمييز عند العرض، لا تتداخل مربعات التحديد ذات المراكز المختلفة هنا: يتم تعيين مقياس مربع التحديد على 0.15 ويتم تعيين ارتفاع وعرض خريطة المعالم على 4. يمكننا أن نرى أن المراكز من مربعات التحديدفي 4 صفوف و 4 أعمدة على الصورة موزعة بشكل منتظم.

```
display_anchors(fmap_w=4, fmap_h=4, s=[0.15])
```



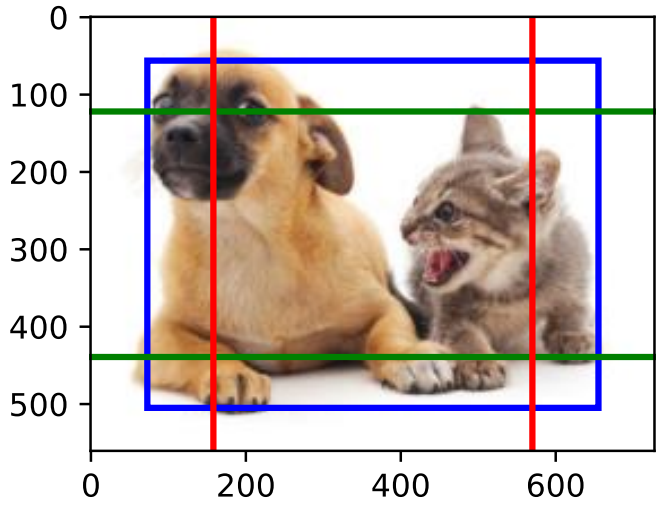
ننتقل إلى تقليل ارتفاع وعرض خريطة المعالم بمقدار النصف واستخدام مربعات تحديد أكبر لاكتشاف الكائنات الأكبر حجمًا. عندما يتم ضبط المقياس على 0.4، ستتداخل بعض مربعات التحديد مع بعضها البعض.

```
display_anchors(fmap_w=2, fmap_h=2, s=[0.4])
```



أخيرًا، قمنا بتقليل ارتفاع وعرض خريطة الميزة بمقدار النصف وزيادة مقياس مربع التحديد إلى 0.8. الآن مركز مربع التحديد هو مركز الصورة.

```
display_anchors(fmap_w=1, fmap_h=1, s=[0.8])
```



14.5.2. كشف متعدد القياسات Multiscale Detection

نظرًا لأننا أنشأنا صناديق تحديد متعددة النطاقات، فسنستخدمها لاكتشاف الكائنات ذات الأحجام المختلفة بمقاييس مختلفة. فيما يلي نقدم طريقة اكتشاف الكائنات متعددة القياسات المستندة إلى CNN والتي سنطبقهافي القسم 14.7.

في بعض المقاييس scales، لنفترض أن لدينا c خرائط معالم للشكل $h \times w$. باستخدام الطريقة الموضحةفي القسم 14.5.1، نقوم بإنشاء hw مجموعات من مربعات التحديد، حيث تحتوي كل مجموعة على صناديق تحديد a مع نفس المركز. على سبيل المثال،في المقياس الأولفي التجاربفي القسم 14.5.1، بالنظر إلى عشرة (عدد القنوات) 4×4 خرائط ميزات، قمنا بإنشاء 16 مجموعة من مربعات التحديد، حيث تحتوي كل مجموعة على 3 مربعات تحديد مع نفس المركز. بعد ذلك، يتم تسمية كل مربع تحديد بالفئة والإزاحة بناءً على مربعات الاحاطة بالحقيقة الاساسية.في المقياس الحالي، يحتاج نموذج اكتشاف الكائن إلى التنبؤ بفئات وإزاحات لـ hwمجموعات مربعات التحديدفي صورة الإدخال، حيث تحتوي المجموعات المختلفة على مراكز مختلفة.

افترض أن c خرائط المعالم هنا هي المخرجات الوسيطة التي تم الحصول عليها عن طريق الانتشار الأمامي لـ CNN بناءً على صورة الإدخال. نظرًا لوجود hw مواقع مكانية مختلفة على كل خريطة معالم، يمكن اعتبار نفس الموقع المكاني على أنه يحتوي على c وحدات. وفقًا لتعريف المجال الاستقبالي receptive fieldفي القسم 7.2، فإن هذه الوحدات الموجودةفي نفس الموقع المكاني لخرائط المعالم لها نفس المجال المستقبلي على صورة الإدخال: فهي تمثل معلومات صورة الإدخالفي نفس المجال الاستقبالي. لذلك، يمكننا تحويل وحدات خرائط المعالمفي نفس الموقع المكاني إلى فئات وإزاحات مربعات التحديد a التي تم إنشاؤها باستخدام هذا الموقع المكاني.في المضمون، نستخدم معلومات صورة الإدخالفي مجال استقبالي معين للتنبؤ بفئات وإزاحات مربعات التحديد القريبة من هذا المجال الاستقباليفي صورة الإدخال.

عندما تحتوي خرائط المعالمفي طبقات مختلفة على حقول متفاوتة الأحجام على صورة الإدخال، يمكن استخدامها لاكتشاف الكائنات ذات الأحجام المختلفة. على سبيل المثال، يمكننا تصميم شبكة عصبية حيث تحتوي وحدات خرائط المعالم الأقرب إلى طبقة المخرجات على حقول استقبالية أوسع، بحيث يمكنها اكتشاف كائنات أكبر من صورة الإدخال.

باختصار، يمكننا الاستفادة من التمثيلات الطبقية للصور على مستويات متعددة بواسطة الشبكات العصبية العميقة لاكتشاف الكائنات متعددة القياسات. سنوضح كيف يعمل هذا من خلال مثال ملموسفي القسم 14.7.

14.5.3. الملخص

- على قياسات متعددة multiple scales، يمكننا إنشاء صناديق (مربعات) تحديد anchor boxes بأحجام مختلفة لاكتشاف الكائنات ذات الأحجام المختلفة.
- من خلال تحديد شكل خرائط المعالم feature maps، يمكننا تحديد مراكز مربعات التحديد التي تم أخذ عينات منها بشكل منتظم على أي صورة.
- نستخدم معلومات صورة الإدخالفي مجال استقبالي receptive field معين للتنبؤ بفئات classes وإزاحات offsets مربعات التحديد القريبة من هذا المجال الاستقبالي في صورة الإدخال.
- من خلال التعلم العميق، يمكننا الاستفادة من التمثيلات الطبقية layerwise representations للصور على مستويات متعددة لاكتشاف الكائنات متعددة النطاقات.

14.5.4. التمارين

1. وفقًا لمناقشاتنافي القسم 8.1، تتعلم الشبكات العصبية العميقة ميزات هرمية مع مستويات متزايدة من التجريد للصور.في اكتشاف الكائنات متعددة القياسات multiscale object detection، هل تتوافق خرائط المعالم بمقاييس مختلفة مع مستويات مختلفة من التجريد؟ لما ولما لا؟
2. في المقياس الأول (`fmap_w=4`، `fmap_h=4`)في التجاربفي القسم 14.5.1، قم بإنشاء مربعات تحديد موزعة بشكل منتظم والتي قد تتداخل.
3. إعطاء متغير خريطة المعالم بالشكل $1 \times c \times h \times w$، حيث c وh وw عدد القنوات، والارتفاع، والعرض لخرائط المعالم، على التوالي. كيف يمكنك تحويل هذا المتغير إلى فئات وإزاحات مربعات التحديد؟ ما هو شكل المخرجات؟

14.6. مجموعة بيانات اكتشاف الكائن The Object Detection Dataset

لا توجد مجموعة بيانات dataset صغيرة مثل MNIST وFashion-MNISTفي مجال اكتشاف الأشياء. من أجل عرض نماذج اكتشاف الكائنات بسرعة، قمنا بتجميع مجموعة بيانات صغيرة وتسميتها. أولاً، التقطنا صور موز مجانية من مكتبنا وأنتجنا 1000 صورة موز بدورات وأحجام مختلفة. ثم وضعنا كل صورة موزةفي موضع عشوائي على صورة خلفية ما.في النهاية، قمنا بتسمية مربعات الاحاطة bounding boxes لتلك الموز على الصور.

14.6.1. تنزيل مجموعة البيانات Downloading the Dataset

يمكن تنزيل مجموعة بيانات اكتشاف الموز مع جميع الصور وملفات تسمية csv مباشرة من الإنترنت.

```
%matplotlib inline
import os
import pandas as pd
from mxnet import gluon, image, np, npx
from d2l import mxnet as d2l

npx.set_np()

#@save
d2l.DATA_HUB['banana-detection'] = (
    d2l.DATA_URL + 'banana-detection.zip',
    '5de26c8fce5ccdea9f91267273464dc968d20d72')
```

14.6.2. قراءة مجموعة البيانات Reading the Dataset

سنقوم بقراءة مجموعة بيانات اكتشاف الموزفي دالة `read_data_bananas` أدناه. تتضمن مجموعة البيانات ملف csv لتسميات فئة الكائن وإحداثيات مربع الاحاطة بالحقيقة الاساسيةفي الزوايا العلوية اليسرى والسفلية اليمنى.

```
#@save
def read_data_bananas(is_train=True):
    """Read the banana detection dataset images and
labels."""
    data_dir = d2l.download_extract('banana-detection')
    csv_fname = os.path.join(data_dir, 'bananas_train'
if is_train
                             else 'bananas_val',
'label.csv')
    csv_data = pd.read_csv(csv_fname)
    csv_data = csv_data.set_index('img_name')
    images, targets = [], []
    for img_name, target in csv_data.iterrows():
        images.append(image.imread(
            os.path.join(data_dir, 'bananas_train' if
is_train else
                         'bananas_val', 'images',
f'{img_name}')))
```

```
        # Here `target` contains (class, upper-left x,
upper-left y,
        # lower-right x, lower-right y), where all the
images have the same
        # banana class (index 0)
        targets.append(list(target))
    return images, np.expand_dims(np.array(targets), 1)
/ 256
```

باستخدام دالة `read_data_bananas` لقراءة الصور والتسميات، ستسمح لنا فئة `BananasDataset` التالية بإنشاء مثيل مجموعة بيانات مخصص لتحميل مجموعة بيانات اكتشاف الموز.

```
#@save
class BananasDataset(gluon.data.Dataset):
    """A customized dataset to load the banana detection
dataset."""
    def __init__(self, is_train):
        self.features, self.labels =
read_data_bananas(is_train)
        print('read ' + str(len(self.features)) + (f'
training examples' if
              is_train else f' validation examples'))

    def __getitem__(self, idx):
        return
(self.features[idx].astype('float32').transpose(2, 0,
1),
                self.labels[idx])

    def __len__(self):
        return len(self.features)
```

أخيرًا، نحدد دالة `load_data_bananas` لإرجاع مثيلين لمكرر البيانات لكل من مجموعات التدريب والاختبار. بالنسبة لمجموعة بيانات الاختبار، ليست هناك حاجة لقراءتها بترتيب عشوائي.

```
#@save
def load_data_bananas(batch_size):
    """Load the banana detection dataset."""
    train_iter =
gluon.data.DataLoader(BananasDataset(is_train=True),
```

```
                                        batch_size,
shuffle=True)
    val_iter =
gluon.data.DataLoader(BananasDataset(is_train=False),
                                          batch_size)
    return train_iter, val_iter
```

دعونا نقرأ الدفعات الصغيرة minibatch ونطبع أشكال كل من الصور والتسميات في هذه الدفعة الصغيرة. يبدو شكل الدفعة الصغيرة للصورة (حجم الدُفعة، عدد القنوات، الارتفاع، العرض) مألوفًا: هو نفسه كما في مهام تصنيف الصور السابقة. شكل التسمية للدفعة الصغيرة هو (حجم الدُفعة، m، 5)، حيث m يمثل أكبر عدد ممكن من مربعات الاحاطة التي تحتوي عليها أي صورة في مجموعة البيانات.

على الرغم من أن الحساب في الدفعات الصغيرة أكثر كفاءة، إلا أنه يتطلب أن تحتوي جميع أمثلة الصور على نفس عدد مربعات الاحاطة لتشكيل الدفعة الصغيرة عبر التسلسل concatenation. بشكل عام، قد تحتوي الصور على عدد متنوع من مربعات الاحاطة؛ وبالتالي، سيتم تعبئة الصور التي تحتوي على عدد أقل من m مربعات الإحاطة بمربعات إحاطة غير قانونية حتى يتم الوصول إليها. ثم يتم تمثيل تسمية كل مربع احاطة بمصفوفة طولها 5. العنصر الأول في المصفوفة هو فئة الكائن في مربع الاحاطة، حيث يشير 1 ـ إلى مربع إحاطة غير قانوني للحشو padding. العناصر الأربعة المتبقية من المصفوفة هي القيم المنسقة (x,y) للركن الأيسر العلوي والزاوية اليمنى السفلية من مربع الاحاطة (النطاق بين 0 و1). بالنسبة لمجموعة بيانات الموز، نظرًا لوجود مربع احاطة واحد فقط في كل صورة، لدينا.

```
batch_size, edge_size = 32, 256
train_iter, _ = load_data_bananas(batch_size)
batch = next(iter(train_iter))
batch[0].shape, batch[1].shape
```

```
Downloading ../data/banana-detection.zip from
http://d2l-data.s3-accelerate.amazonaws.com/banana-
detection.zip...
read 1000 training examples
read 100 validation examples
```

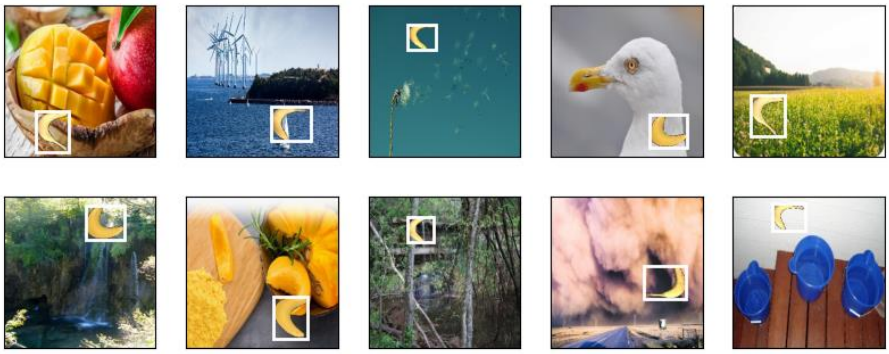
```
((32, 3, 256, 256), (32, 1, 5))
```

14.6.3. التوضيح Demonstration

دعنا نعرض عشر صور مع مربعات الاحاطة للحقيقة الاساسية. يمكننا أن نرى أن دوران الموز وأحجامه ومواضعه تختلف عبر كل هذه الصور. بالطبع، هذه مجرد مجموعة بيانات اصطناعية

artificial dataset بسيطة. من الناحية العملية، عادةً ما تكون مجموعات البيانات الواقعية real-world datasets أكثر تعقيداً.

```
imgs = (batch[0][:10].transpose(0, 2, 3, 1)) / 255
axes = d2l.show_images(imgs, 2, 5, scale=2)
for ax, label in zip(axes, batch[1][:10]):
    d2l.show_bboxes(ax, [label[0][1:5] * edge_size],
colors=['w'])
```



14.6.4. الملخص

- يمكن استخدام مجموعة بيانات اكتشاف الموز التي جمعناها لتوضيح نماذج اكتشاف الكائنات.
- تحميل البيانات لاكتشاف الكائن مشابه لذلك الخاص بتصنيف الصور. ومع ذلك،في اكتشاف الكائن، تحتوي التسميات labels أيضًا على معلومات عن مربعات الاحاطة بالحقيقة الاساسية، وهي مفقودةفي تصنيف الصور.

14.6.5. التمارين

1. اعرض صورًا أخرى باستخدام مربعات للحقيقة الأرضيةفي مجموعة بيانات اكتشاف الموز. كيف تختلف فيما يتعلق بمربعات الاحاطة والأشياء؟
2. لنفترض أننا نريد تطبيق زيادة البيانات data augmentation، مثل الاقتصاص العشوائي random cropping، على اكتشاف الكائنات. كيف يمكن أن يختلف عن ذلكفي تصنيف الصور؟ تلميح: ماذا لو احتوت الصورة التي تم اقتصاصها على جزء صغير فقط من كائن؟

14.7. اكتشاف المربعات المتعددة ذو اللقطة الواحدة Single Shot Multibox Detection

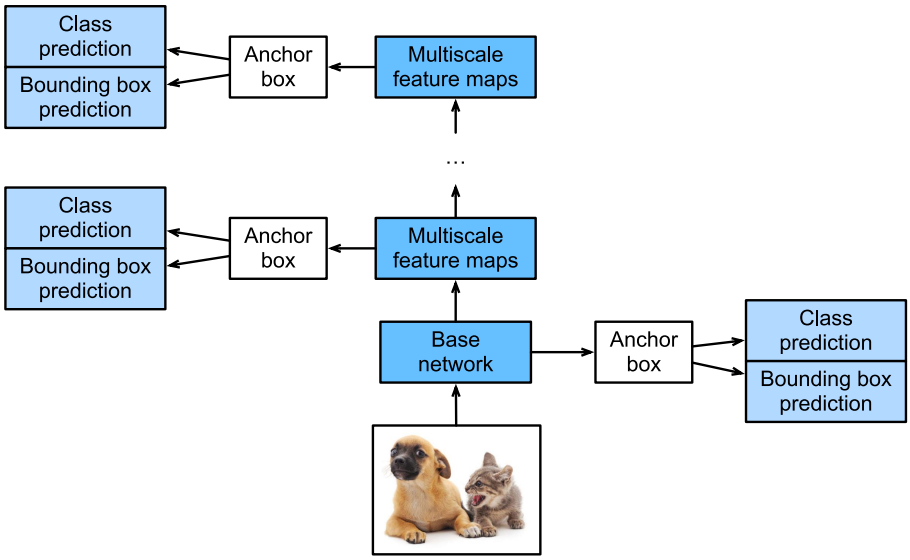
في القسم 14.3 ـ القسم 14.6، قدمنا مربعات الإحاطة bounding boxes ومربعات التحديد anchor boxes واكتشاف الكائنات متعددة القياسات multiscale object detection ومجموعة البيانات لاكتشاف الكائنات dataset for object detection. نحن الآن جاهزون لاستخدام هذه المعرفة الأساسية لتصميم نموذج اكتشاف الكائن object detection model: اكتشاف لقطة واحدة متعددة المربعات single shot multibox detection (SSD)، (Liu et al.,2016). هذا النموذج بسيط وسريع وشائع الاستخدام. على الرغم من أن هذا مجرد نموذج واحد من الكميات الهائلة من نماذج اكتشاف الكائنات، إلا أن بعض مبادئ التصميم وتفاصيل التنفيذ الواردةفي هذا القسم تنطبق أيضًا على نماذج أخرى.

14.7.1. النموذج Model

يقدم الشكل 14.7.1 نظرة عامة على تصميم اكتشاف المربعات المتعددة ذو اللقطة الواحدة. يتكون هذا النموذج بشكل أساسي من شبكة أساسية متبوعة بالعديد من مجموعات خرائط المعالم متعددة القياسات. الشبكة الأساسية مخصصة لاستخراج الميزات من صورة الإدخال، حتى تتمكن من استخدام شبكة CNN عميقة. على سبيل المثال، تتبنى مقالة الكشف عن اكتشاف المربعات المتعددة ذو اللقطة الواحدة الأصلية شبكة VGG مقطوعة قبل طبقة التصنيف (Liu et al.، 2016)، بينما تم أيضًا استخدام ResNet بشكل شائع. من خلال تصميمنا، يمكننا أن نجعل خرائط المعالم أكبر لإخراج الشبكة الأساسية وذلك لإنشاء المزيد من مربعات التحديد لاكتشاف الكائنات الأصغر. بعد ذلك، تقلل كل كتلة خريطة معالم متعددة القياسات (على سبيل المثال، بمقدار النصف) من ارتفاع وعرض خرائط المعالم من الكتلة السابقة، وتمكن كل وحدة من خرائط المعالم من زيادة مجالها المستقبلي على صورة الإدخال.

استرجع تصميم اكتشاف كائن متعدد القياسات من خلال التمثيل الطبقي للصور بواسطة الشبكات العصبية العميقةفي القسم 14.5. نظرًا لأن خرائط الميزات متعددة القياسات الأقرب إلى الجزء العلوي من الشكل 14.7.1 أصغر حجمًا ولكنها تحتوي على حقول استقبالية أكبر، فهي مناسبة للكشف عن كائنات أقل ولكن أكبر.

باختصار، عبر شبكتها الأساسية والعديد من مجموعات خرائط المعالم متعددة القياسات، يُنشئ اكتشاف المربعات المتعددة ذو اللقطة الواحدة عددًا متفاوتًا من مربعات التحديد بأحجام مختلفة، ويكتشف الكائنات ذات الأحجام المتفاوتة من خلال التنبؤ بفئات وإزاحات مربعات التحديد هذه (وبالتالي مربعات الاحاطة)؛ وبالتالي، يعد هذا نموذجًا للكشف عن كائن متعدد القياسات.



الشكل 14.7.1 كنموذج لاكتشاف كائن متعدد القياسات، يتكون اكتشاف المربعات المتعددة ذو اللقطة الواحدة بشكل أساسي من شبكة أساسية متبوعة بالعديد من مجموعات خرائط المعالم متعددة القياسات.

فيما يلي، سنصف تفاصيل تنفيذ المجموعات المختلفةفي الشكل 14.7.1. بادئ ذي بدء، نناقش كيفية تنفيذ التنبؤ بالفئة ومربع الاحاطة.

14.7.1.1. طبقة التنبؤ الطبقي Class Prediction Layer

ليكن عدد فئات الكائن q. ثم تحتوي مربعات الربط على فئات $q + 1$ ، حيث تكون الفئة 0 هي الخلفية.في بعض القياسات، افترض أن ارتفاع وعرض خرائط المعالم هما h و w على التوالي. عندما يتم إنشاء مربعات التحديد مع كل موقع مكاني لخرائط المعالم هذه كمركز لها، يجب تصنيف إجمالي مربعات التحديد hwa. هذا غالبًا ما يجعل التصنيف بطبقات متصلة بالكامل غير ممكن بسبب تكاليف المعلمات الثقيلة المحتملة. تذكر كيف استخدمنا قنوات الطبقات التلافيفية للتنبؤ بالفئاتفي القسم 8.3. يستخدم اكتشاف المربعات المتعددة ذو اللقطة الواحدة نفس الأسلوب لتقليل تعقيد النموذج.

على وجه التحديد، تستخدم طبقة التنبؤ بالفئة طبقة تلافيفية بدون تغيير عرض أو ارتفاع خرائط المعالم. بهذه الطريقة، يمكن أن يكون هناك تطابق واحد لواحد بين المخرجات والمدخلات بنفس الأبعاد المكانية (العرض والارتفاع) لخرائط المعالم. بشكل أكثر تحديداً، تمثل قنوات خرائط معالم المخرجاتفي أي موضع مكاني $(x، y)$ تنبؤات الفئة لجميع مربعات التحديد المتمركزةفي $(x، y)$ من خرائط معالم الإدخال. لإنتاج تنبؤات صحيحة، يجب أن تكون هناك

قنوات إخراج $a(q+1)$ ، حيث تمثل قناة الإخراج مع الفهرس $i(q+1)+j$ ،في نفس الموضع المكاني، تنبؤ الفئة $(0 \leq j \leq q)$ لمربع التحديد $i(0 \leq i < a)$.

نحدد أدناه طبقة التنبؤ بالفئة هذه، مع تحديد a وq عبر الوسيطات num_anchors و num_classes، على التوالي. تستخدم هذه الطبقة طبقة تلافيفية 3×3 مع حشوة 1. يبقى عرض وارتفاع مدخلات ومخرجات هذه الطبقة التلافيفية دون تغيير.

```
%matplotlib inline
from mxnet import autograd, gluon, image, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()

def cls_predictor(num_anchors, num_classes):
    return nn.Conv2D(num_anchors * (num_classes + 1),
kernel_size=3,
                     padding=1)
```

14.7.1.2. طبقة توقع مربع الإحاطة Bounding Box Prediction Layer

يشبه تصميم طبقة التنبؤ بمربع الإحاطة bounding box prediction layer تصميم طبقة التنبؤ بالفئة class prediction layer. يكمن الاختلاف الوحيدفي عدد المخرجات لكل مربع تحديد: هنا نحتاج إلى توقع أربعة ازاحات offsets بدلاً من الفئات $q+1$.

```
def bbox_predictor(num_anchors):
    return nn.Conv2D(num_anchors * 4, kernel_size=3,
padding=1)
```

14.7.1.3. التنبؤات المتسلسلة لمقاييس متعددة Concatenating Predictions for Multiple Scales

كما ذكرنا، يستخدم اكتشاف الصندوق المتعدد ذو اللقطة الواحدة خرائط معالم متعددة النطاقات لإنشاء مربعات التحديد والتنبؤ بفئاتها وإزاحاتها. على مستويات مختلفة، قد تختلف أشكال خرائط المعالم أو عدد مربعات التحديد المتمركزةفي نفس الوحدة. لذلك، قد تختلف أشكال مخرجات التنبؤ بمقاييس مختلفة.

في المثال التالي، نقوم بإنشاء خرائط معالم بمقياسين مختلفين، Y1 و Y2 ، لنفس الدفعة الصغيرة ، حيث يكون ارتفاع وعرض Y2 نصف تلك الخاصة بـ Y1. لنأخذ تنبؤ الفئة كمثال. افترض أنه تم إنشاء 5 و 3 صناديق ربط لكل وحدةفي Y1 و Y2 ، على التوالي. افترض أيضًا أن عدد فئات الكائنات هو 10. بالنسبة لخرائط المعالم Y1 و Y2 ، تكون أعداد القنواتفي

مخرجات تنبؤ الفئة هي $55 = (1 + 10) \times 5$ و$33 = (1 + 10) \times 3$ على التوالي ، حيث يكون أي من شكل المخرجات هو (حجم الدُفعة ، عدد القنوات ، الارتفاع ، العرض).

```
def forward(x, block):
    block.initialize()
    return block(x)

Y1 = forward(np.zeros((2, 8, 20, 20)), cls_predictor(5,
10))
Y2 = forward(np.zeros((2, 16, 10, 10)), cls_predictor(3,
10))
Y1.shape, Y2.shape
```

```
((2, 55, 20, 20), (2, 33, 10, 10))
```

كما نرى، باستثناء أبعاد حجم الدُفعة، فإن الأبعاد الثلاثة الأخرى جميعها لها أحجام مختلفة. لتسلسل مخرجات التنبؤ هذه للحصول على حساب أكثر كفاءة، سنقوم بتحويل هذه الموترات إلى تنسيق أكثر اتساقًا.

لاحظ أن بُعد القناة يحمل تنبؤات مربعات التحديد بنفس المركز. ننقل هذا البعد أولاً إلى الأعمق. نظرًا لأن حجم الدُفعة يظل كما هو بالنسبة للمقاييس المختلفة، فيمكننا تحويل ناتج التنبؤ إلى موتر ثنائي الأبعاد مع الشكل (حجم الدُفعة، العرض × الارتفاع × عدد القنوات). بعد ذلك يمكننا أن نسلسل هذه المخرجات بمقاييس مختلفة على طول البعد 1.

```
def flatten_pred(pred):
    return npx.batch_flatten(pred.transpose(0, 2, 3, 1))

def concat_preds(preds):
    return np.concatenate([flatten_pred(p) for p in
preds], axis=1)
```

بهذه الطريقة، على الرغم من أن Y1 و Y2 لهما أحجام مختلفةفي القنوات والارتفاعات والعرض، فلا يزال بإمكاننا ربط مخرجات التنبؤ هذين بمقياسين مختلفين لنفس الدفعة الصغيرة.

```
concat_preds([Y1, Y2]).shape
```

```
(2, 25300)
```

14.7.1.4. كتلة الاختزال Downsampling Block

لاكتشاف الكائنات على مستويات متعددة، نحدد كتلة الاختزال التالية down_sample_blk التي تقسم ارتفاع وعرض خرائط معالم الإدخال إلى النصف.في الواقع، تطبق هذه الكتلة تصميم كتل VGGفي القسم 8.2.1. بشكل أكثر تحديدًا، تتكون كل كتلة اختزال من طبقتين تلافيفيتين

3×3 مع حشوة 1 متبوعة بطبقة تجميع بحد أقصى 2×2 بخطوة 2. كما نعلم، لا تغير الطبقات التلافيفية 3×3 ذات الحشو 1 شكل خرائط المعالم. ومع ذلك، فإن تجميع بحد أقصى 2×2 اللاحق يقلل من ارتفاع وعرض خرائط معالم الإدخال بمقدار النصف. لكل من خرائط ميزات الإدخال والإخراج لكتلة الاختزال هذه، لأن $1 \times 2 + (3 - 1) + (3 - 1) = 6$, كل وحدةفي المخرجات لها مجال استقبالي 6×6 على الإدخال. لذلك، تعمل كتلة الاختزال على توسيع المجال الاستقبالي لكل وحدةفي خرائط معالم الإخراج الخاصة بها.

```
def down_sample_blk(num_channels):
    blk = nn.Sequential()
    for _ in range(2):
        blk.add(nn.Conv2D(num_channels, kernel_size=3,
padding=1),
                nn.BatchNorm(in_channels=num_channels),
                nn.Activation('relu'))
    blk.add(nn.MaxPool2D(2))
    return blk
```

في المثال التالي، تقوم كتلة الاختزال المُنشأة لدينا بتغيير عدد قنوات الإدخال وتقليل ارتفاع وعرض خرائط معالم الإدخال إلى النصف.

```
forward(np.zeros((2, 3, 20, 20)),
down_sample_blk(10)).shape
```

```
(2, 10, 10, 10)
```

14.7.1.5. كتلة الشبكة الأساسية Base Network Block

يتم استخدام كتلة الشبكة الأساسية لاستخراج الميزات من صور الإدخال. من أجل التبسيط، نقوم ببناء شبكة أساسية صغيرة تتكون من ثلاث كتل اختزال تضاعف عدد القنواتفي كل كتلة. بالنظر إلى صورة الإدخال 256×256 ، تُخرج كتلة الشبكة الأساسية هذه خرائط معالم 32×32 $(256/2^3 = 32)$.

```
def base_net():
    blk = nn.Sequential()
    for num_filters in [16, 32, 64]:
        blk.add(down_sample_blk(num_filters))
    return blk

forward(np.zeros((2, 3, 256, 256)), base_net()).shape
```

```
(2, 64, 32, 32)
```

14.7.1.6. النموذج الكامل The Complete Model

يتكون نموذج الكشف الكامل متعددة الصناديق ذو اللقطة الواحدة من خمس كتل. تُستخدم خرائط المعالم التي تنتجها كل كتلة لكل من (1) إنشاء مربعات التحديد و (2) التنبؤ بفئات وإزاحات مربعات التحديد هذه. من بين هذه الكتل الخمس، الأولى هي كتلة الشبكة الأساسية، والثانية إلى الرابعة هي كتل الاختزال، والكتلة الأخيرة تستخدم للتجميع الحد الأقصى العالمي لتقليل الارتفاع والعرض إلى 1. من الناحية الفنية، من الثانية إلى الخامسة. هي جميع مجموعات خرائط المعالم متعددة القياساتفي الشكل 14.7.1.

```
def get_blk(i):
    if i == 0:
        blk = base_net()
    elif i == 4:
        blk = nn.GlobalMaxPool2D()
    else:
        blk = down_sample_blk(128)
    return blk
```

الآن نحدد الانتشار الأمامي لكل كتلة. تختلف المخرجات هنا عن مهام تصنيف الصور، وتشمل (1) خرائط معالم CNN Y ، (2) مربعات التحديد التي تم إنشاؤها باستخدام Y بالمقياس الحالي ، و (3) الفئات والإزاحات المتوقعة (بناءً على Y) لمربعات التحديد هذه.

```
def blk_forward(X, blk, size, ratio, cls_predictor,
bbox_predictor):
    Y = blk(X)
    anchors = d2l.multibox_prior(Y, sizes=size,
ratios=ratio)
    cls_preds = cls_predictor(Y)
    bbox_preds = bbox_predictor(Y)
    return (Y, anchors, cls_preds, bbox_preds)
```

تذكر أنهفي الشكل 14.7.1، فإن مجموعة خريطة المعالم متعددة القياسات الأقرب إلى الأعلى مخصصة لاكتشاف الأجسام الأكبر؛ وبالتالي، فإنه يحتاج إلى إنشاء صناديق تحديد أكبر.في الانتشار الأمامي أعلاه،في كل كتلة خريطة ميزة متعددة القياسات، نمررفي قائمة من قيمتي مقياس عبر وسيطة `sizes` لدالة `multibox_prior` التي تم استدعاؤها (الموضحةفي القسم 14.4). فيما يلي، يتم تقسيم الفاصل بين 0.2 و 1.05 بالتساوي إلى خمسة أقسام لتحديد قيم المقياس الأصغرفي الكتل الخمس: 0.2 ، 0.37 ، 0.54 ، 0.71 ، و 0.88. ثم يتم إعطاء قيم المقياس الأكبر بواسطة، $\sqrt{0.2 \times 0.37} = 0.272$ و$\sqrt{0.37 \times 0.54} = 0.447$ وهكذا.

```
sizes = [[0.2, 0.272], [0.37, 0.447], [0.54, 0.619],
[0.71, 0.79],
```

```
          [0.88, 0.961]]
ratios = [[1, 2, 0.5]] * 5
num_anchors = len(sizes[0]) + len(ratios[0]) - 1
```

الآن يمكننا تحديد نموذج `TinySSD` الكامل على النحو التالي.

```
class TinySSD(nn.Block):
    def __init__(self, num_classes, **kwargs):
        super(TinySSD, self).__init__(**kwargs)
        self.num_classes = num_classes
        for i in range(5):
            # Equivalent to the assignment statement
`self.blk_i = get_blk(i)`
            setattr(self, f'blk_{i}', get_blk(i))
            setattr(self, f'cls_{i}',
cls_predictor(num_anchors, num_classes))
            setattr(self, f'bbox_{i}',
bbox_predictor(num_anchors))

    def forward(self, X):
        anchors, cls_preds, bbox_preds = [None] * 5,
[None] * 5, [None] * 5
        for i in range(5):
            # Here `getattr(self, 'blk_%d' % i)`
accesses `self.blk_i`
            X, anchors[i], cls_preds[i], bbox_preds[i] =
blk_forward(
                X, getattr(self, f'blk_{i}'), sizes[i],
ratios[i],
                getattr(self, f'cls_{i}'), getattr(self,
f'bbox_{i}'))
        anchors = np.concatenate(anchors, axis=1)
        cls_preds = concat_preds(cls_preds)
        cls_preds = cls_preds.reshape(
            cls_preds.shape[0], -1, self.num_classes +
1)
        bbox_preds = concat_preds(bbox_preds)
        return anchors, cls_preds, bbox_preds
```

نقوم بإنشاء مثيل نموذج ونستخدمه لإجراء انتشار أمامي على دفعة صغيرة من 256×256 الصور X.

كما هو موضح سابقًافي هذا القسم، تتميز مخرجات الكتلة الأولى 32 × 32 بخرائط معالم. تذكر أن كتل الاختزال من الثانية إلى الرابعة تقسم الارتفاع والعرض إلى النصف وأن الكتلة الخامسة تستخدم التجميع العام. نظرًا لأنه يتم إنشاء 4 مربعات تحديد لكل وحدة على طول الأبعاد المكانية لخرائط المعالم،في جميع المقاييس الخمسة، يتم إنشاء إجمالي $(32^2 + 16^2 + 8^2 + 4^2 + 1) \times 4 = 5444$ مربعات التحديد لكل صورة.

```
net = TinySSD(num_classes=1)
net.initialize()
X = np.zeros((32, 3, 256, 256))
anchors, cls_preds, bbox_preds = net(X)

print('output anchors:', anchors.shape)
print('output class preds:', cls_preds.shape)
print('output bbox preds:', bbox_preds.shape)
```

14.7.2. التدريب Training

سنشرح الآن كيفية تدريب اكتشاف المربعات المتعددة ذو اللقطة الواحدة SSD لاكتشاف الكائن.

14.7.2.1. قراءة مجموعة البيانات وتهيئة النموذج Reading the Dataset and Initializing the Model

بادئ ذي بدء، دعنا نقرأ مجموعة بيانات اكتشاف الموز الموضحةفي القسم 14.6.

```
batch_size = 32
train_iter, _ = d2l.load_data_bananas(batch_size)
```

لا يوجد سوى فئة واحدةفي مجموعة بيانات اكتشاف الموز. بعد تحديد النموذج، نحتاج إلى تهيئة معلماته وتحديد خوارزمية التحسين.

```
device, net = d2l.try_gpu(), TinySSD(num_classes=1)
net.initialize(init=init.Xavier(), ctx=device)
trainer = gluon.Trainer(net.collect_params(), 'sgd',
                        {'learning_rate': 0.2,'wd': 5e-4})
```

14.7.2.2. تعريف دوال الخطأ والتقييم Defining Loss and Evaluation Functions

اكتشاف الكائن له نوعان من الخطأ. يتعلق الخطأ الأول بفئات مربعات التحديد: يمكن لحسابها ببساطة إعادة استخدام دالة فقدان الانتروبيا المتقاطعة التي استخدمناها لتصنيف الصور. يتعلق الخطأ الثاني بازاحات مربعات التحديد الإيجابية (غير الخلفية): هذه مشكلة انحدار regression. ومع ذلك، بالنسبة لمشكلة الانحدار هذه، فإننا لا نستخدم الخسارة التربيعية

الموضحةفي القسم 3.1.3. بدلاً من ذلك، نستخدم خطأ المعيار ℓ_1 ، القيمة المطلقة للاختلاف بين التنبؤ والحقيقة الأساسية. يقوم متغير القناع bbox_masks بتصفية مربعات التحديد السالبة ومربعات التحديد غير القانونية (المبطنة padded)في حساب الخطأ.في النهاية، نلخص خطأ فئة مربع التحديد وخطأ ازاحة صندوق التحديد للحصول على دالة الخطأ للنموذج.

```
cls_loss = gluon.loss.SoftmaxCrossEntropyLoss()
bbox_loss = gluon.loss.L1Loss()

def calc_loss(cls_preds, cls_labels, bbox_preds,
bbox_labels, bbox_masks):
    cls = cls_loss(cls_preds, cls_labels)
    bbox = bbox_loss(bbox_preds * bbox_masks,
bbox_labels * bbox_masks)
    return cls + bbox
```

يمكننا استخدام الدقة لتقييم نتائج التصنيف. نظرًا لفقدان المعيار ℓ_1 المستخدم للإزاحة، نستخدم متوسط الخطأ المطلق mean absolute error لتقييم مربعات الاحاطة المتوقعة. يتم الحصول على نتائج التنبؤ هذه من مربعات التحديد التي تم إنشاؤها والازاحات المتوقعة لها.

```
def cls_eval(cls_preds, cls_labels):
    # Because the class prediction results are on the
final dimension,
    # `argmax` needs to specify this dimension
    return float((cls_preds.argmax(axis=-1).astype(
        cls_labels.dtype) == cls_labels).sum())

def bbox_eval(bbox_preds, bbox_labels, bbox_masks):
    return float((np.abs((bbox_labels - bbox_preds) *
bbox_masks)).sum())
```

14.7.2.3. تدريب النموذج Training the Model

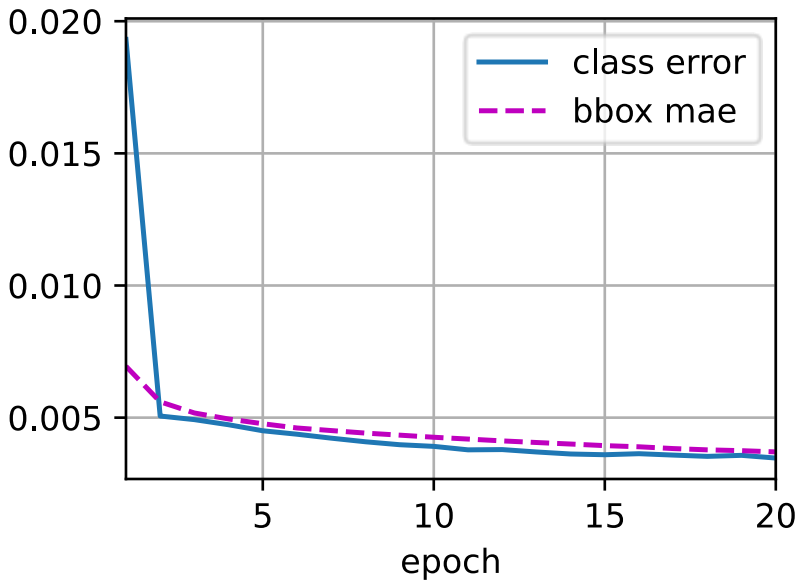
عند تدريب النموذج، نحتاج إلى إنشاء مربعات تحديد متعددة القياسات (anchors) والتنبؤ بفئاتها (cls_preds) والإزاحات (bbox_preds)في الانتشار الأمامي. ثم نقوم بتسمية الفئات (cls_labels) والإزاحات (bbox_labels) لمربعات التحديد التي تم إنشاؤها بناءً على معلومات التسمية Y. أخيرًا، نحسب دالة الخطأ باستخدام القيم المتوقعة والمسماة للفئات والإزاحات. لعمليات التنفيذ المختصرة، تم حذف تقييم مجموعة بيانات الاختبار هنا.

```
num_epochs, timer = 20, d2l.Timer()
animator = d2l.Animator(xlabel='epoch', xlim=[1,
num_epochs],
```

```
                        legend=['class error', 'bbox
mae'])
for epoch in range(num_epochs):
    # Sum of training accuracy, no. of examples in sum
of training accuracy,
    # Sum of absolute error, no. of examples in sum of
absolute error
    metric = d2l.Accumulator(4)
    for features, target in train_iter:
        timer.start()
        X = features.as_in_ctx(device)
        Y = target.as_in_ctx(device)
        with autograd.record():
            # Generate multiscale anchor boxes and
predict their classes and
            # offsets
            anchors, cls_preds, bbox_preds = net(X)
            # Label the classes and offsets of these
anchor boxes
            bbox_labels, bbox_masks, cls_labels =
d2l.multibox_target(anchors,

Y)
            # Calculate the loss function using the
predicted and labeled
            # values of the classes and offsets
            l = calc_loss(cls_preds, cls_labels,
bbox_preds, bbox_labels,
                          bbox_masks)
        l.backward()
        trainer.step(batch_size)
        metric.add(cls_eval(cls_preds, cls_labels),
cls_labels.size,
                   bbox_eval(bbox_preds, bbox_labels,
bbox_masks),
                   bbox_labels.size)
    cls_err, bbox_mae = 1 - metric[0] / metric[1],
metric[2] / metric[3]
    animator.add(epoch + 1, (cls_err, bbox_mae))
print(f'class err {cls_err:.2e}, bbox mae
{bbox_mae:.2e}')
```

```
print(f'{len(train_iter._dataset) / timer.stop():.1f}
examples/sec on '
      f'{str(device)}')
```

```
class err 3.47e-03, bbox mae 3.70e-03
2718.6 examples/sec on gpu(0)
```



14.7.3. التنبؤ Prediction

أثناء التوقع، الهدف هو اكتشاف كل الأشياء ذات الأهميةفي الصورة. فيما يلي نقرأ صورة اختبارية ونغير حجمها، ونحولها إلى موتر رباعي الأبعاد الذي تتطلبه الطبقات التلافيفية.

```
img = image.imread('../img/banana.jpg')
feature = image.imresize(img, 256,
256).astype('float32')
X = np.expand_dims(feature.transpose(2, 0, 1), axis=0)
```

باستخدام دالة `multibox_detection` أدناه ، يتم الحصول على مربعات الإحاطة المتوقعة من مربعات التحديد والإزاحات المتوقعة. ثم يتم استخدام الكبت بدون الحد الأقصى NMS لإزالة مربعات الإحاطة المتوقعة المماثلة.

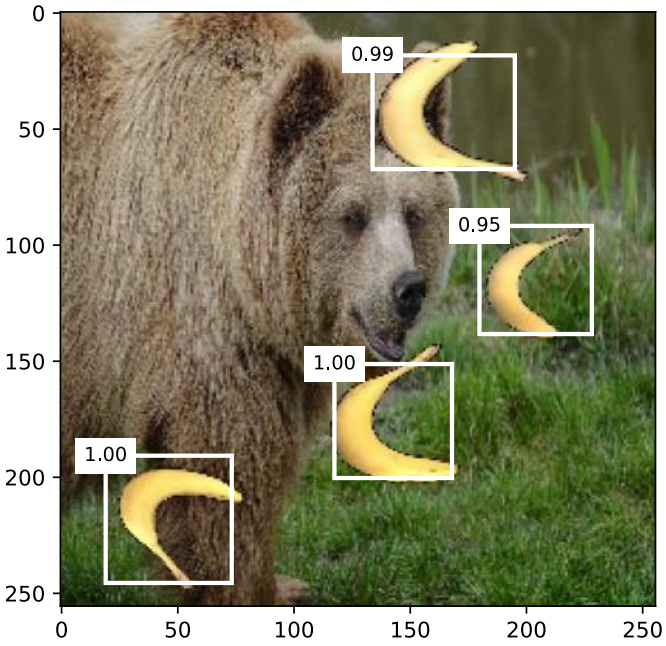
```
def predict(X):
    anchors, cls_preds, bbox_preds =
net(X.as_in_ctx(device))
```

```
    cls_probs = npx.softmax(cls_preds).transpose(0, 2,
1)
    output = d2l.multibox_detection(cls_probs,
bbox_preds, anchors)
    idx = [i for i, row in enumerate(output[0]) if
row[0] != -1]
    return output[0, idx]

output = predict(X)
```

```
[23:23:15] src/operator/nn/./cudnn/./cudnn_algoreg-
inl.h:97: Running performance tests to find the best
convolution algorithm, this can take a while... (set the
environment variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0
to disable)
```

أخيرًا، نعرض جميع مربعات الاحاطة المتوقعة بثقة 0.9 أو أعلى كناتج.

```
def display(img, output, threshold):
    d2l.set_figsize((5, 5))
    fig = d2l.plt.imshow(img.asnumpy())
    for row in output:
        score = float(row[1])
        if score < threshold:
            continue
        h, w = img.shape[:2]
        bbox = [row[2:6] * np.array((w, h, w, h),
ctx=row.ctx)]
        d2l.show_bboxes(fig.axes, bbox, '%.2f' % score,
'w')

display(img, output, threshold=0.9)
```



14.7.4. الملخص

- يعد اكتشاف المربعات المتعددة ذو اللقطة الواحدة Single shot multibox detection نموذجًا لاكتشاف كائن متعدد القياسات multiscale object detection model. عبر شبكتها الأساسية والعديد من كتل خرائط المعالم متعددة القياسات، يُنشئ اكتشاف المربعات المتعددة ذو اللقطة الواحدة عددًا متفاوتًا من مربعات التحديد بأحجام مختلفة، ويكتشف الكائنات ذات الأحجام المتفاوتة من خلال التنبؤ بالفئات والإزاحات لمربعات التحديد هذه (وبالتالي مربعات الاحاطة).
- عند تدريب نموذج اكتشاف المربعات المتعددة ذو اللقطة الواحدة، يتم حساب دالة الخطأ استنادًا إلى القيم المتوقعة والمعنونة لفئات وإزاحات مربع التحديد.

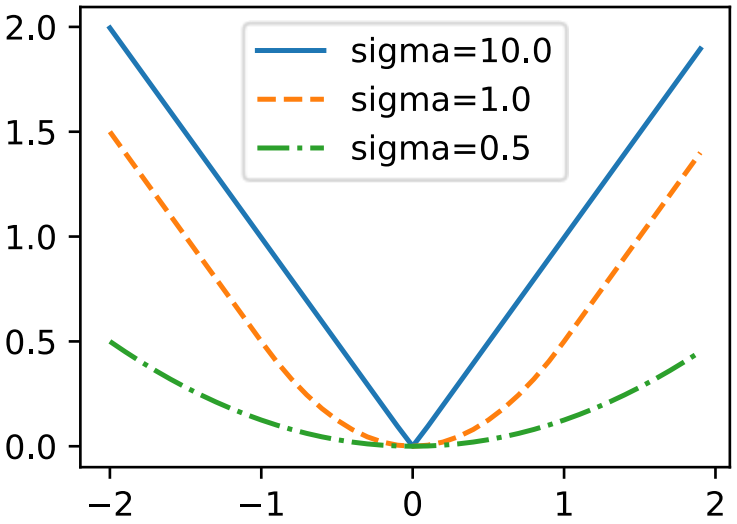
14.7.5. التمارين

1. هل يمكنك تحسين اكتشاف المربعات المتعدد ذو اللقطة الواحدة من خلال تحسين دالة الخطأ؟ على سبيل المثال، استبدل خطأ المعيار ℓ_1 بخطأ المعيار ℓ_1 السلس للازاحات المتوقعة. تستخدم دالة الخطأ هذه دالة مربعة حول الصفر من أجل السلاسة smoothness، والتي يتم التحكم فيها بواسطة المعلمة الفائقة σ:

$$f(x) = \begin{cases} (\sigma x)^2/2, & \text{if } |x| < 1/\sigma^2 \\ |x| - 0.5/\sigma^2, & \text{otherwise} \end{cases}$$

عندما σ يكون هذا الخطأ كبيرة جدًا، فإن هذا الخطأ تشبه خطأ المعيار ℓ_1. عندما تكون قيمتها أصغر، تكون دالة الخطأ أكثر سلاسة.

```
sigmas = [10, 1, 0.5]
lines = ['-', '--', '-.']
x = np.arange(-2, 2, 0.1)
d2l.set_figsize()

for l, s in zip(lines, sigmas):
    y = npx.smooth_l1(x, scalar=s)
    d2l.plt.plot(x.asnumpy(), y.asnumpy(), l,
label='sigma=%.1f' % s)
d2l.plt.legend();
```

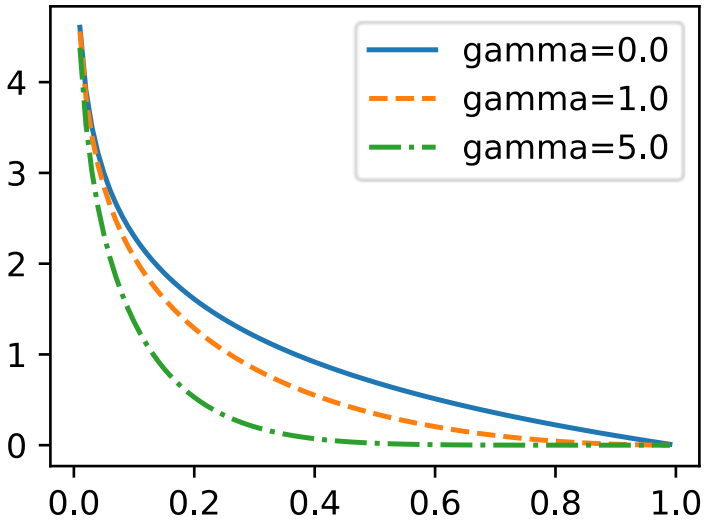


إلى جانب ذلك، استخدمنافي التجربة خطأ الانتروبيا المتقاطعة cross-entropy loss للتنبؤ بالفئة: بالإشارة إلى الاحتمال المتوقع لفئة الحقيقة الأرضية j ، فإن خطأ الانتروبيا المتقاطعة هي $-\log p_j$. يمكننا أيضًا استخدام الخطأ البؤري focal loss (Lin et al.، 2017): بالنظر إلى المعلمات الفائقة $\gamma > 0$ و$\alpha > 0$ يتم تعريف هذا الخطأ على النحو التالي:

$$-\alpha(1-p_j)^{\gamma}\log p_j.$$

كما نرى، يمكن أن تقلل زيادة γ بشكل فعال من الخطأ النسبي للأمثلة المصنفة جيدًا (على سبيل المثال $p_j > 0.5$) بحيث يمكن للتدريب التركيز أكثر على تلك الأمثلة الصعبة التي تم تصنيفها بشكل خاطئ.

```
def focal_loss(gamma, x):
    return -(1 - x) ** gamma * np.log(x)

x = np.arange(0.01, 1, 0.01)
for l, gamma in zip(lines, [0, 1, 5]):
    y = d2l.plt.plot(x.asnumpy(), focal_loss(gamma,
x).asnumpy(), l,
                     label='gamma=%.1f' % gamma)
d2l.plt.legend();
```



2. نظرًا لمحدودية المساحة، فقد أغفلنا بعض تفاصيل التنفيذ الخاصة بنموذج اكتشاف المربعات المتعددة ذو اللقطة الواحدةفي هذا القسم. هل يمكنك تحسين النموذجفي الجوانب التالية:
   1. عندما يكون الكائن أصغر بكثير مقارنة بالصورة، يمكن للنموذج تغيير حجم الصورة المدخلة بشكل أكبر.
   2. يوجد عادةً عددكبير من مربعات التحديد السلبية. لجعل توزيع الفئة أكثر توازناً، يمكننا اختزال عينات مربعات التحديد السلبية.
   3. في دالة الخطأ، قم بتعيين معلمات فائقة مختلفة للوزن لخطأ الفئة وخطأ الازاحة.
   4. استخدم طرقًا أخرى لتقييم نموذج اكتشاف الكائن، مثل تلك الموجودةفي مقالة اكتشاف المربعات المتعددة ذات اللقطة الواحدة (Liu et al.، 2016).

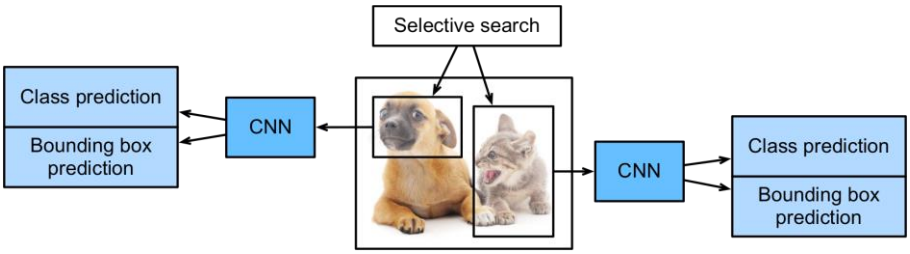
14.8 شبكات CNN القائمة على المنطقة (R-CNNs) Region-based CNNs

إلى جانب اكتشاف المربعات المتعددة ذات اللقطة الواحدة single shot multibox detection الموضحةفي القسم 14.7، تعد شبكات CNN القائمة على المنطقة أو المناطق التي تحتوي على ميزات CNN (R-CNN) أيضًا من بين العديد من الأساليب الرائدة لتطبيق التعلم العميق لاكتشاف الكائن (Girshick et al.، 2014).في هذا القسم، سنقدم R-CNN وسلسلة التحسينات الخاصة بها: fast R-CNN (Girshick، 2015)، وfaster R-CNN (Ren et al.، 2015)، وmask R-CNN (He et al.، 2017). نظرًا للمساحة المحدودة، سنركز فقط على تصميم هذه النماذج.

14.8.1. R-CNNs

تستخرج R-CNN أولاً العديد من مقترحات المنطقة region proposals (على سبيل المثال، 2000) من صورة الإدخال (على سبيل المثال، يمكن أيضًا اعتبار مربعات التحديد anchor boxes كمقترحات للمنطقة)، ووضع علامات على فئاتها ومربعاتها المحيطة bounding boxes (على سبيل المثال، الإزاحات offsets).

ثم يتم استخدام CNN لإجراء انتشار أمامي على مقترح كل منطقة لاستخراج ميزاتها. بعد ذلك، يتم استخدام ميزات كل مقترح منطقة للتنبؤ بالفئة ومربع الاحاطة لمقترح المنطقة هذا.



الشكل 14.8.1 نموذج R-CNN.

يوضح الشكل 14.8.1 نموذج R-CNN. بشكل أكثر تحديدًا، تتكون R-CNN من الخطوات الأربع التالية:

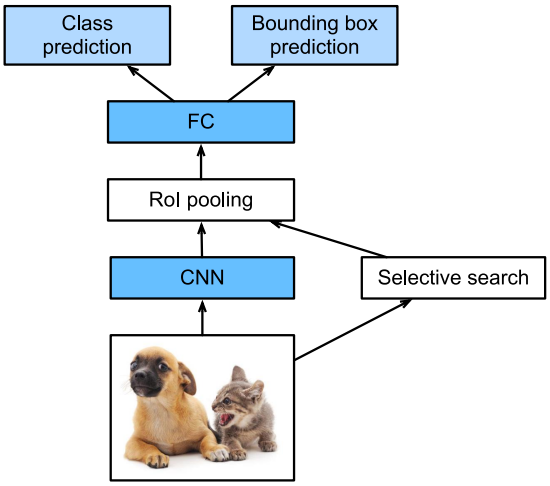
1. قم بإجراء بحث انتقائي selective search لاستخراج العديد من مقترحات المنطقة عالية الجودة على صورة الإدخال (Uijlings et al.، 2013). عادة ما يتم اختيار هذه المناطق المقترحة بمقاييس متعددة بأشكال وأحجام مختلفة. سيتم تسمية كل اقتراح منطقة بفئة ومربع إحاطة بالحقيقة الأساسية.

2. اختر شبكة CNN سابقة التحديد واقطعها قبل طبقة الإخراج. قم بتغيير حجم مقترح كل منطقة إلى حجم المدخلات المطلوبة من قبل الشبكة، وإخراج الميزات المستخرجة لاقتراح المنطقة من خلال الانتشار الأمامي.
3. خذ الميزات المستخرجة والفئة المسماة لكل اقتراح منطقة كمثال. قم بتدريب العديد من آلات متجه الدعم support vector machines لتصنيف الكائنات، حيث تحدد كل آلة متجه داعمة بشكل فردي ما إذا كان المثال يحتوي على فئة معينة.
4. خذ الميزات المستخرجة ومربع الاحاطة المسمى لكل اقتراح منطقة كمثال. تدريب نموذج الانحدار الخطي للتنبؤ بمربع الاحاطة بالحقيقة الأرضية.

على الرغم من أن نموذج R-CNN يستخدم شبكات CNN سابقة التدريب لاستخراج ميزات الصورة بشكل فعال، إلا أنه بطيء. تخيل أننا نختار الآلاف من مقترحات المنطقة من صورة إدخال واحدة: وهذا يتطلب الآلاف من عمليات الانتشار الأمامي لـ CNN لإجراء اكتشاف الكائنات. هذا العبء الحسابي الهائل يجعل من غير المجدي استخدام R-CNNs على نطاق واسع في تطبيقات العالم الحقيقي.

Fast R-CNN .14.8.2

يكمن عنق الزجاجة الرئيسي في أداء R-CNN في نشر CNN المستقل للأمام لكل مقترح منطقة، دون مشاركة الحساب. نظرًا لأن هذه المناطق عادة ما يكون لها تداخل، فإن عمليات استخراج الميزات المستقلة تؤدي إلى الكثير من العمليات الحسابية المتكررة. أحد التحسينات الرئيسية لـ Fast R-CNN من R-CNN هو أن الانتشار الأمامي لـ CNN يتم تنفيذه فقط على الصورة بأكملها (Girshick، 2015).



الشكل 14.8.2 نموذج fast R-CNN.

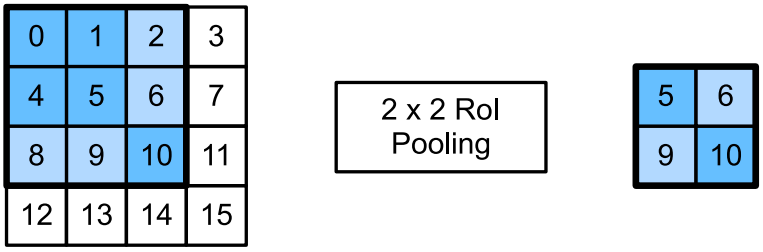
يصف الشكل 14.8.2 نموذج fast R-CNN. حساباته الرئيسية هي كما يلي:

1. بالمقارنة مع R-CNN،في fast R-CNN، يكون إدخال CNN لاستخراج الميزات هو الصورة بأكملها، بدلاً من مقترحات المنطقة الفردية. علاوة على ذلك، فإن شبكة CNN هذه قابلة للتدريب. بالنظر إلى صورة الإدخال، دع شكل إخراج CNN يكون $1 \times c \times h_1 \times w_1$.
2. افترض أن البحث الانتقائي يولد n مقترحات المنطقة. تحدد مقترحات المنطقة هذه (ذات الأشكال المختلفة) مناطق الاهتمام regions of interest (ذات الأشكال المختلفة) على إخراج CNN. بعد ذلك، تستخرج هذه مناطق الاهتمام ميزات من نفس الشكل (لنفترض أنه تم تحديد الارتفاع h_2 والعرض w_2) حتى يمكن ربطها بسهولة. لتحقيق ذلك، تقدم fast R-CNN طبقة تجميع pooling layer منطقة الاهتمام (RoI): يتم إدخال مخرجات CNN ومقترحات المنطقةفي هذه الطبقة، مما يؤدي إلى إخراج معالم متسلسلة concatenated features للشكل $n \times c \times h_2 \times w_2$ يتم استخلاصها بشكل أكبر لجميع مقترحات المنطقة.
3. باستخدام طبقة متصلة بالكامل، قم بتحويل المعالم المتسلسلة إلى إخراج الشكل $n \times d$ $n \times d$ ، حيث d يعتمد على تصميم النموذج.
4. توقع الفئة ومربع الاحاطة لكل من n مقترحات المنطقة. بشكل أكثر تحديداً،في التنبؤ بالفئة ومربع الاحاطة، قم بتحويل ناتج الطبقة المتصلة بالكامل إلى إخراج الشكل $n \times q$ (q هو عدد الفئات) ومخرج الشكل $n \times 4$ ، على التوالي. يستخدم التنبؤ بالفئة انحدار softmax.

تختلف طبقة تجميع منطقة الاهتمام المقترحةفي fast R-CNN عن طبقة التجميع المقدمةفي القسم 7.5.في طبقة التجميع، نتحكم بشكل غير مباشرفي شكل الإخراج عن طريق تحديد أحجام نافذة التجميع والحشو والخطوة.في المقابل، يمكننا تحديد شكل الإخراج مباشرةفي طبقة تجميع المنطقة محل الاهتمام.

على سبيل المثال، دعنا نحدد ارتفاع الإخراج وعرضه لكل منطقة على النحو التالي h_2 و w_2 ، على التوالي. لأي نافذة ROI الشكل $h \times w$ ، يتم تقسيم هذه النافذة إلى $h_2 \times w_2$ شبكة من النوافذ الفرعية، حيث يكون شكل كل نافذة فرعية تقريبًا $(h/h_2) \times (w/w_2)$. من الناحية العملية، يجب تقريب ارتفاع وعرض أي نافذة فرعية، ويجب استخدام أكبر عنصر كناتج للنافذة الفرعية. لذلك، يمكن لطبقة تجميع منطقة الاهتمام استخراج ميزات من نفس الشكل حتى عندما يكون للمناطق ذات الأهمية أشكال مختلفة.

كمثال توضيحي،في الشكل 14.8.3، يتم تحديد المنطقة العلوية اليسرى 3 × 3 ذات الأهمية على أحد المدخلات 4 × 4. بالنسبة إلى منطقة الاهتمام هذه، نستخدم طبقة تجميع 2 × 2 منطقة الاهتمام للحصول على ناتج 2 × 2. لاحظ أن كل من النوافذ الفرعية الأربعة المقسمة تحتوي على العناصر 0 و 1 و 4 و 5 (5 هي الحد الأقصى)؛ 2 و 6 (6 هو الحد الأقصى)؛ 8 و 9 (9 هو الحد الأقصى)؛ و 10.



شكل 14.8.3 طبقة تجميع منطقة الاهتمام 2 × 2.

نوضح أدناه حساب طبقة تجميع منطقة الاهتمام. افترض أن ارتفاع وعرض الميزات X المستخرجة من CNN هما 4، ولا توجد سوى قناة واحدة.

```
from mxnet import np, npx

npx.set_np()

X = np.arange(16).reshape(1, 1, 4, 4)
X
```

```
array([[[[ 0.,  1.,  2.,  3.],
         [ 4.,  5.,  6.,  7.],
         [ 8.,  9., 10., 11.],
         [12., 13., 14., 15.]]]])
```

لنفترض كذلك أن ارتفاع وعرض صورة الإدخال كلاهما 40 بكسل وأن البحث الانتقائي يولد اقتراحين للمنطقة على هذه الصورة. يتم التعبير عن كل اقتراح منطقةفي شكل خمسة عناصر: فئة الكائن الخاصة بها متبوعة بـ (x, y) احداثيات الزوايا العلوية اليسرى والسفلية اليمنى.

```
rois = np.array([[0, 0, 0, 20, 20], [0, 0, 10, 30, 30]])
```

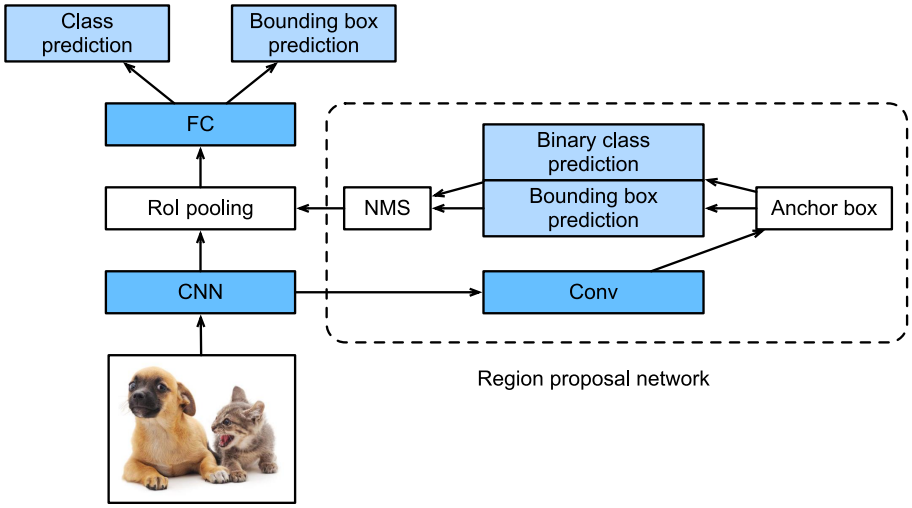
نظرًا لأن ارتفاع وعرض X هما 1/10 من ارتفاع وعرض صورة الإدخال، يتم ضرب إحداثيات عرضي المنطقة بمقدار 0.1 وفقًا للوسيطة spatial_scale المحددة. ثم يتم وضع علامة على منطقتي الاهتمام على X على أنها X[:, :, 0:3, 0:3] و X[:, :, 1:4, 0:4] ، على التوالي. أخيرًافي تجميع منطقة الاهتمام، يتم تقسيم كل منطقة اهتمام إلى شبكة من النوافذ الفرعية لاستخراج ميزات أخرى من نفس الشكل 2 × 2.

```
npx.roi_pooling(X, rois, pooled_size=(2, 2),
spatial_scale=0.1)
```

```
array([[[[ 5.,  6.],
         [ 9., 10.]]],


       [[[ 9., 11.],
         [13., 15.]]]])
```

Faster R-CNN .14.8.3

لكي تكون أكثر دقةفي اكتشاف الأشياء، يتعين على نموذج Faster R-CNN عادةً إنشاء الكثير من مقترحات المنطقةفي البحث الانتقائي. لتقليل مقترحات المنطقة دون فقدان الدقة، تقترح Faster R-CNN استبدال البحث الانتقائي بشبكة اقتراح المنطقة region proposal network (Ren et al.، 2015).



الشكل 14.8.4 نموذج faster R-CNN.

يوضح الشكل 14.8.4 نموذج faster R-CNN. بالمقارنة مع fast R-CNN، فإن faster R-CNN يغير فقط طريقة اقتراح المنطقة من البحث الانتقائي إلى شبكة مقترحات المنطقة. بقي باقي النموذج دون تغيير. تعمل شبكة مقترحات المنطقةفي الخطوات التالية:

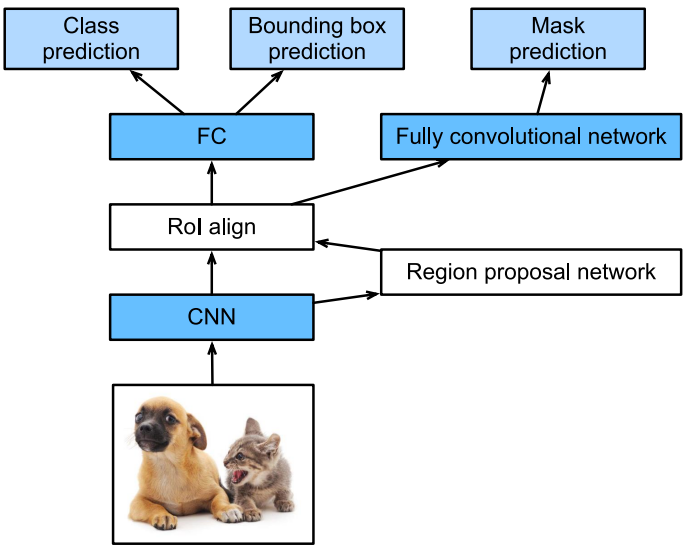
1. استخدم طبقة تلافيفية 3 × 3 مع حشوة 1 لتحويل إخراج CNN إلى إخراج جديد مع القنوات. بهذه الطريقة، تحصل كل وحدة على طول الأبعاد المكانية لخرائط الميزات المستخرجة من CNN على متجه جديد للطول c.

2. قم بتوسيط كل بكسل من خرائط الميزات، وقم بإنشاء مربعات تحديد متعددة بمقاييس ونسب أبعاد مختلفة وقم بتسميتها.
3. باستخدام متجه ميزة الطول c في وسط كل مربع تحديد، توقع الفئة الثنائية (الخلفية أو الكائنات) ومربع الاحاطة لمربع التحديد هذا.
4. ضع في اعتبارك تلك المربعات المحيطة المتوقعة التي تكون فئاتها المتوقعة عبارة عن كائنات. إزالة النتائج المتداخلة باستخدام الكبت بدون الحد الأقصى NMS. مربعات الاحاطة المتوقعة المتبقية للكائنات هي مقترحات المنطقة المطلوبة بواسطة طبقة تجميع المنطقة محل الاهتمام.

تجدر الإشارة إلى أنه كجزء من نموذج faster R-CNN، يتم تدريب شبكة اقتراح المنطقة بشكل مشترك مع بقية النموذج. بمعنى آخر، لا تتضمن دالة الهدف لـ faster R-CNN فقط التنبؤ بالفئة ومربع الاحاطةفي اكتشاف الكائن، ولكن أيضًا التنبؤ بالفئة الثنائية ومربع الاحاطة لمربعات التحديدفي شبكة اقتراح المنطقة. نتيجة للتدريب الشامل end-to-end training، تتعلم شبكة مقترحات المنطقة كيفية إنشاء مقترحات منطقة عالية الجودة، وذلك للبقاء دقيقًافي اكتشاف الكائنات مع تقليل عدد مقترحات المنطقة التي يتم تعلمها من البيانات.

14.8.4. Mask R-CNN

في مجموعة بيانات التدريب، إذا تم أيضًا تسمية مواضع الكائن على مستوى البكسل على الصور، يمكن لـ Mask R-CNN الاستفادة بشكل فعال من هذه التسميات التفصيلية لتحسين دقة اكتشاف الكائن (He et al.، 2017).



الشكل 14.8.5 نموذج Mask R-CNN.

كما هو مبين في الشكل 14.8.5، يتم تعديل Mask R-CNN بناءً على faster R-CNN. على وجه التحديد، يستبدل Mask R-CNN منطقة تجميع منطقة الاهتمام (RoI pooling layer) بطبقة محاذاة منطقة الاهتمام (RoI alignment layer). تستخدم طبقة محاذاة منطقة الاهتمام هذه الاستيفاء ثنائي الخطوط للحفاظ على المعلومات المكانية على خرائط المعالم، والتي تعد أكثر ملاءمة للتنبؤ على مستوى البكسل. يحتوي إخراج هذه الطبقة على خرائط معالم من نفس الشكل لجميع مناطق الاهتمام. يتم استخدامها للتنبؤ ليس فقط بالفئة ومربع الاحاطة لكل منطقة من مناطق الاهتمام، ولكن أيضًا موضع مستوى البكسل للكائن من خلال شبكة تلافيفية إضافية كاملة. سيتم توفير مزيد من التفاصيل حول استخدام شبكة تلافيفية كاملة للتنبؤ بدلالات مستوى البكسل للصورة في الأقسام التالية من هذا الفصل.

14.8.5. الملخص

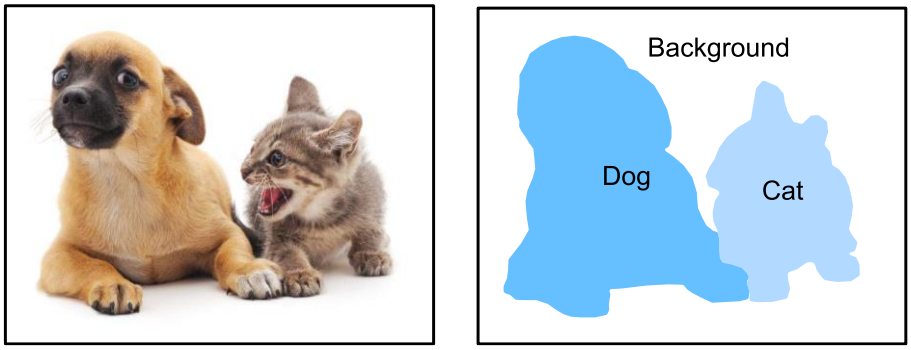
- تستخرج R-CNN العديد من مقترحات المنطقة من صورة الإدخال، وتستخدم CNN لإجراء انتشار أمامي على مقترح كل منطقة لاستخراج ميزاتها، ثم تستخدم هذه الميزات للتنبؤ بالفئة ومربع الاحاطة لمقترح المنطقة هذا.
- أحد التحسينات الرئيسية لـ fast R-CNN من R-CNN هو أن الانتشار الأمامي لـ CNN يتم تنفيذه فقط على الصورة بأكملها. يقدم أيضًا طبقة تجميع المنطقة الاهتمام، بحيث يمكن استخراج ميزات نفس الشكل بشكل أكبر لمناطق الاهتمام التي لها أشكال مختلفة.
- يستبدل faster R-CNN البحث الانتقائي المستخدم في fast R-CNN بشبكة اقتراح منطقة مدربة بشكل مشترك، بحيث يمكن أن يظل الأول دقيقًا في اكتشاف الكائنات مع عدد أقل من مقترحات المنطقة.
- استنادًا إلى faster R-CNN، يقدم mask R-CNN أيضًا شبكة تلافيفية بالكامل، وذلك للاستفادة من التسميات على مستوى البكسل لتحسين دقة اكتشاف الكائنات.

14.8.6. التمارين

1. هل يمكننا تأطير اكتشاف الكائن باعتباره مشكلة انحدار واحدة، مثل التنبؤ بمربعات الاحاطة واحتمالات الفئة؟ يمكنك الرجوع إلى تصميم نموذج YOLO، (Redmon et al., 2016).
2. قارن اكتشاف المربعات المتعددة ذات اللقطة الواحدة بالطرق المقدمة في هذا القسم. ما هي الاختلافات الرئيسية بينهما؟ يمكنك الرجوع إلى الشكل 2 من Zhao et al. (2019).

14.9 التقطيع الدلالي ومجموعة البيانات Semantic Segmentation and the Dataset

عند مناقشة مهام اكتشاف الكائنفي القسم 14.3 ـ القسم 14.8، تُستخدم مربعات الإحاطة bounding boxes المستطيلة لتسمية الكائناتفي الصور والتنبؤ بها. سيناقش هذا القسم مشكلة التقطيع الدلالي semantic segmentation، والتي تركز على كيفية تقسيم الصورة إلى مناطق تنتمي إلى فئات دلالية semantic classes مختلفة. يختلف التقطيع الدلالي عن اكتشاف الكائن، ويتعرف على ما هو موجودفي الصورفي مستوى البكسل ويفهمه: يتم وضع العلامات والتنبؤ بالمناطق الدلالية semantic regionsفي مستوى البكسل. يوضح الشكل 14.9.1 تسميات الكلب والقط وخلفية الصورةفي التقطيع الدلالي. مقارنةً باكتشاف الكائن، من الواضح أن حدود مستوى البكسل الموصوفةفي التقطيع الدلالي تكون أكثر دقة.



الشكل. 14.9.1 تسميات الكلب والقط وخلفية الصورةفي التقطيع الدلالي.

14.9.1. تقطيع الصورة وتقطيع المثيلات Image Segmentation and Instance Segmentation

هناك أيضًا مهمتان مهمتانفي مجال الرؤية الحاسوبية تشبهان التقطيع الدلالي semantic segmentation، وهما تقطيع الصورة image segmentation وتقطيع المثيل instance segmentation. سوف نميزهم بإيجاز عن التقطيع الدلالي على النحو التالي.

- يقسم تقطيع الصورة الصورة إلى عدة مناطق مكونة. عادةً ما تستخدم طرق هذا النوع من المشكلات الارتباط بين وحدات البكسلفي الصورة. لا يحتاج إلى معلومات تسمية حول بكسلات الصورة أثناء التدريب، ولا يضمن أن تحتوي المناطق المقطعة على الدلالات التي نأملفي الحصول عليها أثناء التنبؤ. عند أخذ الصورةفي الشكل 14.9.1 كمدخلات، قد يقسم تقطيع الصورة الكلب إلى منطقتين: أحدهما يغطي الفم والعينين باللون الأسود بشكل أساسي، والأخرى تغطي باقي الجسم وهو أصفر بشكل أساسي.

- يسمى تقطيع المثيل أيضًا الاكتشاف والتقطيع المتزامنين simultaneous detection and segmentation. يدرس كيفية التعرف على مناطق مستوى البكسل لكل مثيل كائن في صورة. يختلف عن التقطيع الدلالي، يحتاج تقطيع المثيل إلى التمييز ليس فقط بين الدلالات، ولكن أيضًا حالات الكائن المختلفة. على سبيل المثال، إذا كان هناك كلبان في الصورة، فإن تقطيع المثيل يحتاج إلى التمييز بين الكلاب التي ينتمي إليها البكسل.

14.9.2. مجموعة بيانات التجزئة الدلالية باسكال The Pascal VOC2012 VOC2012 Semantic Segmentation Dataset

من أهم مجموعة بيانات التقطيع الدلالي باسكال Pascal VOC2012. فيما يلي، سوف نلقي نظرة على مجموعة البيانات هذه.

```
%matplotlib inline
import os
from mxnet import gluon, image, np, npx
from d2l import mxnet as d2l

npx.set_np()
```

يبلغ حجم ملف tar الخاص بمجموعة البيانات حوالي 2 جيجابايت، لذلك قد يستغرق تنزيل الملف بعض الوقت. توجد مجموعة البيانات المستخرجة في `../data/VOCdevkit/VOC2012`.

```
#@save
d2l.DATA_HUB['voc2012'] = (d2l.DATA_URL +
'VOCtrainval_11-May-2012.tar',

'4e443f8a2eca6b1dac8a6c57641b67dd40621a49')

voc_dir = d2l.download_extract('voc2012',
'VOCdevkit/VOC2012')
```

```
Downloading ../data/VOCtrainval_11-May-2012.tar from
http://d2l-data.s3-
accelerate.amazonaws.com/VOCtrainval_11-May-2012.tar...
```

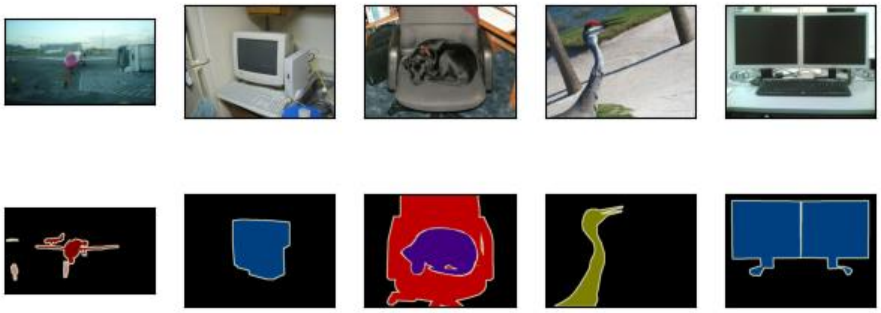
بعد دخول المسار `../data/VOCdevkit/VOC2012` ، يمكننا رؤية المكونات المختلفة لمجموعة البيانات. يحتوي مسار `ImageSets/Segmentation` على ملفات نصية تحدد عينات التدريب والاختبار، بينما يقوم مسار `JPEGImages` و `SegmentationClass` بتخزين صورة الإدخال والتسمية لكل مثال ، على التوالي. التسمية هنا موجود أيضًا بتنسيق

الصورة، بنفس حجم صورة الإدخال المصنفة. إلى جانب ذلك، تنتمي وحدات البكسل التي لها نفس اللونفي أي صورة تسمية إلى نفس الفئة الدلالية. يحدد ما يلي دالة `read_voc_images` لقراءة جميع الصور والتسميات المدخلةفي الذاكرة.

```
#@save
def read_voc_images(voc_dir, is_train=True):
    """Read all VOC feature and label images."""
    txt_fname = os.path.join(voc_dir, 'ImageSets',
'Segmentation',
                             'train.txt' if is_train
else 'val.txt')
    with open(txt_fname, 'r') as f:
        images = f.read().split()
    features, labels = [], []
    for i, fname in enumerate(images):
        features.append(image.imread(os.path.join(
            voc_dir, 'JPEGImages', f'{fname}.jpg')))
        labels.append(image.imread(os.path.join(
            voc_dir, 'SegmentationClass',
f'{fname}.png')))
    return features, labels

train_features, train_labels = read_voc_images(voc_dir,
True)
```

نرسم أول خمس صور إدخال وتسمياتها.في صور التسمية، يمثل الأبيض والأسود الحدود والخلفية، على التوالي، بينما تتوافق الألوان الأخرى مع فئات مختلفة.

```
n = 5
imgs = train_features[:n] + train_labels[:n]
d2l.show_images(imgs, 2, n);
```



بعد ذلك، نقوم بتعداد قيم ألوان RGB وأسماء الفئات لجميع التسمياتفي مجموعة البيانات هذه.

```
#@save
VOC_COLORMAP = [[0, 0, 0], [128, 0, 0], [0, 128, 0], [128, 128, 0],
                [0, 0, 128], [128, 0, 128], [0, 128, 128], [128,
128, 128],
                [64, 0, 0], [192, 0, 0], [64, 128, 0], [192, 128,
0],
                [64, 0, 128], [192, 0, 128], [64, 128, 128], [192,
128, 128],
                [0, 64, 0], [128, 64, 0], [0, 192, 0], [128, 192,
0],
                [0, 64, 128]]

#@save
VOC_CLASSES = ['background', 'aeroplane', 'bicycle', 'bird', 'boat',
               'bottle', 'bus', 'car', 'cat', 'chair', 'cow',
               'diningtable', 'dog', 'horse', 'motorbike', 'person',
               'potted plant', 'sheep', 'sofa', 'train',
'tv/monitor']
```

من خلال الثابتين المحددين أعلاه، يمكننا العثور بسهولة على فهرس الفئة لكل بكسلفي التسمية. نحدد دالة `voc_colormap2label` لبناء التعيين من قيم ألوان RGB المذكورة أعلاه إلى مؤشرات الفئة، ودالة `voc_label_indices` لتعيين أي قيم RGB لمؤشرات فئتهافي مجموعة بيانات Pascal VOC2012 هذه.

```
#@save
def voc_colormap2label():
    """Build the mapping from RGB to class indices for
VOC labels."""
    colormap2label = np.zeros(256 ** 3)
    for i, colormap in enumerate(VOC_COLORMAP):
        colormap2label[
            (colormap[0] * 256 + colormap[1]) * 256 +
colormap[2]] = i
    return colormap2label

#@save
def voc_label_indices(colormap, colormap2label):
    """Map any RGB values in VOC labels to their class
indices."""
```

```
    colormap = colormap.astype(np.int32)
    idx = ((colormap[:, :, 0] * 256 + colormap[:, :, 1])
* 256
           + colormap[:, :, 2])
    return colormap2label[idx]
```

على سبيل المثال،في المثال الأول للصورة، يكون فهرس الفئة للجزء الأمامي من الطائرة هو 1، بينما يكون فهرس الخلفية هو 0.

```
y = voc_label_indices(train_labels[0],
voc_colormap2label())
y[105:115, 130:140], VOC_CLASSES[1]
```

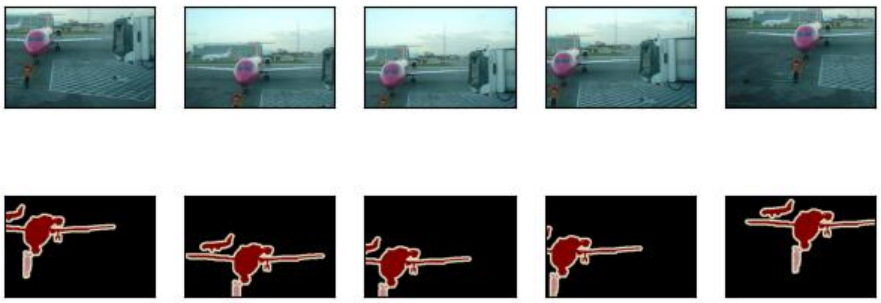
```
(array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
        [0., 0., 0., 0., 0., 0., 0., 1., 1., 1.],
        [0., 0., 0., 0., 0., 0., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
        [0., 0., 0., 0., 1., 1., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 0., 1., 1., 1., 1.],
        [0., 0., 0., 0., 0., 0., 0., 0., 1., 1.]]),
 'aeroplane')
```

14.9.2.1. معالجة البيانات Data Preprocessing

في التجارب السابقة كمافي القسم 8.1—القسم 8.4، يتم إعادة قياس الصور لتلائم شكل الإدخال المطلوب للنموذج. ومع ذلك،في التقطيع الدلالي، يتطلب القيام بذلك إعادة قياس فئات البكسل المتوقعة إلى الشكل الأصلي لصورة الإدخال. قد تكون عملية إعادة القياس هذه غير دقيقة، خاصة بالنسبة للمناطق المقطعة ذات الفئات المختلفة. لتجنب هذه المشكلة، نقوم بقص الصورة إلى شكل ثابت بدلاً من إعادة القياس. على وجه التحديد، باستخدام الاقتصاص العشوائي random cropping من زيادة الصورة image augmentation، نقوم بقص نفس المنطقة من صورة الإدخال والتسمية.

```
#@save
def voc_rand_crop(feature, label, height, width):
    """Randomly crop both feature and label images."""
    feature, rect = image.random_crop(feature, (width,
height))
    label = image.fixed_crop(label, *rect)
    return feature, label
```

```
imgs = []
for _ in range(n):
    imgs += voc_rand_crop(train_features[0],
train_labels[0], 200, 300)
d2l.show_images(imgs[::2] + imgs[1::2], 2, n);
```



14.9.2.2. فئة بيانات التقطيع الدلالي المخصص Custom Semantic Segmentation Dataset Class

نحدد فئة بيانات التقطيع الدلالية المخصص `VOCSegDataset` من خلال وراثة فئة مجموعة البيانات التي توفرها واجهات برمجة التطبيقات API عالية المستوى. من خلال تنفيذ دالة `__getitem__` ، يمكننا الوصول بشكل تعسفي إلى صورة الإدخال المفهرسة على أنها `idx` في مجموعة البيانات وفهرس الفئة لكل بكسل في هذه الصورة. نظرًا لأن حجم بعض الصور في مجموعة البيانات أصغر من حجم إخراج الاقتصاص العشوائي، يتم تصفية هذه الأمثلة بواسطة دالة `filter` مخصصة. بالإضافة إلى ذلك، نحدد أيضًا دالة `normalize_image` لتوحيد قيم قنوات RGB الثلاث لصور الإدخال.

```
#@save
class VOCSegDataset(gluon.data.Dataset):
    """A customized dataset to load the VOC dataset."""
    def __init__(self, is_train, crop_size, voc_dir):
        self.rgb_mean = np.array([0.485, 0.456, 0.406])
        self.rgb_std = np.array([0.229, 0.224, 0.225])
        self.crop_size = crop_size
        features, labels = read_voc_images(voc_dir,
is_train=is_train)
        self.features = [self.normalize_image(feature)
                         for feature in
self.filter(features)]
        self.labels = self.filter(labels)
        self.colormap2label = voc_colormap2label()
```

```
        print('read ' + str(len(self.features)) + '
examples')

    def normalize_image(self, img):
        return (img.astype('float32') / 255 -
self.rgb_mean) / self.rgb_std

    def filter(self, imgs):
        return [img for img in imgs if (
            img.shape[0] >= self.crop_size[0] and
            img.shape[1] >= self.crop_size[1])]

    def __getitem__(self, idx):
        feature, label =
voc_rand_crop(self.features[idx], self.labels[idx],
                                      *self.crop_size)
        return (feature.transpose(2, 0, 1),
                voc_label_indices(label,
self.colormap2label))

    def __len__(self):
        return len(self.features)
```

14.9.2.3. قراءة مجموعة البيانات Reading the Dataset

نستخدم فئة VOCSegDataset المخصصة لإنشاء مثيلات لمجموعة التدريب ومجموعة الاختبار، على التوالي. افترض أننا حددنا أن شكل إخراج الصور التي تم اقتصاصها عشوائيًا هو 320×480. أدناه يمكننا عرض عدد الأمثلة التي تم الاحتفاظ بها في مجموعة التدريب ومجموعة الاختبار.

```
crop_size = (320, 480)
voc_train = VOCSegDataset(True, crop_size, voc_dir)
voc_test = VOCSegDataset(False, crop_size, voc_dir)
```

```
read 1114 examples
read 1078 examples
```

عند تعيين حجم الدُفعة على 64، نحدد مكرر البيانات data iterator لمجموعة التدريب. دعونا نطبع شكل الدفعة الأولى. يختلف عن تصنيف الصور أو اكتشاف الكائن، التسميات هنا هي موترات ثلاثية الأبعاد.

```
batch_size = 64
train_iter = gluon.data.DataLoader(voc_train,
batch_size, shuffle=True,
```

```
                                    last_batch='discard',

num_workers=d2l.get_dataloader_workers())
for X, Y in train_iter:
    print(X.shape)
    print(Y.shape)
    break
```

```
(64, 3, 320, 480)
(64, 320, 480)
```

14.9.2.4. وضع كل شيء معا Putting It All Together

أخيرًا، نحدد دالة `load_data_voc` التالية لتنزيل مجموعة بيانات التقطيع الدلالي Pascal VOC2012 وقراءتها. تقوم بإرجاع مكررات البيانات لكل من مجموعات بيانات التدريب والاختبار.

```
#@save
def load_data_voc(batch_size, crop_size):
    """Load the VOC semantic segmentation dataset."""
    voc_dir = d2l.download_extract('voc2012',
os.path.join(
        'VOCdevkit', 'VOC2012'))
    num_workers = d2l.get_dataloader_workers()
    train_iter = gluon.data.DataLoader(
        VOCSegDataset(True, crop_size, voc_dir),
batch_size,
        shuffle=True, last_batch='discard',
num_workers=num_workers)
    test_iter = gluon.data.DataLoader(
        VOCSegDataset(False, crop_size, voc_dir),
batch_size,
        last_batch='discard', num_workers=num_workers)
    return train_iter, test_iter
```

14.9.3. الملخص

- يتعرف التقطيع الدلالي Semantic segmentation على ما هو موجودفي الصورة بمستوى البكسل ويفهمه عن طريق تقطيع الصورة إلى مناطق تنتمي إلى فئات دلالية مختلفة.
- تعد Pascal VOC2012 واحدة من أهم مجموعة بيانات التقطيع الدلالي.
- في التقطيع الدلالي، نظرًا لأن صورة الإدخال والتسمية تتوافق مع واحد إلى واحد على البكسل، يتم اقتصاص صورة الإدخال عشوائيًا إلى شكل ثابت بدلاً من إعادة قياسها.

14.9.4. التمارين

1. كيف يمكن تطبيق التقطيع الدلالي في المركبات ذاتية القيادة وتشخيصات الصور الطبية؟ هل يمكنك التفكير في تطبيقات أخرى؟
2. أذكر أوصاف زيادة البيانات في القسم 14.1. أي من طرق زيادة الصور image augmentation المستخدمة في تصنيف الصور سيكون غير قابل للتطبيق في التقطيع الدلالي؟

14.10. الالتفاف المنقول Transposed Convolution

طبقات CNN التي رأيناها حتى الآن، مثل الطبقات التلافيفية convolutional layers (القسم 7.2) وطبقات التجميع pooling layers (القسم 7.5)، عادةً ما تقلل (تختزل downsample) الأبعاد المكانية spatial dimensions (الارتفاع والعرض) للمدخلات، أو تبقيها دون تغيير. في التقطيع الدلالي semantic segmentation التي تصنف على مستوى البكسل، سيكون من المناسب إذا كانت الأبعاد المكانية للمدخلات والمخرجات هي نفسها. على سبيل المثال، يمكن لأبعاد القناة عند بكسل إخراج واحد الاحتفاظ بنتائج التصنيف لبكسل الإدخال في نفس الموضع المكاني.

لتحقيق ذلك، خاصة بعد تقليل الأبعاد المكانية بواسطة طبقات CNN، يمكننا استخدام نوع آخر من طبقات CNN التي يمكن أن تزيد (upample) الأبعاد المكانية لخرائط المعالم الوسيطة. في هذا القسم، سنقدم التفافًا منقولا transposed convolution، والذي يسمى أيضًا الالتفاف الجزئي الخطي ractionally-strided convolution (Dumoulin and Visin، 2016)، لعكس عمليات الاختزال بواسطة الالتفاف.

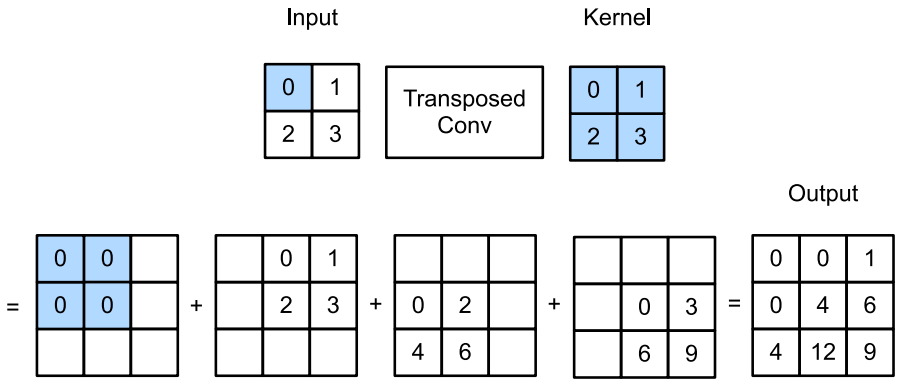
```
from mxnet import init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

14.10.1. عملية أساسية Basic Operation

تجاهل القنوات في الوقت الحالي، فلنبدأ بعملية الالتفاف المنقولة الأساسية transposed convolution بخطوة 1 وبدون حشو. افترض أننا حصلنا على موتر إدخال $n_h \times n_w$ ونواة $k_h \times k_w$. يؤدي تحريك نافذة النواة بخطوة 1 لـ n_w مرات في كل صف وn_h مرات في كل عمود إلى إجمالي $n_h n_w$ للنتائج الوسيطة. كل نتيجة وسيطة هي موتر $(n_h + k_h - 1) \times (n_w + k_w - 1)$ يتم تهيئته كأصفار. لحساب كل موتر وسيط، يتم ضرب كل عنصر في موتر الإدخال بالنواة بحيث يستبدل الموتر الناتج $k_h \times k_w$ جزءًا في كل موتر وسيط. لاحظ أن موضع الجزء

المستبدلفي كل موتر وسيط يتوافق مع موضع العنصرفي موتر الإدخال المستخدمفي الحساب.في النهاية، يتم جمع كل النتائج الوسيطة لإنتاج الناتج.

كمثال، يوضح الشكل 14.10.1 كيف يتم حساب الالتفاف المنقول مع النواة 2 × 2 لموتر الإدخال 2 × 2.



الشكل. 14.10.1 تناوب الالتفاف مع النواة 2 × 2. الأجزاء المظللة هي جزء من موتر وسيط بالإضافة إلى عناصر موتر المدخلات والنواة المستخدمةفي الحساب.

يمكننا تنفيذ عملية الالتفاف الأساسية المنقولة `trans_conv` لمصفوفة إدخال `X` ومصفوفة النواة `K`.

```
def trans_conv(X, K):
    h, w = K.shape
    Y = np.zeros((X.shape[0] + h - 1, X.shape[1] + w -
1))
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            Y[i: i + h, j: j + w] += X[i, j] * K
    return Y
```

على عكس الالتفاف العادي regular convolution (في القسم 7.2) الذي يقلل من عناصر الإدخال عبر النواة، فإن الالتفاف المنقول يبث broadcasts عناصر الإدخال عبر النواة، وبالتالي ينتج مخرجات أكبر من المدخلات. يمكننا إنشاء موتر الإدخال `X` وموتّر النواة `K` من الشكل 14.10.1 للتحقق من صحة إخراج التنفيذ أعلاه لعملية الالتفاف المنقولة ثنائية الأبعاد الأساسية.

```
X = np.array([[0.0, 1.0], [2.0, 3.0]])
K = np.array([[0.0, 1.0], [2.0, 3.0]])
trans_conv(X, K)
```

```
array([[ 0.,  0.,  1.],
```

```
       [ 0.,  4.,  6.],
       [ 4., 12.,  9.]])
```

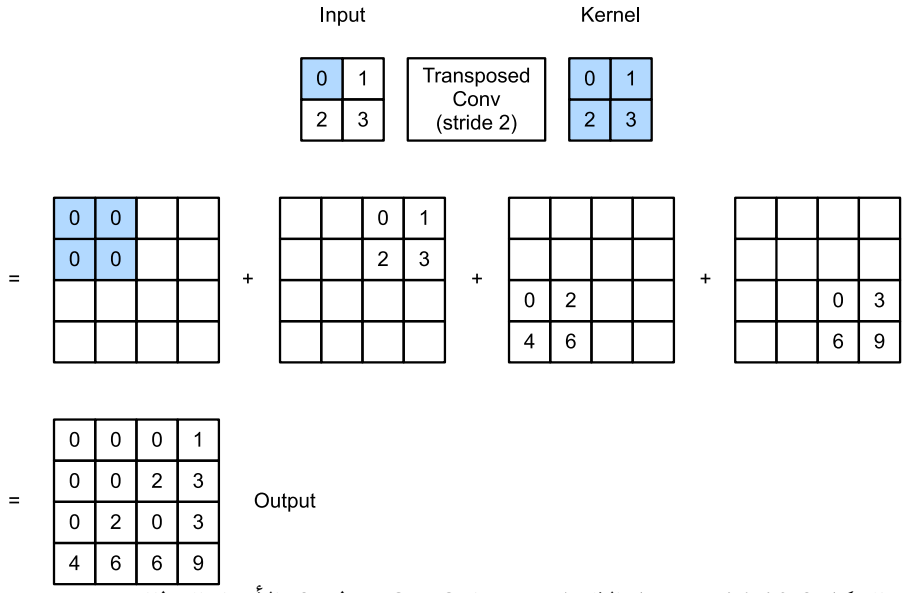
14.10.2. الحشو والخطوات والقنوات المتعددة Padding, Strides, and Multiple Channels

يختلف عن الالتفاف العادي حيث يتم تطبيق الحشو على الإدخال، يتم تطبيقه على الإخراجفي الالتفاف المنقول. على سبيل المثال، عند تحديد رقم المساحة المتروكة على جانبي الارتفاع والعرض كـ 1، ستتم إزالة الصفوف والأعمدة الأولى والأخيرة من ناتج الالتفاف المنقول.

```
tconv = nn.Conv2DTranspose(1, kernel_size=2, padding=1)
tconv.initialize(init.Constant(K))
tconv(X)
```

```
array([[[[4.]]]])
```

في الالتفاف المنقول، يتم تحديد الخطوات للنتائج الوسيطة (وبالتالي الإخراج)، وليس للإدخال. باستخدام نفس موتر المدخلات والنواة من الشكل 14.10.1، يؤدي تغيير الخطوة من 1 إلى 2 إلى زيادة كل من ارتفاع ووزن الموترات الوسيطة، ومن ثم موتر الإخراجفي الشكل 14.10.2.



الشكل 14.10.2 تم تبديل الالتفاف مع نواة 2 × 2 بخطوة 2. الأجزاء المظللة هي جزء من موتر وسيط بالإضافة إلى عناصر موتر المدخلات والنواة المستخدمةفي الحساب.

يمكن لمقتطف الكود التالي التحقق من صحة خرج الالتفاف المنقول لخطوة 2في الشكل 14.10.2.

```
tconv = nn.Conv2DTranspose(1, kernel_size=2, strides=2)
```

```
tconv.initialize(init.Constant(K))
tconv(X)
```

```
array([[[[0., 0., 0., 1.],
         [0., 0., 2., 3.],
         [0., 2., 0., 3.],
         [4., 6., 6., 9.]]]])
```

بالنسبة لقنوات الإدخال والإخراج المتعددة، يعمل الالتفاف المنقول بنفس طريقة الالتفاف العادي. افترض أن الإدخال يحتوي على c_i قنوات، وأن الالتفاف المنقول يخصص $k_h \times k_w$ موتر النواة لكل قناة إدخال. عندما يتم تحديد قنوات إخراج متعددة، سيكون لدينا نواة $c_i \times k_h \times k_w$ لكل قناة إخراج.

كما هو الحال في الكل، إذا قمنا بإدخال X الى الطبقة التلافيفية f للإخراج $Y = f(X)$ وإنشاء طبقة تلافيفية منقولة g بنفس المعلمات الفائقة f باستثناء عدد قنوات الإخراج التي تمثل عدد القنوات في X ، فسيكون لها نفس الشكل كـ X. يمكن توضيح ذلك في المثال التالي.

```
X = np.random.uniform(size=(1, 10, 16, 16))
conv = nn.Conv2D(20, kernel_size=5, padding=2,
strides=3)
tconv = nn.Conv2DTranspose(10, kernel_size=5, padding=2,
strides=3)
conv.initialize()
tconv.initialize()
tconv(conv(X)).shape == X.shape
```

```
True
```

14.10.3. الاتصال بنقل المصفوفة Connection to Matrix Transposition

تتم تسمية الالتفاف المنقول بعد نقل المصفوفة matrix transposition. للتوضيح، دعنا أولاً نرى كيفية تنفيذ عمليات الضرب باستخدام المصفوفات. في المثال أدناه، نحدد 3×3 إدخال X و2×2 نواة الالتفاف K، ثم نستخدم الدالة corr2d لحساب ناتج الالتفاف Y.

```
X = np.arange(9.0).reshape(3, 3)
K = np.array([[1.0, 2.0], [3.0, 4.0]])
Y = d2l.corr2d(X, K)
Y
```

```
array([[27., 37.],
       [57., 67.]])
```

بعد ذلك، نعيد كتابة نواة الالتفاف K كمصفوفة وزن متفرقة W تحتوي على الكثير من الأصفار. شكل مصفوفة الوزن هو (4، 9)، حيث تأتي العناصر غير الصفرية من نواة الالتفاف K.

```
def kernel2matrix(K):
```

```
    k, W = np.zeros(5), np.zeros((4, 9))
    k[:2], k[3:5] = K[0, :], K[1, :]
    W[0, :5], W[1, 1:6], W[2, 3:8], W[3, 4:] = k, k, k,
k
    return W

W = kernel2matrix(K)
W
```

```
array([[1., 2., 0., 3., 4., 0., 0., 0., 0.],
       [0., 1., 2., 0., 3., 4., 0., 0., 0.],
       [0., 0., 0., 1., 2., 0., 3., 4., 0.],
       [0., 0., 0., 0., 1., 2., 0., 3., 4.]])
```

اربط الإدخال X صفًا تلو الآخر للحصول على متجه بطول 9. ثم ضرب المصفوفة W والمتجه X يعطي متجهًا للطول 4. بعد إعادة تشكيله، يمكننا الحصول على نفس النتيجة Y من عملية الالتفاف الأصلية أعلاه: لقد قمنا للتو بتنفيذ عمليات التفافية باستخدام عمليات ضرب المصفوفات.

```
Y == np.dot(W, X.reshape(-1)).reshape(2, 2)
```

```
array([[ True,  True],
       [ True,  True]])
```

وبالمثل، يمكننا تنفيذ الالتفافات المنقولة باستخدام ضرب المصفوفة.في المثال التالي، نأخذ 2×2 الناتج Y من الالتفاف المنتظم أعلاه كمدخل للالتفاف المنقول. لتنفيذ هذه العملية بضرب المصفوفات، نحتاج فقط إلى تبديل مصفوفة الوزن $\mathbf{W}$ بالشكل الجديد (9,4).

```
Z = trans_conv(Y, K)
Z == np.dot(W.T, Y.reshape(-1)).reshape(3, 3)
```

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

ضع في اعتبارك تنفيذ الالتفاف بضرب المصفوفات. بالنظر إلى متجه الإدخال $\mathbf{x}$ ومصفوفة الوزن $\mathbf{W}$ ، يمكن تنفيذ دالة الانتشار الأمامية للالتفاف بضرب مدخلاتها بمصفوفة الوزن وإخراج متجه $\mathbf{y} = \mathbf{W}\mathbf{x}$. نظرًا لأن الانتشار الخلفي يتبع قاعدة السلسلة و $\nabla_{\mathbf{x}}\mathbf{y} = \mathbf{W}^\top$ ، ويمكن تنفيذ دالة الانتشار الخلفي للالتفاف بضرب مدخلاته مع مصفوفة الوزن المنقولة $\mathbf{W}^\top$. لذلك، يمكن للطبقة التلافيفية المنقولة فقط تبادل دالة الانتشار الأمامي ودالة الانتشار الخلفي للطبقة التلافيفية: تضاعف دالتا الانتشار الأمامي والانتشار الخلفي متجه الإدخال مع $\mathbf{W}^\top$ و $\mathbf{W}$ ، على التوالي.

14.10.4. الملخص

- على عكس الالتفاف العادي regular convolution الذي يقلل من عناصر الإدخال عبر النواة، فإن الالتفاف المنقول transposed convolution يبث عناصر الإدخال عبر النواة، وبالتالي ينتج مخرجات أكبر من المدخلات.
- إذا قمنا بإدخال X الى الطبقة التلافيفية f لإخراج $Y = f(X)$ وإنشاء طبقة تلافيفية منقولة g بنفس المعلمات الفائقة مثل f باستثناء عدد قنوات الإخراج التي تمثل عدد القنواتفي X ، فسيكون $g(Y)$ لها نفس الشكل كـ X.
- يمكننا تنفيذ الالتفافات باستخدام ضرب المصفوفات. يمكن للطبقة التلافيفية المنقولة فقط تبادل دالة الانتشار الأمامية ودالة الانتشار الخلفي للطبقة التلافيفية.

14.10.5. التمارين

1. في القسم 14.10.3، يكون لمدخل الالتفاف X ومخرج الالتفاف المنقول Z نفس الشكل. هل لديهم نفس القيمة؟ لماذا؟
2. هل من الفعال استخدام ضرب المصفوفات لتنفيذ الالتفافات؟ لماذا؟

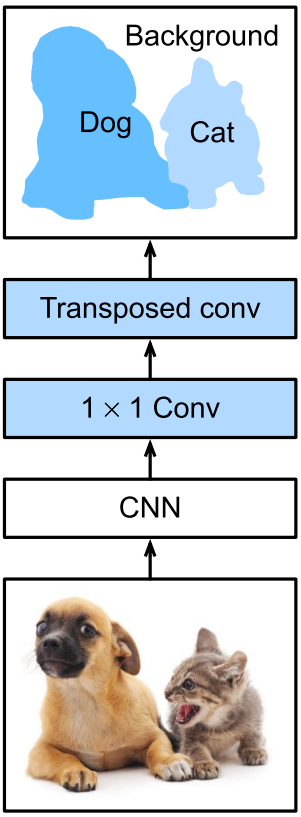
14.11. شبكات تلافيفية بالكامل Fully Convolutional Networks

كما تمت مناقشتهفي القسم 14.9، فإن التقطيع الدلالي semantic segmentation تصنف الصورفي مستوى البكسل. تستخدم الشبكة التلافيفية fully convolutional network (FCN) شبكة عصبية تلافيفية لتحويل بكسلات الصورة إلى فئات بكسل (Long et al.، 2015). على عكس شبكات CNN التي واجهناها سابقًا لتصنيف الصور أو اكتشاف الكائنات، تقوم الشبكة التلافيفية بالكامل بتحويل ارتفاع وعرض خرائط المعالم الوسيطة إلى تلك الخاصة بالصورة المدخلة: يتم تحقيق ذلك من خلال الطبقة التلافيفية المنقولة transposed convolutional layer المقدمةفي القسم 14.10. نتيجة لذلك، يكون لإخراج التصنيف وصورة الإدخال تطابق واحد لواحدفي مستوى البكسل: يحمل بُعد القناةفي أي بكسل إخراج نتائج التصنيف لبكسل الإدخالفي نفس الموضع المكاني.

```
%matplotlib inline
from mxnet import gluon, image, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

14.11.1. النموذج The Model

هنا نصف التصميم الأساسي لنموذج الشبكة التلافيفية بالكامل. كما هو مبين في الشكل 14.11.1، يستخدم هذا النموذج أولاً شبكة CNN لاستخراج ميزات الصورة، ثم يحول عدد القنوات إلى عدد الفئات عبر طبقة تلافيفية، وأخيراً يحول ارتفاع وعرض خرائط المعالم إلى تلك الخاصة بـ الصورة المدخلة عبر الالتفاف المنقول المقدم في القسم 14.10. نتيجة لذلك، يكون لإخراج النموذج نفس ارتفاع وعرض صورة الإدخال، حيث تحتوي قناة الإخراج على الفئات المتوقعة لبكسل الإدخال في نفس الموضع المكاني.



الشكل 14.11.1 شبكة تلافيفية كاملة.

أدناه، نستخدم نموذج ResNet-18 الذي تم اختباره مسبقًا على مجموعة بيانات ImageNet لاستخراج ميزات الصورة والإشارة إلى مثيل النموذج باعتباره `pretrained_net`. تتضمن الطبقات القليلة الأخيرة من هذا النموذج طبقة تجميع متوسط عالمي وطبقة متصلة بالكامل: ليست هناك حاجة إليها في الشبكة التلافيفية بالكامل.

```
pretrained_net =
gluon.model_zoo.vision.resnet18_v2(pretrained=True)
pretrained_net.features[-3:], pretrained_net.output
```

```
HybridSequential(
   (0): Activation(relu)
   (1): GlobalAvgPool2D(size=(1, 1), stride=(1, 1),
padding=(0, 0), ceil_mode=True, global_pool=True,
pool_type=avg, layout=NCHW)
   (2): Flatten
 ),
 Dense(512 -> 1000, linear))
```

بعد ذلك، نقوم بإنشاء شبكة مثيل الشبكة التلافيفية بالكامل. يقوم بنسخ جميع الطبقات سابقة التدريبفي ResNet-18 باستثناء طبقة تجميع المتوسط العالمي النهائية والطبقة المتصلة بالكامل الأقرب إلى الإخراج.

```
net = nn.HybridSequential()
for layer in pretrained_net.features[:-2]:
    net.add(layer)
```

نظرًا لمدخل بارتفاع وعرض 320 و 480 على التوالي، فإن الانتشار الأمامي للشبكة يقلل من ارتفاع الإدخال وعرضه إلى 1/32 من الأصل ، أي 10 و 15.

```
X = np.random.uniform(size=(1, 3, 320, 480))
net(X).shape
```

```
(1, 512, 10, 15)
```

بعد ذلك، نستخدم طبقة تلافيفية 1×1 لتحويل عدد قنوات الإخراج إلى عدد الفئات (21) لمجموعة بيانات Pascal VOC2012. أخيرًا، نحتاج إلى زيادة ارتفاع وعرض خرائط المعالم بمقدار 32 مرة لتغييرها مرة أخرى إلى ارتفاع وعرض صورة الإدخال. تذكر كيفية حساب الشكل الناتج لطبقة تلافيفيةفي القسم 7.3. لان $(320 - 64 + 16 \times 2 + 32)/32 = 10$ ، نقوم ببناء طبقة تلافيفية منقولة بخطوة 32 ، مع ضبط ارتفاع وعرض النواة على الحشو 64. بشكل عام، يمكننا أن نرى أنه بالنسبة للخطوة s ، الحشو $s/2$ (بافتراض أن $s/2$ عدد صحيح)، وارتفاع وعرض النواة $2s$ ، فإن الالتفاف المنقول سيزيد من ارتفاع وعرض المدخلات بـ s مرات.

```
num_classes = 21
net.add(nn.Conv2D(num_classes, kernel_size=1),
        nn.Conv2DTranspose(
            num_classes, kernel_size=64, padding=16,
strides=32))
```

14.11.2. تهيئة الطبقات التلافيفية المنقولة Initializing Transposed Convolutional Layers

نحن نعلم بالفعل أن الطبقات التلافيفية المنقولة يمكن أن تزيد من ارتفاع وعرض خرائط المعالم.في معالجة الصور، قد نحتاج إلى توسيع نطاق الصورة، أي upsampling. يعد الاستيفاء الخطي أحد أساليب upsampling الشائعة الاستخدام. غالبًا ما يستخدم أيضًا لتهيئة الطبقات التلافيفية المنقولة.

لشرح الاستيفاء ثنائي الخطي bilinear interpolation، لنفترض أنه بالنظر إلى صورة الإدخال نريد حساب كل بكسل من صورة الإخراج المكبرة upsampled output image. من أجل حساب بكسل صورة الإخراج عند الإحداثيات (x, y) ، قم أولاً بتعيين (x, y) عند الاحداثيات (x', y') على صورة الإدخال، على سبيل المثال، وفقًا لنسبة حجم الإدخال إلى حجم الإخراج. لاحظ أن x' المعينة و y' هي الأرقام الحقيقية. ثم ابحث عن وحدات البكسل الأربعة الأقرب للاحداثيات (x', y')في صورة الإدخال. أخيرًا، يتم حساب بكسل صورة الإخراج عند الإحداثيات (x, y) بناءً على أقرب وحدات بكسل الأربعة هذه على صورة الإدخال والمسافة النسبية من (x', y').

يمكن تنفيذ اختزال الاستيفاء ثنائي الخطي بواسطة الطبقة التلافيفية المنقولة مع إنشاء النواة بواسطة دالة `bilinear_kernel` التالية. نظرًا لقيود المساحة space limitations، فإننا نقدم فقط تنفيذ دالة `bilinear_kernel` أدناه دون مناقشات حول تصميم الخوارزمية الخاص بها.

```
def bilinear_kernel(in_channels, out_channels,
kernel_size):
    factor = (kernel_size + 1) // 2
    if kernel_size % 2 == 1:
        center = factor - 1
    else:
        center = factor - 0.5
    og = (np.arange(kernel_size).reshape(-1, 1),
          np.arange(kernel_size).reshape(1, -1))
    filt = (1 - np.abs(og[0] - center) / factor) * \
           (1 - np.abs(og[1] - center) / factor)
    weight = np.zeros((in_channels, out_channels,
kernel_size, kernel_size))
    weight[range(in_channels), range(out_channels), :,
:] = filt
    return np.array(weight)
```

دعونا نجرب تكبير upsampling الاستيفاء ثنائي الخطي الذي يتم تنفيذه بواسطة طبقة تلافيفية منقولة. نحن نبني طبقة تلافيفية منقولة تضاعف الطول والوزن، ونهيئ نواتها بدالة `bilinear_kernel`.

```
conv_trans = nn.Conv2DTranspose(3, kernel_size=4,
padding=1, strides=2)
conv_trans.initialize(init.Constant(bilinear_kernel(3,
3, 4)))
```

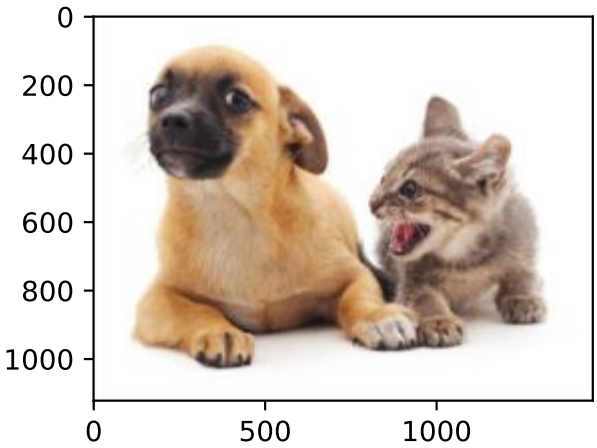
اقرأ الصورة `X` وقم بتعيين إخراج التكبير إلى `Y`. لطباعة الصورة، نحتاج إلى ضبط موضع بُعد القناة.

```
img = image.imread('../img/catdog.jpg')
X = np.expand_dims(img.astype('float32').transpose(2, 0,
1), axis=0) / 255
Y = conv_trans(X)
out_img = Y[0].transpose(1, 2, 0)
```

كما نرى، فإن الطبقة التلافيفية المنقولة تزيد من ارتفاع وعرض الصورة بمعامل اثنين. باستثناء المقاييس المختلفةفي الإحداثيات، فإن الصورة التي تم تكبيرها بواسطة الاستيفاء ثنائي الخطي وتبدو الصورة الأصلية المطبوعةفي القسم 14.3 متشابهة.

```
d2l.set_figsize()
print('input image shape:', img.shape)
d2l.plt.imshow(img.asnumpy());
print('output image shape:', out_img.shape)
d2l.plt.imshow(out_img.asnumpy());
```

```
input image shape: (561, 728, 3)
output image shape: (1122, 1456, 3)
```



في شبكة تلافيفية بالكامل، نقوم بتهيئة الطبقة التلافيفية المنقولة مع اختزال الاستيفاء ثنائي الخطوط. بالنسبة للطبقة التلافيفية 1×1 ، نستخدم تهيئة Xavier.

```
W = bilinear_kernel(num_classes, num_classes, 64)
net[-1].initialize(init.Constant(W))
net[-2].initialize(init=init.Xavier())
```

14.11.3. قراءة مجموعة البيانات Reading the Dataset

نقرأ مجموعة بيانات التقطيع الدلالي كما هو مقدمفي القسم 14.9. يتم تحديد شكل صورة الإخراج للاقتصاص العشوائي على انه 320×480: يمكن قسمة كل من الطول والعرض على 32.

```
batch_size, crop_size = 32, (320, 480)
train_iter, test_iter = d2l.load_data_voc(batch_size,
crop_size)
```

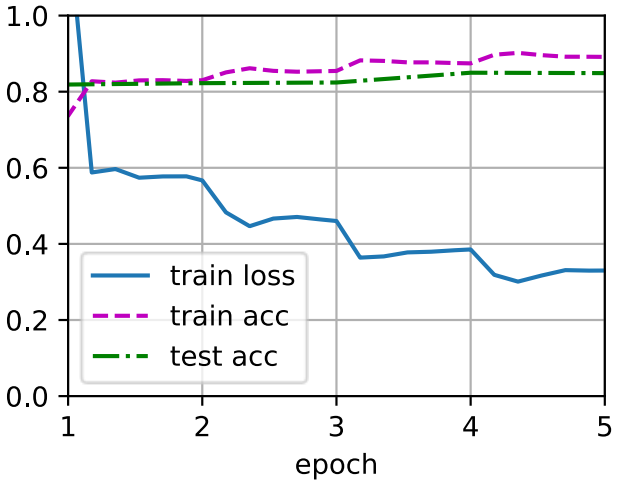
```
read 1114 examples
read 1078 examples
```

14.11.4. التدريب Training

الآن يمكننا تدريب شبكتنا التلافيفية تم إنشاؤها بالكامل. لا تختلف دالة الخطأ وحساب الدقة هنا جوهريًا عن تلك الموجودةفي تصنيف الصورفي الفصول السابقة. نظرًا لأننا نستخدم قناة الإخراج للطبقة التلافيفية المنقولة للتنبؤ بفئة كل بكسل، يتم تحديد بُعد القناةفي حساب الخطأ. بالإضافة إلى ذلك، يتم حساب الدقة بناءً على صحة الفئة المتوقعة لجميع وحدات البكسل.

```
num_epochs, lr, wd, devices = 5, 0.1, 1e-3,
d2l.try_all_gpus()
loss = gluon.loss.SoftmaxCrossEntropyLoss(axis=1)
net.collect_params().reset_ctx(devices)
trainer = gluon.Trainer(net.collect_params(), 'sgd',
                        {'learning_rate': lr, 'wd': wd})
d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs, devices)
```

```
loss 0.330, train acc 0.891, test acc 0.849
195.2 examples/sec on [gpu(0), gpu(1)]
```



14.11.5. التنبؤ Prediction

عند التنبؤ، نحتاج إلى توحيد standardize صورة الإدخال في كل قناة وتحويل الصورة إلى تنسيق الإدخال رباعي الأبعاد الذي تتطلبه شبكة CNN.

```
def predict(img):
    X = test_iter._dataset.normalize_image(img)
    X = np.expand_dims(X.transpose(2, 0, 1), axis=0)
    pred = net(X.as_in_ctx(devices[0])).argmax(axis=1)
    return pred.reshape(pred.shape[1], pred.shape[2])
```

لرسم الفئة المتوقعة لكل بكسل، نقوم بتعيين الفئة المتوقعة مرة أخرى إلى لون التسمية الخاص بها في مجموعة البيانات.

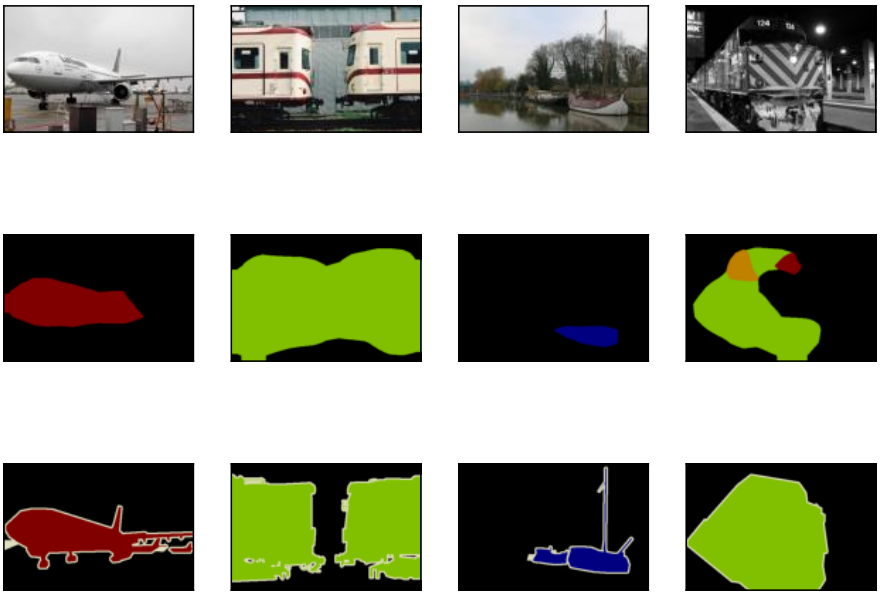
```
def label2image(pred):
    colormap = np.array(d2l.VOC_COLORMAP,
ctx=devices[0], dtype='uint8')
    X = pred.astype('int32')
    return colormap[X, :]
```

تختلف الصور في مجموعة بيانات الاختبار من حيث الحجم والشكل. نظرًا لأن النموذج يستخدم طبقة تلافيفية منقولة بخطوة 32، عندما يكون ارتفاع أو عرض صورة الإدخال غير قابل للتجزئة بمقدار 32، فإن ارتفاع أو عرض الطبقة التلافيفية المنقولة سينحرف عن شكل الصورة المدخلة. لمعالجة هذه المشكلة، يمكننا اقتصاص مناطق مستطيلة متعددة بارتفاع وعرض يمثلان مضاعفات عددية صحيحة لـ 32 في الصورة، وتنفيذ انتشار أمامي على وحدات البكسل في هذه المناطق بشكل منفصل. لاحظ أن اتحاد هذه المساحات المستطيلة يحتاج إلى تغطية الصورة

المدخلة بالكامل. عندما يتم تغطية بكسل بمناطق مستطيلة متعددة، يمكن إدخال متوسط مخرجات الالتفاف المنقولةفي مناطق منفصلة لنفس البكسلفي عملية softmax للتنبؤ بالفئة.

من أجل التبسيط، نقرأ فقط عددًا قليلاً من صور الاختبار الأكبر، ونقص مساحة 480 × 320 للتنبؤ بدءًا من الزاوية العلوية اليسرى للصورة. بالنسبة إلى صور الاختبار هذه، نقوم بطباعة المساحات التي تم اقتصاصها ونتائج التنبؤ والحقيقة الأساسية صفًا تلو الآخر.

```
voc_dir = d2l.download_extract('voc2012',
'VOCdevkit/VOC2012')
test_images, test_labels = d2l.read_voc_images(voc_dir,
False)
n, imgs = 4, []
for i in range(n):
    crop_rect = (0, 0, 480, 320)
    X = image.fixed_crop(test_images[i], *crop_rect)
    pred = label2image(predict(X))
    imgs += [X, pred, image.fixed_crop(test_labels[i],
*crop_rect)]
d2l.show_images(imgs[::3] + imgs[1::3] + imgs[2::3], 3,
n, scale=2);
```



14.11.6. الملخص

- تستخدم الشبكة التلافيفية بالكامل أولاً CNN لاستخراج ميزات الصورة، ثم تقوم بتحويل عدد القنوات إلى عدد الفئات عبر طبقة تلافيفية 1×1 ، وأخيراً تقوم بتحويل ارتفاع وعرض خرائط المعالم إلى تلك الخاصة بالصورة المدخلة عبر الالتفاف المنقول.
- في شبكة تلافيفية بالكامل، يمكننا استخدام الـ upsampling الاستيفاء ثنائي الخطي bilinear interpolation لتهيئة الطبقة التلافيفية المنقولة.

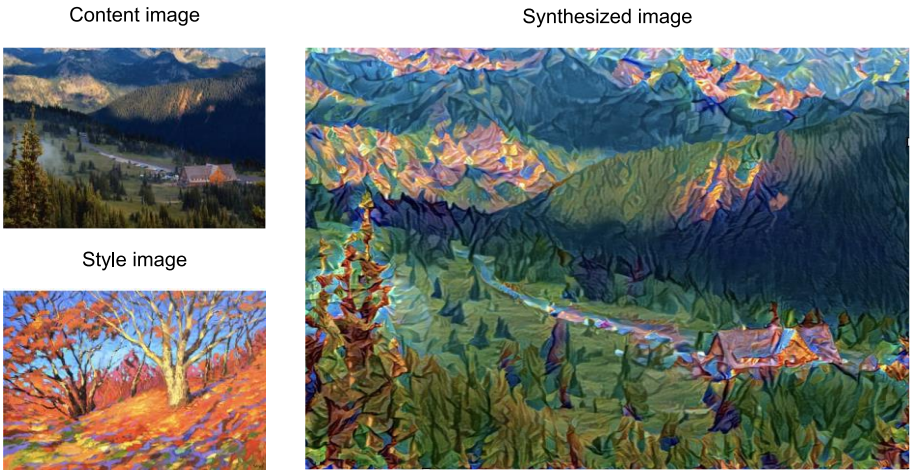
14.11.7. التمارين

1. إذا استخدمنا تهيئة Xavier للطبقة التلافيفية المنقولةفي التجربة فكيف تتغير النتيجة؟
2. هل يمكنك تحسين دقة النموذج من خلال ضبط المعلمات الفائقة؟
3. توقع فئات جميع وحدات البكسلفي صور الاختبار.
4. تستخدم ورقة الشبكة التلافيفية الأصلية أيضًا مخرجات بعض طبقات CNN الوسيطة (Long et al.، 2015). حاول تنفيذ هذه الفكرة.

14.12. نقل النمط العصبي Neural Style Transfer

إذا كنت من عشاق التصوير الفوتوغرافي، فقد تكون على دراية بالفلتر filter. يمكن أن يغير نمط ألوان الصور بحيث تصبح صور المناظر الطبيعية أكثر وضوحًا أو أن تكون الصور الشخصية ذات جلود بيضاء. ومع ذلك، عادةً ما يغير فلتر واحد جانبًا واحدًا فقط من الصورة. لتطبيق نمط مثالي على صورة ما، ربما تحتاج إلى تجربة العديد من مجموعات الفلاتر المختلفة. هذه العملية معقدة مثل ضبط المعلمات الفائقة للنموذج.

في هذا القسم، سنستفيد من التمثيلات الطبقية layerwise representations لشبكة CNN لتطبيق نمط صورة واحدة تلقائيًا على صورة أخرى، أي نقل النمط style transfer (Gatys et al.، 2016). تحتاج هذه المهمة إلى صورتين للإدخال: واحدة هي صورة المحتوى content image والأخرى هي صورة النمط style image. سنستخدم الشبكات العصبية لتعديل صورة المحتوى لجعلها قريبة من صورة النمطفي النمط. على سبيل المثال، صورة المحتوىفي الشكل 14.12.1 هي صورة مناظر طبيعية التقطناهافي حديقة Mount Rainier الوطنيةفي ضواحي سياتل، بينما الصورة النمطية عبارة عن لوحة زيتية مع موضوع أشجار البلوط الخريفية.في الصورة المركبة الناتجة، يتم تطبيق ضربات الفرشاة الزيتية لصورة النمط، مما يؤدي إلى ألوان أكثر إشراقًا، مع الحفاظ على الشكل الرئيسي للكائناتفي صورة المحتوى.

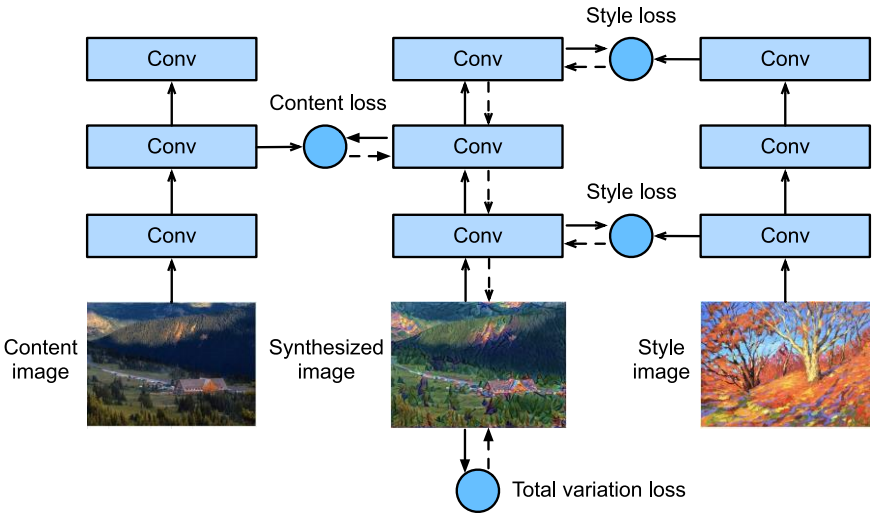


الشكل 14.12.1 بالنظر إلى صور المحتوى والنمط، ينتج عن نقل النمط صورة مركبة synthesized image.

14.12.1. الطريقة Method

يوضح الشكل 14.12.2 طريقة نقل النمط style transfer القائم على CNN مع مثال مبسط. أولاً، نقوم بتهيئة الصورة المركبة synthesized image، على سبيل المثال،في صورة المحتوى. هذه الصورة المركبة هي المتغير الوحيد الذي يحتاج إلى تحديث أثناء عملية نقل النمط، أي معلمات النموذج التي سيتم تحديثها أثناء التدريب. ثم نختار شبكة CNN مدربة مسبقاً لاستخراج ميزات الصورة وتجميد معلمات نموذجها أثناء التدريب. تستخدم شبكة CNN العميقة طبقات متعددة لاستخراج الميزات الهرمية للصور. يمكننا اختيار إخراج بعض هذه الطبقات كميزات محتوى أو ميزات نمط. خذ الشكل 14.12.2 كمثال. تحتوي الشبكة العصبية سابقة التدريب هنا على 3 طبقات تلافيفية، حيث تُخرج الطبقة الثانية ميزات المحتوى، وتخرج الطبقتان الأولى والثالثة ميزات النمط.

بعد ذلك، نحسب دالة الخطأ لنقل النمط من خلال الانتشار الأمامي (اتجاه الأسهم الصلبة)، وتحديث معلمات النموذج (الصورة المركبة للإخراج) من خلال الانتشار الخلفي (اتجاه الأسهم المتقطعة). تتكون دالة الخطأ المستخدمة بشكل شائعفي نقل النمط من ثلاثة أجزاء: (1) خطأ المحتوى content loss يجعل الصورة المركبة وصورة المحتوى قريبة من ميزات المحتوى؛ (2) خطأ النمط style loss يجعل الصورة المركبة وصورة النمط قريبة من ميزات النمط؛ و (3) يساعد خطأ التباين الكلي total variation loss على تقليل الضوضاءفي الصورة المركبة. أخيرًا، عند انتهاء تدريب النموذج، نقوم بإخراج معلمات النموذج الخاصة بنقل النمط لإنشاء الصورة المركبة النهائية.



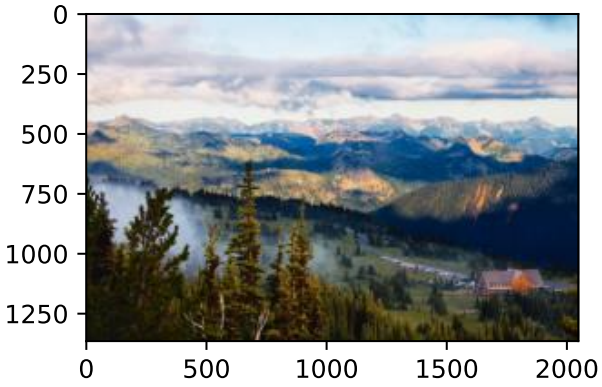
الشكل. 14.12.2 عملية نقل النمط المستندة إلى CNN. تُظهر الخطوط الصلبة اتجاه الانتشار الأمامي وتظهر الخطوط المنقطة انتشارًا خلفياً.

فيما يلي، سنشرح التفاصيل الفنية لنقل النمط عبر تجربة ملموسة.

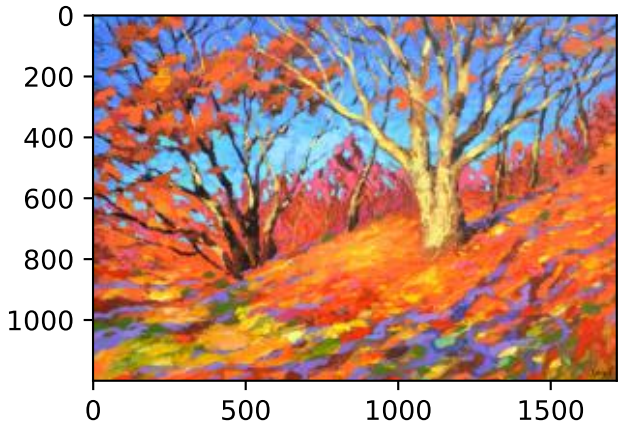
14.12.2. قراءة صور المحتوى والنمط Reading the Content and Style Images

أولا، نقرأ صور المحتوى والنمط. من محاور الإحداثيات المطبوعة، يمكننا معرفة أن هذه الصور لها أحجام مختلفة.

```
%matplotlib inline
from mxnet import autograd, gluon, image, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()

d2l.set_figsize()
content_img = image.imread('../img/rainier.jpg')
d2l.plt.imshow(content_img.asnumpy());
```




```
style_img = image.imread('../img/autumn-oak.jpg')
d2l.plt.imshow(style_img.asnumpy());
```



14.12.3. المعالجة المسبقة واللاحقة Preprocessing and Postprocessing

أدناه، نحدد دالتين للمعالجة المسبقة للصور والمعالجة اللاحقة. تعمل دالة المعالجة المسبقة `preprocess` على توحيد كل من قنوات RGB الثلاث لصورة الإدخال وتحويل النتائج إلى تنسيق إدخال CNN. تعمل دالة المعالجة اللاحقة `postprocess` على استعادة قيم البكسل في صورة الإخراج إلى قيمها الأصلية قبل التوحيد القياسي standardization. نظرًا لأن دالة طباعة الصور تتطلب أن يكون لكل بكسل قيمة فاصلة عائمة من 0 إلى 1، فإننا نستبدل أي قيمة أصغر من 0 أو أكبر من 1 بـ 0 أو 1، على التوالي.

```
rgb_mean = np.array([0.485, 0.456, 0.406])
rgb_std = np.array([0.229, 0.224, 0.225])

def preprocess(img, image_shape):
    img = image.imresize(img, *image_shape)
```

```
    img = (img.astype('float32') / 255 - rgb_mean) /
rgb_std
    return np.expand_dims(img.transpose(2, 0, 1),
axis=0)

def postprocess(img):
    img = img[0].as_in_ctx(rgb_std.ctx)
    return (img.transpose(1, 2, 0) * rgb_std +
rgb_mean).clip(0, 1)
```

14.12.4. استخراج الميزات Extracting Features

نستخدم نموذج VGG-19 الذي تم اختباره مسبقًا على مجموعة بيانات ImageNet لاستخراج ميزات الصورة (Gatys et al.، 2016).

```
pretrained_net =
gluon.model_zoo.vision.vgg19(pretrained=True)
```

لاستخراج ميزات المحتوى وميزات النمط للصورة، يمكننا تحديد إخراج طبقات معينةفي شبكة VGG. بشكل عام، كلما اقتربنا من طبقة الإدخال، أصبح استخراج تفاصيل الصورة أسهل، والعكس صحيح، كلما كان من الأسهل استخراج المعلومات العالمية للصورة. من أجل تجنب الاحتفاظ المفرط بتفاصيل صورة المحتوىفي الصورة المركبة، نختار طبقة VGG الأقرب إلى الإخراج كطبقة محتوى content layer لإخراج ميزات محتوى الصورة. نختار أيضًا إخراج طبقات VGG المختلفة لاستخراج ميزات النمط المحلية والعالمية. تسمى هذه الطبقات أيضًا طبقات النمط style layers. كما هو مذكورفي القسم 8.2، تستخدم شبكة VGG 5 كتل تلافيفية.في التجربة، اخترنا آخر طبقة تلافيفية من الكتلة التلافيفية الرابعة كطبقة محتوى، والطبقة التلافيفية الأولى من كل كتلة تلافيفية كطبقة نمط. يمكن الحصول على مؤشرات هذه الطبقات عن طريق طباعة مثيل `pretrained_net`.

```
style_layers, content_layers = [0, 5, 10, 19, 28], [25]
```

عند استخراج الميزات باستخدام طبقات VGG، نحتاج فقط إلى استخدام كل تلك الميزات من طبقة الإدخال إلى طبقة المحتوى أو طبقة النمط الأقرب إلى طبقة الإخراج. دعنا ننشئ شبكة مثيل شبكة جديدة `net`، والتي تحتفظ فقط بجميع طبقات VGG لاستخدامها لاستخراج الميزات.

```
net = nn.Sequential()
for i in range(max(content_layers + style_layers) + 1):
    net.add(pretrained_net.features[i])
```

بالنظر إلى الإدخال X، إذا استدعينا ببساطة شبكة الانتشار الأمامية net(X)، فيمكننا فقط الحصول على ناتج الطبقة الأخيرة. نظرًا لأننا نحتاج أيضًا إلى مخرجات الطبقات الوسيطة، نحتاج إلى إجراء حساب طبقة تلو طبقة والحفاظ على مخرجات طبقة النمط والمحتوى.

```
def extract_features(X, content_layers, style_layers):
    contents = []
    styles = []
    for i in range(len(net)):
        X = net[i](X)
        if i in style_layers:
            styles.append(X)
        if i in content_layers:
            contents.append(X)
    return contents, styles
```

تم تحديد دالتين أدناه: دالة get_contents تستخرج ميزات المحتوى من صورة المحتوى، وتستخرج دالة get_styles ميزات النمط من صورة النمط. نظرًا لعدم وجود حاجة لتحديث معلمات نموذج VGG المدربة مسبقًا أثناء التدريب، يمكننا استخراج المحتوى وميزات النمط حتى قبل بدء التدريب. نظرًا لأن الصورة المركبة عبارة عن مجموعة من معلمات النموذج التي سيتم تحديثها لنقل النمط، يمكننا فقط استخراج ميزات المحتوى والأسلوب للصورة المركبة عن طريق استدعاء دالة extract_features أثناء التدريب.

```
def get_contents(image_shape, device):
    content_X = preprocess(content_img,
image_shape).copyto(device)
    contents_Y, _ = extract_features(content_X,
content_layers, style_layers)
    return content_X, contents_Y

def get_styles(image_shape, device):
    style_X = preprocess(style_img,
image_shape).copyto(device)
    _, styles_Y = extract_features(style_X,
content_layers, style_layers)
    return style_X, styles_Y
```

14.12.5. تحديد دالة الخطأ Defining the Loss Function

الآن سوف نصف دالة الخطأ لنقل النمط. تتكون دالة الخطأ من خطأ المحتوى وخطأ النمط وخطأ التباين الكلية.

14.12.5.1. خطأالمحتوى Content Loss

على غرار دالة الخطأفي الانحدار الخطي، يقيس خطأ المحتوى الاختلاففي ميزات المحتوى بين الصورة المركبة وصورة المحتوى عبر دالة الخطأ التربيعية. المدخلان لدالة الخطأ التربيعية هما مخرجات طبقة المحتوى المحسوبة بواسطة دالة `extract_features`.

```
def content_loss(Y_hat, Y):
    return np.square(Y_hat - Y).mean()
```

14.12.5.2. خطأالنمط Style Loss

يستخدم خطأ النمط، على غرار خطأ المحتوى، أيضًا دالة الخطأ التربيعية لقياس الاختلاففي النمط بين الصورة المركبة وصورة النمط. للتعبير عن إخراج النمط لأي طبقة نمط، نستخدم أولاً دالة `extract_features` لحساب إخراج طبقة النمط. لنفترض أن الناتج يحتوي على 1 مثال، قنوات c ، ارتفاع h، وعرض w، يمكننا تحويل هذا الناتج إلى مصفوفة $\mathbf{X}$ ذات c صفوف وhw أعمدة. يمكن اعتبار هذه المصفوفة على أنها سلسلة من cمتجهات $\mathbf{x}_1, \ldots, \mathbf{x}_c$، كل منها لديه طول hw. هنا، يمثل المتجه $\mathbf{x}_i$ ميزة نمط القناة i.

في مصفوفة جرام Gram matrix لهذه المتجهات $\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{c \times c}$ ، يكون العنصر x_{ij}في الصف i والعمود j هو حاصل الضرب القياسي للمتجهات $\mathbf{x}_i$ و$\mathbf{x}_j$. إنه يمثل الارتباط بين ميزات أسلوب القنوات i وj. نستخدم مصفوفة جرام لتمثيل إخراج النمط لأي طبقة نمط. لاحظ أنه عندما تكون قيمة hw أكبر، فمن المحتمل أن يؤدي ذلك إلى قيم أكبرفي مصفوفة جرام. لاحظ أيضًا أن ارتفاع وعرض مصفوفة جرام يمثلان عددًا من القنوات. للسماح بخطأ النمط بعدم التأثر بهذه القيم، تقسم الدالة `gram`أدناه مصفوفة غرام على عدد عناصرها، أي chw.

```
def gram(X):
    num_channels, n = X.shape[1], d2l.size(X) //
X.shape[1]
    X = X.reshape((num_channels, n))
    return np.dot(X, X.T) / (num_channels * n)
```

من الواضح أن مدخلي مصفوفة جرام لدالة الخطأ التربيعية لفقدان النمط يعتمدان على مخرجات طبقة النمط للصورة المركبة وصورة النمط. من المفترض هنا أن مصفوفة جرام `gram_Y` بناءً على نمط الصورة قد تم حسابها مسبقًا.

```
def style_loss(Y_hat, gram_Y):
    return np.square(gram(Y_hat) - gram_Y).mean()
```

14.12.5.3. إجمالي خطأ التباين Total Variation Loss

في بعض الأحيان، تحتوي الصورة المركبة المكتسبة learned synthesized image على الكثير من الضوضاء عالية التردد، أي وحدات البكسل الساطعة أو الداكنة بشكل خاص. إحدى

طرق تقليل الضوضاء الشائعة هي التباين الكلي لتقليل الضوضاء total variation denoising.
قم بالإشارة إلى $x_{i,j}$ قيمة البكسل عند الإحداثيات (i,j). تقليل خطأ التباين الكلي:

$$\sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|$$

يجعل قيم وحدات البكسل المجاورة على الصورة المركبة أقرب.

```
def tv_loss(Y_hat):
    return 0.5 * (np.abs(Y_hat[:, :, 1:, :] - Y_hat[:,
:, :-1, :]).mean() +
                  np.abs(Y_hat[:, :, :, 1:] - Y_hat[:,
:, :, :-1]).mean())
```

14.12.5.4. دالة الخطأ Loss Function

دالة الخطأ لنقل النمط هي المجموع المرجح لخطأ المحتوى وخطأ النمط وخطأ التباين الكلي.
من خلال ضبط معلمات الوزن الفائقة هذه، يمكننا الموازنة بين الاحتفاظ بالمحتوى ونقل النمط وتقليل الضوضاء على الصورة المركبة.

```
content_weight, style_weight, tv_weight = 1, 1e4, 10

def compute_loss(X, contents_Y_hat, styles_Y_hat,
contents_Y, styles_Y_gram):
    # Calculate the content, style, and total variance
losses respectively
    contents_l = [content_loss(Y_hat, Y) *
content_weight for Y_hat, Y in zip(
        contents_Y_hat, contents_Y)]
    styles_l = [style_loss(Y_hat, Y) * style_weight for
Y_hat, Y in zip(
        styles_Y_hat, styles_Y_gram)]
    tv_l = tv_loss(X) * tv_weight
    # Add up all the losses
    l = sum(styles_l + contents_l + [tv_l])
    return contents_l, styles_l, tv_l, l
```

14.12.6. تهيئة الصورة المركبة Initializing the Synthesized Image

في نقل النمط، الصورة المركبة هي المتغير الوحيد الذي يحتاج إلى التحديث أثناء التدريب.
وبالتالي، يمكننا تحديد نموذج بسيط ، `SynthesizedImage`، والتعامل مع الصورة المركبة كمعلمات نموذج.في هذا النموذج، يقوم الانتشار الأمامي بإرجاع معلمات النموذج فقط.

```
class SynthesizedImage(nn.Block):
```

```
    def __init__(self, img_shape, **kwargs):
        super(SynthesizedImage, self).__init__(**kwargs)
        self.weight = self.params.get('weight',
shape=img_shape)

    def forward(self):
        return self.weight.data()
```

بعد ذلك، نحدد دالة `get_inits`. تنشئ هذه الدالة مثيلًا لنموذج الصورة المركب وتهيئته إلى الصورة X. يتم حساب مصفوفات غرام لصورة النمطي طبقات النمط المختلفة، `styles_Y_gram`، قبل التدريب.

```
def get_inits(X, device, lr, styles_Y):
    gen_img = SynthesizedImage(X.shape)
    gen_img.initialize(init.Constant(X), ctx=device,
force_reinit=True)
    trainer = gluon.Trainer(gen_img.collect_params(),
'adam',
                            {'learning_rate': lr})
    styles_Y_gram = [gram(Y) for Y in styles_Y]
    return gen_img(), styles_Y_gram, trainer
```

14.12.7. التدريب Training

عند تدريب النموذج على نقل النمط، نستخرج باستمرار ميزات المحتوى وميزات النمط للصورة المركبة، ونحسب دالة الخطأ. يحدد أدناه حلقة التدريب.

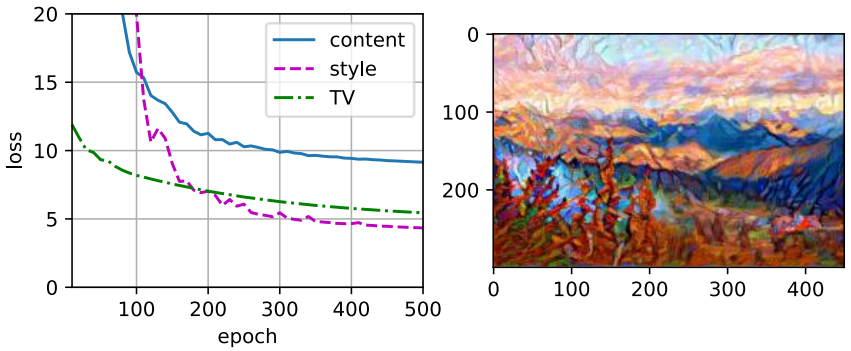
```
def train(X, contents_Y, styles_Y, device, lr,
num_epochs, lr_decay_epoch):
    X, styles_Y_gram, trainer = get_inits(X, device, lr,
styles_Y)
    animator = d2l.Animator(xlabel='epoch',
ylabel='loss',
                            xlim=[10, num_epochs],
ylim=[0, 20],
                            legend=['content', 'style',
'TV'],
                            ncols=2, figsize=(7, 2.5))
    for epoch in range(num_epochs):
        with autograd.record():
            contents_Y_hat, styles_Y_hat =
extract_features(
                X, content_layers, style_layers)
```

```
            contents_l, styles_l, tv_l, l =
compute_loss(
                X, contents_Y_hat, styles_Y_hat,
contents_Y, styles_Y_gram)
        l.backward()
        trainer.step(1)
        if (epoch + 1) % lr_decay_epoch == 0:

trainer.set_learning_rate(trainer.learning_rate * 0.8)
        if (epoch + 1) % 10 == 0:

animator.axes[1].imshow(postprocess(X).asnumpy())
            animator.add(epoch + 1,
[float(sum(contents_l)),

float(sum(styles_l)), float(tv_l)])
    return X
```

الآن نبدأفي تدريب النموذج. نقوم بإعادة قياس ارتفاع وعرض المحتوى ونمط الصور إلى 300 × 450 بكسل. نستخدم صورة المحتوى لتهيئة الصورة المركبة.

```
device, image_shape = d2l.try_gpu(), (450, 300)
net.collect_params().reset_ctx(device)
content_X, contents_Y = get_contents(image_shape,
device)
_, styles_Y = get_styles(image_shape, device)
output = train(content_X, contents_Y, styles_Y, device,
0.9, 500, 50)
```

يمكننا أن نرى أن الصورة المركبة تحافظ على مشهد وكائنات صورة المحتوى، وتنقل لون صورة النمطفي نفس الوقت. على سبيل المثال، تحتوي الصورة المركبة على كتل من الألوان مثل تلك الموجودةفي صورة النمط. تحتوي بعض هذه الكتل على نسيج دقيق من ضربات الفرشاة.



14.12.8. الملخص

- تتكون دالة الخطأ المستخدمة بشكل شائعفي نقل النمط من ثلاثة أجزاء: (1) خطا المحتوى يجعل الصورة المركبة وصورة المحتوى قريبة من ميزات المحتوى؛ (2) خطأ النمط يجعل الصورة المركبة وصورة النمط قريبة من ميزات النمط؛ و (3) يساعد خطأ التباين الكلي على تقليل الضوضاءفي الصورة المركبة.
- يمكننا استخدام شبكة CNN المدربة مسبقاً لاستخراج ميزات الصورة وتقليل دالة الخطأ لتحديث الصورة المركبة باستمرار كمعلمات نموذجية أثناء التدريب.
- نستخدم مصفوفات غرام Gram matrices لتمثيل مخرجات النمط style outputs من طبقات النمط style layers.

14.12.9. التمارين

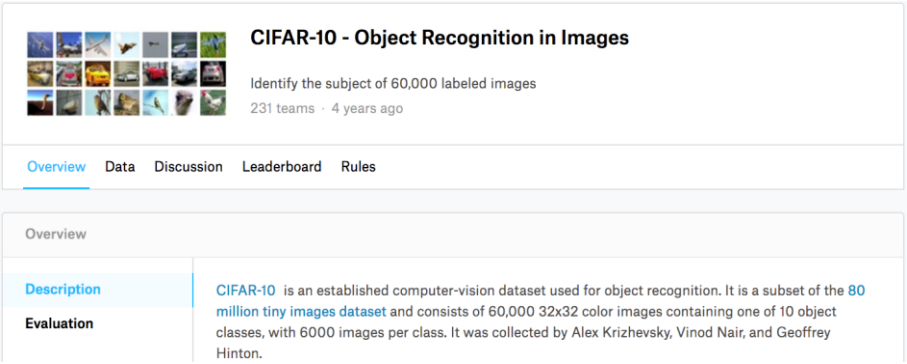
1. كيف يتغير الناتج عند تحديد محتوى وطبقات نمط مختلفة؟
2. اضبط معلمات الوزن الزائدةفي دالة الخسارة. هل يحتفظ الإخراج بمحتوى أكبر أم به ضوضاء أقل؟
3. استخدام صور محتوى وأنماط مختلفة. هل يمكنك إنشاء صور مركبة أكثر إثارة للاهتمام؟
4. هل يمكننا تطبيق نقل النمط style transfer للنص؟ تلميح: يمكنك الرجوع إلى مقالة الاستطلاع التي أعدها Hu et al.، (Hu et al., 2020).

14.13. تصنيف الصور (CIFAR-10) على Image Kaggle Classification (CIFAR-10) on Kaggle

حتى الآن، كنا نستخدم واجهات برمجة التطبيقات API عالية المستوى لأطر التعلم العميق للحصول مباشرة على مجموعات بيانات الصور بتنسيق الموتر. ومع ذلك، غالبًا ما تأتي مجموعات بيانات الصور المخصصةفي شكل ملفات صور.في هذا القسم، سنبدأ من ملفات الصور الأولية، وننظمها ونقرأها ثم نحولها إلى تنسيق موتر خطوة بخطوة.

لقد جربنا مجموعة بيانات CIFAR-10في القسم 14.1، وهي مجموعة بيانات مهمةفي الرؤية الحاسوبية.في هذا القسم، سنطبق المعرفة التي تعلمناهافي الأقسام السابقة لممارسة مسابقة Kaggle لتصنيف الصور CIFAR-10. عنوان الويب للمسابقة هو https://www.kaggle.com/c/cifar-10

يوضح الشكل 14.13.1 المعلومات الموجودة على صفحة الويب الخاصة بالمسابقة. لتقديم النتائج، تحتاج إلى تسجيل حساب Kaggle.



الشكل. 14.13.1 معلومات صفحة ويب مسابقة تصنيف الصور CIFAR-10. يمكن الحصول على مجموعة بيانات المسابقة بالنقر فوق علامة التبويب "البيانات Data".

```
import collections
import math
import os
import shutil
import pandas as pd
from mxnet import gluon, init, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

14.13.1. الحصول على مجموعة البيانات وتنظيمها Obtaining and Organizing the Dataset

تنقسم مجموعة بيانات المسابقة إلى مجموعة تدريب ومجموعة اختبار تحتوي على 50000 و 300000 صورة على التوالي.في مجموعة الاختبار، سيتم استخدام 10000 صورة للتقييم، بينما لن يتم تقييم الصور المتبقية البالغ عددها 290000: يتم تضمينها فقط لجعل الغش صعبًا مع النتائج المصنفة يدويًا manually لمجموعة الاختبار. الصور الموجودةفي مجموعة البيانات هذه كلها ملفات صور بألوان png (قنوات RGB)، يبلغ ارتفاعها وعرضها 32 بكسل. تغطي الصور ما مجموعه 10 فئات هي الطائرات والسيارات والطيور والقطط والغزلان والكلاب والضفادع والخيول والقوارب والشاحنات. يُظهر الزاوية العلوية اليسرى من الشكل 14.13.1 بعض صور الطائرات والسيارات والطيورفي مجموعة البيانات.

14.13.1.1. تنزيل مجموعة البيانات Downloading the Dataset

بعد تسجيل الدخول إلى Kaggle، يمكننا النقر فوق علامة التبويب "البيانات Data"في صفحة ويب مسابقة تصنيف الصور CIFAR-10 الموضحةفي الشكل 14.13.1 وتنزيل مجموعة

البيانات بالنقر فوق الزر "تنزيل الكل Download All". بعد فك ضغط الملف الذي تم تنزيله في `../data` ، وفك ضغط `train.7z` و `test.7z` بداخله ، ستجد مجموعة البيانات بأكملهافي المسارات التالية:

- `../data/cifar-10/train/[1-50000].png`
- `../data/cifar-10/test/[1-300000].png`
- `../data/cifar-10/trainLabels.csv`
- `../data/cifar-10/sampleSubmission.csv`

حيث تحتوي أدلة التدريب والاختبار على صور التدريب والاختبار، على التوالي، يوفر `trainLabels.csv` تسميات لصور التدريب، و `sample_submission.csv` هو نموذج لملف تقديم submission file.

لتسهيل البدء، نقدم عينة صغيرة من مجموعة البيانات التي تحتوي على أول 1000 صورة تدريب و5 صور اختبار عشوائي. لاستخدام مجموعة البيانات الكاملة لمسابقة Kaggle، يلزمك تعيين المتغير التالي `demo` على `False`.

```
#@save
d2l.DATA_HUB['cifar10_tiny'] = (d2l.DATA_URL +
'kaggle_cifar10_tiny.zip',

'2068874e4b9a9f0fb07ebe0ad2b29754449ccacd')

# If you use the full dataset downloaded for the Kaggle
competition, set
# `demo` to False
demo = True

if demo:
    data_dir = d2l.download_extract('cifar10_tiny')
else:
    data_dir = '../data/cifar-10/'
```

```
Downloading ../data/kaggle_cifar10_tiny.zip from
http://d2l-data.s3-
accelerate.amazonaws.com/kaggle_cifar10_tiny.zip...
```

14.13.1.2. تنظيم مجموعة البيانات Organizing the Dataset

نحتاج إلى تنظيم مجموعات البيانات لتسهيل تدريب النموذج والاختبار. دعنا نقرأ أولاً التسميات من ملف csv. تقوم الدالة التالية بإرجاع قاموس يقوم بتعيين الجزء غير الملحق من اسم الملف إلى تسميته.

```
#@save
def read_csv_labels(fname):
    """Read `fname` to return a filename to label
dictionary."""
    with open(fname, 'r') as f:
        # Skip the file header line (column name)
        lines = f.readlines()[1:]
    tokens = [l.rstrip().split(',') for l in lines]
    return dict(((name, label) for name, label in
tokens))

labels = read_csv_labels(os.path.join(data_dir,
'trainLabels.csv'))
print('# training examples:', len(labels))
print('# classes:', len(set(labels.values())))
# training examples: 1000
# classes: 10
```

بعد ذلك، نحدد دالة `reorg_train_valid` لتقسيم مجموعة التحقق من الصحة من مجموعة التدريب الأصلية. الوسيطة `valid_ratio` في هذه الدالة هي نسبة عدد الأمثلةفي مجموعة التحقق من الصحة إلى عدد الأمثلةفي مجموعة التدريب الأصلية. بشكل أكثر تحديدًا، ليكن n هو عدد صور الفئة مع أقل الأمثلة، وr تكون النسبة. ستقوم مجموعة التحقق من الصحة بتقسيم الصور $max(\lfloor nr \rfloor,1)$ لكل فئة. لنستخدم `valid_ratio=0.1` كمثال. نظرًا لأن مجموعة التدريب الأصلية تحتوي على 50000 صورة، فسيكون هناك 45000 صورة مستخدمة للتدريبفي مسار `train_valid_test/train` ، بينما سيتم تقسيم 5000 صورة أخرى على أنها مجموعة تحقق من الصحة في المسار `train_valid_test/valid`. بعد تنظيم مجموعة البيانات، سيتم وضع صور الفئة نفسهافي نفس المجلد.

```
#@save
def copyfile(filename, target_dir):
    """Copy a file into a target directory."""
    os.makedirs(target_dir, exist_ok=True)
    shutil.copy(filename, target_dir)
```

```
#@save
def reorg_train_valid(data_dir, labels, valid_ratio):
    """Split the validation set out of the original
training set."""
    # The number of examples of the class that has the
fewest examples in the
    # training dataset
    n =
collections.Counter(labels.values()).most_common()[-
1][1]
    # The number of examples per class for the
validation set
    n_valid_per_label = max(1, math.floor(n *
valid_ratio))
    label_count = {}
    for train_file in os.listdir(os.path.join(data_dir,
'train')):
        label = labels[train_file.split('.')[0]]
        fname = os.path.join(data_dir, 'train',
train_file)
        copyfile(fname, os.path.join(data_dir,
'train_valid_test',
                                     'train_valid',
label))
        if label not in label_count or
label_count[label] < n_valid_per_label:
            copyfile(fname, os.path.join(data_dir,
'train_valid_test',
                                         'valid',
label))
            label_count[label] = label_count.get(label,
0) + 1
        else:
            copyfile(fname, os.path.join(data_dir,
'train_valid_test',
                                         'train',
label))
    return n_valid_per_label
```

تنظم دالة `reorg_test` أدناه مجموعة الاختبار لتحميل البيانات أثناء التنبؤ.

```
#@save
def reorg_test(data_dir):
```

```
    """Organize the testing set for data loading during
prediction."""
    for test_file in os.listdir(os.path.join(data_dir,
'test')):
        copyfile(os.path.join(data_dir, 'test',
test_file),
                 os.path.join(data_dir,
'train_valid_test', 'test',
                              'unknown'))
```

أخيرًا، نستخدم دالة لاستدعاء دوال `read_csv_labels` و `reorg_train_valid` و `reorg_test` المحددة أعلاه.

```
def reorg_cifar10_data(data_dir, valid_ratio):
    labels = read_csv_labels(os.path.join(data_dir,
'trainLabels.csv'))
    reorg_train_valid(data_dir, labels, valid_ratio)
    reorg_test(data_dir)
```

هنا قمنا فقط بتعيين حجم الدُفعة على 32 للعينة الصغيرة من مجموعة البيانات. عند التدريب واختبار مجموعة البيانات الكاملة لمسابقة Kaggle، يجب تعيين حجم الدفعة على عدد صحيح أكبر، مثل 128. قمنا بتقسيم 10٪ من أمثلة التدريب كمجموعة التحقق من صحة لضبط المعلمات الفائقة.

```
batch_size = 32 if demo else 128
valid_ratio = 0.1
reorg_cifar10_data(data_dir, valid_ratio)
```

14.13.2. زيادة الصورة Image Augmentation

نحن نستخدم زيادة الصورة image augmentation لمعالجة فرط التعلم overfitting. على سبيل المثال، يمكن قلب الصور أفقيًا بشكل عشوائي أثناء التدريب. يمكننا أيضًا إجراء التوحيد القياسي standardization لقنوات RGB الثلاث للصور الملونة. يسرد أدناه بعض هذه العمليات التي يمكنك تعديلها.

```
transform_train = gluon.data.vision.transforms.Compose([
    # Scale the image up to a square of 40 pixels in
both height and width
    gluon.data.vision.transforms.Resize(40),
    # Randomly crop a square image of 40 pixels in both
height and width to
    # produce a small square of 0.64 to 1 times the area
of the original
```

```
    # image, and then scale it to a square of 32 pixels
in both height and
    # width
    gluon.data.vision.transforms.RandomResizedCrop(32,
scale=(0.64, 1.0),

ratio=(1.0, 1.0)),
    gluon.data.vision.transforms.RandomFlipLeftRight(),
    gluon.data.vision.transforms.ToTensor(),
    # Standardize each channel of the image
    gluon.data.vision.transforms.Normalize([0.4914,
0.4822, 0.4465],
                                           [0.2023,
0.1994, 0.2010])])
```

أثناء الاختبار، نقوم فقط بالتوحيد القياسي standardization على الصور لإزالة العشوائيةفي نتائج التقييم.

```
transform_test = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.ToTensor(),
    gluon.data.vision.transforms.Normalize([0.4914,
0.4822, 0.4465],
                                           [0.2023,
0.1994, 0.2010])])
```

14.13.3. قراءة مجموعة البيانات Reading the Dataset

بعد ذلك، نقرأ مجموعة البيانات المنظمة التي تتكون من ملفات صور خام. يتضمن كل مثال صورة وتسمية.

```
train_ds, valid_ds, train_valid_ds, test_ds = [
    gluon.data.vision.ImageFolderDataset(
        os.path.join(data_dir, 'train_valid_test',
folder))
    for folder in ['train', 'valid', 'train_valid',
'test']]
```

أثناء التدريب، نحتاج إلى تحديد جميع عمليات زيادة الصورة المحددة أعلاه. عند استخدام مجموعة التحقق من الصحة لتقييم النموذج أثناء ضبط المعلمة الفائقة، يجب عدم إدخال عشوائية من زيادة الصورة. قبل التنبؤ النهائي، نقوم بتدريب النموذج على مجموعة التدريب المدمجة ومجموعة التحقق من الصحة للاستفادة الكاملة من جميع البيانات المصنفة.

```
train_iter, train_valid_iter = [gluon.data.DataLoader(
```

```
    dataset.transform_first(transform_train),
batch_size, shuffle=True,
    last_batch='discard') for dataset in (train_ds,
train_valid_ds)]

valid_iter = gluon.data.DataLoader(
    valid_ds.transform_first(transform_test),
batch_size, shuffle=False,
    last_batch='discard')

test_iter = gluon.data.DataLoader(
    test_ds.transform_first(transform_test), batch_size,
shuffle=False,
    last_batch='keep')
```

14.13.4. تعريف النموذج Defining the Model

هنا، نبني الكتل المتبقية بناءً على فئة `HybridBlock`، والتي تختلف قليلاً عن التنفيذ الموضح في القسم 8.6. هذا لتحسين الكفاءة الحسابية.

```
class Residual(nn.HybridBlock):
    def __init__(self, num_channels, use_1x1conv=False,
strides=1, **kwargs):
        super(Residual, self).__init__(**kwargs)
        self.conv1 = nn.Conv2D(num_channels,
kernel_size=3, padding=1,
                               strides=strides)
        self.conv2 = nn.Conv2D(num_channels,
kernel_size=3, padding=1)
        if use_1x1conv:
            self.conv3 = nn.Conv2D(num_channels,
kernel_size=1,
                                   strides=strides)
        else:
            self.conv3 = None
        self.bn1 = nn.BatchNorm()
        self.bn2 = nn.BatchNorm()

    def hybrid_forward(self, F, X):
        Y = F.npx.relu(self.bn1(self.conv1(X)))
        Y = self.bn2(self.conv2(Y))
        if self.conv3:
            X = self.conv3(X)
```

```
        return F.npx.relu(Y + X)
```

بعد ذلك، نحدد نموذج ResNet-18.

```
def resnet18(num_classes):
    net = nn.HybridSequential()
    net.add(nn.Conv2D(64, kernel_size=3, strides=1,
padding=1),
            nn.BatchNorm(), nn.Activation('relu'))

    def resnet_block(num_channels, num_residuals,
first_block=False):
        blk = nn.HybridSequential()
        for i in range(num_residuals):
            if i == 0 and not first_block:
                blk.add(Residual(num_channels,
use_1x1conv=True, strides=2))
            else:
                blk.add(Residual(num_channels))
        return blk

    net.add(resnet_block(64, 2, first_block=True),
            resnet_block(128, 2),
            resnet_block(256, 2),
            resnet_block(512, 2))
    net.add(nn.GlobalAvgPool2D(), nn.Dense(num_classes))
    return net
```

نستخدم تهيئة Xavier الموضحةفي القسم 5.4.2.2 قبل بدء التدريب.

```
def get_net(devices):
    num_classes = 10
    net = resnet18(num_classes)
    net.initialize(ctx=devices, init=init.Xavier())
    return net

loss = gluon.loss.SoftmaxCrossEntropyLoss()
```

14.13.5. تحديد دالة التدريب Defining the Training Function

سنختار النماذج ونقوم بضبط المعلمات الفائقة وفقًا لأداء النموذجفي مجموعة التحقق من الصحة. فيما يلي، نحدد نموذج تدريب دالة التدريب `train`.

```
def train(net, train_iter, valid_iter, num_epochs, lr,
wd, devices, lr_period,
```

```
          lr_decay):
    trainer = gluon.Trainer(net.collect_params(), 'sgd',
                            {'learning_rate': lr,
'momentum': 0.9, 'wd': wd})
    num_batches, timer = len(train_iter), d2l.Timer()
    legend = ['train loss', 'train acc']
    if valid_iter is not None:
        legend.append('valid acc')
    animator = d2l.Animator(xlabel='epoch', xlim=[1,
num_epochs],
                            legend=legend)
    for epoch in range(num_epochs):
        metric = d2l.Accumulator(3)
        if epoch > 0 and epoch % lr_period == 0:

trainer.set_learning_rate(trainer.learning_rate *
lr_decay)
        for i, (features, labels) in
enumerate(train_iter):
            timer.start()
            l, acc = d2l.train_batch_ch13(
                net, features, labels.astype('float32'),
loss, trainer,
                devices, d2l.split_batch)
            metric.add(l, acc, labels.shape[0])
            timer.stop()
            if (i + 1) % (num_batches // 5) == 0 or i ==
num_batches - 1:
                animator.add(epoch + (i + 1) /
num_batches,
                             (metric[0] / metric[2],
metric[1] / metric[2],
                              None))
        if valid_iter is not None:
            valid_acc = d2l.evaluate_accuracy_gpus(net,
valid_iter,

d2l.split_batch)
            animator.add(epoch + 1, (None, None,
valid_acc))
    measures = (f'train loss {metric[0] /
metric[2]:.3f}, '
```

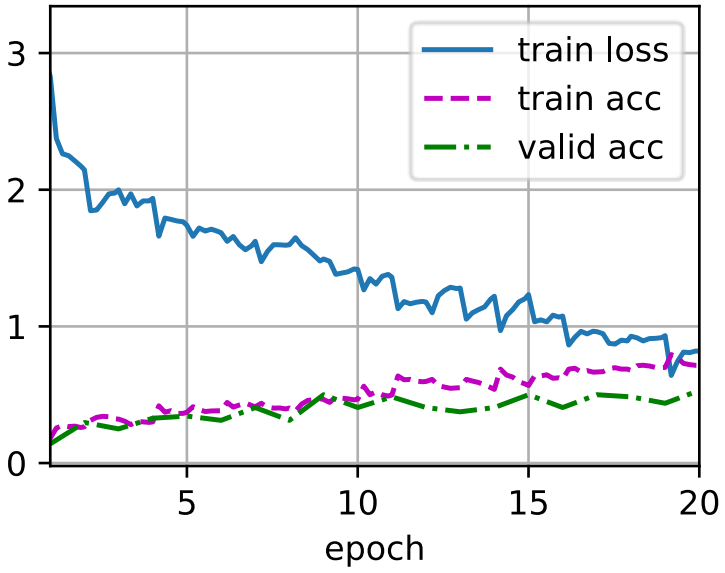
```
                f'train acc {metric[1] /
metric[2]:.3f}')
    if valid_iter is not None:
        measures += f', valid acc {valid_acc:.3f}'
    print(measures + f'\n{metric[2] * num_epochs /
timer.sum():.1f}'
          f' examples/sec on {str(devices)}')
```

14.13.6. التدريب والتحقق من صحة النموذج Training and Validating the Model

الآن، يمكننا تدريب النموذج والتحقق منه. يمكن ضبط جميع المعلمات الفائقة التالية. على سبيل المثال، يمكننا زيادة عدد الفترات. عندما يتم تعيين `lr_period` و `lr_decay` على 4 و 0.9 ، على التوالي ، سيتم ضرب معدل التعلم لخوارزمية التحسين بمقدار 0.9 بعد كل 4 فترات. فقط لسهولة العرض، نقوم هنا بتدريب 20 فترة فقط.

```
devices, num_epochs, lr, wd = d2l.try_all_gpus(), 20,
0.02, 5e-4
lr_period, lr_decay, net = 4, 0.9, get_net(devices)
net.hybridize()
train(net, train_iter, valid_iter, num_epochs, lr, wd,
devices, lr_period,
      lr_decay)
```

```
train loss 0.819, train acc 0.714, valid acc 0.531
1054.9 examples/sec on [gpu(0), gpu(1)]
```

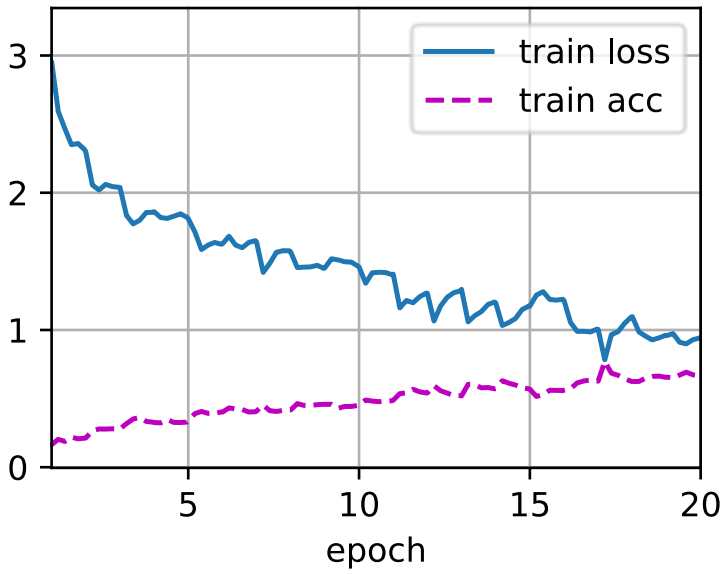


14.13.7. تصنيف مجموعة الاختبار وتقديم النتائج على Classifying Kaggle the Testing Set and Submitting Results on Kaggle

بعد الحصول على نموذج واعد به معلمات فائقة، نستخدم جميع البيانات المصنفة (بمافي ذلك مجموعة التحقق من الصحة) لإعادة تدريب النموذج وتصنيف مجموعة الاختبار.

```
net, preds = get_net(devices), []
net.hybridize()
train(net, train_valid_iter, None, num_epochs, lr, wd,
devices, lr_period,
      lr_decay)

for X, _ in test_iter:
    y_hat = net(X.as_in_ctx(devices[0]))

preds.extend(y_hat.argmax(axis=1).astype(int).asnumpy())
sorted_ids = list(range(1, len(test_ds) + 1))
sorted_ids.sort(key=lambda x: str(x))
df = pd.DataFrame({'id': sorted_ids, 'label': preds})
df['label'] = df['label'].apply(lambda x:
train_valid_ds.synsets[x])
df.to_csv('submission.csv', index=False)
```

```
train loss 0.944, train acc 0.667
1006.7 examples/sec on [gpu(0), gpu(1)]
```



سيقوم الكود أعلاه بإنشاء ملف `submission.csv`، الذي يتوافق تنسيقه مع متطلبات مسابقة Kaggle. طريقة تقديم النتائج إلى Kaggle مماثلة لتلك الواردةفي القسم 5.7.

14.13.8. الملخص

- يمكننا قراءة مجموعات البيانات التي تحتوي على ملفات صور خام بعد تنظيمها بالتنسيق المطلوب.

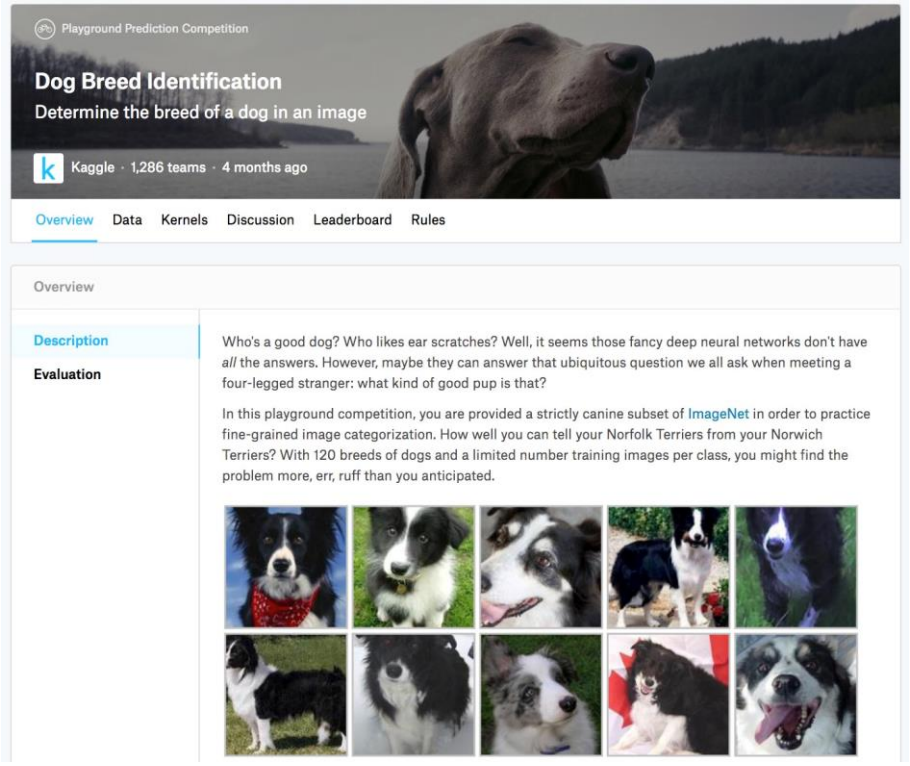
14.13.9. التمارين

1. استخدم مجموعة بيانات CIFAR-10 الكاملة لمسابقة Kaggle هذه. قم بتعيين المعلمات الفائقة على أنها `batch_size = 128`، `num_epochs = 100`، `lr = 0.1`، `lr_period = 50`، و `lr_decay = 0.1`. تعرف على الدقة والترتيب الذي يمكنك تحقيقهفي هذه المسابقة. هل يمكنك تحسينها بشكل أكبر؟
2. ما الدقة التي يمكنك الحصول عليها عند عدم استخدام زيادة الصورة؟

14.14. تحديد سلالة الكلاب (ImageNet Dogs) على Kaggle
Dog Breed Identification (ImageNet Dogs) on Kaggle

في هذا القسم، سنتدرب على مشكلة تحديد سلالة الكلاب على Kaggle. عنوان الويب لهذه المسابقة هو https://www.kaggle.com/c/dog-breed-identification

في هذه المسابقة، سيتم التعرف على 120 سلالة مختلفة من الكلاب.في الواقع، مجموعة البيانات الخاصة بهذه المسابقة هي مجموعة فرعية من مجموعة بيانات ImageNet. على عكس الصور الموجودةفي مجموعة بيانات CIFAR-10في القسم 14.13، تكون الصورفي مجموعة بيانات ImageNet أعلى وأوسع بأبعاد مختلفة. يوضح الشكل 14.14.1 المعلومات الموجودة على صفحة الويب الخاصة بالمسابقة. تحتاج إلى حساب Kaggle لإرسال نتائجك.



الشكل 14.14.1 موقع مسابقة تحديد سلالات الكلاب. يمكن الحصول على مجموعة بيانات المسابقة بالنقر فوق علامة التبويب "البيانات".

```
import os
from mxnet import autograd, gluon, init, npx
from mxnet.gluon import nn
```

```
from d2l import mxnet as d2l

npx.set_np()
```

14.14.1. الحصول على مجموعة البيانات وتنظيمها Obtaining and Organizing the Dataset

تنقسم مجموعة بيانات المنافسة إلى مجموعة تدريب ومجموعة اختبار، والتي تحتوي على صور 10222 و10357 صور JPEG لثلاث قنوات RGB (ملونة)، على التوالي. من بين مجموعة بيانات التدريب، هناك 120 سلالة من الكلاب مثل Labradors وPoodles وDachshunds وSamoyeds وHuskies وChihuahuas وYorkshire Terriers.

14.14.1.1. تنزيل مجموعة البيانات Downloading the Dataset

بعد تسجيل الدخول إلى Kaggle، يمكنك النقر فوق علامة التبويب "البيانات Data "في صفحة ويب المسابقة الموضحةفي الشكل 14.14.1 وتنزيل مجموعة البيانات بالنقر فوق الزر "تنزيل الكل Download All ". بعد فك ضغط الملف الذي تم تنزيلهفي `../data` ، ستجد مجموعة البيانات بأكملهافي المسارات التالية:

- ../data/dog-breed-identification/labels.csv
- ../data/dog-breed-identification/sample_submission.csv
- ../data/dog-breed-identification/train
- ../data/dog-breed-identification/test

ربما لاحظت أن الهيكل أعلاه مشابه لهيكل مسابقة CIFAR-10في القسم 14.13، حيث تتدرب المجلدات `train/` و`test/` تحتوي على صور تدريب واختبار الكلاب، على التوالي، ويحتوي `labels.csv` على تسميات لصور التدريب. وبالمثل، لتسهيل البدء، نقدم عينة صغيرة من مجموعة البيانات المذكورة أعلاه: `train_valid_test_tiny.zip`. إذا كنت ستستخدم مجموعة البيانات الكاملة لمسابقة Kaggle، فأنت بحاجة إلى تغيير المتغير أدناه `demo` إلى `False`.

```
#@save
d2l.DATA_HUB['dog_tiny'] = (d2l.DATA_URL +
'kaggle_dog_tiny.zip',

'0cb91d09b814ecdc07b50f31f8dcad3e81d6a86d')

# If you use the full dataset downloaded for the Kaggle
competition, change
# the variable below to `False`
```

```
demo = True
if demo:
    data_dir = d2l.download_extract('dog_tiny')
else:
    data_dir = os.path.join('..', 'data', 'dog-breed-
identification')
```

```
Downloading ../data/kaggle_dog_tiny.zip from http://d2l-
data.s3-accelerate.amazonaws.com/kaggle_dog_tiny.zip...
```

14.14.1.2. تنظيم مجموعة البيانات Organizing the Dataset

يمكننا تنظيم مجموعة البيانات بشكل مشابه لما فعلناهفي القسم 14.13، أي تقسيم مجموعة التحقق من الصحة من مجموعة التدريب الأصلية، ونقل الصور إلى مجلدات فرعية مجمعة حسب التسميات.

تقرأ دالة `reorg_dog_data` أدناه تسميات بيانات التدريب ، وتقسم مجموعة التحقق من الصحة، وتنظم مجموعة التدريب.

```
def reorg_dog_data(data_dir, valid_ratio):
    labels = d2l.read_csv_labels(os.path.join(data_dir,
'labels.csv'))
    d2l.reorg_train_valid(data_dir, labels, valid_ratio)
    d2l.reorg_test(data_dir)


batch_size = 32 if demo else 128
valid_ratio = 0.1
reorg_dog_data(data_dir, valid_ratio)
```

14.14.2. زيادة الصورة Image Augmentation

تذكر أن مجموعة بيانات سلالة الكلاب dog breed هذه هي مجموعة فرعية من مجموعة بيانات ImageNet، والتي تكون صورها أكبر من تلك الموجودةفي مجموعة بيانات CIFAR-10في القسم 14.13. يسرد ما يلي بعض عمليات زيادة الصورة image augmentation التي قد تكون مفيدة للصور الأكبر نسبيًا.

```
transform_train = gluon.data.vision.transforms.Compose([
    # Randomly crop the image to obtain an image with an
area of 0.08 to 1 of
    # the original area and height-to-width ratio
between 3/4 and 4/3. Then,
    # scale the image to create a new 224 x 224 image
```

```
    gluon.data.vision.transforms.RandomResizedCrop(224,
scale=(0.08, 1.0),

ratio=(3.0/4.0, 4.0/3.0)),
    gluon.data.vision.transforms.RandomFlipLeftRight(),
    # Randomly change the brightness, contrast, and
saturation

gluon.data.vision.transforms.RandomColorJitter(brightnes
s=0.4,

contrast=0.4,

saturation=0.4),
    # Add random noise
    gluon.data.vision.transforms.RandomLighting(0.1),
    gluon.data.vision.transforms.ToTensor(),
    # Standardize each channel of the image
    gluon.data.vision.transforms.Normalize([0.485,
0.456, 0.406],
                                           [0.229,
0.224, 0.225])])
```

أثناء التنبؤ، نستخدم فقط عمليات المعالجة المسبقة للصور بدون عشوائية.

```
transform_test = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.Resize(256),
    # Crop a 224 x 224 square area from the center of
the image
    gluon.data.vision.transforms.CenterCrop(224),
    gluon.data.vision.transforms.ToTensor(),
    gluon.data.vision.transforms.Normalize([0.485,
0.456, 0.406],
                                           [0.229,
0.224, 0.225])])
```

14.14.3. قراءة مجموعة البيانات Reading the Dataset

كمافي القسم 14.13، يمكننا قراءة مجموعة البيانات المنظمة التي تتكون من ملفات صور خام.

```
train_ds, valid_ds, train_valid_ds, test_ds = [
    gluon.data.vision.ImageFolderDataset(
        os.path.join(data_dir, 'train_valid_test',
folder))
```

```
    for folder in ('train', 'valid', 'train_valid',
'test')]
```

أدناه نقوم بإنشاء مثيلات مكرر البيانات data iterator بنفس الطريقة كمافي القسم 14.13.

```
train_iter, train_valid_iter = [gluon.data.DataLoader(
    dataset.transform_first(transform_train),
batch_size, shuffle=True,
    last_batch='discard') for dataset in (train_ds,
train_valid_ds)]

valid_iter = gluon.data.DataLoader(
    valid_ds.transform_first(transform_test),
batch_size, shuffle=False,
    last_batch='discard')

test_iter = gluon.data.DataLoader(
    test_ds.transform_first(transform_test), batch_size,
shuffle=False,
    last_batch='keep')
```

14.14.4. الضبط الدقيق لنموذج مسبق التدريب Fine-Tuning a Pretrained Model

مرة أخرى، مجموعة البيانات الخاصة بهذه المسابقة هي مجموعة فرعية من مجموعة بيانات ImageNet. لذلك، يمكننا استخدام النهج الذي تمت مناقشتهفي القسم 14.2 لتحديد نموذج تم اختباره مسبقًا على مجموعة بيانات ImageNet الكاملة واستخدامه لاستخراج ميزات الصورة ليتم إدخالهافي شبكة إخراج مخصصة صغيرة الحجم. توفر واجهات برمجة التطبيقات API عالية المستوى لأطر التعلم العميق مجموعة واسعة من النماذج التي تم اختبارها مسبقًا على مجموعة بيانات ImageNet. هنا، نختار نموذج ResNet-34 الذي تم اختباره مسبقًا، حيث نقوم ببساطة بإعادة استخدام مدخلات طبقة إخراج هذا النموذج (أي الميزات المستخرجة). ثم يمكننا استبدال طبقة المخرجات الأصلية بشبكة إخراج مخصصة صغيرة يمكن تدريبها، مثل تكديس طبقتين متصلتين بالكامل. يختلف عن التجربةفي القسم 14.2، ما يلي لا يعيد تدريب النموذج قبل التدريب المستخدم لاستخراج الميزة. هذا يقلل من وقت التدريب والذاكرة لتخزين الانحدارات gradients.

تذكر أننا قمنا بتوحيد standardized الصور باستخدام المتوسطات والانحرافات المعيارية لقنوات RGB الثلاث لمجموعة بيانات ImageNet الكاملة.في الواقع، يتوافق هذا أيضًا مع عملية التوحيد القياسية standardization بواسطة النموذج الذي تم اختباره مسبقًا على ImageNet.

```
def get_net(devices):
    finetune_net =
gluon.model_zoo.vision.resnet34_v2(pretrained=True)
    # Define a new output network
    finetune_net.output_new =
nn.HybridSequential(prefix='')
    finetune_net.output_new.add(nn.Dense(256,
activation='relu'))
    # There are 120 output categories
    finetune_net.output_new.add(nn.Dense(120))
    # Initialize the output network
    finetune_net.output_new.initialize(init.Xavier(),
ctx=devices)
    # Distribute the model parameters to the CPUs or
GPUs used for computation
    finetune_net.collect_params().reset_ctx(devices)
    return finetune_net
```

قبل حساب الخطأ، نحصل أولاً على مدخلات طبقة إخراج النموذج الذي تم اختباره مسبقًا، أي الميزة المستخرجة. ثم نستخدم هذه الميزة كمدخلات لشبكة الإخراج المخصصة الصغيرة الخاصة بنا لحساب الخطأ.

```
loss = gluon.loss.SoftmaxCrossEntropyLoss()

def evaluate_loss(data_iter, net, devices):
    l_sum, n = 0.0, 0
    for features, labels in data_iter:
        X_shards, y_shards = d2l.split_batch(features,
labels, devices)
        output_features = [net.features(X_shard) for
X_shard in X_shards]
        outputs = [net.output_new(feature) for feature
in output_features]
        ls = [loss(output, y_shard).sum() for output,
y_shard
              in zip(outputs, y_shards)]
        l_sum += sum([float(l.sum()) for l in ls])
        n += labels.size
    return l_sum / n
```

14.14.5. تعريف دالة التدريب Defining the Training Function

سنختار النموذج ونقوم بضبط المعلمات الفائقة وفقًا لأداء النموذجفي مجموعة التحقق من الصحة. تقوم دالة التدريب النموذج `train` بتدريب معلمات شبكة الإخراج المخصصة الصغيرة فقط.

```
def train(net, train_iter, valid_iter, num_epochs, lr,
wd, devices, lr_period,
          lr_decay):
    # Only train the small custom output network
    trainer =
gluon.Trainer(net.output_new.collect_params(), 'sgd',
                              {'learning_rate': lr,
'momentum': 0.9, 'wd': wd})
    num_batches, timer = len(train_iter), d2l.Timer()
    legend = ['train loss']
    if valid_iter is not None:
        legend.append('valid loss')
    animator = d2l.Animator(xlabel='epoch', xlim=[1,
num_epochs],
                            legend=legend)
    for epoch in range(num_epochs):
        metric = d2l.Accumulator(2)
        if epoch > 0 and epoch % lr_period == 0:

trainer.set_learning_rate(trainer.learning_rate *
lr_decay)
        for i, (features, labels) in
enumerate(train_iter):
            timer.start()
            X_shards, y_shards =
d2l.split_batch(features, labels, devices)
            output_features = [net.features(X_shard) for
X_shard in X_shards]
            with autograd.record():
                outputs = [net.output_new(feature)
                           for feature in
output_features]
                ls = [loss(output, y_shard).sum() for
output, y_shard
                      in zip(outputs, y_shards)]
            for l in ls:
```

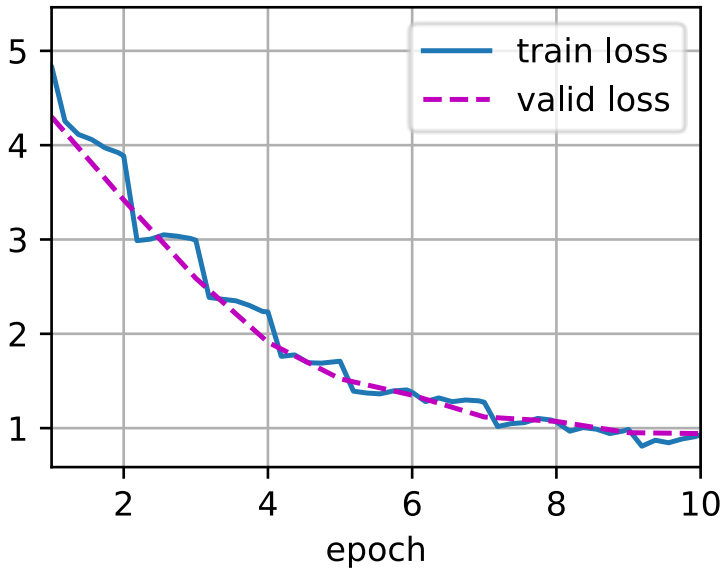
```
                l.backward()
            trainer.step(batch_size)
            metric.add(sum([float(l.sum()) for l in
ls]), labels.shape[0])
            timer.stop()
            if (i + 1) % (num_batches // 5) == 0 or i ==
num_batches - 1:
                animator.add(epoch + (i + 1) /
num_batches,
                             (metric[0] / metric[1],
None))
        if valid_iter is not None:
            valid_loss = evaluate_loss(valid_iter, net,
devices)
            animator.add(epoch + 1, (None, valid_loss))
    measures = f'train loss {metric[0] / metric[1]:.3f}'
    if valid_iter is not None:
        measures += f', valid loss {valid_loss:.3f}'
    print(measures + f'\n{metric[1] * num_epochs /
timer.sum():.1f}'
          f' examples/sec on {str(devices)}')
```

14.14.6. التدريب والتحقق من صحة النموذج Training and Validating the Model

الآن يمكننا تدريب النموذج والتحقق منه. جميع المعلمات الفائقة التالية قابلة للضبط tunable. على سبيل المثال، يمكن زيادة عدد الفترات. نظرًا لأنه تم ضبط `lr_period` و `lr_decay` على 2 و 0.9 ، على التوالي ، سيتم ضرب معدل التعلم لخوارزمية التحسين بمقدار 0.9 بعد كل فترتين.

```
devices, num_epochs, lr, wd = d2l.try_all_gpus(), 10,
5e-3, 1e-4
lr_period, lr_decay, net = 2, 0.9, get_net(devices)
net.hybridize()
train(net, train_iter, valid_iter, num_epochs, lr, wd,
devices, lr_period,
      lr_decay)
```

```
train loss 0.928, valid loss 0.942
205.5 examples/sec on [gpu(0), gpu(1)]
```



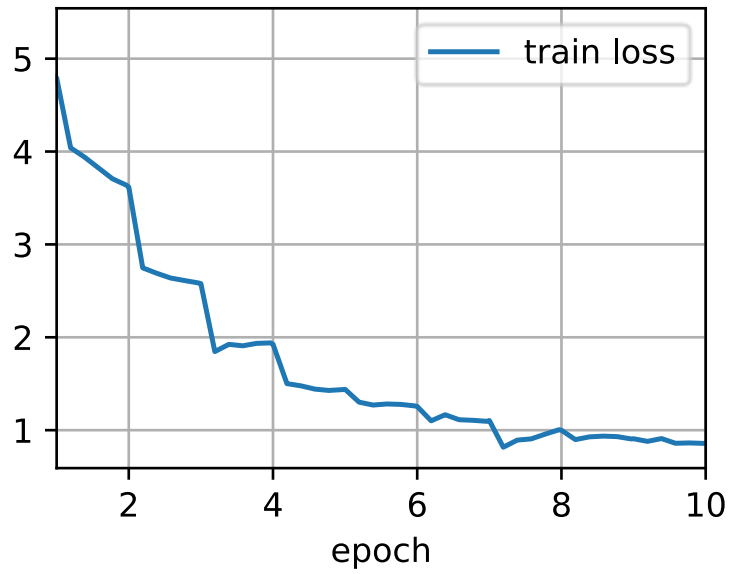
14.14.7. تصنيف مجموعة الاختبار وتقديم النتائج على Classifying Kaggle the Testing Set and Submitting Results on Kaggle

على غرار الخطوة الأخيرةفي القسم 14.13،في النهاية، تُستخدم جميع البيانات المصنفة (بمافي ذلك مجموعة التحقق من الصحة) لتدريب النموذج وتصنيف مجموعة الاختبار. سوف نستخدم شبكة الإخراج المخصصة المدربة من أجل التصنيف.

```
net = get_net(devices)
net.hybridize()
train(net, train_valid_iter, None, num_epochs, lr, wd,
devices, lr_period,
      lr_decay)

preds = []
for data, label in test_iter:
    output_features =
net.features(data.as_in_ctx(devices[0]))
    output =
npx.softmax(net.output_new(output_features))
    preds.extend(output.asnumpy())
ids = sorted(os.listdir(
    os.path.join(data_dir, 'train_valid_test', 'test',
'unknown')))
with open('submission.csv', 'w') as f:
```

```
    f.write('id,' + ','.join(train_valid_ds.synsets) +
'\n')
    for i, output in zip(ids, preds):
        f.write(i.split('.')[0] + ',' + ','.join(
            [str(num) for num in output]) + '\n')
```

```
train loss 0.860
180.4 examples/sec on [gpu(0), gpu(1)]
```



ستُنشئ الشفرة أعلاه ملف `submission.csv` ليتم إرساله إلى Kaggle بنفس الطريقة الموضحةفي القسم 5.7.

14.14.8. الملخص

- الصورفي مجموعة بيانات ImageNet أكبر (بأبعاد مختلفة) من صور CIFAR-10. قد نقوم بتعديل عمليات زيادة الصورة للمهام الموجودة على مجموعة بيانات مختلفة.
- لتصنيف مجموعة فرعية من مجموعة بيانات ImageNet، يمكننا الاستفادة من النماذج المدربة مسبقًا pre-trained models على مجموعة بيانات ImageNet الكاملة لاستخراج الميزات وتدريب شبكة إخراج مخصصة صغيرة الحجم فقط. سيؤدي ذلك إلى تقليل الوقت الحسابي وتكلفة الذاكرة.

14.14.9. التمارين

1. عند استخدام مجموعة بيانات منافسة Kaggle الكاملة، ما هي النتائج التي يمكنك تحقيقها عند زيادة `batch_size` (حجم الدُفعة) و`num_epochs` (عدد الفترات) أثناء تعيين بعض المعلمات الفائقة الأخرى مثل `lr = 0.01`، و`lr_period = 10`، و`lr_decay = 0.1`؟
2. هل تحصل على نتائج أفضل إذا كنت تستخدم نموذجًا أعمق تم اختباره مسبقًا؟ كيف تقوم بضبط المعلمات الفائقة؟ هل يمكنك تحسين النتائج؟

15

المعالجة اللغوية الطبيعية: التدريب المسبق

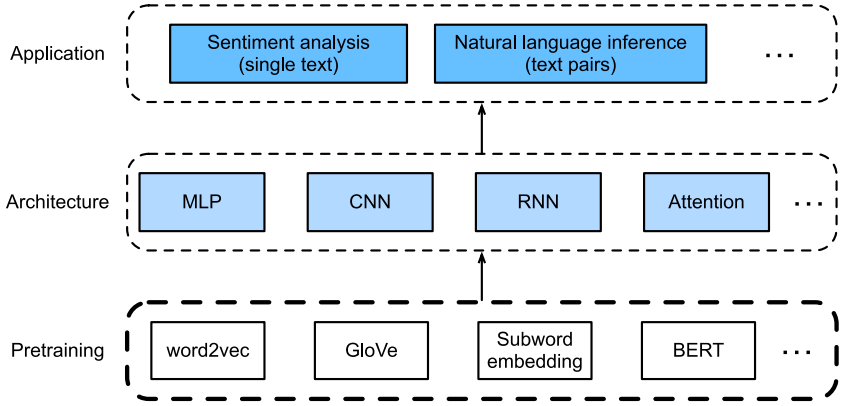
15. المعالجة اللغوية الطبيعية: التدريب المسبقNatural Language Processing: Pretraining

يحتاج البشر إلى التواصل. من هذه الحاجة الأساسية للحالة البشرية، تم إنشاء قدر كبير من النصوص المكتوبة على أساس يومي. نظرًا للنص الغنيفي الوسائط الاجتماعية وتطبيقات الدردشة ورسائل البريد الإلكتروني ومراجعات المنتجات والمقالات الإخبارية والأوراق البحثية والكتب، يصبح من الضروري تمكين أجهزة الكمبيوتر من فهمها لتقديم المساعدة أو اتخاذ القرارات بناءً على اللغات البشرية.

تدرس المعالجة اللغوية الطبيعية Natural language processing التفاعلات بين أجهزة الكمبيوتر والبشر باستخدام اللغات الطبيعية. من الناحية العملية، من الشائع جدًا استخدام تقنيات المعالجة اللغوية الطبيعية لمعالجة وتحليل بيانات النص (لغة الإنسان الطبيعية)، مثل نماذج اللغة language modelsفي القسم 9.3 ونماذج الترجمة الآلية machine translation modelsفي القسم 10.5.

لفهم النص، يمكننا أن نبدأ من خلال تعلم تمثيلاته. من خلال الاستفادة من التسلسلات النصية الحالية من مجموعات كبيرة، تم استخدام التعلم تحت الإشراف الذاتي self-supervised learning على نطاق واسع لإجراء تمثيلات نصية مسبقة، مثل التنبؤ ببعض الأجزاء المخفية من النص باستخدام جزء آخر من النص المحيط بها. بهذه الطريقة، تتعلم النماذج من خلال الإشراف من البيانات النصية الضخمة massive text data دون بذل جهود باهظةفي وضع العلامات (التسميات) expensive labeling efforts!

كما سنرىفي هذا الفصل، عند معالجة كل كلمة أو كلمة فرعية كرمز فردي individual token، يمكن تدريب تمثيل كل رمز مسبقًا باستخدام نماذج word2vec أو GloVe أو تضمين الكلمات الفرعية على مجموعة كبيرة large corpora. بعد التدريب المسبق pretraining، يمكن أن يكون تمثيل كل رمز متجهًا vector، ومع ذلك، فإنه يظل كما هو بغض النظر عن السياق. على سبيل المثال، تمثيل المتجه لكلمة "bank" هو نفسهفي كل من " go to the bank to deposit some money" و" go to the bank to sit down". وبالتالي، فإن العديد من نماذج ما قبل التدريب الحديثة تكيف تمثيل نفس الرمزفي سياقات مختلفة. من بينها BERT، وهو نموذج أعمق يخضع للإشراف الذاتي يعتمد على مشفر المحولات transformer encoder.في هذا الفصل، سوف نركز على كيفية إجراء مثل هذه التمثيلات للنص، كما هو موضحفي الشكل 15.1.



الشكل 15.1 يمكن تغذية تمثيلات النص المحددة مسبقًا إلى العديد من بُنيات التعلم العميق لمختلف تطبيقات المعالجة اللغوية الطبيعية. يركز هذا الفصل على التدريب المسبق على تمثيل النص الأولي.

لرؤية الصورة الكبيرة، يوضح الشكل 15.1 أنه يمكن تغذية تمثيلات النص التي تم اختبارها مسبقًا لمجموعة متنوعة من بُنيات التعلم العميق لتطبيقات المعالجة اللغوية الطبيعية المختلفة. سوف نغطيهافي القسم 16.

15.1. تضمين كلمة Word Embedding (word2vec)

اللغة الطبيعية Natural language هي نظام معقد يستخدم للتعبير عن المعاني meanings. في هذا النظام، الكلمات words هي الوحدة الأساسية للمعنى. كما يوحي الاسم، فإن متجهات الكلمات word vectors هي متجهات تستخدم لتمثيل الكلمات، ويمكن أيضًا اعتبارها متجهات خاصية feature vectors أو تمثيلات للكلمات. تسمى تقنية تعيين الكلمات إلى المتجهات الحقيقية تضمين الكلمات word embedding.في السنوات الأخيرة، أصبح تضمين الكلمات تدريجيًا المعرفة الأساسية للمعالجة اللغوية الطبيعية.

15.1.1. تعتبر متجهات واحد ساخن خيارًا سيئًا One-Hot Vectors Are a Bad Choice

استخدمنا متجهًا واحدًا ساخنًا one-hot vectors لتمثيل الكلمات (الأحرف عبارة عن كلمات)في القسم 9.5. افترض أن عدد الكلمات المختلفةفي القاموس (حجم القاموس) هو N، وأن كل كلمة تتوافق مع عدد صحيح مختلف (فهرس index) من 0 إلى $N-1$. للحصول على تمثيل متجه واحد ساخن لأي كلمة ذات فهرس i، نقوم بإنشاء متجه طولN مع جميع 0 ونضع العنصرفي الموضع i الى 1. بهذه الطريقة، يتم تمثيل كل كلمة كمتجه للطول N، ويمكن أن تكون تستخدم مباشرة من قبل الشبكات العصبية.

على الرغم من سهولة إنشاء متجهات الكلمة الواحدة الساخنة، إلا أنها ليست خيارًا جيدًافي العادة. السبب الرئيسي هو أن متجهات الكلمات الساخنة لا يمكنها التعبير بدقة عن التشابه بين الكلمات المختلفة، مثل تشابه جيب التمام cosine similarity الذي نستخدمه غالبًا. بالنسبة إلى المتجهات $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$، فإن تشابه جيب التمام هو جيب تمام الزاوية بينهما:

$$\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1,1].$$

نظرًا لأن تشابه جيب التمام بين متجهات واحد ساخن لأي كلمتين مختلفتين هو 0، فلا يمكن لمتجهات واحد ساخن ترميز أوجه التشابه بين الكلمات.

15.1.2. word2vec الإشراف الذاتي Self-Supervised word2vec

تم اقتراح أداة word2vec لمعالجة المشكلة المذكورة أعلاه. يقوم بتعيين كل كلمة إلى متجه ثابت الطول، ويمكن لهذه المتجهات أن تعبر بشكل أفضل عن علاقة التشابه similarity والتناظر analogy بين الكلمات المختلفة. تحتوي أداة word2vec على نموذجين، وهما تخطي–جرام skip-gram، (2013,Mikolov et al.) وحقيبة الكلمات المستمرة continuous bag of words (CBOW)، (2013,(Mikolov et al.). بالنسبة للتمثيلات ذات المعنى الدلالي، يعتمد تدريبهم على الاحتمالات الشرطية التي يمكن اعتبارها توقعًا لبعض الكلمات باستخدام بعض الكلمات المحيطة بهافي corpora. نظرًا لأن الإشراف يأتي من البيانات بدون تسميات، فإن كلا من تخطي–جرام وحقيبة الكلمات المستمرة هي نماذج تخضع للإشراف الذاتي.

فيما يلي، سوف نقدم هذين النموذجين وطرق التدريب الخاصة بهما.

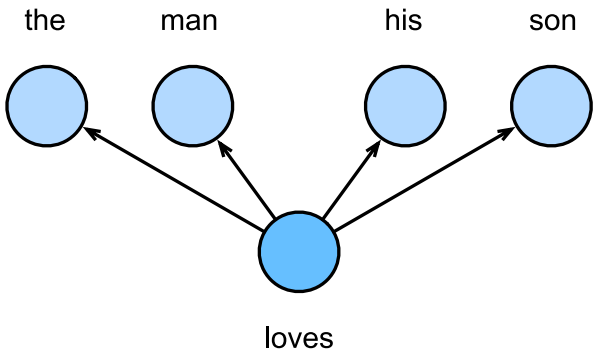
15.1.3. نموذج تخطي-جرام The Skip-Gram Model

يفترض نموذج تخطي–جرام أنه يمكن استخدام كلمة لتوليد الكلمات المحيطة بهافي تسلسل نصي. خذ التسلسل النصي "the"، "man"، "loves"، "his"، "son"كمثال. دعنا نختار "loves" ككلمة مركزية center word ونضبط حجم نافذة السياق context window size على 2. كما هو موضحفي الشكل 15.1.1، بالنظر إلى الكلمة المركزية "loves"، يأخذ نموذج تخطي–جرامفي الاعتبار الاحتمال الشرطي لتوليد كلمات السياق context words: "the" و "man" و "his" و "son"، والتي لا تبعد أكثر من كلمتين عن الكلمة المركزية:

$$P(\text{"the"},\text{"man"},\text{"his"},\text{"son"} \mid \text{"loves"}).$$

افترض أن كلمات السياق يتم إنشاؤها بشكل مستقل نظرًا للكلمة المركزية (أي الاستقلال الشرطي conditional independence).في هذه الحالة، يمكن إعادة كتابة الاحتمال الشرطي أعلاه

$$P(\text{"the"} \mid \text{"loves"}) \cdot P(\text{"man"} \mid \text{"loves"}) \cdot P(\text{"his"} \mid \text{"loves"}) \cdot P(\text{"son"} \mid \text{"loves"}).$$



الشكل 15.1.1 يأخذ نموذج تخطي–جرام في الاعتبار الاحتمال الشرطي لتوليد كلمات السياق المحيطة بكلمة مركزية.

في نموذج تخطي–جرام، تحتوي كل كلمة على تمثيلات لمتجه ذات أبعاد d لحساب الاحتمالات الشرطية. بشكل أكثر تحديداً، بالنسبة لأي كلمة بها فهرس iفي القاموس، قم بالإشارة إليها بواسطة $\mathbf{v}_i \in \mathbb{R}^d$ و $\mathbf{u}_i \in \mathbb{R}^d$ متجهيها عند استخدامها ككلمة مركزية center word وكلمة سياق context word، على التوالي. يمكن نمذجة الاحتمال الشرطي لتوليد أي كلمة سياق w_o (مع فهرس oفي القاموس) بالنظر إلى الكلمة المركزية w_c (مع فهرس cفي القاموس) من خلال عملية softmax على عمليات ضرب نقطية متجهية:

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

حيث يتم تعيين فهرس المفردات (vocabulary index) $\mathcal{V} = \{0, 1, \ldots, |\mathcal{V}| - 1\}$. إعطاء تسلسل نصي للطول T ، حيث يتم الإشارة إلى الكلمةفي الخطوة الزمنية t كـ $w^{(t)}$. افترض أن كلمات السياق يتم إنشاؤها بشكل مستقلفي ضوء أي كلمة مركزية. بالنسبة لحجم نافذة السياق m ، فإن دالة الاحتمال لنموذج تخطي–جرام هي احتمال إنشاء كل كلمات السياق مع إعطاء أي كلمة مركزية:

$$\prod_{t=1}^{T} \prod_{-m \le j \le m,\ j \ne 0} P(w^{(t+j)} \mid w^{(t)}),$$

حيث يمكن حذف أي خطوة زمنية أقل 1 من أو أكبر من T.

15.1.3.1. التدريب Training

معلمات نموذج skip-gram هي متجه الكلمات المركزي ومتجه الكلمات السياق لكل كلمةفي المفردات.في التدريب، نتعلم معلمات النموذج من خلال تعظيم دالة الاحتمال likelihood function (أي تقدير الاحتمال الأقصى maximum likelihood estimation). هذا يعادل تقليل دالة الخطأ التالية:

$$-\sum_{t=1}^{T} \sum_{-m \le j \le m,\ j \ne 0} \log P(w^{(t+j)} \mid w^{(t)}).$$

عند استخدام الانحدارالاشتقاقي العشوائي لتقليل الخطأ،في كل تكرار يمكننا بشكل عشوائي أخذ عينة لاحقة أقصر لحساب الانحدار gradient (العشوائي stochastic) لهذه النتيجة اللاحقة لتحديث معلمات النموذج. لحساب هذا الانحدار (العشوائي)، نحتاج إلى الحصول على انحدارات الاحتمال الشرطي اللوغاريتمي log conditional probability فيما يتعلق بمتجه الكلمات المركزي ومتجه كلمة السياق. بشكل عام، وفقًا لـ (15.1.4)، يكون الاحتمال الشرطي الاحتمال الشرطي اللوغاريتمي الذي يتضمن أي زوج من الكلمة المركزية w_c وكلمة السياق w_o

$$\log\ P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\,(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)).$$

من خلال التفاضل، يمكننا الحصول على الانحدار بالنسبة لمتجه الكلمات المركزي $\mathbf{v}_c$ مثل

$$\begin{aligned}\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j \in \mathcal{V}} \exp\,(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp\,(\mathbf{u}_i^\top \mathbf{v}_c)} \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\right) \mathbf{u}_j \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} P(w_j \mid w_c)\mathbf{u}_j\,.\end{aligned}$$

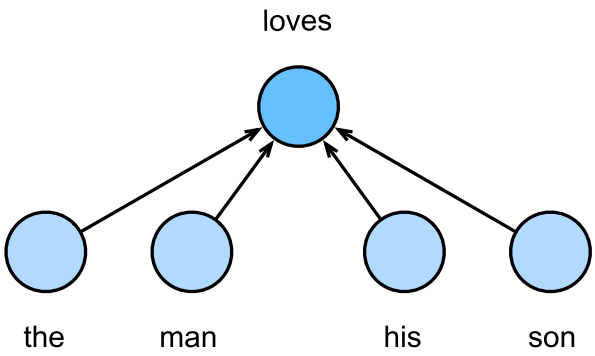
لاحظ أن العملية الحسابيةفي (15.1.8) تتطلب الاحتمالات الشرطية لجميع الكلماتفي القاموس باستخدام الكلمة المركزية w_c. يمكن الحصول على انحدارات متجهات الكلمة الأخرى بنفس الطريقة.

بعد التدريب، لأي كلمة بها فهرس i في القاموس، نحصل على متجهات الكلمات $\mathbf{v}_i$ (مثل الكلمة المركزية) و $\mathbf{u}_i$ (ككلمة السياق).في تطبيقات المعالجة اللغوية الطبيعية، يتم استخدام متجهات الكلمات المركزية لنموذج تخطي–جرام عادةً لتمثيل الكلمات.

15.1.4. نموذج حقيبة الكلمات المستمرة (CBOW) The Continuous Bag of Words (CBOW) Model

يشبه نموذج حقيبة الكلمات المستمرة (CBOW) نموذج تخطي–جرام. يتمثل الاختلاف الرئيسي عن نموذج تخطي–جرامفي أن نموذج حقيبة الكلمات المستمر يفترض أن كلمة مركزية يتم إنشاؤها بناءً على كلمات السياق المحيطة بهافي تسلسل النص. على سبيل المثال،في نفس التسلسل النصي "the" و "man" و "loves" و "his" و "son"، مع "loves" كالكلمة المركزية وحجم نافذة السياق هو 2، حقيبة الكلمات المستمرة يأخذ النموذجفي الاعتبار الاحتمال الشرطي لتوليد الكلمة المركزية "loves" بناءً على كلمات السياق "the" و "man" و "his" و "son" (كما هو موضحفي الشكل 15.1.2)، وهو

$$P(\text{"loves"} \mid \text{"the","man","his","son"}).$$



الشكل 15.1.2 نموذج حقيبة الكلمات المستمرة يأخذفي الاعتبار الاحتمال الشرطي لتوليد الكلمة المركزية بالنظر إلى كلمات السياق المحيطة بها.

نظرًا لوجود كلمات سياق متعددةفي نموذج حقيبة الكلمات المستمرة، يتم حساب متوسط متجهات كلمات السياق هذهفي حساب الاحتمال الشرطي. على وجه التحديد، بالنسبة لأي كلمة تحتوي على فهرس i في القاموس، قم بالإشارة إليها بواسطة $\mathbf{v}_i \in \mathbb{R}^d$ و $\mathbf{u}_i \in \mathbb{R}^d$ متجهيها عند استخدامها ككلمة سياق وكلمة مركزية (يتم تبديل المعانيفي نموذج skip-gram)، على التوالي. يمكن نمذجة الاحتمال الشرطي لتوليد أي كلمة مركزية w_c (مع فهرس c في القاموس) بالنظر إلى كلمات السياق المحيطة بها $w_{o_1}, \ldots, w_{o_{2m}}$ (مع فهرس $o_1, \ldots, o_{2m}$ في القاموس) بواسطة

$$P(w_c \mid w_{o_1}, \ldots, w_{o_{2m}}) = \frac{\exp(\frac{1}{2m}\mathbf{u}_c^\top(\mathbf{v}_{o_1} + \cdots + \mathbf{v}_{o_{2m}}))}{\sum_{i\in\mathcal{V}} \exp(\frac{1}{2m}\mathbf{u}_i^\top(\mathbf{v}_{o_1} + \cdots + \mathbf{v}_{o_{2m}}))}.$$

للإيجاز، ليكن $\mathcal{W}_o = \{w_{o_1}, \ldots, w_{o_{2m}}\}$ و $\bar{\mathbf{v}}_o = (\mathbf{v}_{o_1} + \cdots + \mathbf{v}_{o_{2m}})/(2m)$. ثم يمكن تبسيط (15.1.10) كـ

$$P(w_c \mid \mathcal{W}_o) = \frac{\exp(\mathbf{u}_c^\top \bar{\mathbf{v}}_o)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)}.$$

إعطاء تسلسل نصي للطول T، حيث يتم الإشارة إلى الكلمةفي الخطوة الزمنية t كـ $w^{(t)}$. بالنسبة لحجم نافذة السياق m، فإن دالة الاحتمال لنموذج حقيبة الكلمات المستمرة هي احتمال إنشاء كل الكلمات المركزية بالنظر إلى كلمات السياق الخاصة بها:

$$\prod_{t=1}^{T} P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}).$$

15.1.4.1. التدريب Training

تدريب نماذج حقيبة الكلمات المستمرة هو نفسه تقريبًا مثل نماذج تخطي–جرام التدريبية. الحد الأقصى لتقدير الاحتمالية لنموذج حقيبة الكلمات المستمرة يعادل تقليل دالة الخطأ التالية:

$$-\sum_{t=1}^{T} \log P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)}).$$

لاحظ أن

$$\log P(w_c \mid \mathcal{W}_o) = \mathbf{u}_c^\top \bar{\mathbf{v}}_o - \log(\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)).$$

من خلال التفاضل، يمكننا الحصول على تدرجه (انحداره) فيما يتعلق بأي متجه لكلمة سياق مثل $\mathbf{v}_{o_i}(i = 1, \ldots, 2m)$

$$\begin{aligned}\frac{\partial \log P(w_c \mid \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} &= \frac{1}{2m}(\mathbf{u}_c - \sum_{j\in\mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)}) \\ &= \frac{1}{2m}(\mathbf{u}_c - \sum_{j\in\mathcal{V}} P(w_j \mid \mathcal{W}_o)\mathbf{u}_j).\end{aligned}$$

يمكن الحصول على انحدارات متجهات الكلمة الأخرى بنفس الطريقة. على عكس نموذج تخطي–جرام، فإن نموذج حقيبة الكلمات المستمرة يستخدم عادةً متجهات كلمات السياق مثل تمثيلات الكلمة.

15.1.5. الملخص

- متجهات الكلمات word vectors هي متجهات تستخدم لتمثيل الكلمات ، ويمكن اعتبارها أيضًا متجهات للميزات أو تمثيلات للكلمات. تسمى تقنية تعيين الكلمات إلى المتجهات الحقيقية تضمين الكلمات word embedding.
- تحتوي أداة word2vec على كلا من نموذجي تخطي–جرام skip-gram وحقيبة الكلمات المستمرة continuous bag of words models.
- يفترض نموذج تخطي–جرام أنه يمكن استخدام كلمة لتوليد الكلمات المحيطة بها في تسلسل نصي ؛ بينما يفترض نموذج حقيبة الكلمات المستمرة أن الكلمة المركزية center word يتم إنشاؤها بناءً على كلمات السياق context words المحيطة بها.

15.1.6. التمارين

1. ما هو التعقيد الحسابي لحساب كل انحدار؟ ماذا يمكن أن تكون المشكلة إذا كان حجم القاموس ضخمًا؟
2. تتكون بعض العبارات الثابتة في اللغة الإنجليزية من كلمات متعددة ، مثل "new york". كيفية تدريب متجهات كلمتهم؟ تلميح: انظر القسم 4 في مقالة word2vec، (Mikolov et al.,2013).
3. دعونا نفكر في تصميم word2vec من خلال أخذ نموذج تخطي–جرام كمثال. ما العلاقة بين حاصل الضرب القياسي لمتجهي كلمتين في نموذج تخطي–جرام وتشابه جيب التمام؟ بالنسبة إلى زوج من الكلمات ذات دلالات متشابهة ، لماذا قد يكون تشابه جيب التمام لمتجهات كلمتهم (التي تم تدريبها بواسطة نموذج تخطي–جرام) مرتفعًا؟

15.2. التدريب التقريبي Approximate Training

تذكر مناقشاتنافي القسم 15.1. الفكرة الرئيسية لنموذج تخطي–جرام هي استخدام عمليات softmax لحساب الاحتمال الشرطي لتوليد كلمة سياق w_o بناءً على الكلمة المركزية w_c المعطاةفي (15.1.4)، والتي تُعطى الخسارة اللوغاريتمية المقابلة لها بعكس (15.1.7).

نظرًا لطبيعة عملية softmax، نظرًا لأن كلمة السياق قد تكون أي شخصفي القاموس $\mathcal{V}$، فإن عكس (15.1.7) يحتوي على مجموع العناصر بقدر الحجم الكامل للمفردات. وبالتالي، فإن

حساب الانحدار لنموذج التخطي في (15.1.8) ونموذج حقيبة الكلمات المستمرة في (15.1.15) يحتويان على الجمع summation. لسوء الحظ، فإن التكلفة الحسابية لمثل هذه الانحدارات التي تصل إلى قاموس كبير (غالبًا بمئات الآلاف أو ملايين الكلمات) ضخمة!

من أجل تقليل التعقيد الحسابي المذكور أعلاه، سيقدم هذا القسم طريقتين تدريب تقريبيين approximate training: أخذ العينات السلبية negative sampling وsoftmax الهرمي hierarchical softmax. نظرًا للتشابه بين نموذج تخطي–جرام ونموذج حقيبة الكلمات المستمرة، سنأخذ نموذج تخطي – جرام كمثال لوصف هاتين الطريقتين التدريبيتين التقريبيتين.

15.2.1. أخذ العينات السلبية Negative Sampling

أخذ العينات السلبية Negative Sampling يعدل دالة الهدف الأصلية. بالنظر إلى نافذة السياق لكلمة مركزية w_c، فإن حقيقة أن أي كلمة w_o (سياق) تأتي من نافذة السياق هذه تعتبر حدثًا مع الاحتمالية التي تم نمذجتها بواسطة

$$P(D = 1 \mid w_c, w_o) = \sigma(\mathbf{u}_o^\top \mathbf{v}_c),$$

حيث يستخدم تعريف دالة sigmoid:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

لنبدأ بمضاعفة الاحتمالية المشتركة لجميع هذه الأحداث في التسلسل النصي لتدريب عمليات تضمين الكلمات. على وجه التحديد، بالنظر إلى تسلسل نصي للطول T ، قم بالإشارة إلى $w^{(t)}$ الكلمة في خطوة زمنية t واترك حجم نافذة السياق m، ضع في اعتبارك تعظيم الاحتمال المشترك joint probability

$$\prod_{t=1}^{T} \prod_{-m \le j \le m,\, j \ne 0} P(D = 1 \mid w^{(t)}, w^{(t+j)}).$$

ومع ذلك، (15.2.3) يأخذ بعين الاعتبار فقط الأحداث التي تنطوي على أمثلة إيجابية. نتيجة لذلك، يتم تكبير الاحتمال المشترك في (15.2.3) إلى 1 فقط إذا كانت جميع متجهات الكلمات مساوية لما لا نهاية. بالطبع، هذه النتائج لا معنى لها. لجعل دالة الهدف أكثر جدوى، يضيف أخذ العينات السلبية negative sampling أمثلة سلبية مأخوذة من توزيع محدد مسبقًا.

يشير S الى الحدث إلى أن كلمة سياق w_o تأتي من نافذة سياق كلمة مركزية w_c. بالنسبة لهذا الحدث الذي يتضمن w_o ، من عينة K لتوزيع محدد مسبقًا، كلمات ضوضاء noise words ليست من نافذة السياق هذه. قم بالإشارة الى N_k للحدث إلى أن كلمة ضوضاء $(k = 1, \ldots, K)$ w_k لا تأتي من نافذة سياق لـ w_c. افترض أن هذه الأحداث التي تتضمن المثال الإيجابي والأمثلة

السلبية $S, N_1, \ldots, N_K$ مستقلة عن بعضها البعض. يعيد أخذ العينات السلبية كتابة الاحتمال المشترك (الذي يتضمن أمثلة إيجابية فقط)في (15.2.3) كـ

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{(t+j)} \mid w^{(t)}),$$

حيث يتم تقريب الاحتمال الشرطي من خلال الأحداث $S, N_1, \ldots, N_K$:

$$P(w^{(t+j)} \mid w^{(t)}) = P(D=1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k \sim P(w)}^{K} P(D=0 \mid w^{(t)}, w_k).$$

قم بالإشارة بواسطة i_t وh_k مؤشرات الكلمة $w^{(t)}$ في الخطوة الزمنية t لتسلسل النص وكلمة الضوضاء w_k، على التوالي. الخسارة اللوغاريتمية فيما يتعلق بالاحتمالات الشرطيةفي (15.2.5) هي

$$\begin{aligned}
-\log P(w^{(t+j)} \mid w^{(t)}) &= -\log P(D=1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log P(D=0 \mid \\
&= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)\right) \\
&= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right).
\end{aligned}$$

يمكننا أن نرى الآن أن التكلفة الحسابية للانحداراتفي كل خطوة تدريب لا علاقة لها بحجم القاموس، ولكنها خطيا تعتمد على K. عند ضبط المعلمة الفائقة K على قيمة أصغر، تكون التكلفة الحسابية للانحداراتفي كل خطوة تدريب مع أخذ العينات السلبية أقل.

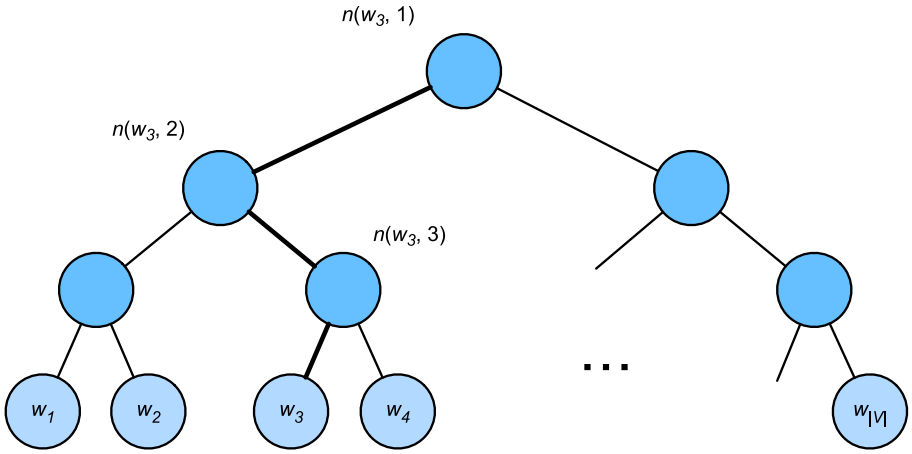
15.2.2. Softmax الهرمي Hierarchical Softmax

كطريقة تدريب تقريبية بديلة، يستخدم softmax الهرمي الشجرة الثنائية binary tree، وهي بُنية بيانات موضحةفي الشكل 15.2.1، حيث تمثل كل عقدة ورقيةفي الشجرة كلمةفي القاموس.

قم بالإشارة بواسطة $L(w)$ إلى عدد العقد (بمافي ذلك كلا الطرفين) على المسار من العقدة الجذرية إلى العقدة الطرفية التي تمثل كلمةفي الشجرة الثنائية. لتكن $n(w,j)$ العقدة j^{th} على هذا المسار، مع سياق متجه الكلمات $\mathbf{u}_{n(w,j)}$. على سبيل المثال،في الشكل 15.2.1. يقارب softmax الهرمي الاحتمال الشرطيفي (15.1.4) مثل

$$P(w_o \mid w_c) = \prod_{j=1}^{L(w_o)-1} \sigma\left(\llbracket n(w_o, j+1) = \text{leftChild}(n(w_o, j)) \rrbracket \cdot \mathbf{u}_{n(w_o,j)}^\top \mathbf{v}_c\right),$$

حيث يتم تعريف الدالة σفي (15.2.2)، و$\text{leftChild}(n)$ هي العقدة الفرعية اليسرى للعقدة n: إذا كان x صحيح، يكون $[[x]] = 1$ وخلاف ذلك يكون $[[x]] = -1$.



الشكل 15.2.1 softmax الهرمي للتدريب التقريبي، حيث تمثل كل عقدة ورقةفي الشجرة كلمةفي القاموس.

للتوضيح، دعونا نحسب الاحتمال الشرطي لتوليد الكلمة w_3 معطى الكلمة w_cفي الشكل 15.2.1.. يتطلب هذا عمليات ضرب نقطية بين متجه الكلمات $\mathbf{v}_c$ لـ w_c ومتجهات العقدة غير الورقية على المسار (المسار بالخط الغامقفي الشكل 15.2.1) من الجذر إلى w_3 ، والذي يتم اجتيازه يسارًا ثم يمينًا ثم يسارًا:

$$P(w_3 \mid w_c) = \sigma(\mathbf{u}_{n(w_3,1)}^\top \mathbf{v}_c) \cdot \sigma(-\mathbf{u}_{n(w_3,2)}^\top \mathbf{v}_c) \cdot \sigma(\mathbf{u}_{n(w_3,3)}^\top \mathbf{v}_c).$$

لان $\sigma(x) + \sigma(-x) = 1$ ، تنص على أن الاحتمالات الشرطية لتوليد جميع الكلماتفي القاموس $\mathcal{V}$ بناءً على مجموع أي كلمة w_c يصل إلى واحد:

$$\sum_{w \in \mathcal{V}} P(w \mid w_c) = 1. \qquad (15.2.9)$$

لحسن الحظ، نظرًا لأن $L(w_o) - 1$ بترتيب $\mathcal{O}(\log_2 |\mathcal{V}|)$ بسبب بُنية الشجرة الثنائية، عندما يكون حجم القاموس $\mathcal{V}$ ضخمًا، يتم تقليل التكلفة الحسابية لكل خطوة تدريب باستخدام softmax الهرمي بشكل كبير مقارنةً بذلك بدون تدريب تقريبي.

15.2.3. الملخص

- يبني أخذ العينات السلبية Negative sampling دالة الخطأ من خلال النظر في الأحداث المستقلة بشكل متبادل والتي تتضمن أمثلة إيجابية وسلبية. تعتمد التكلفة الحسابية للتدريب خطيًا على عدد كلمات الضوضاء noise words في كل خطوة.
- يبني softmax الهرمي دالة الخطأ باستخدام المسار من عقدة الجذر إلى عقدة الورقة في الشجرة الثنائية. تعتمد التكلفة الحسابية للتدريب على لوغاريتم حجم القاموس في كل خطوة.

15.2.4. التمارين

1. كيف يمكننا أخذ عينات من كلمات الضوضاء في اخذ العينات السلبية؟
2. تحقق من أن (15.2.9) صحيحة.
3. كيف يتم تدريب نموذج حقيبة الكلمات المستمر باستخدام اخذ العينات السالبة والتسلسل الهرمي softmax على التوالي؟

15.3. مجموعة البيانات الخاصة بالتدريب المسبق لتضمين الكلمات
The Dataset for Pretraining Word Embeddings

الآن بعد أن عرفنا التفاصيل الفنية لنماذج word2vec وطرق التدريب التقريبية approximate training methods، دعنا نتصفح عمليات تنفيذها. على وجه التحديد، سوف نأخذ نموذج تخطي-جرام skip-gram في القسم 15.1 وأخذ العينات السلبية negative sampling في القسم 15.2 كمثال. في هذا القسم، نبدأ بمجموعة البيانات الخاصة بالتدريب المسبق على نموذج تضمين كلمة pretraining the word embedding model: سيتم تحويل التنسيق الأصلي للبيانات إلى دفعات صغيرة minibatches يمكن تكرارها أثناء التدريب.

```
import collections
import math
import os
import random
from mxnet import gluon, np
from d2l import mxnet as d2l
```

15.3.1. قراءة مجموعة البيانات Reading the Dataset

مجموعة البيانات dataset التي نستخدمها هنا هي Penn Tree Bank (PTB). تم أخذ عينات هذه المجموعة corpus من مقالات وول ستريت جورنال، وتم تقسيمها إلى مجموعات التدريب training set والتحقق من الصحة validation set والاختبار test set. في التنسيق

الأصلي، يمثل كل سطر من الملف النصي جملة من الكلمات مفصولة بمسافات. هنا نتعامل مع كل كلمة كرمز token.

```
#@save
d2l.DATA_HUB['ptb'] = (d2l.DATA_URL + 'ptb.zip',

'319d85e578af0cdc590547f26231e4e31cdf1e42')

#@save
def read_ptb():
    """Load the PTB dataset into a list of text
lines."""
    data_dir = d2l.download_extract('ptb')
    # Read the training set
    with open(os.path.join(data_dir, 'ptb.train.txt'))
as f:
        raw_text = f.read()
    return [line.split() for line in
raw_text.split('\n')]

sentences = read_ptb()
f'# sentences: {len(sentences)}'
```

```
Downloading ../data/ptb.zip from http://d2l-data.s3-
accelerate.amazonaws.com/ptb.zip...
```

```
'# sentences: 42069'
```

بعد قراءة مجموعة التدريب، نقوم ببناء مفردات للمجموعة، حيث يتم استبدال أي كلمة تظهر أقل من 10 مرات بالرمز "<unk>". لاحظ أن مجموعة البيانات الأصلية تحتوي أيضًا على رموز "<unk>" التي تمثل كلمات نادرة (غير معروفة unknown).

```
vocab = d2l.Vocab(sentences, min_freq=10)
f'vocab size: {len(vocab)}'
```

15.3.2. أخذ العينات الفرعية Subsampling

تحتوي البيانات النصية عادةً على كلمات عالية التردد (التكرار high-frequency words) مثل "the" و "a" و "in": قد تظهر بلايين المرات في مجموعات كبيرة جدًا. ومع ذلك، غالبًا ما تتزامن هذه الكلمات مع العديد من الكلمات المختلفة في إطارات السياق، مما يوفر القليل من الإشارات المفيدة. على سبيل المثال، ضع في اعتبارك كلمة "chip" في نافذة السياق: من البديهي

أن تواجدها مع الكلمة ذات التكرار المنخفض "intel" أكثر فائدةفي التدريب من التواجد المشترك مع الكلمة عالية التكرار "a". علاوة على ذلك، فإن التدريب بكميات هائلة من الكلمات (عالية التردد) بطيء. وبالتالي، عند تدريب نماذج تضمين الكلمات، يمكن اخذ عينات فرعية subsampled الكلمات عالية التردد (Mikolov et al.، 2013). على وجه التحديد، سيتم تجاهل كل كلمة مفهرسة w_iفي مجموعة البيانات باحتمالية

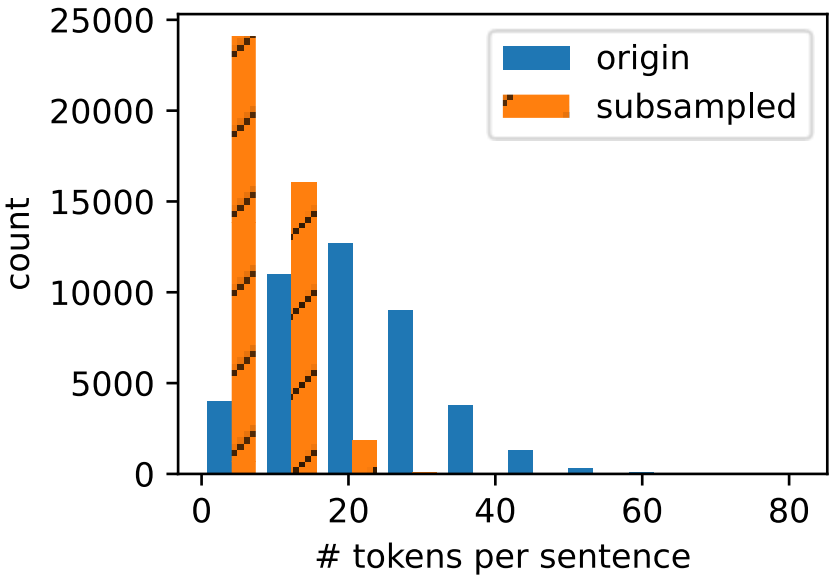
$$P(w_i) = max(1 - \sqrt{\frac{t}{f(w_i)}}, 0),$$

حيث $f(w_i)$ هي نسبة عدد الكلمات w_i إلى العدد الإجمالي للكلماتفي مجموعة البيانات، والثابت هو معلمة فائقة (10^{-4} في التجربة). يمكننا أن نرى أنه فقط عندما يمكن تجاهل التردد النسبي $f(w_i) > t$ للكلمة (عالية التردد) w_i ، وكلما زاد التردد النسبي للكلمة، زاد احتمال التخلص منها.

```
#@save
def subsample(sentences, vocab):
    """Subsample high-frequency words."""
    # Exclude unknown tokens ('<unk>')
    sentences = [[token for token in line if
vocab[token] != vocab.unk]
                 for line in sentences]
    counter = collections.Counter([
        token for line in sentences for token in line])
    num_tokens = sum(counter.values())

    # Return True if `token` is kept during subsampling
    def keep(token):
        return(random.uniform(0, 1) <
               math.sqrt(1e-4 / counter[token] *
num_tokens))

    return ([[token for token in line if keep(token)]
for line in sentences],
            counter)

subsampled, counter = subsample(sentences, vocab)
```

يرسم مقتطف الكود التالي الرسم البياني لعدد الرموز لكل جملة قبل وبعد أخذ العينات الفرعية. كما هو متوقع، يؤدي اخذ العينات الفرعي إلى تقصير الجمل بشكل كبير عن طريق إسقاط الكلمات عالية التردد، مما يؤدي إلى تسريع التدريب.

```
d2l.show_list_len_pair_hist(['origin', 'subsampled'], '#
tokens per sentence',
                            'count', sentences,
subsampled);
```



بالنسبة للرموز المنفردة individual tokens، يكون معدل أخذ العينات للكلمة عالية التردد "the" أقل من 20/1.

```
def compare_counts(token):
    return (f'# of "{token}": '
            f'before={sum([l.count(token) for l in
sentences])}, '
            f'after={sum([l.count(token) for l in
subsampled])}')

compare_counts('the')
```

```
'# of "the": before=50770, after=2095'
```

في المقابل، يتم الاحتفاظ بالكلمات منخفضة التردد "join" تمامًا.

```
compare_counts('join')
```

```
'# of "join": before=45, after=45'
```

بعد أخذ العينات الفرعية، نقوم بتعيين الرموز لمؤشراتها للمجموعة.

```
corpus = [vocab[line] for line in subsampled]
corpus[:3]
```

```
[[], [27, 4127, 6612, 3228, 710, 1773], [3922, 1922,
4743]]
```

15.3.3. استخراج كلمات المركز وكلمات السياق Extracting Center Words and Context Words

تستخرج دالة `get_centers_and_contexts` التالية جميع كلمات المركز وكلمات السياق الخاصة بها من المجموعة. يقوم بشكل منتظم بأخذ عينات عدد صحيح بين 1 و `max_window_size` عشوائيًا مثل حجم نافذة السياق. بالنسبة إلى أي كلمة مركزية، فإن الكلمات التي لا تتجاوز المسافة بينها وبين حجم نافذة سياق العينة هي كلمات السياق الخاصة بها.

```
#@save
def get_centers_and_contexts(corpus, max_window_size):
    """Return center words and context words in skip-
gram."""
    centers, contexts = [], []
    for line in corpus:
        # To form a "center word--context word" pair,
each sentence needs to
        # have at least 2 words
        if len(line) < 2:
            continue
        centers += line
        for i in range(len(line)):  # Context window
centered at `i`
            window_size = random.randint(1,
max_window_size)
            indices = list(range(max(0, i -
window_size),
                                 min(len(line), i + 1 +
window_size)))
            # Exclude the center word from the context
words
            indices.remove(i)
```

```
            contexts.append([line[idx] for idx in
indices])
    return centers, contexts
```

بعد ذلك، نقوم بإنشاء مجموعة بيانات اصطناعية تحتوي على جملتين من 7 و 3 كلمات، على التوالي. اجعل الحد الأقصى لحجم نافذة السياق هو 2 واطبع جميع الكلمات المركزية وكلمات السياق الخاصة بها.

```
tiny_dataset = [list(range(7)), list(range(7, 10))]
print('dataset', tiny_dataset)
for center, context in
zip(*get_centers_and_contexts(tiny_dataset, 2)):
    print('center', center, 'has contexts', context)
```

```
dataset [[0, 1, 2, 3, 4, 5, 6], [7, 8, 9]]
center 0 has contexts [1]
center 1 has contexts [0, 2]
center 2 has contexts [1, 3]
center 3 has contexts [1, 2, 4, 5]
center 4 has contexts [2, 3, 5, 6]
center 5 has contexts [3, 4, 6]
center 6 has contexts [5]
center 7 has contexts [8]
center 8 has contexts [7, 9]
center 9 has contexts [7, 8]
```

عند التدريب على مجموعة بيانات PTB، قمنا بتعيين الحد الأقصى لحجم نافذة السياق على 5. يستخرج التالي كل الكلمات المركزية وكلمات السياق الخاصة بهافي مجموعة البيانات.

```
all_centers, all_contexts =
get_centers_and_contexts(corpus, 5)
f'# center-context pairs: {sum([len(contexts) for
contexts in all_contexts])}'
```

```
'# center-context pairs: 1505018'
```

15.3.4. أخذ العينات السلبية Negative Sampling

نستخدم اخذ العينات السلبية Negative Sampling للتدريب التقريبي. لأخذ عينات من كلمات الضوضاء وفقًا لتوزيع محدد مسبقًا، نحدد فئة `RandomGenerator` التالية ، حيث يتم تمرير توزيع أخذ العينات (الذي قد يكون غير طبيعي) عبر وسيطة `sampling_weights`.

```
#@save
class RandomGenerator:
```

```
    """Randomly draw among {1, ..., n} according to n
sampling weights."""
    def __init__(self, sampling_weights):
        # Exclude
        self.population = list(range(1,
len(sampling_weights) + 1))
        self.sampling_weights = sampling_weights
        self.candidates = []
        self.i = 0

    def draw(self):
        if self.i == len(self.candidates):
            # Cache `k` random sampling results
            self.candidates = random.choices(
                self.population, self.sampling_weights,
k=10000)
            self.i = 0
        self.i += 1
        return self.candidates[self.i - 1]
```

على سبيل المثال، يمكننا رسم 10 متغيرات عشوائية X بين المؤشرات 1 و 2 و 3 مع احتمالات أخذ العينات $P(X = 1) = 2/9, P(X = 2) = 3/9$ و $P(X = 3) = 4/9$ على النحو التالي.

بالنسبة إلى زوج من الكلمات المركزية وكلمة السياق، قمنا بأخذ عينات عشوائية من كلمات الضوضاء K (5في التجربة). وفقًا للاقتراحات الواردةفي مقالة word2vec، يتم تعيين احتمالية أخذ العينات$P(w)$ لكلمة ضوضاء w على التردد النسبيفي القاموس الذي تم رفعه إلى قوة 0.75 (Mikolov et al.، 2013).

```
#@save
def get_negatives(all_contexts, vocab, counter, K):
    """Return noise words in negative sampling."""
    # Sampling weights for words with indices 1, 2, ...
(index 0 is the
    # excluded unknown token) in the vocabulary
    sampling_weights =
[counter[vocab.to_tokens(i)]**0.75
                          for i in range(1, len(vocab))]
    all_negatives, generator = [],
RandomGenerator(sampling_weights)
    for contexts in all_contexts:
        negatives = []
```

```
        while len(negatives) < len(contexts) * K:
            neg = generator.draw()
            # Noise words cannot be context words
            if neg not in contexts:
                negatives.append(neg)
        all_negatives.append(negatives)
    return all_negatives

all_negatives = get_negatives(all_contexts, vocab,
counter, 5)
```

15.3.5. تحميل أمثلة التدريب في الدفعات الصغيرة Loading Training Examples in Minibatches

بعد استخراج كل الكلمات المركزية مع كلمات السياق وكلمات الضوضاء التي تم أخذ عينات منها، سيتم تحويلها إلى مجموعات صغيرة من الأمثلة التي يمكن تحميلها بشكل متكرر أثناء التدريب.

في الدفعات الصغيرة، يشتمل المثال i^{th} على كلمة مركزية وكلمات سياق وكلمات ضوضاء. نظرًا لاختلاف أحجام نافذة السياق $n_i + m_i$ ، فإنها تختلف باختلاف i. وبالتالي، بالنسبة لكل مثال، نجمع كلمات السياق وكلمات الضوضاءفي متغير `contexts_negatives`، وأصفار الحشو pad zeros حتى يصل طول السلسلة إلى $max_i n_i + m_i$ (`max_len`). لاستبعاد الحشواتفي حساب الخطأ، نحدد متغير القناع `masks`. هناك تطابق واحد لواحد بين العناصر في `masks` والعناصرفي `contexts_negatives` ، حيث تتوافق الأصفار (بخلاف تلك الموجودةفي `masks`) مع الحشواتفي `contexts_negatives`.

للتمييز بين الأمثلة الإيجابية والسلبية، نقوم بفصل كلمات السياق عن كلمات الضوضاءفي `contexts_negatives` عبر متغير `labels`. على غرار `masks`، هناك أيضًا تطابق واحد لواحد بين العناصرفي `labels` والعناصرفي `contexts_negatives` ، حيث تتوافق الآحاد (بخلاف الأصفار)في التسميات مع كلمات السياق (الأمثلة الإيجابية)في `contexts_negatives`.

يتم تنفيذ الفكرة أعلاهفي دالة `batchify` التالية. بيانات الإدخال الخاصة به عبارة عن قائمة بطول يساوي حجم الدُفعة، حيث يكون كل عنصر مثالاً يتكون من كلمات المركز `center` وسياق كلمات `context` وكلمات الضوضاء الخاصة به `negative`. تقوم هذه الدالة بإرجاع الدفعات الصغيرة minibatch الذي يمكن تحميله للحسابات أثناء التدريب، مثل تضمين متغير `mask`.

```
#@save
def batchify(data):
    """Return a minibatch of examples for skip-gram with
negative sampling."""
    max_len = max(len(c) + len(n) for _, c, n in data)
    centers, contexts_negatives, masks, labels = [], [],
[], []
    for center, context, negative in data:
        cur_len = len(context) + len(negative)
        centers += [center]
        contexts_negatives += [context + negative + [0]
* (max_len - cur_len)]
        masks += [[1] * cur_len + [0] * (max_len -
cur_len)]
        labels += [[1] * len(context) + [0] * (max_len -
len(context))]
    return (np.array(centers).reshape((-1, 1)),
np.array(
        contexts_negatives), np.array(masks),
np.array(labels))
```

دعونا نختبر هذه الدالة باستخدام دفعات صغيرة من مثالين.

```
x_1 = (1, [2, 2], [3, 3, 3, 3])
x_2 = (1, [2, 2, 2], [3, 3])
batch = batchify((x_1, x_2))

names = ['centers', 'contexts_negatives', 'masks',
'labels']
for name, data in zip(names, batch):
    print(name, '=', data)
```

```
centers = [[1.]
 [1.]]
contexts_negatives = [[2. 2. 3. 3. 3. 3.]
 [2. 2. 2. 3. 3. 0.]]
masks = [[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 0.]]
labels = [[1. 1. 0. 0. 0. 0.]
 [1. 1. 1. 0. 0. 0.]]
```

15.3.6. وضع كل شيء معا Putting It All Together

أخيرًا، نحدد دالة `load_data_ptb` التي تقرأ مجموعة بيانات PTB وتعيد مكرر البيانات والمفردات.

```
#@save
def load_data_ptb(batch_size, max_window_size,
num_noise_words):
    """Download the PTB dataset and then load it into
memory."""
    sentences = read_ptb()
    vocab = d2l.Vocab(sentences, min_freq=10)
    subsampled, counter = subsample(sentences, vocab)
    corpus = [vocab[line] for line in subsampled]
    all_centers, all_contexts =
get_centers_and_contexts(
        corpus, max_window_size)
    all_negatives = get_negatives(
        all_contexts, vocab, counter, num_noise_words)
    dataset = gluon.data.ArrayDataset(
        all_centers, all_contexts, all_negatives)
    data_iter = gluon.data.DataLoader(
        dataset, batch_size,
shuffle=True,batchify_fn=batchify,
        num_workers=d2l.get_dataloader_workers())
    return data_iter, vocab
```

دعنا نطبع الدفعة الصغيرة الأولى من مكرر البيانات.

```
data_iter, vocab = load_data_ptb(512, 5, 5)
for batch in data_iter:
    for name, data in zip(names, batch):
        print(name, 'shape:', data.shape)
    break
```

```
centers shape: (512, 1)
contexts_negatives shape: (512, 60)
masks shape: (512, 60)
labels shape: (512, 60)
```

15.3.7. الملخص

- قد لا تكون الكلمات عالية التردد مفيدة جدًا في التدريب. يمكننا أخذ عينات فرعية منها للتعجيل في التدريب.
- من أجل الكفاءة الحسابية ، نقوم بتحميل الأمثلة في الدفعات الصغيرة minibatches. يمكننا تحديد متغيرات أخرى لتمييز الحشوات paddings من غير الحشوات non-paddings ، والأمثلة الإيجابية من السلبية.

15.3.8. التمارين

1. كيف يتغير وقت تشغيل الكود في هذا القسم إذا لم يتم استخدام اخذ العينات الجزئي subsampling ؟
2. تخزن فئة `RandomGenerator` نتائج أخذ العينات العشوائية `k` مؤقتًا. اضبط `k` على قيم أخرى وانظر كيف تؤثر على سرعة تحميل البيانات.
3. ما هي المعلمات الفائقة الأخرى في كود هذا القسم التي قد تؤثر على سرعة تحميل البيانات؟

15.4. التدريب المسبق لـ word2vec Pretraining word2vec

نواصل تنفيذ نموذج تخطي-جرام skip-gram المحددفي القسم 15.1. ثم سنقوم بإجراء اختبار مسبق لـ word2vec باستخدام اخذ عينات سلبية على مجموعة بيانات PTB. بادئ ذي بدء، دعنا نحصل على مكرر البيانات والمفردات لمجموعة البيانات هذه عن طريق استدعاء الدالة `d2l.load_data_ptb` ، والتي تم وصفهافي القسم 15.3

```
import math
from mxnet import autograd, gluon, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()

batch_size, max_window_size, num_noise_words = 512, 5, 5
data_iter, vocab = d2l.load_data_ptb(batch_size,
max_window_size,
                                     num_noise_words)
```

15.4.1. نموذج تخطي-جرام The Skip-Gram Model

نقوم بتنفيذ نموذج تخطي-جرام skip-gram باستخدام طبقات التضمين embedding layers وضرب المصفوفة الدفعية batch matrix multiplications. أولاً، دعنا نراجع كيفية عمل تضمين الطبقات.

15.4.1.1. طبقة التضمين Embedding Layer

كما هو موضحفي القسم 10.7، تقوم طبقة التضمين بتعيين فهرس الرمز token's index إلى متجه المعالم الخاص بها. وزن هذه الطبقة عبارة عن مصفوفة عدد صفوفها يساوي حجم القاموس (`input_dim`) وعدد الأعمدة يساوي البعد المتجه لكل رمز (`output_dim`). بعد تدريب نموذج تضمين الكلمة، هذا الوزن هو ما نحتاجه.

```
embed = nn.Embedding(input_dim=20, output_dim=4)
embed.initialize()
embed.weight
```

```
Parameter embedding0_weight (shape=(20, 4),
dtype=float32)
```

مدخلات طبقة التضمين هي فهرس الرمز (كلمة). لأي فهرس الرمز i ، يمكن الحصول على تمثيل المتجه من صف i^{th} مصفوفة الوزنفي طبقة التضمين. نظرًا لأنه تم ضبط البعد المتجه (output_dim) على 4، فإن طبقة التضمين ترجع متجهات ذات شكل (2, 3, 4) لمجموعة صغيرة من مؤشرات الرمز token indices بالشكل (2, 3).

```
x = np.array([[1, 2, 3], [4, 5, 6]])
embed(x)
```

```
rray([[[ 0.01438687,  0.05011239,  0.00628365,
0.04861524],
        [-0.01068833,  0.01729892,  0.02042518, -
0.01618656],
        [-0.00873779, -0.02834515,  0.05484822, -
0.06206018]],

       [[ 0.06491279, -0.03182812, -0.01631819, -
0.00312688],
        [ 0.0408415 ,  0.04370362,  0.00404529, -
0.0028032 ],
        [ 0.00952624, -0.01501013,  0.05958354,
0.04705103]]])
```

15.4.1.2. تعريف الانتشار الأمامي Defining the Forward Propagation

في الانتشار الأمامي، يشتمل إدخال نموذج تخطي-جرام skip-gram على مؤشرات الكلمات المركزية center للشكل (batch size, 1) والسياق المتسلسل ومؤشرات كلمة الضوضاء contexts_and_negatives للشكل (batch size, max_len) ، حيث يتم تعريف max_len في القسم 15.3.5. يتم تحويل هذين المتغيرين أولاً من مؤشرات الرمز إلى متجهات عبر طبقة التضمين، ثم يقوم ضرب مصفوفة الدُفعات (الموضحفي القسم 11.2.4.1) بإرجاع إخراج الشكل (batch size, 1, max_len). كل عنصرفي الإخراج هو الضرب النقطي لمتجه الكلمات المركزية أو متجه كلمة السياق او الضوضاء.

```
def skip_gram(center, contexts_and_negatives, embed_v,
embed_u):
    v = embed_v(center)
    u = embed_u(contexts_and_negatives)
    pred = npx.batch_dot(v, u.swapaxes(1, 2))
```

```
    return pred
```

دعنا نطبع شكل الإخراج لدالة `skip_gram` لبعض أمثلة المدخلات.

```
skip_gram(np.ones((2, 1)), np.ones((2, 4)), embed,
embed).shape
```

```
(2, 1, 4)
```

15.4.2. التدريب Training

قبل تدريب نموذج تخطي-جرام بأخذ عينات سلبية، دعنا أولاً نحدد دالة الخطأ الخاصة به.

15.4.2.1. الخسارة الثنائية عبر الانتروبيا Binary Cross-Entropy Loss

وفقًا لتعريف دالة الخسارة لأخذ العينات السالبفي القسم 15.2.1، سوف نستخدم الخسارة الثنائية عبر الانتروبيا binary cross-entropy loss.

```
loss = gluon.loss.SigmoidBCELoss()
```

استرجع وصفنا لمتغير القناع mask ومتغير التسمية labelفي القسم 15.3.5. يحسب التالي الخسارة الثنائية عبر الانتروبيا للمتغيرات المحددة.

```
pred = np.array([[1.1, -2.2, 3.3, -4.4]] * 2)
label = np.array([[1.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0,
0.0]])
mask = np.array([[1, 1, 1, 1], [1, 1, 0, 0]])
loss(pred, label, mask) * mask.shape[1] /
mask.sum(axis=1)
```

```
array([0.93521017, 1.8462094 ])
```

يوضح أدناه كيفية حساب النتائج المذكورة أعلاه (بطريقة أقل كفاءة) باستخدام دالة التنشيط sigmoidفي خسارة الانتروبيا الثنائية. يمكننا اعتبار المخرجين كخسارتين طبيعيتين يتم حساب متوسطهما على التنبؤات غير المقنعة.

```
def sigmd(x):
    return -math.log(1 / (1 + math.exp(-x)))

print(f'{(sigmd(1.1) + sigmd(2.2) + sigmd(-3.3) +
sigmd(4.4)) / 4:.4f}')
print(f'{(sigmd(-1.1) + sigmd(-2.2)) / 2:.4f}')
```

```
0.9352
1.8462
```

15.4.2.2. تهيئة معلمات النموذج Initializing Model Parameters

نحدد طبقتين من طبقات التضمين لجميع الكلماتفي المفردات عند استخدامهاككلمات مركزية وكلمات سياق، على التوالي. تم تعيين بُعد متجه الكلمة `embed_size` على 100.

```
embed_size = 100
net = nn.Sequential()
net.add(nn.Embedding(input_dim=len(vocab),
output_dim=embed_size),
        nn.Embedding(input_dim=len(vocab),
output_dim=embed_size))
```

15.4.2.3. تعريف حلقة التدريب Defining the Training Loop

تم تعريف حلقة التدريب أدناه. بسبب وجود الحشو، يختلف حساب دالة الخسارة قليلاً مقارنة بدوال التدريب السابقة.

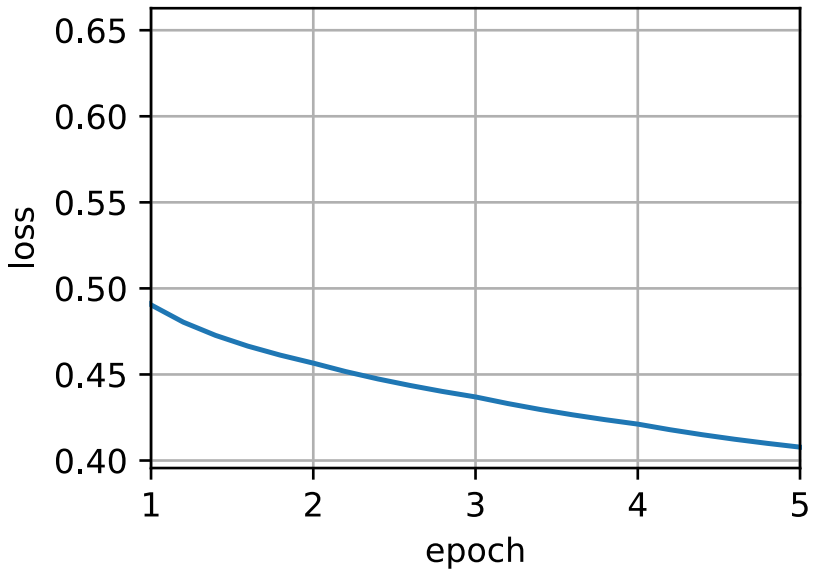
```
def train(net, data_iter, lr, num_epochs,
device=d2l.try_gpu()):
    net.initialize(ctx=device, force_reinit=True)
    trainer = gluon.Trainer(net.collect_params(),
'adam',
                            {'learning_rate': lr})
    animator = d2l.Animator(xlabel='epoch',
ylabel='loss',
                            xlim=[1, num_epochs])
    # Sum of normalized losses, no. of normalized losses
    metric = d2l.Accumulator(2)
    for epoch in range(num_epochs):
        timer, num_batches = d2l.Timer(), len(data_iter)
        for i, batch in enumerate(data_iter):
            center, context_negative, mask, label = [
                data.as_in_ctx(device) for data in
batch]
            with autograd.record():
                pred = skip_gram(center,
context_negative, net[0], net[1])
                l = (loss(pred.reshape(label.shape),
label, mask) *
                     mask.shape[1] / mask.sum(axis=1))
            l.backward()
            trainer.step(batch_size)
            metric.add(l.sum(), l.size)
            if (i + 1) % (num_batches // 5) == 0 or i ==
num_batches - 1:
                animator.add(epoch + (i + 1) /
num_batches,
                             (metric[0] / metric[1],))
```

```
    print(f'loss {metric[0] / metric[1]:.3f}, '
          f'{metric[1] / timer.stop():.1f} tokens/sec on
{str(device)}')
```

الآن يمكننا تدريب نموذج تخطي-جرام باستخدام اخذ العينات السلبية.

```
lr, num_epochs = 0.002, 5
train(net, data_iter, lr, num_epochs)
```

```
loss 0.408, 103058.8 tokens/sec on gpu(0)
```



15.4.3. تطبيق تضمينات الكلمات Applying Word Embeddings

بعد تدريب نموذج word2vec، يمكننا استخدام تشابه جيب التمام لمتجهات الكلمات من النموذج المدرب للعثور على الكلمات من القاموس الأكثر تشابهًا دلالياً مع كلمة الإدخال.

```
def get_similar_tokens(query_token, k, embed):
    W = embed.weight.data()
    x = W[vocab[query_token]]
    # Compute the cosine similarity. Add 1e-9 for
numerical stability
    cos = np.dot(W, x) / np.sqrt(np.sum(W * W, axis=1) *
np.sum(x * x) + 1e-9)
    topk = npx.topk(cos, k=k+1,
ret_typ='indices').asnumpy().astype('int32')
    for i in topk[1:]:  # Remove the input words
```

```
        print(f'cosine sim={float(cos[i]):.3f}:
{vocab.to_tokens(i)}')

get_similar_tokens('chip', 3, net[0])
```

```
cosine sim=0.665: intel
cosine sim=0.623: graphics
cosine sim=0.615: microprocessor
```

15.4.4. الملخص

- يمكننا تدريب نموذج تخطي-جرام بأخذ عينات سالبة باستخدام طبقات التضمين وخسارة الانتروبيا الثنائية.
- تتضمن تطبيقات تضمينات الكلمات العثور على كلمات متشابهة لغويًا لكلمة معينة بناءً على تشابه جيب التمام لمتجهات الكلمات.

15.4.5. التمارين

1. باستخدام النموذج المدرب، ابحث عن كلمات متشابهة لغويًا لكلمات الإدخال الأخرى. هل يمكنك تحسين النتائج بضبط المعلمات الفائقة؟
2. عندما تكون مجموعة التدريب ضخمة ، فإننا غالبًا ما نقوم بأخذ عينات من كلمات السياق وكلمات الضوضاء للكلمات المركزية في الدفعة الصغيرة الحالية عند تحديث معلمات النموذج. بمعنى آخر ، قد تحتوي نفس الكلمة المركزية على كلمات سياق مختلفة أو كلمات ضوضاء في فترات تدريب مختلفة. ما هي فوائد هذه الطريقة؟ حاول تنفيذ طريقة التدريب هذه.

15.5. تضمين الكلمة مع المتجهات العالمية (GloVe) Word Embedding with Global Vectors

قد يحمل التواجد المشترك للكلمة داخل إطارات السياق معلومات دلالية غنية. على سبيل المثال، في مجموعة كبيرة من المرجح أن تتزامن كلمة "solid" مع "ice" أكثر من كلمة "steam"، لكن كلمة "gas" ربما تتزامن مع "steam" أكثر من كلمة "ice". إلى جانب ذلك، يمكن حساب إحصائيات المجموعة العالمية global corpus statistics لمثل هذه co-occurrences: يمكن أن يؤدي ذلك إلى تدريب أكثر كفاءة. للاستفادة من المعلومات الإحصائيةفي المجموعة الكاملة لتضمين الكلمات، دعنا أولاً نعيد النظرفي نموذج تخطي ـ جرام skip-gramفي القسم 15.1.3، ولكن تفسيره باستخدام إحصائيات المجموعة العالمية مثل عدد مرات التواجد المشترك co-occurrence counts.

15.5.1. تخطي-جرام مع إحصائيات المجموعة العالمية Skip-Gram with Global Corpus Statistics

للدلالة على q_{ij} الاحتمال الشرطي $P(w_j \mid w_i)$ لكلمة w_j كلمة معينة w_i في نموذج تخطي-جرام، لدينا

$$q_{ij} = \frac{\exp(\mathbf{u}_j^\top \mathbf{v}_i)}{\sum_{k \in \mathcal{V}} \exp(\mathbf{u}_k^\top \mathbf{v}_i)},$$

حيث $\mathbf{v}_i$ تمثل أي متجهات للفهرس i و$\mathbf{u}_i$ تمثل الكلمة w_i على أنها الكلمة المركزية وكلمة السياق، على التوالي، و$\mathcal{V} = \{0,1,\ldots,|\mathcal{V}|-1\}$ هي مجموعة الفهرس للمفردات.

ضعفي اعتبارك الكلمة w_i التي قد ترد عدة مراتفي المجموعة corpus.في المجموعة بأكملها، تشكل جميع كلمات السياق حيثما يتم أخذ w_i على أنها الكلمة المركزية الخاصة بها مجموعة متعددة (multiset) $\mathcal{C}_i$ من مؤشرات من الكلمات التي تسمح بمثيلات متعددة من نفس العنصر. بالنسبة لأي عنصر، يُطلق على عدد مثيلاته التعددية multiplicity. للتوضيح بمثال، افترض أن الكلمة w_i تحدث مرتينفي مجموعة النصوص وأن مؤشرات كلمات السياق تأخذ w_i والتي تتخذ مركزًا لهافي نافذتي السياق هي k,j,m,k وk,l,k,j . وبالتالي، المجموعة المتعددة $\mathcal{C}_i = \{j,j,k,k,k,k,l,m\}$ ، حيث تكون تعدد multiplicities العناصر j,k,l,m هو 2، 4، 1، 1، على التوالي.

الآن دعنا نشير إلى تعدد العناصر jفي المجموعة المتعددة $\mathcal{C}_i$ كـx_{ij} . هذا هو عدد التواجد المشترك العالمي للكلمة w_j (ككلمة السياق) والكلمة w_i (مثل الكلمة المركزية)في نفس نافذة السياقفي المجموعة بأكملها. باستخدام مثل هذه الإحصائيات العالمية، فإن دالة الخسارةفي نموذج تخطي ـ جرام تعادل

$$-\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ij} \log q_{ij}.$$

نشير أيضًا إلى x_i عدد كل كلمات السياقفي نوافذ السياق حيث w_i تحدث ككلمة مركزية، والتي تعادل $|\mathcal{C}_i|$. ليكن p_{ij} الاحتمال الشرطي x_{ij}/x_i لتوليد كلمة سياق w_j معينة كلمة مركزية w_i، (15.5.2) يمكن إعادة كتابتها كـ

$$-\sum_{i \in \mathcal{V}} x_i \sum_{j \in \mathcal{V}} p_{ij} \log q_{ij}.$$

في (15.5.3)، $-\sum_{j\in\mathcal{V}} p_{ij}\log q_{ij}$ تحسب الانتروبيا المتقاطعة للتوزيع الشرطي p_{ij} لإحصائيات المجموعة العالمية والتوزيع الشرطي q_{ij} لتنبؤات النموذج. يتم اخذ وزن هذه الخطأ أيضًا بواسطة x_i موضح أعلاه. سيسمح تقليل دالة الخسارةفي (15.5.3) للتوزيع الشرطي المتوقع بالاقتراب من التوزيع الشرطي من إحصائيات المجموعة العالمية.

على الرغم من استخدامها بشكل شائع لقياس المسافة بين التوزيعات الاحتمالية، إلا أن دالة خسارة الانتروبيا المتقاطعة قد لا تكون خيارًا جيدًا هنا. من ناحية، كما ذكرنافي القسم 15.2، ينتج عن تكلفة التسوية q_{ij} بشكل صحيح مجموع المفردات بأكملها، والذي يمكن أن يكون مكلفًا من الناحية الحسابية. من ناحية أخرى، غالبًا ما يتم تصميم عدد كبير من الأحداث النادرة من مجموعة كبيرة من خلال خسارة الانتروبيا المتقاطعة ليتم تخصيصها بوزن كبير.

15.5.2. نموذج GloVe The GloVe Model

في ضوء ذلك، يقوم نموذج GloVe بإجراء ثلاث تغييرات على نموذج تخطي-جرام بناءً على الخسارة التربيعية (Pennington et al.، 2014):

1. استخدم المتغيرات $p'_{ij} = x_{ij}$ و$q'_{ij} = \exp(\mathbf{u}_j^\top \mathbf{v}_i)$ التي ليست توزيعات احتمالية وتأخذ لوغاريتم كلاهما ، لذا فإن مصطلح الخسارة التربيعية هو
$$(\log p'_{ij} - \log q'_{ij})^2 = (\mathbf{u}_j^\top \mathbf{v}_i - \log x_{ij})^2$$
2. أضف معلمتين للنموذج العددي لكل كلمة w_i: تحيز الكلمة المركزية b_i وتحيز كلمة السياق c_i.
3. استبدل وزن كل مصطلح خسارة بدالة الوزن $h(x_{ij})$ ، حيث $h(x)$ يزداد في الفترة $[0,1]$.

بتجميع كل الأشياء معًا، فإن تدريب GloVe هو تقليل دالة الخسارة التالية:

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} h(x_{ij})(\mathbf{u}_j^\top \mathbf{v}_i + b_i + c_j - \log x_{ij})^2.$$

بالنسبة لدالة الوزن، فإن الخيار المقترح هو: $h(x) = (x/c)^\alpha$ (على سبيل المثال $\alpha = 0.75$) إذا $x < c$ (على سبيل المثال $c = 100$)؛ خلاف ذلك $h(x) = 1$.في هذه الحالة، لأن $h(0) = 0$ لأي $x_{ij} = 0$ يمكن حذف مصطلح الخسارة التربيعية من الكفاءة الحسابية. على سبيل المثال، عند استخدام الانحدار الاشتقاقي العشوائي المصغر minibatch SGD للتدريب، في كل تكرار نقوم بشكل عشوائي بأخذ عينة صغيرة من غير الصفر لحساب الانحدارات وتحديث معلمات النموذج. لاحظ أن هذه العناصر غير الصفرية هي إحصائيات مجمعة عالمية مسبقة

الحساب؛ وبالتالي، يسمى النموذج GloVe للمتجهات العالمية GloVe for Global Vectors.

يجب التأكيد على أنه إذا ظهرت الكلمة w_iفي نافذة سياق الكلمة w_j، فعندئذٍ العكس بالعكس. وبالتالي،$x_{ij} = x_{ji}$. على عكس word2vec الذي يناسب الاحتمال الشرطي غير المتماثل p_{ij}، فإن GloVe يناسب المتماثل $\log x_{ij}$. لذلك، فإن متجه الكلمات المركزية ومتجه كلمات السياق لأي كلمة مكافئين رياضيًافي نموذج GloVe. ومع ذلك، من الناحية العملية، نظرًا لقيم التهيئة المختلفة، قد تظل الكلمة نفسها تحصل على قيم مختلفةفي هذين المتجهين بعد التدريب: يلخصها GloVe على أنها متجه الإخراج.

15.5.3. تفسير GloVe من نسبة احتمالات التواجد المشترك Interpreting GloVe from the Ratio of Co-occurrence Probabilities

يمكننا أيضًا تفسير نموذج GloVe من منظور آخر. باستخدام نفس الترميزفي القسم 15.5.1، ليكن $p_{ij} \stackrel{\text{def}}{=} P(w_j \mid w_i)$ هو الاحتمال الشرطي لتوليد كلمة السياق w_j المعطاة w_i باعتبارها الكلمة المركزيةفي المجموعة. يسرد القسم 15.5.3 العديد من احتمالات التواجد المشترك مع الأخذفي الاعتبار الكلمات "ice" و "steam" ونسبها بناءً على إحصائيات من مجموعة كبيرة.

w_k	solid	gas	water	fashion
$p_1 = P(w_k \mid \text{ice})$	0.00019	0.000066	0.003	0.000017
$p_2 = P(w_k \mid \text{steam})$	0.000022	0.00078	0.0022	0.000018
p_1/p_2	8.9	0.085	1.36	0.96

الجدول: احتمالات التواجد المشترك للكلمة_الكلمة ونسبها من مجموعة كبيرة (مقتبس من الجدول 1في Pennington et al. (2014))

يمكننا ملاحظة ما يلي من القسم 15.5.3:

- بالنسبة للكلمة w_k التي ترتبط بـ "ice" ولكنها غير مرتبطة بـ "steam"، مثل $w_k =$ solid، نتوقع نسبة أكبر من احتمالات التواجد المشترك ، مثل 8.9.

- بالنسبة للكلمة w_k المرتبطة بـ "steam" ولكنها غير مرتبطة بـ "ice"، مثل $w_k = \text{gas}$، نتوقع نسبة أصغر من احتمالات التواجد المشترك ، مثل 0.085.
- بالنسبة للكلمة w_k التي ترتبط بكل من "ice" و "steam" ، مثل $w_k = \text{water}$، نتوقع نسبة احتمالات التواجد المشترك قريبة من 1 ، مثل 1.36.
- بالنسبة للكلمة w_k التي لا علاقة لها بكل من "ice" و "steam"، مثل ، نتوقع نسبة احتمالات التواجد المشترك قريبة من 1 ، مثل 0.96.

يمكن ملاحظة أن نسبة احتمالات التواجد المشترك يمكن أن تعبر بشكل حدسي عن العلاقة بين الكلمات. وبالتالي، يمكننا تصميم دالة من متجهات من ثلاث كلمات لتناسب هذه النسبة. بالنسبة لنسبة احتمالات التكرار p_{ij}/p_{ik} مع كونها الكلمة المركزية w_j وكونها كلمات السياق w_k، نريد أن نلائم هذه النسبة باستخدام بعض الدوال f:

$$f(\mathbf{u}_j, \mathbf{u}_k, \mathbf{v}_i) \approx \frac{p_{ij}}{p_{ik}}.$$

من بين العديد من التصميمات الممكنة لـ f ، نختار فقط خيارًا معقولًا فيما يلي. نظرًا لأن نسبة احتمالات التكرار هي قيمة قياسية scalar، فنحن نطلب أن تكون f دالة عددية، مثل $f(\mathbf{u}_j, \mathbf{u}_k, \mathbf{v}_i) = f((\mathbf{u}_j - \mathbf{u}_k)^\top \mathbf{v}_i)$. تبديل مؤشرات الكلمات j وkفي (15.5.5)، يجب أن يتم الاحتفاظ بـ $f(x)f(-x) = 1$ ، لذا فإن أحد الاحتمالات هو $f(x) = \exp(x)$، على سبيل المثال،

$$f(\mathbf{u}_j, \mathbf{u}_k, \mathbf{v}_i) = \frac{\exp(\mathbf{u}_j^\top \mathbf{v}_i)}{\exp(\mathbf{u}_k^\top \mathbf{v}_i)} \approx \frac{p_{ij}}{p_{ik}}.$$

الآن دعنا نختار $\exp(\mathbf{u}_j^\top \mathbf{v}_i) \approx \alpha p_{ij}$، حيث α ثابت. لان $p_{ij} = x_{ij}/x_i$، بعد أخذ اللوغاريتم على كلا الجانبين نحصل على $\mathbf{u}_j^\top \mathbf{v}_i \approx \log\alpha + \log x_{ij} - \log x_i$. قد نستخدم مصطلحات تحيز إضافية لتلائم $-\log\alpha + \log x_i$، مثل انحياز الكلمة المركزية b_i وتحيز كلمة السياق c_j:

$$\mathbf{u}_j^\top \mathbf{v}_i + b_i + c_j \approx \log x_{ij}. \quad (15.5.7)$$

قياس الخطأ التربيعي (15.5.7) بالأوزان، يتم الحصول على دالة خسارة GloVe في (15.5.4).

15.5.4. الملخص

- يمكن تفسير نموذج تخطي-جرام skip-gram باستخدام إحصائيات المجموعة العالمية global corpus statistics مثل عدد التكرار المشترك للكلمة word-word co-occurrence counts.

- قد لا تكون خسارة الانتروبيا خيارًا جيدًا لقياس الفرق بين توزيعين احتماليين ، خاصة لمجموعة كبيرة. يستخدم GloVe الخسارة التربيعية لملاءمة إحصائيات المجموعة العالمية مسبقة الحوسبة.
- متجه الكلمات المركزي ومتجه الكلمات السياق متكافئان رياضيًا لأي كلمة في GloVe.
- يمكن تفسير GloVe من نسبة احتمالات التواجد المشترك للكلمة_ الكلمة.

15.5.5. التمارين

1. إذا كانت الكلمات w_i وw_j تحدث بشكل مشترك في نفس نافذة السياق ،كيف يمكننا استخدام المسافة بينهما في متسلسلة النص لإعادة تصميم طريقة حساب الاحتمال الشرطي p_{ij}؟ تلميح: انظر القسم 4.2 من مقالة GloVe، (Pennington et al.,2014).
2. لأي كلمة، هل تحيز الكلمة المركزية وتحيز كلمة السياق متكافئان رياضياً في GloVe؟ لماذا؟

15.6 تضمين الكلمات الفرعية Subword Embedding

في اللغة الإنجليزية، كلمات مثل "helps" و "helped" و "helping" هي أشكال تصريفية لنفس الكلمة "help". العلاقة بين "dog" و "dogs" هي نفسها العلاقة بين "cat" و "cats"، والعلاقة بين "boy" و "boyfriend" هي نفسها العلاقة بين "girl" و "girlfriend".في لغات أخرى مثل الفرنسية والإسبانية، تحتوي العديد من الأفعال على أكثر من 40 صيغة تصريف، بينمافي الفنلندية، قد يصل الاسم إلى 15 حالة.في علم اللغة linguistics، تدرس علم التصريف morphology تكوين الكلمات والعلاقات بين الكلمات. ومع ذلك، لم يتم استكشاف البُنية الداخلية للكلماتفي word2vec ولافي GloVe.

15.6.1. نموذج fastText The fastText Model

تذكر كيف يتم تمثيل الكلماتفي word2vec.في كل من نموذج تخطي_جرام skip-gram ونموذج حقيبة الكلمات المستمر continuous bag-of-words، يتم تمثيل أشكال مختلفة من نفس الكلمة مباشرة بواسطة متجهات مختلفة بدون معلمات مشتركة. لاستخدام المعلومات المورفولوجية morphological information، اقترح نموذج FastText طريقة تضمين الكلمات الفرعية subword embedding، حيث تكون الكلمة الفرعية عبارة عن n_غرام (Bojanowski et al.، 2017). بدلاً من تعلم تمثيلات المتجهات على مستوى الكلمات، يمكن اعتبار fastText بمثابة تخطي_جرام على مستوى الكلمات الفرعية، حيث يتم تمثيل كل كلمة مركزية بمجموع متجهات الكلمات الفرعية الخاصة بها.

دعنا نوضح كيفية الحصول على كلمات فرعية لكل كلمة مركزيةفي fastText باستخدام كلمة "where". أولاً، أضف الأحرف الخاصة ">" and "<"في بداية الكلمة ونهايتها لتمييز البادئات prefixes واللواحق suffixes عن الكلمات الفرعية الأخرى. بعد ذلك، استخرج الأحرف n-grams من الكلمة. على سبيل المثال، عندما $n = 3$ نحصل على جميع الكلمات الفرعية ذات الطول 3: "<wh", "whe", "her", "ere", "re>" ، والكلمة الفرعية الخاصة "<where>".

في FastText، لأي كلمة w ، يُرمز إليها بواسطة $\mathcal{G}_w$ اتحاد كل كلماتها الفرعية التي يتراوح طولها بين 3 و 6 وكلماتها الفرعية الخاصة. المفردات هي اتحاد الكلمات الفرعية لكل الكلمات. إذا فرضنا $\mathbf{z}_g$ متجه الكلمة الفرعية gفي القاموس، فإن متجه الكلمة $\mathbf{v}_w$ ككلمة مركزية wفي نموذج تخطي–جرام هو مجموع متجهات الكلمات الفرعية الخاصة به:

$$\mathbf{v}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g.$$

ما تبقى من fastText هو نفس نموذج تخطي–جرام. بالمقارنة مع نموذج تخطي–جرام، تكون المفرداتفي fastText أكبر، مما يؤدي إلى مزيد من معلمات النموذج. بالإضافة إلى ذلك، لحساب تمثيل كلمة ما، يجب تلخيص جميع متجهات الكلمات الفرعية الخاصة بها، مما يؤدي إلى زيادة التعقيد الحسابي. ومع ذلك، بفضل المعلمات المشتركة من الكلمات الفرعية بين الكلمات ذات الهياكل المتشابهة، قد تحصل الكلمات النادرة وحتى الكلمات غير المفردات على تمثيلات متجهية أفضلفي fastText.

15.6.2. ترميز Byte Pair Byte Pair Encoding

في fastText، يجب أن تكون جميع الكلمات الفرعية المستخرجة ذات أطوال محددة، مثل 3 الى 6 ، وبالتالي لا يمكن تحديد حجم المفردات مسبقًا. للسماح بكلمات فرعية متغيرة الطول في مفردات ذات حجم ثابت، يمكننا تطبيق خوارزمية ضغط تسمى byte pair encoding (BPE) لاستخراج كلمات فرعية (Sennrich et al.، 2015).

يقوم ترميز Byte pair بإجراء تحليل إحصائي لمجموعة بيانات التدريب لاكتشاف الرموز الشائعة داخل الكلمة، مثل الأحرف المتتالية ذات الطول التعسفي. بدءًا من الرموز ذات الطول 1، يدمج ترميز Byte pair بشكل متكرر الزوج الأكثر شيوعًا من الرموز المتتالية لإنتاج رموز أطول جديدة. لاحظ أنه من أجل الكفاءة، لا يتم النظرفي الأزواج التي تتخطى حدود الكلمات.في النهاية، يمكننا استخدام رموز مثل الكلمات الفرعية لتقسيم الكلمات. تم استخدام ترميز Byte pair ومتغيراته لتمثيل المدخلاتفي نماذج التدريب المسبق للمعالجة اللغوية الطبيعية الشائعة

مثل GPT-2، (2019,Radford et al.) وRoBERTa، (2019,Liu et al.). فيما يلي، سنوضح كيفية عمل ترميز Byte pair.

أولاً، نقوم بتهيئة مفردات الرموز مثل جميع الأحرف الإنجليزية الصغيرة، ورمز خاص في نهاية الكلمة `'_'` ، ورمز خاص غير معروف `'[UNK]'`.

```
import collections

symbols = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
'j', 'k', 'l', 'm',
           'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z',
           '_', '[UNK]']
```

نظرًا لأننا لا نأخذ في الاعتبار أزواج الرموز التي تتخطى حدود الكلمات، فإننا نحتاج فقط إلى قاموس `raw_token_freqs` يقوم بتعيين الكلمات لترددتها (عدد مرات الحدوث) في مجموعة بيانات. لاحظ أن الرمز الخاص `'_'` يتم إلحاقه بكل كلمة حتى نتمكن بسهولة من استعادة تسلسل الكلمات (على سبيل المثال، "a taller man") من سلسلة من رموز الإخراج (على سبيل المثال، "a_ long er_ man"). نظرًا لأننا نبدأ عملية الدمج من مفردات الأحرف الفردية والرموز الخاصة فقط، يتم إدخال مسافة بين كل زوج من الأحرف المتتالية داخل كل كلمة (مفاتيح القاموس `token_freqs`). بمعنى آخر، المسافة هي الفاصل بين الرموز داخل الكلمة.

```
raw_token_freqs = {'fast_': 4, 'faster_': 3, 'tall_': 5,
'taller_': 4}
token_freqs = {}
for token, freq in raw_token_freqs.items():
    token_freqs[' '.join(list(token))] =
raw_token_freqs[token]
token_freqs
```

```
{'f a s t _': 4, 'f a s t e r _': 3, 't a l l _': 5, 't
a l l e r _': 4}
```

نحدد دالة `get_max_freq_pair` التالية التي تُرجع الزوج الأكثر شيوعًا من الرموز المتتالية داخل الكلمة ، حيث تأتي الكلمات من مفاتيح قاموس الإدخال `token_freqs`.

```
def get_max_freq_pair(token_freqs):
    pairs = collections.defaultdict(int)
    for token, freq in token_freqs.items():
        symbols = token.split()
```

```
        for i in range(len(symbols) - 1):
            # Key of `pairs` is a tuple of two
consecutive symbols
            pairs[symbols[i], symbols[i + 1]] += freq
    return max(pairs, key=pairs.get)  # Key of `pairs`
with the max value
```

كنهج جشع يعتمد على تكرار الرموز المتتالية، سيستخدم ترميز byte pair دالة `merge_symbols` التالية لدمج الزوج الأكثر شيوعًا من الرموز المتتالية لإنتاج رموز جديدة.

```
def merge_symbols(max_freq_pair, token_freqs, symbols):
    symbols.append(''.join(max_freq_pair))
    new_token_freqs = dict()
    for token, freq in token_freqs.items():
        new_token = token.replace('
'.join(max_freq_pair),

''.join(max_freq_pair))
        new_token_freqs[new_token] = token_freqs[token]
    return new_token_freqs
```

الآن نقوم بإجراء خوارزمية ترميز byte pair بشكل متكرر على مفاتيح القاموس `token_freqs`.في التكرار الأول، الزوج الأكثر شيوعًا من الرموز المتتالية هما `'t'` و `'a'`، وبالتالي يدمجهما ترميز byte pair لإنتاج رمز جديد `'ta'`.في التكرار الثاني، يستمر ترميز byte pairفي دمج `'ta'` و `'l'` لينتج عن رمز جديد آخر `'tal'`.

```
num_merges = 10
for i in range(num_merges):
    max_freq_pair = get_max_freq_pair(token_freqs)
    token_freqs = merge_symbols(max_freq_pair,
token_freqs, symbols)
    print(f'merge #{i + 1}:', max_freq_pair)
```

```
merge #1: ('t', 'a')
merge #2: ('ta', 'l')
merge #3: ('tal', 'l')
merge #4: ('f', 'a')
merge #5: ('fa', 's')
merge #6: ('fas', 't')
merge #7: ('e', 'r')
merge #8: ('er', '_')
merge #9: ('tall', '_')
merge #10: ('fast', '_')
```

بعد 10 تكرارات من ترميز byte pair، يمكننا أن نرى أن رموز القائمة تحتوي الآن على 10 رموز أخرى يتم دمجها بشكل متكرر من رموز أخرى.

```
print(symbols)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z', '_', '[UNK]', 'ta', 'tal', 'tall',
'fa', 'fas', 'fast', 'er', 'er_', 'tall_', 'fast_']
```

بالنسبة لمجموعة البيانات نفسها المحددةفي مفاتيح القاموس raw_token_freqs، يتم الآن تقسيم كل كلمةفي مجموعة البيانات حسب الكلمات الفرعية "fast_" و "fast" و "er_" و "tall_" و "tall" نتيجة خوارزمية ترميز byte pair. على سبيل المثال، يتم تجزئة الكلمتين "faster_" و "taller_" إلى "fast er_" و "tall er_"، على التوالي.

```
print(list(token_freqs.keys()))
```

```
['fast_', 'fast er_', 'tall_', 'tall er_']
```

لاحظ أن نتيجة ترميز byte pair يعتمد على مجموعة البيانات المستخدمة. يمكننا أيضًا استخدام الكلمات الفرعية التي تم تعلمها من مجموعة بيانات واحدة لتقسيم كلمات مجموعة بيانات أخرى. كنهج جشع، تحاول دالة `segment_BPE` التالية تقسيم الكلمات إلى أطول كلمات فرعية ممكنة من رموز وسيطة الإدخال.

```
def segment_BPE(tokens, symbols):
    outputs = []
    for token in tokens:
        start, end = 0, len(token)
        cur_output = []
        # Segment token with the longest possible
subwords from symbols
        while start < len(token) and start < end:
            if token[start: end] in symbols:
                cur_output.append(token[start: end])
                start = end
                end = len(token)
            else:
                end -= 1
        if start < len(token):
            cur_output.append('[UNK]')
        outputs.append(' '.join(cur_output))
    return outputs
```

فيما يلي، نستخدم الكلمات الفرعيةفي القائمة symbols، والتي يتم تعلمها من مجموعة البيانات المذكورة أعلاه، لتقسيم `tokens` التي تمثل مجموعة بيانات أخرى.

```
tokens = ['tallest_', 'fatter_']
print(segment_BPE(tokens, symbols))
```

```
['tall e s t _', 'fa t t er_']
```

15.6.3. الملخص

- يقترح نموذج FastText طريقة تضمين الكلمات الفرعية subword embedding approach. استنادًا إلى نموذج تخطي-جرام في word2vec، فإنه يمثل كلمة مركزية كمجموع متجهات الكلمات الفرعية الخاصة بها.
- يقوم ترميز Byte pair بإجراء تحليل إحصائي لمجموعة بيانات التدريب لاكتشاف الرموز الشائعة common symbols داخل الكلمة. كنهج جشع ، يدمج ترميز Byte pair بشكل متكرر الزوج الأكثر شيوعًا من الرموز المتتالية.
- قد يؤدي تضمين الكلمات الفرعية إلى تحسين جودة تمثيلات الكلمات النادرة والكلمات خارج القاموس.

15.6.4. التمارين

1. على سبيل المثال ، هناك حوالي 3×10^8 6-جرام ممكنة في اللغة الإنجليزية. ما هي المشكلة عندما يكون هناك الكثير من الكلمات الفرعية؟ كيف تعالج المشكلة؟ تلميح: راجع نهاية القسم 3.2 من مقالة fastText (Bojanowski et al.، 2017).
2. كيف يتم تصميم نموذج تضمين الكلمات الفرعية بناءً على نموذج حقيبة الكلمات المستمرة؟
3. للحصول على مفردات ذات الحجم m ، كم عدد عمليات الدمج المطلوبة عندما يكون حجم مفردات الرمز الأولي n ؟
4. كيف تمتد فكرة ترميز byte pair لاستخراج العبارات extract phrases؟

15.7. تشابه الكلمات وقياسها Word Similarity and Analogy

في القسم 15.4، قمنا بتدريب نموذج word2vec على مجموعة بيانات صغيرة، وقمنا بتطبيقه للعثور على كلمات متشابهة دلالياً لكلمة إدخال.في الممارسة العملية، يمكن تطبيق متجهات الكلمات التي تم تدريبها مسبقًافي مجموعات كبيرة على مهام معالجة اللغة الطبيعية اللاحقة، والتي سيتم تناولها لاحقًافي القسم 16. لإثبات دلالات متجهات الكلمات سابقة التدريب من مجموعات كبيرة بطريقة مباشرة، دعنا نطبقهافي مهام تشابه الكلمات وقياسها.

```
import os
```

```
from mxnet import np, npx
from d2l import mxnet as d2l

npx.set_np()
```

15.7.1. تحميل متجهات الكلمات المدربة مسبقًا Loading Pretrained Word Vectors

فيما يلي قوائم بتضمينات GloVe مدربة مسبقاً ذات أبعاد 50 و100 و300، والتي يمكن تنزيلها من موقع GloVe على الويب. تتوفر تضمينات fastText المدربة مسبقًا بعدة لغات. هنا نعتبر نسخة إنجليزية واحدة (300-dimensional "wiki.en") يمكن تحميلها من موقع fastText.

```
#@save
d2l.DATA_HUB['glove.6b.50d'] = (d2l.DATA_URL +
'glove.6B.50d.zip',

'0b8703943ccdb6eb788e6f091b8946e82231bc4d')

#@save
d2l.DATA_HUB['glove.6b.100d'] = (d2l.DATA_URL +
'glove.6B.100d.zip',

'cd43bfb07e44e6f27cbcc7bc9ae3d80284fdaf5a')

#@save
d2l.DATA_HUB['glove.42b.300d'] = (d2l.DATA_URL +
'glove.42B.300d.zip',

'b5116e234e9eb9076672cfeabf5469f3eec904fa')

#@save
d2l.DATA_HUB['wiki.en'] = (d2l.DATA_URL + 'wiki.en.zip',

'c1816da3821ae9f43899be655002f6c723e91b88')
```

`TokenEmbedding` لتحميل تضمينات GloVe و fastText المدربة مسبقاً هذه، نحدد فئة التالية.

```
#@save
class TokenEmbedding:
    """Token Embedding."""
    def __init__(self, embedding_name):
```

```
        self.idx_to_token, self.idx_to_vec =
self._load_embedding(
            embedding_name)
        self.unknown_idx = 0
        self.token_to_idx = {token: idx for idx, token
in

enumerate(self.idx_to_token)}

    def _load_embedding(self, embedding_name):
        idx_to_token, idx_to_vec = ['<unk>'], []
        data_dir = d2l.download_extract(embedding_name)
        # GloVe website:
https://nlp.stanford.edu/projects/glove/
        # fastText website: https://fasttext.cc/
        with open(os.path.join(data_dir, 'vec.txt'),
'r') as f:
            for line in f:
                elems = line.rstrip().split(' ')
                token, elems = elems[0], [float(elem)
for elem in elems[1:]]
                # Skip header information, such as the
top row in fastText
                if len(elems) > 1:
                    idx_to_token.append(token)
                    idx_to_vec.append(elems)
        idx_to_vec = [[0] * len(idx_to_vec[0])] +
idx_to_vec
        return idx_to_token, np.array(idx_to_vec)

    def __getitem__(self, tokens):
        indices = [self.token_to_idx.get(token,
self.unknown_idx)
                   for token in tokens]
        vecs = self.idx_to_vec[np.array(indices)]
        return vecs

    def __len__(self):
        return len(self.idx_to_token)
```

أدناه نقوم بتحميل تضمينات GloVe 50 بعدًا (تم اختبارها مسبقًا على مجموعة فرعية من Wikipedia). عند إنشاء مثيل `TokenEmbedding`، يجب تنزيل ملف التضمين المحدد إذا لم يكن قد تم تنزيله بعد.

```
glove_6b50d = TokenEmbedding('glove.6b.50d')
```

```
Downloading ../data/glove.6B.50d.zip from http://d2l-
data.s3-accelerate.amazonaws.com/glove.6B.50d.zip...
```

إخراج حجم المفردات. تحتوي المفردات على 400000 كلمة (الرموز tokens) ورمز غير معروف.

```
len(glove_6b50d)
```

```
400001
```

يمكننا الحصول على فهرس الكلمةفي المفردات والعكس صحيح.

```
glove_6b50d.token_to_idx['beautiful'],
glove_6b50d.idx_to_token[3367]
```

```
(3367, 'beautiful')
```

15.7.2. تطبيق متجهات الكلمات المدربة مسبقًا Applying Pretrained Word Vectors

باستخدام متجهات GloVe المحملة، سوف نوضح دلالاتها من خلال تطبيقهافي مهام التشابه والقياسفي الكلمة word similarity and analogy التالية.

15.7.2.1. تشابه الكلمات Word Similarity

على غرار القسم 15.4.3، من أجل العثور على كلمات متشابهة دلالياً لكلمة إدخال بناءً على أوجه التشابه بين جيب التمام بين متجهات الكلمات، نقوم بتنفيذ دالة knn التالية (k أقرب جيران).

```
def knn(W, x, k):
    # Add 1e-9 for numerical stability
    cos = np.dot(W, x.reshape(-1,)) / (
        np.sqrt(np.sum(W * W, axis=1) + 1e-9) *
np.sqrt((x * x).sum()))
    topk = npx.topk(cos, k=k, ret_typ='indices')
    return topk, [cos[int(i)] for i in topk]
```

بعد ذلك، نبحث عن كلمات مماثلة باستخدام متجهات الكلمات المدربة مسبقًا من تضمين `embed` مثيل `TokenEmbedding`.

```
def get_similar_tokens(query_token, k, embed):
```

```
    topk, cos = knn(embed.idx_to_vec,
embed[[query_token]], k + 1)
    for i, c in zip(topk[1:], cos[1:]):  # Exclude the
input word
        print(f'cosine sim={float(c):.3f}:
{embed.idx_to_token[int(i)]}')
```

تحتوي مفردات متجهات الكلمات المدربة مسبقًا في `glove_6b50d` على 400000 كلمة ورمز غير معروف. باستثناء كلمة الإدخال والرمز غير المعروف، فلنجد من بين هذه المفردات ثلاث كلمات متشابهة لغويًا لكلمة "chip".

```
get_similar_tokens('chip', 3, glove_6b50d)
```

```
cosine sim=0.856: chips
cosine sim=0.749: intel
cosine sim=0.749: electronics
```

فيما يلي نواتج كلمات مشابهة لكلمات "baby" و "beautiful".

```
get_similar_tokens('baby', 3, glove_6b50d)
```

```
cosine sim=0.839: babies
cosine sim=0.800: boy
cosine sim=0.792: girl
```

```
get_similar_tokens('beautiful', 3, glove_6b50d)
```

```
cosine sim=0.921: lovely
cosine sim=0.893: gorgeous
cosine sim=0.830: wonderful
```

15.7.2.2. قياس الكلمات Word Analogy

إلى جانب البحث عن كلمات متشابهة، يمكننا أيضًا تطبيق متجهات الكلمات على مهام قياس الكلمات word analogy. على سبيل المثال، "man":"woman" :: "son":"daughter" هي شكل من أشكال قياس الكلمة: "man" لكلمة "woman" حيث أن "son" تعني "daughter". على وجه التحديد، يمكن تعريف مهمة إكمال القياس على النحو التالي: لقياس الكلمة $a:b::c:d$ ، بالنظر إلى الكلمات الثلاث الأولى a، b و c ، ابحث عن d. دلالة على متجه الكلمة w بواسطة $\text{vec}(w)$. لإكمال القياس، سنجد الكلمة التي يكون متجهها أكثر تشابهًا مع نتيجة $\text{vec}(c) + \text{vec}(b) - \text{vec}(a)$.

```
def get_analogy(token_a, token_b, token_c, embed):
    vecs = embed[[token_a, token_b, token_c]]
    x = vecs[1] - vecs[0] + vecs[2]
    topk, cos = knn(embed.idx_to_vec, x, 1)
    return embed.idx_to_token[int(topk[0])]  # Remove
unknown words
```

دعنا نتحقق من القياس "male-female" باستخدام متجهات الكلمات المحملة.

```
get_analogy('man', 'woman', 'son', glove_6b50d)
```
```
'daughter'
```

يكمل أدناه قياس "capital-country" :"beijing" :"china" ::"tokyo" :"japan". يوضح هذا الدلالاتفي متجهات الكلمات التي تم اختبارها مسبقًا.

```
get_analogy('beijing', 'china', 'tokyo', glove_6b50d)
```
```
'japan'
```

بالنسبة لقياس "adjective-superlative adjective" مثل "bad" :"worst" ::"big" :"biggest" ، يمكننا أن نرى أن متجهات الكلمات المدربة مسبقًا قد تلتقط المعلومات النحوية.

```
get_analogy('bad', 'worst', 'big', glove_6b50d)
```
```
'biggest'
```

لإظهار الفكرة الملتقطة من الزمن الماضيفي متجهات الكلمات التي تم اختبارها مسبقًا، يمكننا اختبار الصيغة باستخدام تشبيه "present tense-past tense": "go": "go" :: "did": "do".

```
get_analogy('do', 'did', 'go', glove_6b50d)
```
```
'went'
```

15.7.3. الملخص

- في الممارسة العملية ، يمكن تطبيق متجهات الكلمات المدربة مسبقًا في مجموعات كبيرة على مهام معالجة اللغة الطبيعية النهائية.
- يمكن تطبيق متجهات الكلمات المدربة مسبقًا على مهام التشابه similarity والقياس analogy.

15.7.4. التمارين

1. اختبر نتائج fastText باستخدام `TokenEmbedding('wiki.en')`.
2. عندما تكون المفردات كبيرة للغاية ، كيف يمكننا العثور على كلمات متشابهة أو إكمال قياس الكلمات بشكل أسرع؟

15.8 تمثيلات التشفير ثنائي الاتجاه من المحولات Bidirectional Encoder Representations from (BERT) Transformers

لقد قدمنا العديد من نماذج تضمين الكلمات لفهم اللغة الطبيعية. بعد التدريب المسبق pretraining، يمكن اعتبار المخرجات بمثابة مصفوفة حيث يكون كل صف متجهًا يمثل كلمة من المفردات المدربة مسبقًا.في الواقع، جميع نماذج تضمين الكلمات هذه مستقلة عن السياق context-independent. لنبدأ بتوضيح هذه الخاصية.

15.8.1. من مستقل للسياق إلى حساس للسياق From Context-Independent to Context-Sensitive

راجع التجارب الواردةفي القسم 15.4 والقسم 15.7. على سبيل المثال، يقوم كل من word2vec و GloVe بتعيين نفس المتجه المحدد مسبقًا لنفس الكلمة بغض النظر عن سياق الكلمة (إن وجد). بشكل رسمي، التمثيل المستقل عن السياق لأي رمز x هو دالة $f(x)$ تأخذ فقط xكمدخلاتها. نظرًا لوفرة تعدد المعاني والدلالات المعقدةفي اللغات الطبيعية، فإن التمثيلات المستقلة عن السياق لها حدود واضحة. على سبيل المثال، كلمة "crane"في سياقات "a crane is flying" و " crane driver came" لها معاني مختلفة تمامًا؛ وبالتالي، قد يتم تخصيص تمثيلات مختلفة للكلمة نفسها اعتمادًا على السياقات.

هذا يحفز تطوير تمثيلات الكلمات الحساسة للسياق، حيث تعتمد تمثيلات الكلمات على سياقاتها. ومن ثم، فإن التمثيل الحساس للسياق للرمز x هو دالة $f(x, c(x))$ تعتمد على x وسياقه $c(x)$. تتضمن التمثيلات الشائعة الحساسة للسياق TagLM (أداة تمييز التسلسل المعزز لنموذج اللغة language-model-augmented sequence tagger) (Peters et al.، 2017)، و CoVe (متجهات السياق Context Vectors) (McCann et al.، 2017)، و ELMo(التضمين من نماذج اللغة Embeddings from Language Models) (Peters et al.، 2018).

على سبيل المثال، بأخذ التسلسل بأكمله كمدخل، فإن ELMo هي دالة تقوم بتعيين تمثيل لكل كلمة من تسلسل الإدخال. على وجه التحديد، يجمع ELMo بين جميع تمثيلات الطبقة المتوسطة من LSTM ثنائي الاتجاه سابق التدريب باعتباره تمثيل الإخراج. بعد ذلك، ستتم إضافة تمثيل ELMo إلى النموذج الخاضع للإشراف الحالي للمهمة النهائية كميزات إضافية، مثل ربط تمثيل ELMo والتمثيل الأصلي (على سبيل المثال، GloVe) من الرموزفي النموذج الحالي. من ناحية أخرى، يتم تجميد جميع الأوزانفي نموذج LSTM ثنائي الاتجاه الذي تم

اختباره مسبقًا بعد إضافة تمثيلات ELMo. من ناحية أخرى، يتم تخصيص النموذج الخاضع للإشراف الحالي خصيصًا لمهمة معينة. بالاستفادة من أفضل النماذج المختلفة للمهام المختلفة في ذلك الوقت، أدت إضافة ELMo إلى تحسين أحدث ما توصلت إليه التكنولوجيا عبر ست مهام لمعالجة اللغة الطبيعية: تحليل المشاعر sentiment analysis، والاستدلال اللغوي الطبيعي natural language inference، ووضع العلامات على الأدوار الدلالية semantic role labeling، ودقة coreference، والتعرف على الكيانات المسماة named entity recognition، والإجابة على الأسئلة question answering.

15.8.2. من مهمة محددة إلى مهمة غير محددة From Context-Independent to Context-Sensitive

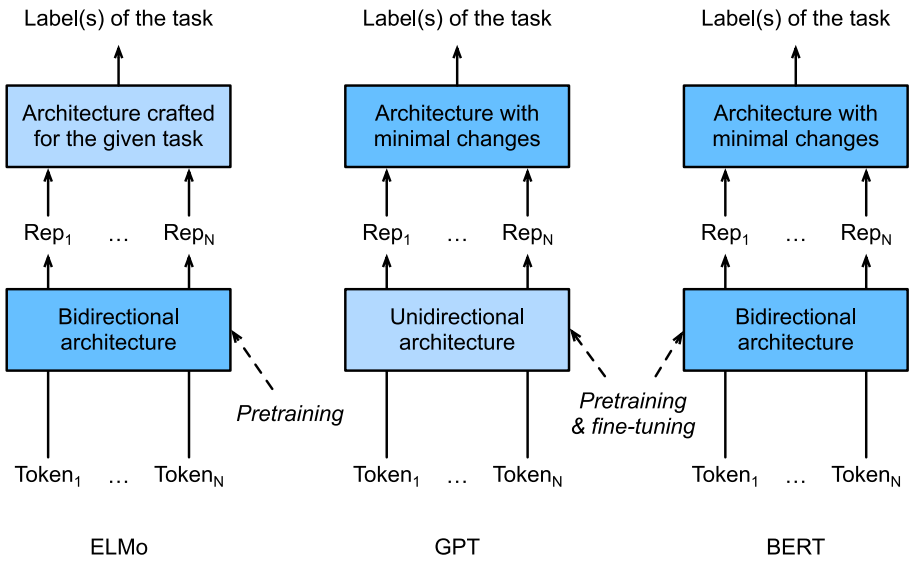
على الرغم من أن ELMo قامت بتحسين الحلول لمجموعة متنوعة من مهام معالجة اللغة الطبيعية، إلا أن كل حل لا يزال يتوقف على بُنية خاصة بمهمة معينة task-specific architecture. ومع ذلك، فمن غير العادي عمليا صياغة بُنية محددة لكل مهمة معالجة لغة طبيعية. يمثل نموذج GPT (التدريب المسبق التوليدي Generative Pre-Training) جهدًا في تصميم نموذج عام غير محدد للمهام task-agnostic للتمثيلات الحساسة للسياق (Radford et al.، 2018). بنيت على وحدة فك ترميز المحولات transformer decoder، GPT تدرس نموذجًا للغة سيتم استخدامه لتمثيل التسلسلات النصية. عند تطبيق GPT على مهمة مصب downstream task، سيتم تغذية ناتج نموذج اللغةفي طبقة إخراج خطية مضافة للتنبؤ بتسمية المهمة.في تناقض حاد مع ELMo الذي يجمد معلمات النموذج الذي تم اختباره مسبقًا، تعمل GPT على ضبط جميع المعلماتفي وحدة فك ترميز المحولات الجاهزة أثناء التعلم الخاضع للإشراف لمهمة المصب. تم تقييم GPT على اثني عشر مهمة من استدلال اللغة الطبيعية، والإجابة على الأسئلة وتشابه الجمل، والتصنيف، وتحسين أحدث ما توصلت إليه التكنولوجيافي تسعة منها مع الحد الأدنى من التغييراتفي بنية النموذج.

ومع ذلك، نظرًا لطبيعة الانحدار الذاتي autoregressive لنماذج اللغة، فإن GPT تتطلع فقط إلى الأمام (من اليسار إلى اليمين).في السياقات "i went to the bank to deposit cash " و "i went to the bank to sit down"، نظرًا لأن "bank " حساس للسياق الموجود على يساره، ستعيد GPT نفس التمثيل لـ "bank"، على الرغم من انه يملك معاني مختلفة.

15.8.3. بيرت: الجمع بين أفضل ما في العالمين BERT: Combining the Best of Both Worlds

كما رأينا، يقوم ELMo بترميز السياق بشكل ثنائي الاتجاه ولكنه يستخدم بنى خاصة بالمهمة؛ بينما يكون GPT غير محدد المهام ولكنه يرمز السياق من اليسار إلى اليمين. من خلال الجمع

بين أفضل مافي العالمين، يقوم BERT (تمثيلات التشفير ثنائية الاتجاه من المحولات) بترميز السياق ثنائي الاتجاه ويتطلب الحد الأدنى من التغييرات الهيكلية لمجموعة واسعة من مهام معالجة اللغة الطبيعية (Devlin et al.، 2018). باستخدام مشفر محول مسبق التدريب، يمكن لـ BERT تمثيل أي رمز بناءً على سياقه ثنائي الاتجاه. أثناء التعلم الخاضع للإشراف للمهام النهائية، يكون BERT مشابهًا لـ GPTفي جانبين. أولاً، سيتم تغذية تمثيلات BERTفي طبقة الإخراج المضافة، مع الحد الأدنى من التغييراتفي بُنية النموذج اعتمادًا على طبيعة المهام، مثل التنبؤ بكل رمز مقابل التنبؤ بالتسلسل بأكمله. ثانيًا، يتم ضبط جميع معلمات وحدة ترميز المحولات المحولة بدقة، بينما يتم تدريب طبقة الإخراج الإضافية من نقطة الصفر. يوضح الشكل 15.8.1 الاختلافات بين ELMo وGPT وBERT.



الشكل 15.8.1 مقارنة بين ELMo وGPT وBERT.

قام BERT أيضًا بتحسين أحدث ما توصلت إليه التكنولوجيافي إحدى عشرة مهمة معالجة للغة الطبيعية ضمن فئات واسعة من (1) تصنيف نص واحد single text classification (على سبيل المثال، تحليل المشاعر)، (2) تصنيف زوج النص text pair classification (على سبيل المثال، استدلال اللغة الطبيعية)، (3) الإجابة على السؤال، (4) وضع علامات على النص text tagging (على سبيل المثال، التعرف على الكيانات المسماة named entity recognition). كل ما تم اقتراحهفي عام 2018، من ELMo الحساسة للسياق إلى GPT وBERT غير المحدودين للمهام، فإن التدريب المسبق البسيط من الناحية المفاهيمية ولكنه

قوي تجريبيًا للتمثيلات العميقة للغات الطبيعية أحدث ثورةفي الحلول لمختلف مهام معالجة اللغة الطبيعية.

في بقية هذا الفصل، سوف نتعمقفي التدريب المسبق لـ BERT. عندما يتم شرح تطبيقات المعالجة اللغوية الطبيعيةفي القسم 16، سنقوم بتوضيح ضبط BERT لتطبيقات المصب downstream applications.

```
from mxnet import gluon, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

15.8.4. تمثيل المدخلات Input Representation

في معالجة اللغة الطبيعية، تأخذ بعض المهام (على سبيل المثال، تحليل المشاعر) نصًا منفردًا كمدخل، بينمافي بعض المهام الأخرى (على سبيل المثال، استدلال اللغة الطبيعية)، يكون الإدخال عبارة عن زوج من تسلسلات النص. يمثل تسلسل إدخال BERT بشكل لا لبس فيه نصًا منفردًا وأزواجًا نصية.في السابق، يكون تسلسل إدخال BERT عبارة عن سلسلة لرمز التصنيف الخاص "<cls>"، وعلامات تسلسل نصي، ورمز الفصل الخاص "<sep>".في الأخير، تسلسل إدخال BERT هو عبارة عن سلسلة من "<cls>"، الرموز لتسلسل النص الأول، "<sep>"، الرموز لتسلسل النص الثاني، و"<sep>". سوف نميز باستمرار المصطلحات "تسلسل إدخال BERT" عن الأنواع الأخرى من "التسلسلات". على سبيل المثال، قد يتضمن تسلسل إدخال BERT إما تسلسل نصي واحد أو تسلسلين نصيين.

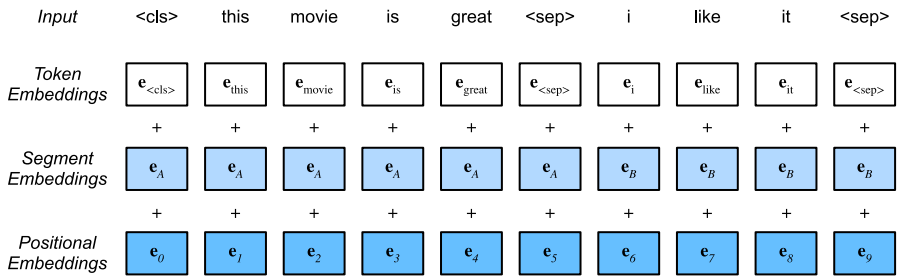
للتمييز بين أزواج النص text pairs، تتم إضافة التضمينات التي تم تعلمها $\mathbf{e}_A$ و $\mathbf{e}_B$ إلى التضمينات الرمزية للتسلسل الأول والتسلسل الثاني، على التوالي. بالنسبة لمدخلات النص الفردي، يتم استخدام $\mathbf{e}_A$ فقط.

تأخذ `get_tokens_and_segments` التالية إما جملة واحدة أو جملتين كمدخلات ، ثم تعيد الرموز لتسلسل إدخال BERT ومعرفات المقطع المقابلة لها.

```
#@save
def get_tokens_and_segments(tokens_a, tokens_b=None):
    """Get tokens of the BERT input sequence and their
segment IDs."""
    tokens = ['<cls>'] + tokens_a + ['<sep>']
    # 0 and 1 are marking segment A and B, respectively
    segments = [0] * (len(tokens_a) + 2)
```

```
    if tokens_b is not None:
        tokens += tokens_b + ['<sep>']
        segments += [1] * (len(tokens_b) + 1)
    return tokens, segments
```

تختار BERT مشفر المحول على أنه معمارية ثنائية الاتجاه. شائعفي مشفر المحولات، تتم إضافة التضمينات الموضعيةفي كل موضع من تسلسل إدخال BERT. ومع ذلك، يختلف عن مشفر المحولات الأصلي، يستخدم BERT تضمينات موضعية قابلة للتعلم. للتلخيص، يوضح الشكل 15.8.2 أن التضميناتفي تسلسل إدخال BERT هي مجموع تضمينات الرمز، والتضمينات المقطعية segment embeddings، والتضمينات الموضعية positional embeddings.



الشكل 15.8.2 إن عمليات التضمينفي تسلسل إدخال BERT هي مجموع تضمينات الرمز، والتضمينات المقطعية، والتضمينات الموضعية.

```
#@save
class BERTEncoder(nn.Block):
    """BERT encoder."""
    def __init__(self, vocab_size, num_hiddens,
ffn_num_hiddens, num_heads,
                 num_blks, dropout, max_len=1000,
**kwargs):
        super(BERTEncoder, self).__init__(**kwargs)
        self.token_embedding = nn.Embedding(vocab_size,
num_hiddens)
        self.segment_embedding = nn.Embedding(2,
num_hiddens)
        self.blks = nn.Sequential()
        for _ in range(num_blks):
            self.blks.add(d2l.TransformerEncoderBlock(
                num_hiddens, ffn_num_hiddens, num_heads,
dropout, True))
```

```
        # In BERT, positional embeddings are learnable,
thus we create a
        # parameter of positional embeddings that are
long enough
        self.pos_embedding =
self.params.get('pos_embedding',
                                                shape=(1,
max_len, num_hiddens))

    def forward(self, tokens, segments, valid_lens):
        # Shape of `X` remains unchanged in the
following code snippet:
        # (batch size, max sequence length,
`num_hiddens`)
        X = self.token_embedding(tokens) +
self.segment_embedding(segments)
        X = X + self.pos_embedding.data(ctx=X.ctx)[:,
:X.shape[1], :]
        for blk in self.blks:
            X = blk(X, valid_lens)
        return X
```

افترض أن حجم المفردات هو 10000. لإثبات الاستدلال الأمامي لـ BERTEncoder، دعنا ننشئ مثيلًا له ونبدأفي تهيئة معلماته.

```
vocab_size, num_hiddens, ffn_num_hiddens, num_heads =
10000, 768, 1024, 4
num_blks, dropout = 2, 0.2
encoder = BERTEncoder(vocab_size, num_hiddens,
ffn_num_hiddens, num_heads,
                      num_blks, dropout)
encoder.initialize()
```

نحدد الرموز على أنها 2 متسلسلات إدخال BERT بطول 8، حيث يكون كل رمز مؤشرًا للمفردات. يُرجع الاستدلال الأمامي لـ BERTEncoder مع الرموز للإدخال النتيجة المشفرة حيث يتم تمثيل كل رمز بواسطة متجه يتم تحديد طوله مسبقًا بواسطة المعلمة الفائقة num_hiddens. يُشار إلى هذه المعلمة الفائقة عادةً بالحجم المخفي hidden size (عدد الوحدات المخفية) لجهاز تشفير المحولات transformer encoder.

```
tokens = np.random.randint(0, vocab_size, (2, 8))
segments = np.array([[0, 0, 0, 0, 1, 1, 1, 1], [0, 0, 0,
1, 1, 1, 1, 1]])
encoded_X = encoder(tokens, segments, None)
```

```
encoded_X.shape
```

```
(2, 8, 768)
```

15.8.5. التدريب المسبق للمهام Pretraining Tasks

يعطي الاستدلال الأمامي لـ `BERTEncoder` تمثيل BERT لكل رمز لنص الإدخال والرموز المدرجة "<cls>" و "<seq>". بعد ذلك، سوف نستخدم هذه التمثيلات لحساب دالة الخسارة للتدريب المسبق لـ BERT. يتكون التدريب المسبق من المهمتين التاليتين: نمذجة اللغة المقنعة Masked Language Modeling وتنبؤ الجملة التالية next sentence prediction.

15.8.5.1. نمذجة اللغة المقنعة masked language modeling

كما هو موضحفي القسم 9.3، يتنبأ نموذج اللغة برمز باستخدام السياق الموجود على يساره. لتشفير السياق ثنائي الاتجاه لتمثيل كل رمز، يقوم BERT بإخفاء الرموز بشكل عشوائي واستخدام الرموز من السياق ثنائي الاتجاه للتنبؤ بالرموز المقنعة masked tokens بطريقة تخضع للإشراف الذاتي. يشار إلى هذه المهمة كنموذج لغة مقنع masked language model.

في هذه المهمة مسبقة التدريب، سيتم اختيار 15٪ من الرموز بشكل عشوائي كرموز مقنعة للتنبؤ. للتنبؤ بالرمز المميز المقنع دون الغش باستخدام التسمية، فإن أحد الأساليب المباشرة هو استبداله دائمًا برمز "<mask>"في تسلسل إدخال BERT. ومع ذلك، فإن الرمز الاصطناعي الخاص "<mask>" لن يظهر أبدًافي الضبط الدقيق. لتجنب مثل هذا عدم التطابق بين التدريب المسبق والضبط الدقيق fine-tuning، إذا تم إخفاء رمز للتنبؤ (على سبيل المثال، تم تحديد "great" لإخفائه والتنبؤ بهفي "this movie is great")، فسيتم استبدالهفي الإدخال بـ:

- الرمز "<mask>" لمدة 80٪ من الوقت (على سبيل المثال ، "this movie is great"يصبح "this movie is <mask>") ؛
- الرمز العشوائي لمدة 10٪ من الوقت (على سبيل المثال ، "this movie is great" يصبح "this movie is drink") ؛
- الرمز التي لم يتم تغييره لمدة 10٪ من الوقت (على سبيل المثال ، "this movie is great" تصبح "this movie is great").

لاحظ أنه لمدة 10٪ من 15٪ من الوقت يتم إدخال رمز عشوائي. تشجع هذه الضوضاء العرضية BERT على أن تكون أقل تحيزًا تجاه الرمز المقنع (خاصةً عندما يظل الرمز لتسمية دون تغيير) في ترميز السياق ثنائي الاتجاه الخاص به.

نقوم بتنفيذ فئة `MaskLM` التالية للتنبؤ بالرموز المقنعةفي مهمة نموذج اللغة المقنعة للتدريب المسبق لـ BERT. يستخدم التنبؤ MLP ذو الطبقة الواحدة المخفية (`self.mlp`).في الاستدلال الأمامي، يتطلب الأمر مدخلين: النتيجة المشفرة لـ `BERTEncoder`ومواضع الرمز للتنبؤ. الناتج هو نتائج التنبؤفي هذه المواضع.

```
#@save
class MaskLM(nn.Block):
    """The masked language model task of BERT."""
    def __init__(self, vocab_size, num_hiddens,
**kwargs):
        super(MaskLM, self).__init__(**kwargs)
        self.mlp = nn.Sequential()
        self.mlp.add(
            nn.Dense(num_hiddens, flatten=False,
activation='relu'))
        self.mlp.add(nn.LayerNorm())
        self.mlp.add(nn.Dense(vocab_size,
flatten=False))

    def forward(self, X, pred_positions):
        num_pred_positions = pred_positions.shape[1]
        pred_positions = pred_positions.reshape(-1)
        batch_size = X.shape[0]
        batch_idx = np.arange(0, batch_size)
        # Suppose that `batch_size` = 2,
`num_pred_positions` = 3, then
        # `batch_idx` is `np.array([0, 0, 0, 1, 1, 1])`
        batch_idx = np.repeat(batch_idx,
num_pred_positions)
        masked_X = X[batch_idx, pred_positions]
        masked_X = masked_X.reshape((batch_size,
num_pred_positions, -1))
        mlm_Y_hat = self.mlp(masked_X)
        return mlm_Y_hat
```

لإثبات الاستدلال الأمامي لـ `MaskLM`، نقوم بإنشاء مثيله `mlm`وتهيئته. تذكر أن `encoded_X` من الاستدلال الأمامي لـ `BERTEncoder`يمثل تسلسلين إدخال BERT. نحدد `mlm_positions` على أنها المؤشرات الثلاثة للتنبؤ بأي تسلسل إدخال BERT من `encoded_X`. يعيد الاستدلال الأمامي لـ `mlm` نتائج التنبؤ `mlm_Y_hat`في جميع المواضع

المقنعة `mlm_positions` من `encoded_X`. لكل توقع ، حجم النتيجة يساوي حجم المفردات.

```
mlm = MaskLM(vocab_size, num_hiddens)
mlm.initialize()
mlm_positions = np.array([[1, 5, 2], [6, 1, 5]])
mlm_Y_hat = mlm(encoded_X, mlm_positions)
mlm_Y_hat.shape
```

```
(2, 3, 10000)
```

باستخدام تسميات الحقيقة الأساسية `mlm_Y` للرموز المتوقعة `mlm_Y_hat` تحت الأقنعة ، يمكننا حساب خسارة الانتروبيا المتقاطعة لمهمة نموذج اللغة المقنعةفي تدريب BERT المسبق.

```
mlm_Y = np.array([[7, 8, 9], [10, 20, 30]])
loss = gluon.loss.SoftmaxCrossEntropyLoss()
mlm_l = loss(mlm_Y_hat.reshape((-1, vocab_size)),
mlm_Y.reshape(-1))
mlm_l.shape
```

```
(6,)
```

15.8.5.2. توقع الجملة التالية Next Sentence Prediction

على الرغم من أن نمذجة اللغة المقنعة قادرة على ترميز سياق ثنائي الاتجاه لتمثيل الكلمات، إلا أنها لا تمثل بشكل صريح العلاقة المنطقية بين أزواج النص. للمساعدةفي فهم العلاقة بين تسلسلين نصيين، يعتبر BERT مهمة تصنيف ثنائي، توقع الجملة التالية،في التدريب المسبق. عند إنشاء أزواج من الجمل للتدريب المسبق، تكونفي الواقع جمل متتاليةفي نصف الوقت تحمل التسمية "True"؛ بينمافي النصف الآخر من الوقت، يتم أخذ عينات الجملة الثانية بشكل عشوائي من المجموعة مع تسمية "False".

تستخدم فئة `NextSentencePred` التالية MLP ذات طبقة واحدة مخفية للتنبؤ بما إذا كانت الجملة الثانية هي الجملة التالية من الجملة الأولىفي تسلسل إدخال BERT. بسبب الانتباه الذاتيفي مشفر المحولات، فإن تمثيل BERT للرمز "<cls>" يشفر الجملتين من المدخلات. ومن ثم، فإن طبقة الإخراج (المخرجات الذاتية) لمصنف MLP تأخذ X كمدخل، حيث X هو ناتج الطبقة المخفية MLP التي يكون مدخلها هو الرمز "<cls>" المشفر.

```
#@save
class NextSentencePred(nn.Block):
    """The next sentence prediction task of BERT."""
    def __init__(self, **kwargs):
```

```
        super(NextSentencePred, self).__init__(**kwargs)
        self.output = nn.Dense(2)

    def forward(self, X):
        # `X` shape: (batch size, `num_hiddens`)
        return self.output(X)
```

يمكننا أن نرى أن الاستدلال الأمامي لمثيل `NextSentencePred` يُرجع تنبؤات ثنائية لكل تسلسل إدخال BERT.

```
nsp = NextSentencePred()
nsp.initialize()
nsp_Y_hat = nsp(encoded_X)
nsp_Y_hat.shape
```

```
(2, 2)
```

يمكن أيضًا حساب خسارة الانتروبيافي التصنيفات الثنائية.

```
nsp_y = np.array([0, 1])
nsp_l = loss(nsp_Y_hat, nsp_y)
nsp_l.shape
```

```
(2,)
```

من الجدير بالذكر أنه يمكن الحصول على جميع التسمياتفي كل من مهام التدريب المسبق المذكورة أعلاه بشكل عادي من مجموعة ما قبل التدريب دون بذل جهد يدوي لوضع التسميات. تم اختبار BERT الأصلي مسبقًا على تسلسل BookCorpus، (Zhu et al.,2015) ويكيبيديا الإنجليزية. هاتان المجموعتان النصيتان كبيرتان: تحتويان على 800 مليون كلمة و 2.5 مليار كلمة، على التوالي.

15.8.6. وضع كل شيء معا Putting It All Together

عند التدريب المسبق لـ BERT، تكون دالة الخسارة النهائية عبارة عن مزيج خطي من دوال الخسارة لنمذجة اللغة المقنعة وتنبؤ الجملة التالية. الآن يمكننا تحديد فئة `BERTModel`من خلال إنشاء مثيل للفئات الثلاث `BERTEncoder`و `MaskLM`و `NextSentencePred`. يُرجع الاستدلال الأمامي تمثيلات BERT المشفرة `encoded_X`، وتنبؤات نمذجة اللغة المقنعة `mlm_Y_hat`، وتنبؤات الجملة التالية `nsp_Y_hat`.

```
#@save
class BERTModel(nn.Block):
    """The BERT model."""
    def __init__(self, vocab_size, num_hiddens,
ffn_num_hiddens, num_heads,
```

```
                 num_blks, dropout, max_len=1000):
        super(BERTModel, self).__init__()
        self.encoder = BERTEncoder(vocab_size,
num_hiddens, ffn_num_hiddens,
                                   num_heads, num_blks,
dropout, max_len)
        self.hidden = nn.Dense(num_hiddens,
activation='tanh')
        self.mlm = MaskLM(vocab_size, num_hiddens)
        self.nsp = NextSentencePred()

    def forward(self, tokens, segments, valid_lens=None,
pred_positions=None):
        encoded_X = self.encoder(tokens, segments,
valid_lens)
        if pred_positions is not None:
            mlm_Y_hat = self.mlm(encoded_X,
pred_positions)
        else:
            mlm_Y_hat = None
        # The hidden layer of the MLP classifier for
next sentence prediction.
        # 0 is the index of the '<cls>' token
        nsp_Y_hat = self.nsp(self.hidden(encoded_X[:, 0,
:]))
        return encoded_X, mlm_Y_hat, nsp_Y_hat
```

15.8.7. الملخص

- تعد نماذج تضمين الكلمات مثل word2vec و GloVe مستقلة عن السياق. يخصصون نفس المتجه المحدد مسبقًا لنفس الكلمة بغض النظر عن سياق الكلمة (إن وجد). يصعب عليهم التعامل مع تعدد المعاني أو دلالات معقدة في اللغات الطبيعية.
- بالنسبة لتمثيلات الكلمات الحساسة للسياق مثل ELMo و GPT ، تعتمد تمثيلات الكلمات على سياقاتها.
- يقوم ELMo بتشفير السياق ثنائي الاتجاه ولكنه يستخدم معماريات خاصة بالمهمة (ومع ذلك ، فمن غير العادي عمليا صياغة بُنية محددة لكل مهمة معالجة لغة طبيعية)؛ بينما يكون GPT غير محدد المهام ولكنه يشفر السياق من اليسار إلى اليمين.
- يجمع BERT بين أفضل ما في العالمين: فهو يشفر السياق بشكل ثنائي الاتجاه ويتطلب الحد الأدنى من التغييرات المعمارية لمجموعة واسعة من مهام معالجة اللغة الطبيعية.

- إن تضمينات في تسلسل إدخال BERT هي مجموع تضمينات الرمز، والتضمينات المقطعية ، والتضمينات الموضعية.
- يتكون تدريب BERT المسبق من مهمتين: نمذجة اللغة المقنعة وتنبؤ الجملة التالية. الأول قادر على ترميز سياق ثنائي الاتجاه لتمثيل الكلمات ، في حين أن الأخير يصمم بوضوح العلاقة المنطقية بين أزواج النص.

15.8.8. التمارين

1. عند تساوي جميع الأشياء الأخرى ، هل سيتطلب نموذج اللغة المقنعة أكثر أو أقل من خطوات ما قبل التدريب للتقارب من نموذج اللغة من اليسار إلى اليمين؟ لماذا؟
2. في التطبيق الأصلي لـ BERT ، تستخدم شبكة التغذية الأمامية الموضعية positionwise feed-forward network في `BERTEncoder` (عبر `d2l.TransformerEncoderBlock`) والطبقة المتصلة بالكامل في `MaskLM` كلاهما وحدة خطأ غاوسي الخطية Gaussian error linear unit (GELU) ، (Hendrycks and Gimpel, 2016) كدالة التنشيط. ابحث في الفرق بين GELU و ReLU.

15.9. مجموعة البيانات الخاصة بالتدريب المسبق لبيرت The Dataset for Pretraining BERT

لإجراء تدريب مسبق لنموذج BERT كما تم تنفيذهفي القسم 15.8، نحتاج إلى إنشاء مجموعة البيانات بالتنسيق المثالي لتسهيل مهمتي التدريب المسبق: نمذجة اللغة المقنعة وتنبؤ الجملة التالية. من ناحية أخرى، تم اختبار نموذج BERT الأصلي مسبقًا على تسلسل مجموعتين ضخمتين من BookCorpus وWikipedia الإنجليزية (انظر القسم 15.8.5)، مما يجعل من الصعب تشغيله لمعظم قراء هذا الكتاب. من ناحية أخرى، قد لا يصلح نموذج BERT الجاهز الجاهز للتطبيقات من مجالات معينة مثل الطب. وبالتالي، أصبح من الشائع إجراء اختبار BERT مسبقًا على مجموعة بيانات مخصصة. لتسهيل عرض تدريب BERT المسبق، نستخدم مجموعة أصغر WikiText-2 (Merity et al., 2016).

بالمقارنة مع مجموعة بيانات PTB المستخدمة للتدريب المسبق على word2vecفي القسم 15.3، يحتفظ (i) WikiText-2 بعلامات الترقيم الأصلية، مما يجعلها مناسبة للتنبؤ بالجمل التالية؛ (2) يحتفظ بالحالة والأرقام الأصلية؛ (3) أكبر مرتين.

```
import os
import random
from mxnet import gluon, np, npx
```

```
from d2l import mxnet as d2l

npx.set_np()
```

في مجموعة بيانات WikiText-2، يمثل كل سطر فقرة حيث يتم إدخال مسافة بين أي علامة ترقيم والرمز السابق لها. يتم الاحتفاظ بالفقرات التي تحتوي على جملتين على الأقل. لتقسيم الجمل، نستخدم النقطة فقط كمحدد delimiter للبساطة. نترك مناقشات حول تقنيات تقسيم الجمل الأكثر تعقيدًافي التمارينفي نهاية هذا القسم.

```
#@save
d2l.DATA_HUB['wikitext-2'] = (

'https://s3.amazonaws.com/research.metamind.io/wikitext/
'
    'wikitext-2-v1.zip',
'3c914d17d80b1459be871a5039ac23e752a53cbe')

#@save
def _read_wiki(data_dir):
    file_name = os.path.join(data_dir,
'wiki.train.tokens')
    with open(file_name, 'r') as f:
        lines = f.readlines()
    # Uppercase letters are converted to lowercase ones
    paragraphs = [line.strip().lower().split(' . ')
                  for line in lines if len(line.split('
. ')) >= 2]
    random.shuffle(paragraphs)
    return paragraphs
```

15.9.1. تعريف دوال المساعد لمهام التدريب المسبق Defining Helper Functions for Pretraining Tasks

فيما يلي، نبدأ بتنفيذ دوال المساعد helper functions لمهمتي تدريب BERT: توقع الجملة التالية ونمذجة اللغة المقنعة. سيتم استدعاء هذه دوال المساعد لاحقًا عند تحويل مجموعة النص الخام إلى مجموعة بيانات بالتنسيق المثالي لإجراء BERT مسبقًا.

15.9.1.1. إنشاء مهمة توقع الجملة التالية Generating the Next Sentence Prediction Task

وفقًا للأوصاف الواردةفي القسم 15.8.5.2، تُنشئ دوال `_get_next_sentence` مثالاً تدريبيًا لمهمة التصنيف الثنائي.

```
#@save
def _get_next_sentence(sentence, next_sentence,
paragraphs):
    if random.random() < 0.5:
        is_next = True
    else:
        # `paragraphs` is a list of lists of lists
        next_sentence =
random.choice(random.choice(paragraphs))
        is_next = False
    return sentence, next_sentence, is_next
```

تُنشئ الدالة التالية أمثلة تدريبية للتنبؤ بالجمل التالية من فقرة الإدخال عن طريق استدعاء دالة `_get_next_sentence`. هنا `paragraph` هي قائمة الجمل، حيث كل جملة عبارة عن قائمة من الرموز tokens. تحدد الوسيطة `max_len` الحد الأقصى لطول تسلسل إدخال BERT أثناء التدريب المسبق.

```
#@save
def _get_nsp_data_from_paragraph(paragraph, paragraphs,
vocab, max_len):
    nsp_data_from_paragraph = []
    for i in range(len(paragraph) - 1):
        tokens_a, tokens_b, is_next =
_get_next_sentence(
            paragraph[i], paragraph[i + 1], paragraphs)
        # Consider 1 '<cls>' token and 2 '<sep>' tokens
        if len(tokens_a) + len(tokens_b) + 3 > max_len:
            continue
        tokens, segments =
d2l.get_tokens_and_segments(tokens_a, tokens_b)
        nsp_data_from_paragraph.append((tokens,
segments, is_next))
    return nsp_data_from_paragraph
```

15.9.1.2. إنشاء مهمة نمذجة اللغة المقنعة Generating the Masked Language Modeling Task

من أجل إنشاء أمثلة تدريبية لمهمة نمذجة اللغة المقنعة من تسلسل إدخال BERT، نحدد دالة `_replace_mlm_tokens` التالية.في مدخلاتها، تعد الرموز `tokens` عبارة عن قائمة من الرموز التي تمثل تسلسل إدخال BERT ، و `candidate_pred_positions` عبارة عن قائمة بمؤشرات الرموز لتسلسل إدخال BERT باستثناء الرموز الخاصة special tokens (لا يتم توقع الرموزفي مهمة نمذجة اللغة المقنعة) ، ويشير `num_mlm_preds` عدد التنبؤات

(15٪ من الرموز العشوائية للتنبؤ). باتباع تعريف مهمة نمذجة اللغة المقنعةفي القسم 15.8.5.1، في كل موضع تنبؤ، يمكن استبدال الإدخال برمز "<mask>" أو رمز عشوائي، أو يظل دون تغيير. في النهاية، تُرجع الدالة المميزة للإدخال بعد الاستبدال المحتمل، ومؤشرات الرمز حيث تحدث التنبؤات وتسميات هذه التنبؤات.

```
#@save
def _replace_mlm_tokens(tokens,
candidate_pred_positions, num_mlm_preds,
                        vocab):
    # For the input of a masked language model, make a
new copy of tokens and
    # replace some of them by '<mask>' or random tokens
    mlm_input_tokens = [token for token in tokens]
    pred_positions_and_labels = []
    # Shuffle for getting 15% random tokens for
prediction in the masked
    # language modeling task
    random.shuffle(candidate_pred_positions)
    for mlm_pred_position in candidate_pred_positions:
        if len(pred_positions_and_labels) >=
num_mlm_preds:
            break
        masked_token = None
        # 80% of the time: replace the word with the
'<mask>' token
        if random.random() < 0.8:
            masked_token = '<mask>'
        else:
            # 10% of the time: keep the word unchanged
            if random.random() < 0.5:
                masked_token = tokens[mlm_pred_position]
            # 10% of the time: replace the word with a
random word
            else:
                masked_token =
random.choice(vocab.idx_to_token)
        mlm_input_tokens[mlm_pred_position] =
masked_token
        pred_positions_and_labels.append(
            (mlm_pred_position,
tokens[mlm_pred_position]))
```

```
    return mlm_input_tokens, pred_positions_and_labels
```

من خلال استدعاء دالة `_replace_mlm_tokens` المذكورة أعلاه ، تأخذ الدالة التالية تسلسل إدخال BERT (`tokens`) كمدخلات وترجع مؤشرات رموز الإدخال (بعد استبدال الرمز كما هو موضحفي القسم 15.8.5.1) ، مؤشرات الرمز حيث تحدث التنبؤات، ومؤشرات التسمية لهذه التنبؤات.

```
#@save
def _get_mlm_data_from_tokens(tokens, vocab):
    candidate_pred_positions = []
    # `tokens` is a list of strings
    for i, token in enumerate(tokens):
        # Special tokens are not predicted in the masked
language modeling
        # task
        if token in ['<cls>', '<sep>']:
            continue
        candidate_pred_positions.append(i)
    # 15% of random tokens are predicted in the masked
language modeling task
    num_mlm_preds = max(1, round(len(tokens) * 0.15))
    mlm_input_tokens, pred_positions_and_labels =
_replace_mlm_tokens(
        tokens, candidate_pred_positions, num_mlm_preds,
vocab)
    pred_positions_and_labels =
sorted(pred_positions_and_labels,
                                        key=lambda x:
x[0])
    pred_positions = [v[0] for v in
pred_positions_and_labels]
    mlm_pred_labels = [v[1] for v in
pred_positions_and_labels]
    return vocab[mlm_input_tokens], pred_positions,
vocab[mlm_pred_labels]
```

15.9.2. تحويل النص إلى مجموعة بيانات التدريب المسبق Transforming Text into the Pretraining Dataset

نحن الآن جاهزون تقريبًا لتخصيص فئة Dataset للتدريب المسبق على BERT. قبل ذلك، ما زلنا بحاجة إلى تعريف دالة مساعدة `_pad_bert_inputs` لإلحاق الرموز "<pad>" بالمدخلات. تحتوي أمثلة الوسيطات الخاصة بها على مخرجات من الدالتين المساعدتين

`_get_mlm_data_from_tokens` و `_get_nsp_data_from_paragraph` لمهمتي التدريب المسبق.

```
#@save
def _pad_bert_inputs(examples, max_len, vocab):
    max_num_mlm_preds = round(max_len * 0.15)
    all_token_ids, all_segments, valid_lens,  = [], [],
[]
    all_pred_positions, all_mlm_weights, all_mlm_labels
= [], [], []
    nsp_labels = []
    for (token_ids, pred_positions, mlm_pred_label_ids,
segments,
         is_next) in examples:
        all_token_ids.append(np.array(token_ids +
[vocab['<pad>']] * (
            max_len - len(token_ids)), dtype='int32'))
        all_segments.append(np.array(segments + [0] * (
            max_len - len(segments)), dtype='int32'))
        # `valid_lens` excludes count of '<pad>' tokens
        valid_lens.append(np.array(len(token_ids),
dtype='float32'))

all_pred_positions.append(np.array(pred_positions + [0]
* (
            max_num_mlm_preds - len(pred_positions)),
dtype='int32'))
        # Predictions of padded tokens will be filtered
out in the loss via
        # multiplication of 0 weights
        all_mlm_weights.append(
            np.array([1.0] * len(mlm_pred_label_ids) +
[0.0] * (
                max_num_mlm_preds -
len(pred_positions)), dtype='float32'))

all_mlm_labels.append(np.array(mlm_pred_label_ids + [0]
* (
            max_num_mlm_preds -
len(mlm_pred_label_ids)), dtype='int32'))
        nsp_labels.append(np.array(is_next))
```

```
    return (all_token_ids, all_segments, valid_lens,
all_pred_positions,
            all_mlm_weights, all_mlm_labels, nsp_labels)
```

من خلال وضع دوال المساعد لتوليد أمثلة تدريبية لمهمتي التدريب المسبق، ودالة المساعد لحشو المدخلات معًا، نقوم بتخصيص فئة `_WikiTextDataset` التالية على أنها مجموعة بيانات WikiText-2 للتدريب المسبق لـ BERT. من خلال تنفيذ دالة `__getitem__`، يمكننا الوصول بشكل تعسفي إلى أمثلة التدريب المسبق (نمذجة اللغة المقنعة وتنبؤ الجملة التالية) التي تم إنشاؤها من زوج من الجمل من مجموعة WikiText-2.

يستخدم نموذج BERT الأصلي زخارف WordPiece التي يبلغ حجم مفرداتها 30000 (Wu et al.، 2016). طريقة الترميزفي WordPiece هي تعديل طفيف لخوارزمية ترميز byte pair الأصليفي القسم 15.6.2. للتبسيط، نستخدم دالة `d2l.tokenize` للترميز tokenization. الرموز غير المتكررة Infrequent tokens التي تظهر أقل من خمس مرات يتم تصفيتها.

```
#@save
class _WikiTextDataset(gluon.data.Dataset):
    def __init__(self, paragraphs, max_len):
        # Input `paragraphs[i]` is a list of sentence
strings representing a
        # paragraph; while output `paragraphs[i]` is a
list of sentences
        # representing a paragraph, where each sentence
is a list of tokens
        paragraphs = [d2l.tokenize(
            paragraph, token='word') for paragraph in
paragraphs]
        sentences = [sentence for paragraph in
paragraphs
                     for sentence in paragraph]
        self.vocab = d2l.Vocab(sentences, min_freq=5,
reserved_tokens=[
            '<pad>', '<mask>', '<cls>', '<sep>'])
        # Get data for the next sentence prediction task
        examples = []
        for paragraph in paragraphs:

examples.extend(_get_nsp_data_from_paragraph(
                paragraph, paragraphs, self.vocab,
max_len))
```

```
        # Get data for the masked language model task
        examples = [(_get_mlm_data_from_tokens(tokens,
self.vocab)
                     + (segments, is_next))
                    for tokens, segments, is_next in
examples]
        # Pad inputs
        (self.all_token_ids, self.all_segments,
self.valid_lens,
         self.all_pred_positions, self.all_mlm_weights,
         self.all_mlm_labels, self.nsp_labels) =
_pad_bert_inputs(
            examples, max_len, self.vocab)

    def __getitem__(self, idx):
        return (self.all_token_ids[idx],
self.all_segments[idx],
                self.valid_lens[idx],
self.all_pred_positions[idx],
                self.all_mlm_weights[idx],
self.all_mlm_labels[idx],
                self.nsp_labels[idx])

    def __len__(self):
        return len(self.all_token_ids)
```

باستخدام دالة `_read_wiki` وفئة `_WikiTextDataset` ، نحدد `load_data_wiki` التالية لتنزيل مجموعة بيانات WikiText-2 وإنشاء أمثلة للتدريب المسبق منها.

```
#@save
def load_data_wiki(batch_size, max_len):
    """Load the WikiText-2 dataset."""
    num_workers = d2l.get_dataloader_workers()
    data_dir = d2l.download_extract('wikitext-2',
'wikitext-2')
    paragraphs = _read_wiki(data_dir)
    train_set = _WikiTextDataset(paragraphs, max_len)
    train_iter = gluon.data.DataLoader(train_set,
batch_size, shuffle=True,

num_workers=num_workers)
```

```
    return train_iter, train_set.vocab
```

بضبط حجم الدُفعة على 512 والحد الأقصى لطول تسلسل إدخال BERT ليكون 64، نقوم بطباعة أشكال دفعة صغيرة من أمثلة التدريب المسبق لـ BERT. لاحظ أنهفي كل تسلسل إدخال BERT، (64 × 0.15)10 يتم توقع المواضع لمهمة نمذجة اللغة المقنعة.

```
batch_size, max_len = 512, 64
train_iter, vocab = load_data_wiki(batch_size, max_len)

for (tokens_X, segments_X, valid_lens_x,
pred_positions_X, mlm_weights_X,
     mlm_Y, nsp_y) in train_iter:
    print(tokens_X.shape, segments_X.shape,
valid_lens_x.shape,
          pred_positions_X.shape, mlm_weights_X.shape,
mlm_Y.shape,
          nsp_y.shape)
    break
```

```
Downloading ../data/wikitext-2-v1.zip from
https://s3.amazonaws.com/research.metamind.io/wikitext/w
ikitext-2-v1.zip...
(512, 64) (512, 64) (512,) (512, 10) (512, 10) (512, 10)
(512,)
```

في النهاية، دعونا نلقي نظرة على حجم المفردات. حتى بعد تصفية الرموز النادرة، فإنها لا تزال أكبر بمرتين من مجموعة بيانات PTB.

```
len(vocab)
```

```
20256
```

15.9.3. الملخص

- بالمقارنة مع مجموعة بيانات PTB ، تحتفظ مجموعة بيانات WikiText-2 بعلامات الترقيم والحالة والأرقام الأصلية، وهي أكبر مرتين.
- يمكننا الوصول بشكل تعسفي إلى أمثلة التدريب المسبق (نمذجة اللغة المقنعة وتنبؤ الجملة التالية) الناتجة عن زوج من الجمل من مجموعة WikiText-2.

15.9.4. التمارين

1. للتبسيط، يتم استخدام النقطة كمحدد delimiter وحيد لتقسيم الجمل. جرب تقنيات أخرى لتقسيم الجمل، مثل spaCy وNLTK. خذ NLTK كمثال. تحتاج إلى تثبيت NLTK أولاً: `pip install nltk`. في الكود، قم أولاً بالاستيراد `import nltk`. بعد ذلك، قم بتنزيل `nltk.download('punkt')`.

لتقسيم الجمل مثل `sentences = 'This is great ! Why not?'`، استدعاء `nltk.tokenize.sent_tokenize(sentences)` سيعيد قائمة من سلسلتين جملتين: `['This is great !', 'Why not ?']`.

2. ما هو حجم المفردات إذا لم نقم بتصفية أي رمز غير متكرر infrequent token؟

15.10. التدريب المسبق لبيرت Pretraining BERT

مع نموذج BERT المنفذ في القسم 15.8 وأمثلة التدريب المسبق التي تم إنشاؤها من مجموعة بيانات WikiText-2 في القسم 15.9، سنقوم بإجراء اختبار BERT مسبقًا على مجموعة بيانات WikiText-2 في هذا القسم.

```
from mxnet import autograd, gluon, init, np, npx
from d2l import mxnet as d2l

npx.set_np()
```

للبدء، نقوم بتحميل مجموعة بيانات WikiText-2 كمجموعات صغيرة من أمثلة التدريب المسبق لنمذجة اللغة المقنعة masked language modeling وتنبؤ الجملة التالية next sentence prediction. حجم الدُفعة هو 512 والحد الأقصى لطول تسلسل إدخال BERT هو 64. لاحظ أنه في نموذج BERT الأصلي، يكون الحد الأقصى للطول هو 512.

```
batch_size, max_len = 512, 64
train_iter, vocab = d2l.load_data_wiki(batch_size,
max_len)
```

15.10.1. التدريب المسبق لبيرت Pretraining BERT

يحتوي BERT الأصلي على نسختين بأحجام مختلفة للنموذج (Devlin et al.، 2018). يستخدم النموذج الأساسي ($\text{BERT}_{\text{BASE}}$) 12 طبقة (كتل تشفير المحولات transformer encoder blocks) مع 768 وحدة مخفية (حجم مخفي hidden size) و12 رأس انتباه ذاتي self-attention heads. النموذج الكبير ($\text{BERT}_{\text{LARGE}}$) يستخدم 24 طبقة مع 1024 وحدة مخفية و16 رأس انتباه ذاتي. والجدير بالذكر أن الأول يحتوي على 110 ملايين معلمة بينما يحتوي الأخير على 340 مليون معلمة. للتوضيح بسهولة، نحدد BERT صغيرًا، باستخدام طبقتين، و128 وحدة مخفية، ورأسين للانتباه الذاتي.

```
net = d2l.BERTModel(len(vocab), num_hiddens=128,
ffn_num_hiddens=256,
                    num_heads=2, num_blks=2,
dropout=0.2)
devices = d2l.try_all_gpus()
net.initialize(init.Xavier(), ctx=devices)
```

```
loss = gluon.loss.SoftmaxCELoss()
```

قبل تحديد حلقة التدريب، نحدد الدالة المساعدة `_get_batch_loss_bert`. بالنظر إلى جزء من أمثلة التدريب، تحسب هذه الدالة الخسارة لكل من نمذجة اللغة المقنعة ومهام التنبؤ بالجمل التالية. لاحظ أن الخسارة النهائية للتدريب المسبق لـ BERT هي مجرد مجموع كل من خسارة نمذجة اللغة المقنعة وخسارة توقع الجملة التالية.

```
#@save
def _get_batch_loss_bert(net, loss, vocab_size,
tokens_X_shards,
                         segments_X_shards,
valid_lens_x_shards,
                         pred_positions_X_shards,
mlm_weights_X_shards,
                         mlm_Y_shards, nsp_y_shards):
    mlm_ls, nsp_ls, ls = [], [], []
    for (tokens_X_shard, segments_X_shard,
valid_lens_x_shard,
         pred_positions_X_shard, mlm_weights_X_shard,
mlm_Y_shard,
         nsp_y_shard) in zip(
        tokens_X_shards, segments_X_shards,
valid_lens_x_shards,
        pred_positions_X_shards, mlm_weights_X_shards,
mlm_Y_shards,
        nsp_y_shards):
        # Forward pass
        _, mlm_Y_hat, nsp_Y_hat = net(
            tokens_X_shard, segments_X_shard,
valid_lens_x_shard.reshape(-1),
            pred_positions_X_shard)
        # Compute masked language model loss
        mlm_l = loss(
            mlm_Y_hat.reshape((-1, vocab_size)),
mlm_Y_shard.reshape(-1),
            mlm_weights_X_shard.reshape((-1, 1)))
        mlm_l = mlm_l.sum() / (mlm_weights_X_shard.sum()
+ 1e-8)
        # Compute next sentence prediction loss
        nsp_l = loss(nsp_Y_hat, nsp_y_shard)
        nsp_l = nsp_l.mean()
        mlm_ls.append(mlm_l)
```

```
            nsp_ls.append(nsp_l)
            ls.append(mlm_l + nsp_l)
            npx.waitall()
        return mlm_ls, nsp_ls, ls
```

من خلال استدعاء دالتي المساعد المذكورتين أعلاه، تحدد دالة `train_bert` التالية إجراء التدريب المسبق (`net`) BERT على مجموعة بيانات (`train_iter`) WikiText-2. يمكن أن يستغرق تدريب BERT وقتًا طويلاً. بدلاً من تحديد عدد فترات التدريب كما هو الحال في دالة `train_ch13` (انظر القسم 14.1) ، يحدد إدخال `num_steps` للدالة التالية عدد خطوات التكرار للتدريب.

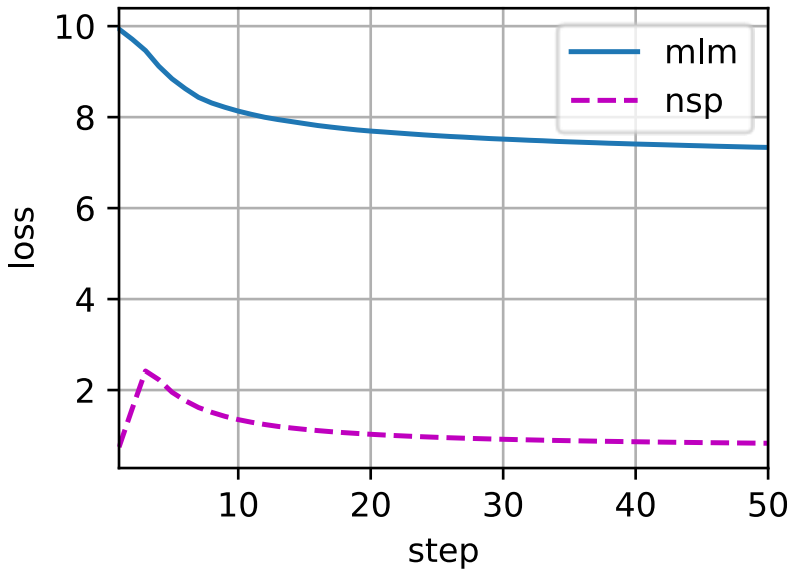
```
def train_bert(train_iter, net, loss, vocab_size,
devices, num_steps):
    trainer = gluon.Trainer(net.collect_params(),
'adam',
                            {'learning_rate': 0.01})
    step, timer = 0, d2l.Timer()
    animator = d2l.Animator(xlabel='step',
ylabel='loss',
                            xlim=[1, num_steps],
legend=['mlm', 'nsp'])
    # Sum of masked language modeling losses, sum of
next sentence prediction
    # losses, no. of sentence pairs, count
    metric = d2l.Accumulator(4)
    num_steps_reached = False
    while step < num_steps and not num_steps_reached:
        for batch in train_iter:
            (tokens_X_shards, segments_X_shards,
valid_lens_x_shards,
             pred_positions_X_shards,
mlm_weights_X_shards,
             mlm_Y_shards, nsp_y_shards) =
[gluon.utils.split_and_load(
                elem, devices, even_split=False) for
elem in batch]
            timer.start()
            with autograd.record():
                mlm_ls, nsp_ls, ls =
_get_batch_loss_bert(
                    net, loss, vocab_size,
tokens_X_shards, segments_X_shards,
```

```
                    valid_lens_x_shards,
pred_positions_X_shards,
                    mlm_weights_X_shards, mlm_Y_shards,
nsp_y_shards)
            for l in ls:
                l.backward()
            trainer.step(1)
            mlm_l_mean = sum([float(l) for l in mlm_ls])
/ len(mlm_ls)
            nsp_l_mean = sum([float(l) for l in nsp_ls])
/ len(nsp_ls)
            metric.add(mlm_l_mean, nsp_l_mean,
batch[0].shape[0], 1)
            timer.stop()
            animator.add(step + 1,
                         (metric[0] / metric[3],
metric[1] / metric[3]))
            step += 1
            if step == num_steps:
                num_steps_reached = True
                break

    print(f'MLM loss {metric[0] / metric[3]:.3f}, '
          f'NSP loss {metric[1] / metric[3]:.3f}')
    print(f'{metric[2] / timer.sum():.1f} sentence
pairs/sec on '
          f'{str(devices)}')
```

يمكننا رسم كل من خسارة نمذجة اللغة المقنعة وخسارة توقع الجملة التالية أثناء تدريب BERT.

```
train_bert(train_iter, net, loss, len(vocab), devices,
50)
```

```
MLM loss 7.334, NSP loss 0.831
7321.7 sentence pairs/sec on [gpu(0), gpu(1)]
```



15.10.2. تمثيل النص باستخدام بيرت Representing Text with BERT

بعد التدريب المسبق لـ BERT، يمكننا استخدامه لتمثيل نص واحد أو أزواج نصية أو أي رمز فيها. تقوم الدالة التالية بإرجاع تمثيلات BERT (net) لجميع الرموزفي tokens_a وtokens_b.

```
def get_bert_encoding(net, tokens_a, tokens_b=None):
    tokens, segments =
d2l.get_tokens_and_segments(tokens_a, tokens_b)
    token_ids = np.expand_dims(np.array(vocab[tokens],
ctx=devices[0]),
                               axis=0)
    segments = np.expand_dims(np.array(segments,
ctx=devices[0]), axis=0)
    valid_len = np.expand_dims(np.array(len(tokens),
ctx=devices[0]), axis=0)
    encoded_X, _, _ = net(token_ids, segments,
valid_len)
    return encoded_X
```

تأمل الجملة "a crane is flying". راجع تمثيل مدخلات BERT كما تمت مناقشتهفي القسم 15.8.4. بعد إدخال الرموز الخاصة "<cls>" (المستخدمة للتصنيف) و "<sep>" (المستخدمة للفصل)، يبلغ طول تسلسل إدخال BERT ستة. نظرًا لأن الصفر هو فهرس الرمز "<cls>"، فإن

`encoded_text[:, 0, :]` هو تمثيل BERT لجملة الإدخال بأكملها. لتقييم الرمز متعدد المعاني "crane"، نطبع أيضًا العناصر الثلاثة الأولى لتمثيل BERT للرمز.

```
tokens_a = ['a', 'crane', 'is', 'flying']
encoded_text = get_bert_encoding(net, tokens_a)
# Tokens: '<cls>', 'a', 'crane', 'is', 'flying', '<sep>'
encoded_text_cls = encoded_text[:, 0, :]
encoded_text_crane = encoded_text[:, 2, :]
encoded_text.shape, encoded_text_cls.shape,
encoded_text_crane[0][:3]
```

```
((1, 6, 128),
 (1, 128),
 array([ 1.0648314,  1.4775515, -1.3542705],
ctx=gpu(0)))
```

الآن فكر في زوج الجملة "a crane driver came" و "he just left". وبالمثل، فإن `encoded_pair[:, 0, :]` هي النتيجة المشفرة لزوج الجملة بالكامل من BERT المُدرب مسبقًا. لاحظ أن العناصر الثلاثة الأولى من رمز تعدد المعاني "crane" تختلف عن تلك التي تختلف عندما يكون السياق مختلفًا. هذا يدعم أن تمثيلات BERT حساسة للسياق context-sensitive.

```
tokens_a, tokens_b = ['a', 'crane', 'driver', 'came'],
['he', 'just', 'left']
encoded_pair = get_bert_encoding(net, tokens_a,
tokens_b)
# Tokens: '<cls>', 'a', 'crane', 'driver', 'came',
'<sep>', 'he', 'just',
# 'left', '<sep>'
encoded_pair_cls = encoded_pair[:, 0, :]
encoded_pair_crane = encoded_pair[:, 2, :]
encoded_pair.shape, encoded_pair_cls.shape,
encoded_pair_crane[0][:3]
```

```
((1, 10, 128),
 (1, 128),
 array([ 1.0649579,  1.4775581, -1.3542051],
ctx=gpu(0)))
```

في القسم 16، سنقوم بضبط نموذج BERT الذي تم اختباره مسبقًا لتطبيقات معالجة اللغة الطبيعية المصب downstream natural language processing applications.

15.10.3. الملخص

- يحتوي BERT الأصلي على نسختين ، حيث يحتوي النموذج الأساسي على 110 مليون معلمة والنموذج الكبير يحتوي على 340 مليون معلمة.
- بعد التدريب المسبق لـ BERT ، يمكننا استخدامه لتمثيل نص واحد أو أزواج نصية أو أي رمز فيها.
- في التجربة، يكون للرمز نفسه تمثيل BERT مختلف عندما تختلف سياقاتهم. هذا يدعم أن تمثيلات BERT حساسة للسياق.

15.10.4. التمارين

1. في التجربة ، يمكننا أن نرى أن خسارة نمذجة اللغة المقنعة أعلى بكثير من خسارة توقع الجملة التالية. لماذا؟
2. اضبط الحد الأقصى لطول تسلسل إدخال BERT ليكون 512 (مثل نموذج BERT الأصلي). استخدم تكوينات نموذج BERT الأصلي مثل $\text{BERT}_{\text{LARGE}}$. هل واجهت أي خطأ عند تشغيل هذا القسم؟ لماذا؟

16

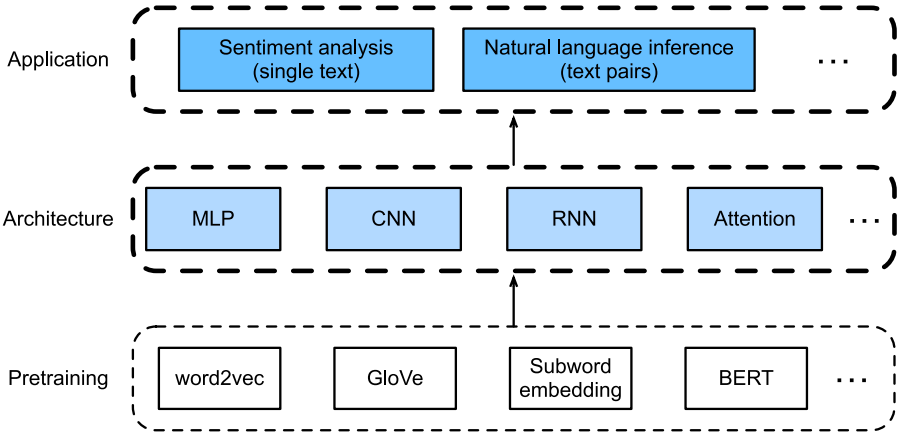
المعالجة اللغوية الطبيعية: التطبيقات

16. المعالجة اللغوية الطبيعية: التطبيقات Natural Language Processing: Applications

لقد رأينا كيفية تمثيل الرموزفي تسلسلات نصية وتدريب تمثيلاتهافي القسم 15. يمكن تغذية تمثيلات النص المدربة مسبقًا لنماذج مختلفة لمهام المعالجة اللغوية الطبيعية المختلفة.

في الواقع، لقد ناقشت الفصول السابقة بالفعل بعض تطبيقات المعالجة اللغوية الطبيعية دون الحاجة إلى تدريب مسبق، فقط لشرح هياكل التعلم العميق. على سبيل المثال،في القسم 9، اعتمدنا على RNNs لتصميم نماذج اللغة لإنشاء نص يشبه الرواية.في القسم 10 والقسم 11، قمنا أيضًا بتصميم نماذج تستند إلى RNNs وآليات الانتباه للترجمة الآلية.

ومع ذلك، لا ينوي هذا الكتاب تغطية جميع هذه التطبيقات بطريقة شاملة. بدلاً من ذلك، ينصب تركيزنا على كيفية تطبيق التعلم التمثيلي representation learning (العميق) للغات لمعالجة مشاكل المعالجة اللغوية الطبيعية. بالنظر إلى تمثيلات النص المدربة مسبقًا، سيستكشف هذا الفصل مهمتين شائعتين وتمثيليتين للمعالجة اللغوية الطبيعية: تحليل المشاعر sentiment analysis واستنتاج اللغة الطبيعية natural language inference، والتي تحلل النص الفردي وعلاقات أزواج النص، على التوالي.



الشكل 16.1 يمكن تغذية تمثيلات النص المدربة مسبقًا إلى العديد من بُنيات التعلم العميق لمختلف تطبيقات معالجة اللغة الطبيعيةفي المصب. يركز هذا الفصل على كيفية تصميم نماذج لتطبيقات المعالجة اللغوية الطبيعية المختلفة.

كما هو موضحفي الشكل 16.1، يركز هذا الفصل على وصف الأفكار الأساسية لتصميم نماذج المعالجة اللغوية الطبيعية باستخدام أنواع مختلفة من بُنى التعلم العميق، مثل MLPs و CNNs

و RNNs والانتباه attention. على الرغم من أنه من الممكن دمج أي تمثيلات نصية سابقة التدريب مع أي معمارية لأي تطبيقفي الشكل 16.1، فإننا نختار بعض التوليفات التمثيلية. على وجه التحديد، سوف نستكشف البنى الشعبية القائمة على شبكات RNN وCNN لتحليل المشاعر. لاستدلال اللغة الطبيعية، نختار الانتباه وMLPs لشرح كيفية تحليل أزواج النص.في النهاية، نقدم كيفية ضبط نموذج BERT مسبق التدريب لمجموعة واسعة من تطبيقات المعالجة اللغوية الطبيعية، على سبيل المثال على مستوى التسلسل sequence level (تصنيف النص الفردي وتصنيف أزواج النص) ومستوى الرمز token level (وضع علامات على النص والإجابة على الأسئلة). كحالة تجريبية ملموسة، سنقوم بضبط BERT لاستنتاج اللغة الطبيعية.

كما قدمنافي القسم 15.8، تتطلب BERT تغييرات طفيفةفي البُنية لمجموعة واسعة من تطبيقات معالجة اللغة الطبيعية. ومع ذلك، تأتي هذه الميزة على حساب ضبط عدد كبير من معلمات BERT لتطبيقات المصب. عندما تكون المساحة أو الوقت محدودة، فإن تلك النماذج المصممة بناءً على MLPs وCNNs وRNNs والاهتمام تكون أكثر جدوى. فيما يلي، نبدأ بتطبيق تحليل المشاعر ونوضح تصميم النموذج بناءً على شبكات RNN وCNN، على التوالي.

16.1. تحليل المشاعر ومجموعة البيانات Sentiment Analysis and the Dataset

مع انتشار وسائل التواصل الاجتماعي ومنصات المراجعة على الإنترنت، تم تسجيل عدد كبير من البيانات ذات الرأي opinionated data، مما يحمل إمكانات كبيرة لدعم عمليات صنع القرار. يدرس تحليل المشاعر Sentiment analysis مشاعر الناسفي نصوصهم المنتجة، مثل مراجعات المنتجات product reviews وتعليقات المدونة blog comments ومناقشات المنتدى forum discussions. يتمتع بتطبيقات واسعةفي مجالات متنوعة مثل السياسة politics (على سبيل المثال، تحليل المشاعر العامة تجاه السياسات)، والتمويل finance (على سبيل المثال، تحليل مشاعر السوق)، والتسويق marketing (على سبيل المثال، البحث عن المنتجات وإدارة العلامات التجارية).

نظرًا لأنه يمكن تصنيف المشاعر على أنها أقطاب polarities أو مقاييس scales منفصلة (على سبيل المثال، إيجابية وسلبية)، يمكننا اعتبار تحليل المشاعر مهمة تصنيف نص، والتي تحول تسلسل نص متغير الطول إلى فئة نصية بطول ثابت.في هذا الفصل، سنستخدم مجموعة بيانات مراجعة الأفلام الكبيرة (large movie review dataset) من ستانفورد لتحليل المشاعر. يتكون من مجموعة تدريب ومجموعة اختبار، تحتوي إما على 25000 مراجعة فيلم تم تنزيلها

من IMDb.في كلتا مجموعتي البيانات، هناك عدد متساوٍ من التسميات "positive" و "negative"، مما يشير إلى استقطاب المشاعر المختلفة.

```
import os
from mxnet import np, npx
from d2l import mxnet as d2l

npx.set_np()
```

16.1.1. قراءة مجموعة البيانات Reading the Dataset

أولاً، قم بتنزيل واستخراج مجموعة بيانات مراجعة IMDb هذهفي المسار `../data/aclImdb`.

```
#@save
d2l.DATA_HUB['aclImdb'] = (d2l.DATA_URL +
'aclImdb_v1.tar.gz',

'01ada507287d82875905620988597833ad4e0903')

data_dir = d2l.download_extract('aclImdb', 'aclImdb')
```

```
Downloading ../data/aclImdb_v1.tar.gz from http://d2l-
data.s3-accelerate.amazonaws.com/aclImdb_v1.tar.gz...
```

بعد ذلك، اقرأ التدريب واختبار مجموعات البيانات. كل مثال عبارة عن مراجعة وتسميتها: 1 لكلمة "positive" و0 لكلمة "negative".

```
#@save
def read_imdb(data_dir, is_train):
    """Read the IMDb review dataset text sequences and
labels."""
    data, labels = [], []
    for label in ('pos', 'neg'):
        folder_name = os.path.join(data_dir, 'train' if
is_train else 'test',
                                   label)
        for file in os.listdir(folder_name):
            with open(os.path.join(folder_name, file),
'rb') as f:
                review = f.read().decode('utf-
8').replace('\n', '')
                data.append(review)
```

```
                labels.append(1 if label == 'pos' else
0)
    return data, labels

train_data = read_imdb(data_dir, is_train=True)
print('# trainings:', len(train_data[0]))
for x, y in zip(train_data[0][:3], train_data[1][:3]):
    print('label:', y, 'review:', x[:60])
```

```
# trainings: 25000
label: 1 review: Henry Hathaway was daring, as well as
enthusiastic, for his
label: 1 review: An unassuming, subtle and lean film,
"The Man in the White S
label: 1 review: Eddie Murphy really made me laugh my
ass off on this HBO sta
```

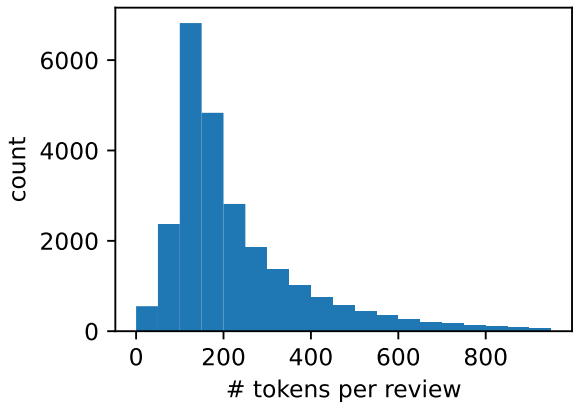
16.1.2. المعالجة المسبقة لمجموعة البيانات Preprocessing the Dataset

معالجة كل كلمة كرمز وتصفية الكلمات التي تظهر أقل من 5 مرات، نقوم بإنشاء مفردات من مجموعة بيانات التدريب.

```
train_tokens = d2l.tokenize(train_data[0], token='word')
vocab = d2l.Vocab(train_tokens, min_freq=5,
reserved_tokens=['<pad>'])
```

بعد الترميز، دعنا نرسم المدرج التكراري histogram لأطوال المراجعة بالرموز tokens.

```
d2l.set_figsize()
d2l.plt.xlabel('# tokens per review')
d2l.plt.ylabel('count')
d2l.plt.hist([len(line) for line in train_tokens],
bins=range(0, 1000, 50));
```



كما توقعنا، فإن المراجعات لها أطوال متفاوتة. لمعالجة دفعة صغيرة من هذه المراجعات في كل مرة، قمنا بتعيين طول كل مراجعة على 500 مع الاقتطاع truncation والحشو padding، وهو ما يشبه خطوة المعالجة المسبقة لمجموعة بيانات الترجمة الآلية machine translation dataset في القسم 10.5.

```
num_steps = 500  # sequence length
train_features = np.array([d2l.truncate_pad(
    vocab[line], num_steps, vocab['<pad>']) for line in
train_tokens])
print(train_features.shape)
```

```
(25000, 500)
```

16.1.3. إنشاء مكررات البيانات Creating Data Iterators

الآن يمكننا إنشاء مكررات البيانات. في كل تكرار، يتم إرجاع دفعة صغيرة من الأمثلة.

```
train_iter = d2l.load_array((train_features,
train_data[1]), 64)

for X, y in train_iter:
    print('X:', X.shape, ', y:', y.shape)
    break
print('# batches:', len(train_iter))
```

```
X: (64, 500) , y: (64,)
# batches: 391
```

16.1.4. وضع كل شيء معا Putting It All Together

أخيرًا، نختتم الخطوات المذكورة أعلاه في دالة `load_data_imdb`. تقوم بإرجاع مكررات بيانات التدريب والاختبار ومفردات مجموعة بيانات مراجعة IMDb.

```
#@save
def load_data_imdb(batch_size, num_steps=500):
    """Return data iterators and the vocabulary of the
IMDb review dataset."""
    data_dir = d2l.download_extract('aclImdb',
'aclImdb')
    train_data = read_imdb(data_dir, True)
    test_data = read_imdb(data_dir, False)
    train_tokens = d2l.tokenize(train_data[0],
token='word')
    test_tokens = d2l.tokenize(test_data[0],
token='word')
```

```
    vocab = d2l.Vocab(train_tokens, min_freq=5)
    train_features = np.array([d2l.truncate_pad(
        vocab[line], num_steps, vocab['<pad>']) for line
in train_tokens])
    test_features = np.array([d2l.truncate_pad(
        vocab[line], num_steps, vocab['<pad>']) for line
in test_tokens])
    train_iter = d2l.load_array((train_features,
train_data[1]), batch_size)
    test_iter = d2l.load_array((test_features,
test_data[1]), batch_size,
                               is_train=False)
    return train_iter, test_iter, vocab
```

16.1.5. الملخص

- يدرس تحليل المشاعر Sentiment analysis مشاعر الناسفي نصهم المنتج، والذي يعتبر بمثابة مشكلة تصنيف النص التي تحول تسلسل نص متفاوت الطول إلى فئة نصية ذات طول ثابت.
- بعد المعالجة المسبقة preprocessing، يمكننا تحميل مجموعة بيانات مراجعة الأفلام الكبيرةفي ستانفورد (مجموعة بيانات مراجعة IMDb)في مكررات البيانات مع مفردات.

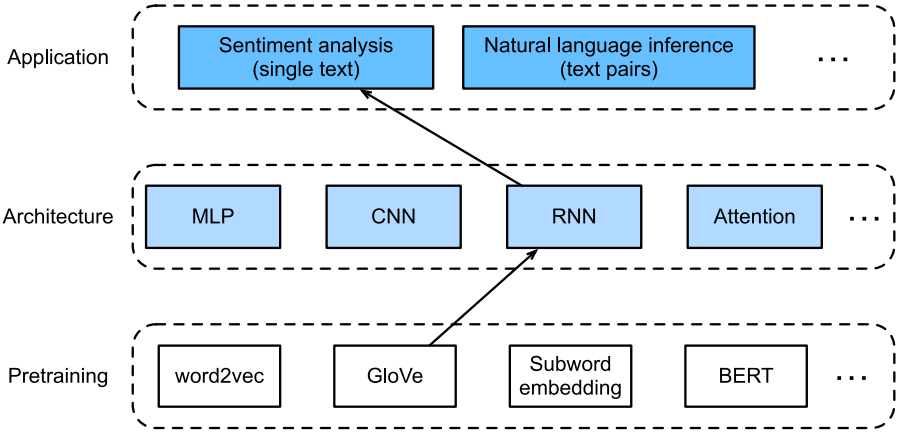
16.1.6. التمارين

1. ما هي المعلمات الفائقةفي هذا القسم التي يمكننا تعديلها لتسريع نماذج تحليل مشاعر التدريب؟
2. هل يمكنك تنفيذ دالة لتحميل مجموعة البيانات الخاصة بمراجعات Amazonفي مكررات البيانات والتسميات لتحليل المشاعر؟

16.2. تحليل المشاعر: استخدام الشبكات العصبية المتكررة Sentiment Analysis: Using Recurrent Neural Networks

مثل مهام تشابه الكلمات والقياس، يمكننا أيضًا تطبيق متجهات الكلمات المحددة مسبقًا لتحليل المشاعر. نظرًا لأن مجموعة بيانات مراجعة IMDbفي القسم 16.1 ليست كبيرة جدًا، فإن استخدام تمثيلات النص التي تم اختبارها مسبقًافي مجموعة كبيرة الحجم قد يقلل من فرط التخصيص overfitting النموذج. كمثال محدد موضحفي الشكل 16.2.1، سنقوم بتمثيل كل رمز باستخدام نموذج GloVe الذي تم اختباره مسبقًا ، وسنقوم بتغذية تمثيلات الرمز هذهفي RNN متعدد الطبقات ثنائي الاتجاه للحصول على تمثيل تسلسل النص ، والذي سيتم تحويله

إلى مخرجات تحليل المشاعر (Maas et al.، 2011). بالنسبة لنفس تطبيق downstream، سننظرفي خيار معماري مختلف لاحقًا.



الشكل 16.2.1 يغذي هذا القسم GloVe المُدرب مسبقًا إلى بُنية قائمة على RNN لتحليل المشاعر.

```
from mxnet import gluon, init, np, npx
from mxnet.gluon import nn, rnn
from d2l import mxnet as d2l

npx.set_np()

batch_size = 64
train_iter, test_iter, vocab =
d2l.load_data_imdb(batch_size)
```

16.2.1. تمثيل نص واحد مع RNNs Representing Single Text with RNNs

في مهام تصنيف النص، مثل تحليل المشاعر، سيتم تحويل تسلسل نص متغير الطول إلى فئات ذات طول ثابت.في فئة `BiRNN` التالية ، بينما يحصل كل رمزفي تسلسل نصي على تمثيل GloVe الفردي الذي تم اختباره مسبقًا عبر طبقة التضمين (`self.embedding`)، يتم ترميز التسلسل بالكامل بواسطة RNN ثنائي الاتجاه (`self.encoder`). بشكل أكثر تحديدًا، يتم ربط الحالات المخفية (في الطبقة الأخيرة) لـ LSTM ثنائي الاتجاهفي كل من خطوات الوقت الأولي والنهائي على أنها تمثيل لتسلسل النص. يتم بعد ذلك تحويل تمثيل النص الفردي هذا إلى فئات مخرجات بواسطة طبقة متصلة بالكامل (`self.decoder`) بمخرجين ("positive" و "negative").

```
class BiRNN(nn.Block):
    def __init__(self, vocab_size, embed_size,
num_hiddens,
                 num_layers, **kwargs):
        super(BiRNN, self).__init__(**kwargs)
        self.embedding = nn.Embedding(vocab_size,
embed_size)
        # Set `bidirectional` to True to get a
bidirectional RNN
        self.encoder = rnn.LSTM(num_hiddens,
num_layers=num_layers,
                                bidirectional=True,
input_size=embed_size)
        self.decoder = nn.Dense(2)

    def forward(self, inputs):
        # The shape of `inputs` is (batch size, no. of
time steps). Because
        # LSTM requires its input's first dimension to
be the temporal
        # dimension, the input is transposed before
obtaining token
        # representations. The output shape is (no. of
time steps, batch size,
        # word vector dimension)
        embeddings = self.embedding(inputs.T)
        # Returns hidden states of the last hidden layer
at different time
        # steps. The shape of `outputs` is (no. of time
steps, batch size,
        # 2 * no. of hidden units)
        outputs = self.encoder(embeddings)
        # Concatenate the hidden states at the initial
and final time steps as
        # the input of the fully connected layer. Its
shape is (batch size,
        # 4 * no. of hidden units)
        encoding = np.concatenate((outputs[0], outputs[-
1]), axis=1)
        outs = self.decoder(encoding)
        return outs
```

دعونا نبني RNN ثنائي الاتجاه مع طبقتين مخفيتين لتمثيل نص واحد لتحليل المشاعر.

```
embed_size, num_hiddens, num_layers, devices = 100, 100,
2, d2l.try_all_gpus()
net = BiRNN(len(vocab), embed_size, num_hiddens,
num_layers)

net.initialize(init.Xavier(), ctx=devices)
```

16.2.2. تحميل متجهات الكلمات المعرفة مسبقًا Loading Pretrained Word Vectors

أدناه نقوم بتحميل 100بُعد (يجب أن تكون متسقة مع `embed_size`)تضمينات GloVe للرموزفي المفردات.

```
glove_embedding = d2l.TokenEmbedding('glove.6b.100d')
```

اطبع شكل المتجهات لجميع الرموزفي المفردات.

```
embeds = glove_embedding[vocab.idx_to_token]
embeds.shape
```

```
(49346, 100)
```

نحن نستخدم متجهات الكلمات سابقة التدريب هذه لتمثيل الرموزفي المراجعات ولن نقوم بتحديث هذه المتجهات أثناء التدريب.

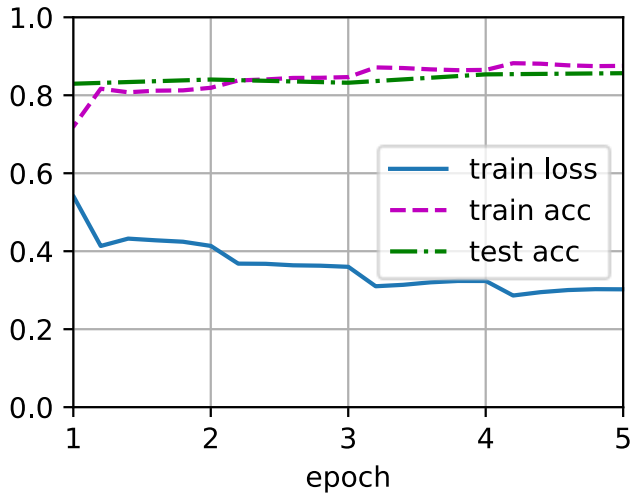
```
net.embedding.weight.set_data(embeds)
net.embedding.collect_params().setattr('grad_req',
'null')
```

16.2.3. تدريب وتقييم النموذج Training and Evaluating the Model

الآن يمكننا تدريب RNN ثنائي الاتجاه لتحليل المشاعر.

```
lr, num_epochs = 0.01, 5
trainer = gluon.Trainer(net.collect_params(), 'adam',
{'learning_rate': lr})
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs, devices)
```

```
loss 0.302, train acc 0.875, test acc 0.857
782.0 examples/sec on [gpu(0), gpu(1)]
```



نحدد الدالة التالية للتنبؤ بمشاعر تسلسل النص باستخدام النموذج المدرب net.

```
#@save
def predict_sentiment(net, vocab, sequence):
    """Predict the sentiment of a text sequence."""
    sequence = np.array(vocab[sequence.split()],
ctx=d2l.try_gpu())
    label = np.argmax(net(sequence.reshape(1, -1)),
axis=1)
    return 'positive' if label == 1 else 'negative'
```

أخيرًا، دعنا نستخدم النموذج المدرب للتنبؤ بالمشاعر لجملتين بسيطتين.

```
predict_sentiment(net, vocab, 'this movie is so great')
```

```
'positive'
```

```
predict_sentiment(net, vocab, 'this movie is so bad')
```

```
'negative'
```

16.2.4. الملخص

- يمكن أن تمثل متجهات الكلمات المحددة مسبقًا رموزًا فرديةفي تسلسل نصي.
- يمكن أن تمثل RNNs ثنائية الاتجاه تسلسلًا نصيًا، على سبيل المثال عبر تسلسل حالاتها المخفيةفي خطوات الوقت الأولية والنهائية. يمكن تحويل تمثيل النص الفردي هذا إلى فئات باستخدام طبقة متصلة بالكامل.

16.2.5. التمارين

1. قم بزيادة عدد الفترات. هل يمكنك تحسين دقة التدريب والاختبار؟ ماذا عن ضبط المعلمات الفائقة الأخرى؟
2. استخدم متجهات الكلمات الأكبر حجمًا، مثل التضمينات GloVe ذات الأبعاد 300. هل يحسن دقة التصنيف؟
3. هل يمكننا تحسين دقة التصنيف باستخدام رمز spaCy؟ تحتاج إلى تثبيت spaCy (`pip install spacy`) وتثبيت الحزمة الإنجليزية (`python -m spacy download en`). في الكود، أولاً، قم باستيراد spaCy (`import spacy`). بعد ذلك، قم بتحميل حزمة spaCy English (`spacy_en = spacy.load('en')`). أخيرًا، قم بتعريف دالة `def tokenizer(text)`:

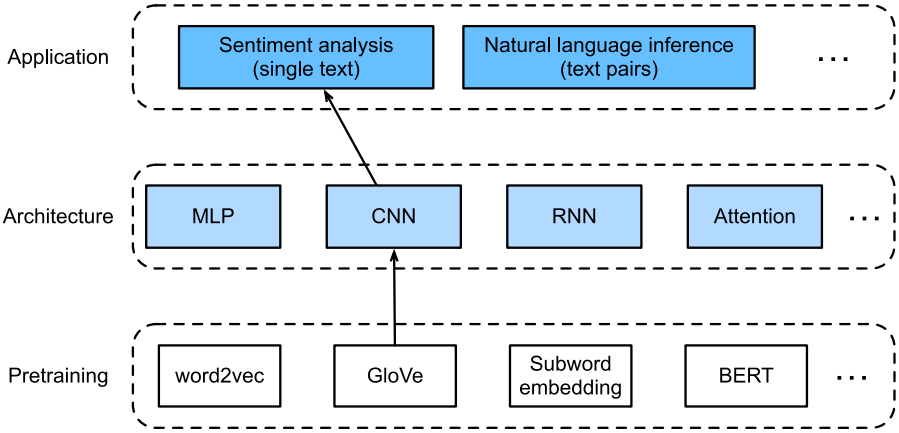
```
return [tok.text for tok in spacy_en.tokenizer(tex
t)]
```

واستبدل دالة `tokenizer` الأصلية. لاحظ الأشكال المختلفة من الرموز للعبارات في GloVe و spaCy. على سبيل المثال، تأخذ عبارة الرمز "new york" شكل "new-york" في GloVe وشكل "new york" بعد ترميز spaCy.

16.3. تحليل المشاعر: استخدام الشبكات العصبية التلافيفية Sentiment Analysis: Using Convolutional Neural Networks

في القسم 7، بحثنا في آليات معالجة بيانات الصورة ثنائية الأبعاد باستخدام شبكات CNN ثنائية الأبعاد، والتي تم تطبيقها على الميزات المحلية مثل وحدات البكسل المجاورة. على الرغم من أنها مصممة في الأصل للرؤية الحاسوبية، إلا أن شبكات CNN تستخدم أيضًا على نطاق واسع للمعالجة اللغوية الطبيعية. ببساطة، فكر في أي تسلسل نصي كصورة أحادية البعد. بهذه الطريقة، يمكن لشبكات CNN أحادية البعد معالجة الميزات المحلية مثل nـ جرام في النص.

في هذا القسم، سنستخدم نموذج textCNN لشرح كيفية تصميم بُنية CNN لتمثيل نص واحد (Kim، 2014). مقارنة بالشكل 16.2.1 الذي يستخدم بُنية RNN مع تدريب GloVe لتحليل المشاعر، يكمن الاختلاف الوحيد في الشكل 16.3.1 في اختيار البُنية.

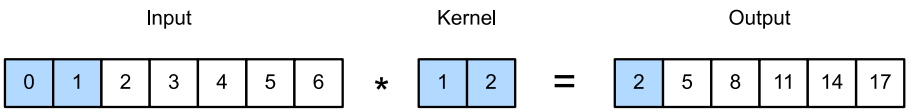


الشكل 16.3.1 يغذي هذا القسم GloVe المُدرب مسبقًا إلى بُنية قائمة على CNN لتحليل المشاعر.

```
from mxnet import gluon, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()

batch_size = 64
train_iter, test_iter, vocab =
d2l.load_data_imdb(batch_size)
```

16.3.1. الالتفافات أحادية البعد One-Dimensional Convolutions

قبل تقديم النموذج، دعنا نرى كيف يعمل الالتفاف أحادي البعد one-dimensional convolution. ضعفي اعتبارك أنها مجرد حالة خاصة من الالتفاف ثنائي الأبعاد two-dimensional convolution استنادًا إلى عملية الارتباط المتبادل cross-correlation.



الشكل 16.3.2 عملية الارتباط المتبادل أحادية البعد. الأجزاء المظللة هي أول عنصر إخراج بالإضافة إلى عناصر موتر الإدخال والنواة المستخدمة لحساب الإخراج: $0 \times 1 + 1 \times 2 = 2$.

كما هو مبين في الشكل 16.3.2، في الحالة أحادية البعد، تنزلق نافذة الالتفاف convolution window من اليسار إلى اليمين عبر موتر الإدخال. أثناء الانزلاق، يتم ضرب موتر الإدخال

الفرعي (على سبيل المثال، 0 و1في الشكل 16.3.2) الموجودفي نافذة الالتفاففي موضع معين وموتّر النواة (على سبيل المثال،1 و2في الشكل 16.3.2) بشكل عنصري. مجموع هذه عمليات الضرب يعطي القيمة القياسية المفردة (على سبيل المثال $0 \times 1 + 1 \times 2 = 2$ ،في الشكل 16.3.2)في الموضع المقابل لموتّر الاخراج.

نقوم بتنفيذ الارتباط المتبادل أحادي البعدفي دالة `corr1d` التالية. بالنظر إلى موتر الإدخال `X` وموتّر النواة `K`، فإنه يُرجع موتر الإخراج `Y`.

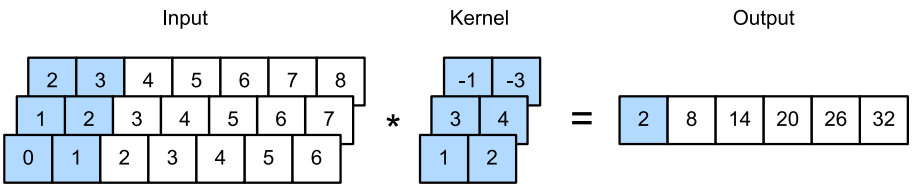
```
def corr1d(X, K):
    w = K.shape[0]
    Y = np.zeros((X.shape[0] - w + 1))
    for i in range(Y.shape[0]):
        Y[i] = (X[i: i + w] * K).sum()
    return Y
```

يمكننا إنشاء موتر الإدخال `X` وموتر النواة `K` من الشكل 16.3.2 للتحقق من صحة إخراج تنفيذ الارتباط المتبادل أحادي البعد أعلاه.

```
X, K = np.array([0, 1, 2, 3, 4, 5, 6]), np.array([1, 2])
corr1d(X, K)
```

```
array([ 2.,  5.,  8., 11., 14., 17.])
```

لأي إدخال أحادي البعد مع قنوات متعددة، تحتاج نواة الالتفاف إلى نفس عدد قنوات الإدخال. ثم بالنسبة لكل قناة، قم بإجراء عملية الارتباط المتبادل على موتر أحادي البعد للمدخل والموتر أحادي البعد لنواة الالتفاف، مع جمع النتائج عبر جميع القنوات لإنتاج موتر الإخراج أحادي البعد. يوضح الشكل 16.3.3 عملية الارتباط المتبادل أحادية البعد مع 3 قنوات دخل.



الشكل 16.3.3 عملية الارتباط المتبادل أحادية البعد بثلاث قنوات دخل. الأجزاء المظللة هي أول عنصر إخراج بالإضافة إلى عناصر موتر الإدخال والنواة المستخدمة لحساب الإخراج:

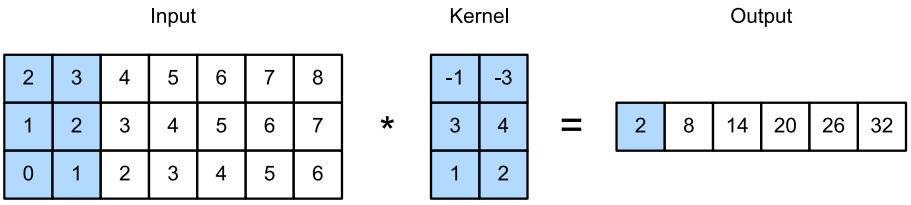
$$0 \times 1 + 1 \times 2 + 1 \times 3 + 2 \times 4 + 2 \times (-1) + 3 \times (-3) = 2.$$

يمكننا تنفيذ عملية الارتباط المتبادل أحادية البعد لقنوات الإدخال المتعددة والتحقق من صحة النتائجفي الشكل 16.3.3.

```
def corr1d_multi_in(X, K):
```

```
    # First, iterate through the 0th dimension (channel
dimension) of `X` and
    # `K`. Then, add them together
    return sum(corr1d(x, k) for x, k in zip(X, K))

X = np.array([[0, 1, 2, 3, 4, 5, 6],
              [1, 2, 3, 4, 5, 6, 7],
              [2, 3, 4, 5, 6, 7, 8]])
K = np.array([[1, 2], [3, 4], [-1, -3]])
corr1d_multi_in(X, K)
array([ 2.,  8., 14., 20., 26., 32.])
```

لاحظ أن الارتباطات المتبادلة أحادية البعد متعددة المدخلات تعادل الارتباطات المتبادلة ثنائية الأبعاد ذات المدخل الفردي. للتوضيح، شكل مكافئ للارتباط المتبادل أحادي البعد لقناة متعددة المدخلاتفي الشكل 16.3.3 هو الارتباط المتبادل ثنائي الأبعاد لقناة المدخلات الأحاديةفي الشكل 16.3.4، حيث ارتفاع يجب أن تكون نواة الالتفاف هي نفسها الخاصة بموتر الإدخال.



الشكل 16.3.4 عملية الارتباط المتبادل ثنائي الأبعاد بقناة إدخال واحدة. الأجزاء المظللة هي أول عنصر إخراج بالإضافة إلى عناصر موتر الإدخال والنواة المستخدمة لحساب الإخراج:

$$2 \times (-1) + 3 \times (-3) + 1 \times 3 + 2 \times 4 + 0 \times 1 + 1 \times 2 = 2.$$

كل من المخرجاتفي الشكل 16.3.2 والشكل 16.3.3 لها قناة واحدة فقط. تمامًا مثل الالتفافات ثنائية الأبعاد مع قنوات الإخراج المتعددة الموضحةفي القسم 7.4.2، يمكننا أيضًا تحديد قنوات إخراج متعددة للالتفافات أحادية البعد.

16.3.2 تجميع الحد الأقصى بمرور الوقت Max-Over-Time Pooling

وبالمثل، يمكننا استخدام التجميع pooling لاستخراج أعلى قيمة من تمثيلات التسلسل كأهم ميزة عبر الخطوات الزمنية. يعمل تجميع الحد الأقصى بمرور الوقت max-over-time pooling المستخدمفي textCNN مثل التجميع العالمي أحادي البعد (Collobert et al.، 2011). بالنسبة للإدخال متعدد القنوات حيث تخزن كل قناة القيمفي خطوات زمنية مختلفة،

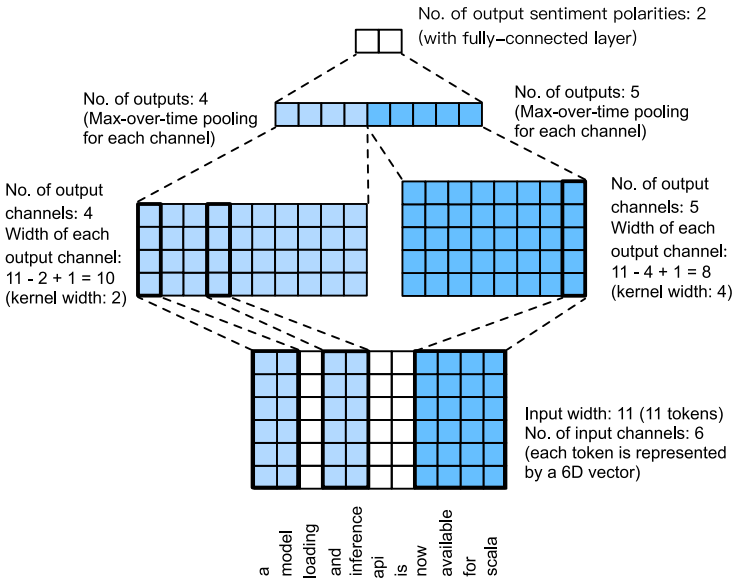
يكون الإخراجفي كل قناة هو القيمة القصوى لتلك القناة. لاحظ أن تجميع الحد الأقصى بمرور الوقت يسمح بعدد مختلف من الخطوات الزمنيةفي قنوات مختلفة.

16.3.3. نموذج textCNN

باستخدام الالتفاف أحادي البعد وتجميع الحد الأقصى للوقت، يأخذ نموذج textCNN تمثيلات رمزية فردية مُدَرَّبة مسبقًا كمدخلات، ثم يحصل على تمثيلات التسلسل ويحولها لتطبيق downstream.

بالنسبة لتسلسل نصي واحد مع الرموز n التي يتم تمثيلها بواسطة متجهات الأبعاد d ، يكون عرض، وارتفاع، وعدد قنوات موتر الإدخال هي n ،1 وd ، على التوالي. يحول نموذج textCNN الإدخال إلى الإخراج على النحو التالي:

1. قم بتحديد نوى التفاف متعددة أحادية البعد وإجراء عمليات التفاف بشكل منفصل على المدخلات. قد تلتقط نوى الالتفاف ذات العروض المختلفة ميزات محلية بين أعداد مختلفة من الرموز المجاورة.
2. قم بإجراء تجميع حد أقصى للوقت على جميع قنوات الإخراج، ثم قم بتجميع جميع مخرجات التجميع القياسي scalar pooling outputs كمتجه.
3. قم بتحويل المتجه المتسلسل إلى فئات الإخراج باستخدام الطبقة المتصلة بالكامل. يمكن استخدام التسرب Dropout لتقليل فرط التجهيز overfitting.



الشكل 16.3.5. البُنية النموذجية لـ textCNN.

يوضح الشكل 16.3.5 بُنية نموذج textCNN مع مثال ملموس. الإدخال عبارة عن جملة تحتوي على 11 رمزًا، حيث يتم تمثيل كل رمز من خلال متجهات سداسية الأبعاد. إذن لدينا مدخل من 6 قنوات بعرض 11. حدد نواتين التفاف أحادي البعد بعرض 2 و4، مع 4 و5 قنوات إخراج، على التوالي. إنها تنتج 4 قنوات إخراج بعرض $10 = 1 + 2 - 11$ و5 قنوات إخراج بعرض $8 = 1 + 4 - 11$. على الرغم من العروض المختلفة لهذه القنوات التسعة، إلا أن الحد تجميع الحد الأقصى بمرور الوقت يعطي متجهًا متسلسلًا مكونًا من 9 أبعاد، والذي يتم تحويله أخيرًا إلى متجه إخراج ثنائي الأبعاد لتنبؤات المشاعر الثنائية.

16.3.3.1. تعريف النموذج Defining the Model

نقوم بتنفيذ نموذج textCNNفي الفئة التالية. بالمقارنة مع نموذج RNN ثنائي الاتجاهفي القسم 16.2، بالإضافة إلى استبدال الطبقات المتكررة بطبقات تلافيفية، نستخدم أيضًا طبقتين من التضمين: واحدة بأوزان قابلة للتدريب والأخرى بأوزان ثابتة.

```
class TextCNN(nn.Block):
    def __init__(self, vocab_size, embed_size,
kernel_sizes, num_channels,
                 **kwargs):
        super(TextCNN, self).__init__(**kwargs)
        self.embedding = nn.Embedding(vocab_size,
embed_size)
        # The embedding layer not to be trained
        self.constant_embedding =
nn.Embedding(vocab_size, embed_size)
        self.dropout = nn.Dropout(0.5)
        self.decoder = nn.Dense(2)
        # The max-over-time pooling layer has no
parameters, so this instance
        # can be shared
        self.pool = nn.GlobalMaxPool1D()
        # Create multiple one-dimensional convolutional
layers
        self.convs = nn.Sequential()
        for c, k in zip(num_channels, kernel_sizes):
            self.convs.add(nn.Conv1D(c, k,
activation='relu'))

    def forward(self, inputs):
        # Concatenate two embedding layer outputs with
shape (batch size, no.
```

```
        # of tokens, token vector dimension) along
vectors
        embeddings = np.concatenate((
            self.embedding(inputs),
self.constant_embedding(inputs)), axis=2)
        # Per the input format of one-dimensional
convolutional layers,
        # rearrange the tensor so that the second
dimension stores channels
        embeddings = embeddings.transpose(0, 2, 1)
        # For each one-dimensional convolutional layer,
after max-over-time
        # pooling, a tensor of shape (batch size, no. of
channels, 1) is
        # obtained. Remove the last dimension and
concatenate along channels
        encoding = np.concatenate([
            np.squeeze(self.pool(conv(embeddings)),
axis=-1)
            for conv in self.convs], axis=1)
        outputs = self.decoder(self.dropout(encoding))
        return outputs
```

فلنقم بإنشاء مثيل textCNN. يحتوي على 3 طبقات تلافيفية بعرض النواة 3 و 4 و 5، وكلها تحتوي على 100 قناة إخراج.

```
embed_size, kernel_sizes, nums_channels = 100, [3, 4,
5], [100, 100, 100]
devices = d2l.try_all_gpus()
net = TextCNN(len(vocab), embed_size, kernel_sizes,
nums_channels)
net.initialize(init.Xavier(), ctx=devices)
```

16.3.3.2. تحميل متجهات الكلمات المدربة مسبقًا Loading Pretrained Word Vectors

كما هو الحال في القسم 16.2، نقوم بتحميل تضمينات GloVe المكونة من 100 بُعد والتي تم تدريبها مسبقًا كتمثيلات رمزية مهيأة. سيتم تدريب تمثيلات الرمز (أوزان التضمين embedding weights) في `embedding` وتثبيتها في `constant_embedding`.

```
glove_embedding = d2l.TokenEmbedding('glove.6b.100d')
embeds = glove_embedding[vocab.idx_to_token]
net.embedding.weight.set_data(embeds)
```

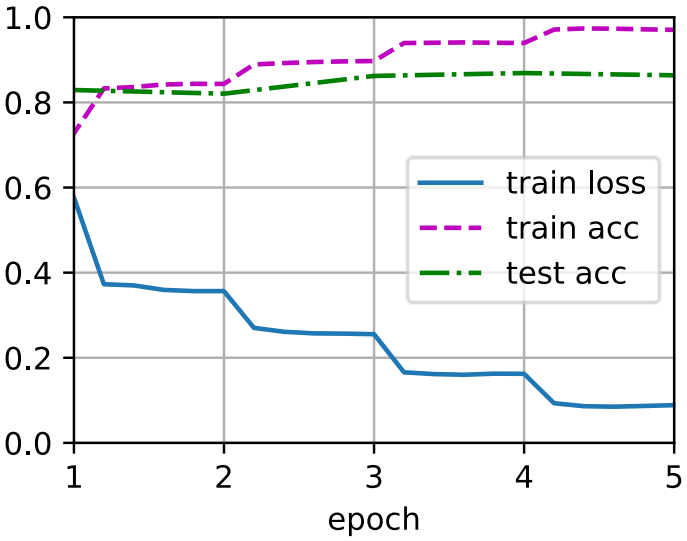
```
net.constant_embedding.weight.set_data(embeds)
net.constant_embedding.collect_params().setattr('grad_re
q', 'null')
```

16.3.3.3. تدريب وتقييم النموذج Training and Evaluating the Model

الآن يمكننا تدريب نموذج textCNN لتحليل المشاعر.

```
lr, num_epochs = 0.001, 5
trainer = gluon.Trainer(net.collect_params(), 'adam',
{'learning_rate': lr})
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs, devices)
```

```
loss 0.089, train acc 0.970, test acc 0.864
3701.4 examples/sec on [gpu(0), gpu(1)]
```



أدناه نستخدم النموذج المدرب للتنبؤ بالمشاعر لجملتين بسيطتين.

```
d2l.predict_sentiment(net, vocab, 'this movie is so
great')
```

```
'positive'
```

```
d2l.predict_sentiment(net, vocab, 'this movie is so
bad')
```

```
'negative'
```

16.3.4. الملخص

- يمكن لشبكات CNN أحادية البعد معالجة الميزات المحلية مثل $n-$grams في النص.
- تعد الارتباطات المتبادلة أحادية البعد متعددة المدخلات مكافئة للارتباطات المتبادلة ثنائية الأبعاد ذات المدخل الفردي.
- يسمح تجميع الحد الأقصى بمرور الوقت بأعداد مختلفة من الخطوات الزمنيةفي قنوات مختلفة.
- يحول نموذج textCNN تمثيلات الرموز الفردية إلى مخرجات تطبيق downstream باستخدام طبقات تلافيفية أحادية البعد وطبقات تجميع الحد الاقصى بمرور الوقت.

16.3.5. التمارين

1. قم بضبط المعلمات الفائقة وقارن بُنيتي تحليل المشاعرفي القسم 16.2 وفي هذا القسم، مثل دقة التصنيف والكفاءة الحسابية.
2. هل يمكنك زيادة تحسين دقة تصنيف النموذج باستخدام الأساليب المقدمةفي تمارين القسم 16.2؟
3. أضف الترميز الموضعي positional encodingفي تمثيلات الإدخال. هل يحسن دقة التصنيف؟

16.4. الاستنباط اللغوي الطبيعي ومجموعة البيانات Natural Language Inference and the Dataset

في القسم 16.1، ناقشنا مشكلة تحليل المشاعر. تهدف هذه المهمة إلى تصنيف تسلسل نصي واحد إلى فئات محددة مسبقًا، مثل مجموعة من استقطاب المشاعر. ومع ذلك، عندما تكون هناك حاجة لتقرير ما إذا كان يمكن استنتاج جملة ما من جملة أخرى، أو التخلص من التكرار من خلال تحديد الجمل المتكافئة لغويًا، فإن معرفة كيفية تصنيف تسلسل نصي واحد غير كافٍ. بدلاً من ذلك، نحتاج إلى أن نكون قادرين على التفكيرفي أزواج من التسلسلات النصية.

16.4.1. الاستنباط اللغوي الطبيعي Natural Language Inference

يدرس الاستنباط (الاستدلال) اللغوي الطبيعي ما إذا كان يمكن استنتاج الفرضية hypothesis من المقدمة premise، حيث يكون كلاهما تسلسل نصي. بمعنى آخر، يحدد الاستنباط اللغوي الطبيعي العلاقة المنطقية بين زوج من التسلسلات النصية. تنقسم هذه العلاقات عادةً إلى ثلاثة أنواع:

- Entailment: يمكن استنتاج الفرضية hypothesis من المقدمة premise.
- Contradiction: يمكن الاستدلال على نفي الفرضية من المقدمة.
- Neutral: جميع الحالات الأخرى.

فيما يلي مثال على التناقض contradiction حيث تشير عبارة " running the coding example" إلى "not sleeping " بدلاً من "sleeping ".

Premise: A man is running the coding example from Dive into Deep Learning.

Hypothesis: The man is sleeping.

يوضح المثال الثالث علاقة حيادية neutrality لأنه لا يمكن الاستدلال على "famous " أو " not famous " من حقيقة أن "are performing for us ".

Premise: The musicians are performing for us.

Hypothesis: The musicians are famous.

كان الاستدلال اللغوي الطبيعي موضوعًا رئيسيًا لفهم اللغة الطبيعية. يتمتع بتطبيقات واسعة تتراوح من استرجاع المعلومات إلى الإجابة على أسئلة المجال المفتوح. لدراسة هذه المشكلة، سنبدأ بالتحقيق في مجموعة بيانات مرجعية شائعة لاستدلال اللغة الطبيعية.

16.4.2. مجموعة بيانات ستانفورد لاستدلال اللغة الطبيعية (SNLI) The Stanford Natural Language Inference (SNLI) Dataset

مجموعة استدلال اللغة الطبيعية في ستانفورد (SNLI) عبارة عن مجموعة من أكثر من 500000 زوج من الجمل الإنجليزية المسماة (Bowman et al.، 2015). نقوم بتنزيل وتخزين مجموعة بيانات SNLI المستخرجة في المسار `../data/snli_1.0`.

```
import os
import re
from mxnet import gluon, np, npx
from d2l import mxnet as d2l

npx.set_np()

#@save
d2l.DATA_HUB['SNLI'] = (
```

```
'https://nlp.stanford.edu/projects/snli/snli_1.0.zip',
    '9fcde07509c7e87ec61c640c1b2753d9041758e4')

data_dir = d2l.download_extract('SNLI')
```

16.4.2.1. قراءة مجموعة البيانات Reading the Dataset

تحتوي مجموعة بيانات SNLI الأصلية على معلومات أكثر ثراءً مما نحتاجه حقًّافي تجاربنا. وبالتالي، فإننا نحدد دالة `read_snli` لاستخراج جزء فقط من مجموعة البيانات ، ثم نعيد قوائم المقدمات premises والفرضيات hypotheses وتسمياتها.

```
#@save
def read_snli(data_dir, is_train):
    """Read the SNLI dataset into premises, hypotheses,
and labels."""
    def extract_text(s):
        # Remove information that will not be used by us
        s = re.sub('\\(', '', s)
        s = re.sub('\\)', '', s)
        # Substitute two or more consecutive whitespace
with space
        s = re.sub('\\s{2,}', ' ', s)
        return s.strip()
    label_set = {'entailment': 0, 'contradiction': 1,
'neutral': 2}
    file_name = os.path.join(data_dir,
'snli_1.0_train.txt'
                             if is_train else
'snli_1.0_test.txt')
    with open(file_name, 'r') as f:
        rows = [row.split('\t') for row in
f.readlines()[1:]]
    premises = [extract_text(row[1]) for row in rows if
row[0] in label_set]
    hypotheses = [extract_text(row[2]) for row in rows
if row[0] in label_set]
    labels = [label_set[row[0]] for row in rows if
row[0] in label_set]
    return premises, hypotheses, labels
```

دعنا الآن نطبع أول 3 أزواج من المقدمة والفرضية، بالإضافة إلى تسمياتهم ("0" و "1" و "2" تتوافق مع "entailment " و "contradiction " و "neutral "، على التوالي).

```
train_data = read_snli(data_dir, is_train=True)
for x0, x1, y in zip(train_data[0][:3],
train_data[1][:3], train_data[2][:3]):
    print('premise:', x0)
    print('hypothesis:', x1)
    print('label:', y)
```

```
premise: A person on a horse jumps over a broken down
airplane .
hypothesis: A person is training his horse for a
competition .
label: 2
premise: A person on a horse jumps over a broken down
airplane .
hypothesis: A person is at a diner , ordering an
omelette .
label: 1
premise: A person on a horse jumps over a broken down
airplane .
hypothesis: A person is outdoors , on a horse .
label: 0
```

مجموعة التدريب لديها حوالي 550000 زوج، ومجموعة الاختبار بها حوالي 10000 زوج. يوضح ما يلي أن التسميات الثلاثة "entailment" و "contradiction" و "neutral" متوازنة في كل من مجموعة التدريب ومجموعة الاختبار.

```
test_data = read_snli(data_dir, is_train=False)
for data in [train_data, test_data]:
    print([[row for row in data[2]].count(i) for i in
range(3)])
```

```
[183416, 183187, 182764]
[3368, 3237, 3219]
```

16.4.2.2. تحديد فئة لتحميل مجموعة البيانات Defining a Class for Loading the Dataset

نحدد أدناه فئة لتحميل مجموعة بيانات SNLI بالوراثة من فئة `Dataset` في Gluon. تحدد الوسيطة `num_steps` في مُنشئ الفئة طول تسلسل النص بحيث يكون لكل دفعة صغيرة من التسلسلات نفس الشكل. بمعنى آخر، يتم قطع الرموز بعد `num_steps` الأولى في التسلسل الأطول، بينما يتم إلحاق الرموز الخاصة "<pad>" بالتسلسلات الأقصر حتى يصبح طولها `num_steps`. من خلال تنفيذ دالة `__getitem__`، يمكننا الوصول بشكل تعسفي إلى المقدمة والفرضية والتسمية باستخدام الفهرس `idx`.

```
#@save
class SNLIDataset(gluon.data.Dataset):
    """A customized dataset to load the SNLI dataset."""
    def __init__(self, dataset, num_steps, vocab=None):
        self.num_steps = num_steps
        all_premise_tokens = d2l.tokenize(dataset[0])
        all_hypothesis_tokens = d2l.tokenize(dataset[1])
        if vocab is None:
            self.vocab = d2l.Vocab(all_premise_tokens +
all_hypothesis_tokens,
                                   min_freq=5,
reserved_tokens=['<pad>'])
        else:
            self.vocab = vocab
        self.premises = self._pad(all_premise_tokens)
        self.hypotheses =
self._pad(all_hypothesis_tokens)
        self.labels = np.array(dataset[2])
        print('read ' + str(len(self.premises)) + '
examples')

    def _pad(self, lines):
        return np.array([d2l.truncate_pad(
            self.vocab[line], self.num_steps,
self.vocab['<pad>'])
                         for line in lines])

    def __getitem__(self, idx):
        return (self.premises[idx],
self.hypotheses[idx]), self.labels[idx]

    def __len__(self):
        return len(self.premises)
```

16.4.2.3. وضع كل شيء معا Putting It All Together

الآن يمكننا استدعاء دالة `read_snli` وفئة `SNLIDataset` لتنزيل مجموعة بيانات SNLI وإرجاع مثيلات `DataLoader` لكل من مجموعات التدريب والاختبار ، جنبًا إلى جنب مع مفردات مجموعة التدريب. من الجدير بالذكر أنه يجب علينا استخدام المفردات المكونة من مجموعة التدريب كمفردات مجموعة الاختبار. نتيجة لذلك، فإن أي رمز جديد من مجموعة الاختبار سيكون غير معروف للنموذج الذي تم تدريبه على مجموعة التدريب.

```
#@save
def load_data_snli(batch_size, num_steps=50):
    """Download the SNLI dataset and return data
iterators and vocabulary."""
    num_workers = d2l.get_dataloader_workers()
    data_dir = d2l.download_extract('SNLI')
    train_data = read_snli(data_dir, True)
    test_data = read_snli(data_dir, False)
    train_set = SNLIDataset(train_data, num_steps)
    test_set = SNLIDataset(test_data, num_steps,
train_set.vocab)
    train_iter = gluon.data.DataLoader(train_set,
batch_size, shuffle=True,

num_workers=num_workers)
    test_iter = gluon.data.DataLoader(test_set,
batch_size, shuffle=False,

num_workers=num_workers)
    return train_iter, test_iter, train_set.vocab
```

هنا قمنا بتعيين حجم الدُفعة على 128 وطول التسلسل على 50، واستدعاء دالة `load_data_snli` للحصول على مكررات البيانات والمفردات. ثم نقوم بطباعة حجم المفردات.

```
train_iter, test_iter, vocab = load_data_snli(128, 50)
len(vocab)
```

```
read 549367 examples
read 9824 examples
```

```
18678
```

الآن نقوم بطباعة شكل الدفعة الأولى. على عكس تحليل المشاعر، لدينا مدخلين `X[0]` و `X[1]` يمثلان أزواج من المقدمات والفرضيات.

```
for X, Y in train_iter:
    print(X[0].shape)
    print(X[1].shape)
    print(Y.shape)
    break
```

```
(128, 50)
(128, 50)
```

```
(128,)
```

16.4.3. الملخص

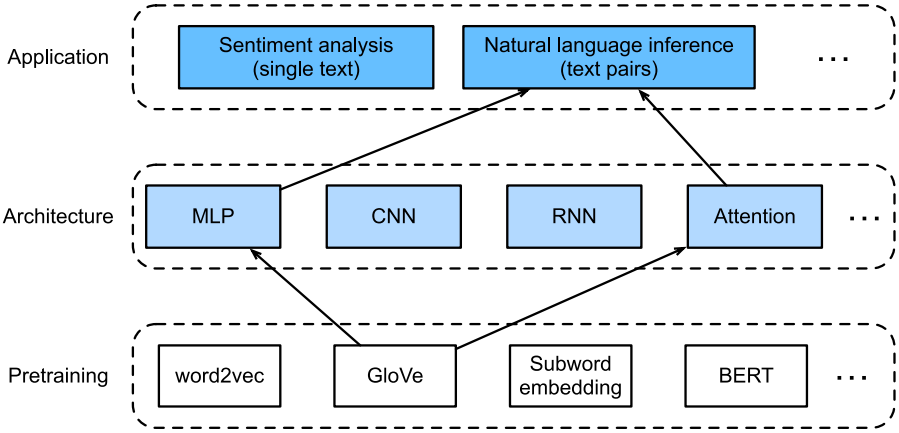
- يدرس الاستدلال اللغوي الطبيعي Natural language inference ما إذا كان يمكن استنتاج الفرضية hypothesis من مقدمة premise، حيث يكون كلاهما تسلسل نصي.
- في الاستدلال اللغوي الطبيعي، تشمل العلاقات بين المقدمات والفرضيات الاستنتاج entailment والتناقض contradiction والحيادية neutral.
- مجموعة استدلال اللغة الطبيعيةفي ستانفورد (SNLI) هي مجموعة بيانات مرجعية شائعة لاستدلال اللغة الطبيعية.

16.4.4. التمارين

1. لطالما تم تقييم الترجمة الآلية بناءً على مطابقة $n-gram$ السطحية بين ترجمة المخرجات وترجمة الحقيقة الأساسية. هل يمكنك تصميم مقياس لتقييم نتائج الترجمة الآلية باستخدام الاستدلال اللغوي الطبيعي؟
2. كيف يمكننا تغيير المعلمات الفائقة لتقليل حجم المفردات؟

16.5. الاستدلال اللغوي الطبيعي: استخدام الانتباه Natural Language Inference: Using Attention

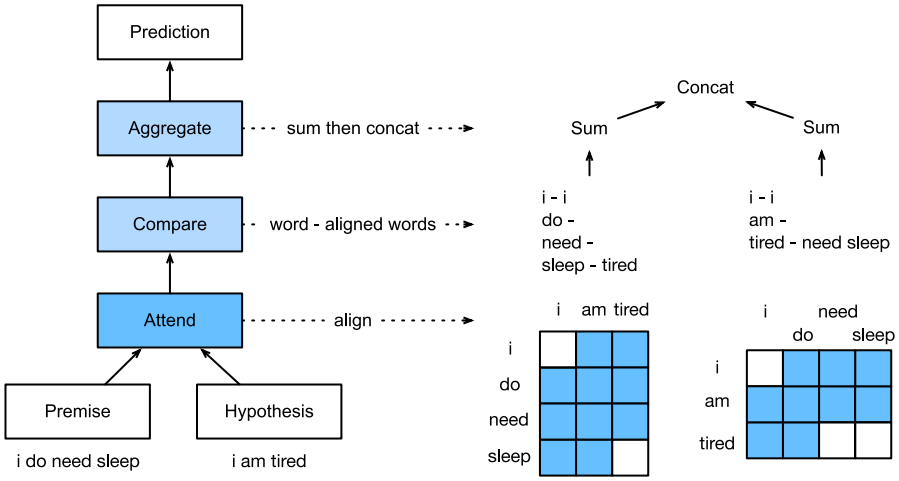
قدمنا مهمة الاستدلال اللغوي الطبيعي natural language inference ومجموعة بيانات SNLIفي القسم 16.4. في ضوء العديد من النماذج التي تستند إلى معماريات معقدة وعميقة، اقترح Parikh et al معالجة الاستدلال اللغوي الطبيعي بآليات الانتباه attention mechanisms وأطلق عليه "نموذج الانتباه القابل للتحلل decomposable attention mode " (Parikh et al.، 2016). ينتج عن هذا نموذج بدون طبقات متكررة أو تلافيفية، وتحقيق أفضل نتيجةفي ذلك الوقت على مجموعة بيانات SNLI بمعلمات أقل بكثير.في هذا القسم، سنصف وننفذ هذه الطريقة القائمة على الانتباه (باستخدام MLPs) للاستدلال اللغوي الطبيعي، كما هو موضحفي الشكل 16.5.1.



الشكل 16.5.1 يغذي هذا القسم GloVe المدرب مسبقًا إلى بُنية قائمة على الانتباه وMLPs للاستدلال اللغوي الطبيعي.

16.5.1. النموذج The Model

أبسط من الحفاظ على ترتيب الرموزفي المقدمات premises والفرضيات hypotheses، يمكننا فقط محاذاة الرموزفي تسلسل نصي واحد لكل رمزفي الآخر، والعكس صحيح، ثم مقارنة هذه المعلومات وتجميعها للتنبؤ بالعلاقات المنطقية بين المقدمات والفرضيات. على غرار محاذاة الرموز بين الجمل المصدر والهدففي الترجمة الآلية، يمكن تحقيق محاذاة الرموز بين المقدمات والفرضيات بدقة من خلال آليات الانتباه.



الشكل 16.5.2 الاستدلال اللغوي الطبيعي باستخدام آليات الانتباه.

يصور الشكل 16.5.2 طريقة الاستدلال اللغوي الطبيعية باستخدام آليات الانتباه. على مستوى عالٍ، يتكون من ثلاث خطوات مدربة بشكل مشترك: الحضور attending والمقارنة comparing والتجميع aggregating. سوف نوضح لهم خطوة بخطوة فيما يلي.

```
from mxnet import gluon, init, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

16.5.1.1. الحضور Attending

تتمثل الخطوة الأولىفي محاذاة الرموزفي تسلسل نصي واحد لكل رمزفي التسلسل الآخر. افترض أن الافتراض هو " i do need sleep" والفرضية هي " i am tired". بسبب التشابه الدلالي semantical similarity، قد نرغبفي محاذاة "i"في الفرضية مع "i"في المقدمة، ومحاذاة "tired"في الفرضية مع "sleep"في المقدمة. وبالمثل، قد نرغبفي محاذاة "i"في المقدمة مع "i"في الفرضية، ومحاذاة "need" و"sleep"في المقدمة مع "tired"في الفرضية. لاحظ أن هذا المحاذاة يكون ناعمًا soft باستخدام متوسط الاوزان، حيث يتم ربط الأوزان الكبيرة بشكل مثالي بالرموز المراد محاذاتها. لسهولة العرض، يوضح الشكل 16.5.2 مثل هذه المحاذاة بطريقة صعبة hard.

الآن نصف المحاذاة الناعمة باستخدام آليات الانتباه بمزيد من التفصيل. قم بالإشارة إلى $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ و $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ المقدمة والفرضية، والتي يكون عدد الرموز بها m و n ، على التوالي، حيث $\mathbf{a}_i, \mathbf{b}_j \in \mathbb{R}^d (i = 1, \ldots, m, j = 1, \ldots, n)$ عبارة عن متجه كلمة ذي أبعاد d. من أجل المحاذاة الناعمة، نحسب أوزان الانتباه $e_{ij} \in \mathbb{R}$ على أنها

$$e_{ij} = f(\mathbf{a}_i)^\top f(\mathbf{b}_j),$$

حيث تكون الدالة عبارة عن MLP محددةفي دالة `mlp` التالية. يتم تحديد بُعد الإخراج لـ f بواسطة وسيطة `num_hiddens` الخاصة بـ `mlp`.

```
def mlp(num_hiddens, flatten):
    net = nn.Sequential()
    net.add(nn.Dropout(0.2))
    net.add(nn.Dense(num_hiddens, activation='relu',
flatten=flatten))
    net.add(nn.Dropout(0.2))
    net.add(nn.Dense(num_hiddens, activation='relu',
flatten=flatten))
    return net
```

وتجدر الإشارة إلى أنهفي (16.5.1) f تأخذ المدخلات $\mathbf{a}_i$ و$\mathbf{b}_j$ بشكل منفصل بدلاً من أخذ زوج منها معًا كمدخلات. تؤدي خدعة التحلل decomposition هذه إلى تطبيقات $m+n$ فقط (التعقيد الخطي linear complexity) لـ f بدلاً من التطبيقات mn (التعقيد التربيعي quadratic complexity).

بتسوية أوزان الانتباهفي (16.5.1)، نحسب المتوسط المرجح لجميع متجهات الرمزفي الفرضية للحصول على تمثيل للفرضية التي تتم محاذاتها بنعومة softly aligned مع الرمز المفهرس بواسطة iفي المقدمة:

$$\boldsymbol{\beta}_i = \sum_{j=1}^{n} \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \mathbf{b}_j.$$

وبالمثل، نحسب المحاذاة الناعمة للرموز لكل رمز مفهرس بواسطة jفي الفرضية:

$$\boldsymbol{\alpha}_j = \sum_{i=1}^{m} \frac{\exp(e_{ij})}{\sum_{k=1}^{m} \exp(e_{kj})} \mathbf{a}_i.$$

نحدد أدناه فئة `Attend` لحساب المحاذاة الناعمة للفرضيات (`beta`) مع أماكن الإدخال `A` والمحاذاة الناعمة للمقدمات (`alpha`) مع فرضيات الإدخال `B`.

```
class Attend(nn.Block):
    def __init__(self, num_hiddens, **kwargs):
        super(Attend, self).__init__(**kwargs)
        self.f = mlp(num_hiddens=num_hiddens,
flatten=False)

    def forward(self, A, B):
        # Shape of `A`/`B`: (b`atch_size`, no. of tokens
in sequence A/B,
        # `embed_size`)
        # Shape of `f_A`/`f_B`: (`batch_size`, no. of
tokens in sequence A/B,
        # `num_hiddens`)
        f_A = self.f(A)
        f_B = self.f(B)
        # Shape of `e`: (`batch_size`, no. of tokens in
sequence A,
```

```
        # no. of tokens in sequence B)
        e = npx.batch_dot(f_A, f_B, transpose_b=True)
        # Shape of `beta`: (`batch_size`, no. of tokens
in sequence A,
        # `embed_size`), where sequence B is softly
aligned with each token
        # (axis 1 of `beta`) in sequence A
        beta = npx.batch_dot(npx.softmax(e), B)
        # Shape of `alpha`: (`batch_size`, no. of tokens
in sequence B,
        # `embed_size`), where sequence A is softly
aligned with each token
        # (axis 1 of `alpha`) in sequence B
        alpha = npx.batch_dot(npx.softmax(e.transpose(0,
2, 1)), A)
        return beta, alpha
```

16.5.1.2. المقارنة Comparing

في الخطوة التالية، نقارن رمزًافي تسلسل واحد مع التسلسل الآخر الذي يتم محاذاته بنعومة مع هذا الرمز. لاحظ أنهفي المحاذاة الناعمة، ستتم مقارنة جميع الرموز من تسلسل واحد، على الرغم من اختلاف أوزان الانتباه بها، برمزفي التسلسل الآخر. لسهولة العرض، أزواج الشكل 16.5.2 مع الرموز المحاذاة بطريقة صعبة. على سبيل المثال، افترض أن خطوة الحضور تحدد أن "need" و "sleep"في المقدمة تتماشى مع "tired"في الفرضية، ستتم مقارنة الزوجين "tired-need sleep".

في خطوة المقارنة، نقوم بتغذية سلسلة (عامل [·,·]) الرموز من تسلسل واحد والرموز المحاذية من التسلسل الآخر إلى دالة g (MLP):

$$\mathbf{v}_{A,i} = g([\mathbf{a}_i, \boldsymbol{\beta}_i]), i = 1, \ldots, m$$
$$\mathbf{v}_{B,j} = g([\mathbf{b}_j, \boldsymbol{\alpha}_j]), j = 1, \ldots, n.$$

في (16.5.4)،$\mathbf{v}_{A,i}$ هي المقارنة بين الرمز iفي المقدمة وجميع الرموز للفرضية التي يتم محاذاتها بنعومة مع الرمز i ؛ طالما $\mathbf{v}_{B,j}$ هي المقارنة بين الرمز jفي الفرضية وجميع الرموز للمقدمة التي يتم محاذاتها بنعومة مع الرمز j. يعرّف فئة `Compare` التالية مثل خطوة المقارنة.

```
class Compare(nn.Block):
    def __init__(self, num_hiddens, **kwargs):
        super(Compare, self).__init__(**kwargs)
        self.g = mlp(num_hiddens=num_hiddens,
flatten=False)
```

```
    def forward(self, A, B, beta, alpha):
        V_A = self.g(np.concatenate([A, beta], axis=2))
        V_B = self.g(np.concatenate([B, alpha], axis=2))
        return V_A, V_B
```

16.5.1.3. التجميع Aggregating

مع وجود مجموعتين من متجهات المقارنة $\mathbf{v}_{A,i}(i = 1, \ldots, m)$ و $\mathbf{v}_{B,j}(j = 1, \ldots, n)$ في متناول اليد، سنقوم في الخطوة الأخيرة بتجميع هذه المعلومات لاستنتاج العلاقة المنطقية. نبدأ بتلخيص كلتا المجموعتين:

$$\mathbf{v}_A = \sum_{i=1}^{m} \mathbf{v}_{A,i}, \mathbf{v}_B = \sum_{j=1}^{n} \mathbf{v}_{B,j}.$$

بعد ذلك نقوم بتغذية تسلسل كل من نتائج التلخيص إلى دالة h (MLP) للحصول على نتيجة التصنيف للعلاقة المنطقية:

$$\hat{\mathbf{y}} = h([\mathbf{v}_A, \mathbf{v}_B]).$$

يتم تحديد خطوة التجميع في فئة `Aggregate` التالية.

```
class Aggregate(nn.Block):
    def __init__(self, num_hiddens, num_outputs,
**kwargs):
        super(Aggregate, self).__init__(**kwargs)
        self.h = mlp(num_hiddens=num_hiddens,
flatten=True)
        self.h.add(nn.Dense(num_outputs))

    def forward(self, V_A, V_B):
        # Sum up both sets of comparison vectors
        V_A = V_A.sum(axis=1)
        V_B = V_B.sum(axis=1)
        # Feed the concatenation of both summarization
results into an MLP
        Y_hat = self.h(np.concatenate([V_A, V_B],
axis=1))
        return Y_hat
```

16.5.1.4. وضع كل شيء معا Putting It All Together

من خلال وضع خطوات الحضور والمقارنة والتجميع معًا، نحدد نموذج الانتباه القابل للتحلل لتدريب هذه الخطوات الثلاث بشكل مشترك.

```
class DecomposableAttention(nn.Block):
    def __init__(self, vocab, embed_size, num_hiddens,
**kwargs):
        super(DecomposableAttention,
self).__init__(**kwargs)
        self.embedding = nn.Embedding(len(vocab),
embed_size)
        self.attend = Attend(num_hiddens)
        self.compare = Compare(num_hiddens)
        # There are 3 possible outputs: entailment,
contradiction, and neutral
        self.aggregate = Aggregate(num_hiddens, 3)

    def forward(self, X):
        premises, hypotheses = X
        A = self.embedding(premises)
        B = self.embedding(hypotheses)
        beta, alpha = self.attend(A, B)
        V_A, V_B = self.compare(A, B, beta, alpha)
        Y_hat = self.aggregate(V_A, V_B)
        return Y_hat
```

16.5.2. تدريب وتقييم النموذج Training and Evaluating the Model

سنقوم الآن بتدريب وتقييم نموذج الانتباه القابل للتحلل المحدد على مجموعة بيانات SNLI. نبدأ بقراءة مجموعة البيانات.

16.5.2.1. قراءة مجموعة البيانات Reading the dataset

نقوم بتنزيل وقراءة مجموعة بيانات SNLI باستخدام الدالة المحددةفي القسم 16.4. يتم تعيين حجم الدُفعة وطول التسلسل على 256 و50 ، على التوالي.

```
batch_size, num_steps = 256, 50
train_iter, test_iter, vocab =
d2l.load_data_snli(batch_size, num_steps)
```

```
Downloading ../data/snli_1.0.zip from
https://nlp.stanford.edu/projects/snli/snli_1.0.zip...
read 549367 examples
read 9824 examples
```

16.5.2.2. إنشاء النموذج Creating the Model

نحن نستخدم تضمين GloVe ذو البعد 100 المدرب مسبقًا لتمثيل الرموز للإدخال. وبالتالي، قمنا بتعريف أبعاد المتجهات $\mathbf{a}_i$ و $\mathbf{b}_j$ مسبقًا في (16.5.1) على أنه 100. يتم تعيين بُعد الإخراج للدوال f في (16.5.1) و g في (16.5.4) على 200. ثم نقوم بإنشاء مثيل نموذج، ونهيأ المعلمات، وتحميل تضمين GloVe لتهيئة متجهات الرموز للإدخال.

```
embed_size, num_hiddens, devices = 100, 200,
d2l.try_all_gpus()
net = DecomposableAttention(vocab, embed_size,
num_hiddens)
net.initialize(init.Xavier(), ctx=devices)
glove_embedding = d2l.TokenEmbedding('glove.6b.100d')
embeds = glove_embedding[vocab.idx_to_token]
net.embedding.weight.set_data(embeds)
```

```
Downloading ../data/glove.6B.100d.zip from http://d2l-
data.s3-accelerate.amazonaws.com/glove.6B.100d.zip...
```

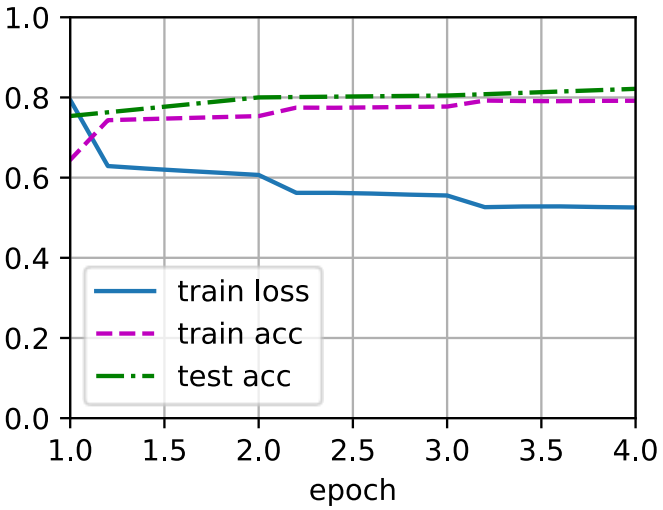
16.5.2.3. تدريب وتقييم النموذج Training and Evaluating the Model

على النقيض من دالة `split_batch` في القسم 13.5 التي تأخذ مدخلات فردية مثل تسلسل النص (أو الصور) ، فإننا نحدد دالة `split_batch_multi_inputs` لأخذ مدخلات متعددة مثل المقدمات والفرضيات في الدفعات الصغيرة minibatches.

يمكننا الآن تدريب النموذج وتقييمه على مجموعة بيانات SNLI.

```
lr, num_epochs = 0.001, 4
trainer = gluon.Trainer(net.collect_params(), 'adam',
{'learning_rate': lr})
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs, devices,
               split_batch_multi_inputs)
```

```
loss 0.526, train acc 0.792, test acc 0.821
9241.2 examples/sec on [gpu(0), gpu(1)]
```



16.5.2.4. استخدام النموذج Using the Model

أخيرًا، حدد دالة التنبؤ لإخراج العلاقة المنطقية بين زوج من المقدمة والفرضية.

```
#@save
def predict_snli(net, vocab, premise, hypothesis):
    """Predict the logical relationship between the
premise and hypothesis."""
    premise = np.array(vocab[premise],
ctx=d2l.try_gpu())
    hypothesis = np.array(vocab[hypothesis],
ctx=d2l.try_gpu())
    label = np.argmax(net([premise.reshape((1, -1)),
                           hypothesis.reshape((1, -
1))]), axis=1)
    return 'entailment' if label == 0 else
'contradiction' if label == 1 \
           else 'neutral'
```

يمكننا استخدام النموذج المدرَّب للحصول على نتيجة الاستدلال اللغوي الطبيعي لعينة زوج من الجمل.

```
predict_snli(net, vocab, ['he', 'is', 'good', '.'],
['he', 'is', 'bad', '.'])
```

```
'contradiction'
```

16.5.3. الملخص

- يتكون نموذج الانتباه القابل للتحلل decomposable attention model من ثلاث خطوات للتنبؤ بالعلاقات المنطقية بين المقدمات والفرضيات: الحضور attending والمقارنة comparing والتجميع aggregating.
- باستخدام آليات الانتباه، يمكننا محاذاة الرموزفي تسلسل نصي واحد لكل رمزفي الآخر، والعكس صحيح. تكون هذه المحاذاة ناعمة باستخدام المتوسط المرجح، حيث ترتبط الأوزان الكبيرة بشكل مثالي بالرموز التي يجب محاذاتها.
- تؤدي خدعة التحلل decomposition trick إلى تعقيد خطي مرغوب فيه أكثر من التعقيد التربيعي عند حساب أوزان الانتباه.
- يمكننا استخدام متجهات الكلمات التي تم اختبارها مسبقًا كتمثيل إدخال لمهمة معالجة اللغة الطبيعية مثل الاستدلال اللغوي الطبيعي.

16.5.4. التمارين

1. درب النموذج مع مجموعات أخرى من المعلمات الفائقة. هل يمكنك الحصول على دقة أفضلفي مجموعة الاختبار؟
2. ما هي العيوب الرئيسية لنموذج الانتباه القابل للتحلل لاستدلال اللغة الطبيعية؟
3. افترض أننا نريد الحصول على مستوى التشابه الدلالي (على سبيل المثال، قيمة مستمرة بين 0 و1) لأي زوج من الجمل. كيف سنقوم بجمع مجموعة البيانات وتسميتها؟ هل يمكنك تصميم نموذج بآليات الانتباه؟

16.6. الضبط الدقيق لـ BERT لتطبيقات مستوى التسلسل ومستوى الرمز Fine-Tuning BERT for Sequence-Level and Token-Level Applications

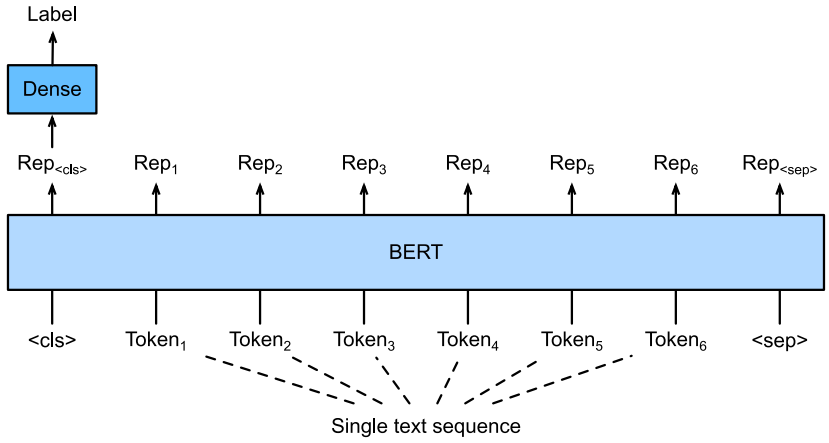
في الأقسام السابقة من هذا الفصل، قمنا بتصميم نماذج مختلفة لتطبيقات المعالجة اللغوية الطبيعية، مثل القائمة على RNNs وCNNs والانتباه وMLPs. هذه النماذج مفيدة عندما يكون هناك قيود على المكان أو الوقت، ومع ذلك، فإن صياغة نموذج معين لكل مهمة معالجة لغة طبيعية أمر غير عملي عمليًا.في القسم 15.8، قدمنا نموذجًا للتدريب المسبق BERT، والذي يتطلب الحد الأدنى من التغييرات المعمارية لمجموعة واسعة من مهام المعالجة اللغويةالطبيعية. من ناحية،في وقت اقتراحها، قامت BERT بتحسين أحدث ما وصلت اليه التكنولوجيافي مختلف مهام معالجة اللغة الطبيعية. من ناحية أخرى، كما هو مذكورفي القسم 15.10، يأتي الإصداران من طراز BERT الأصلي مع 110 مليون و340 مليون معلمة.

وبالتالي، عندما تكون هناك موارد حسابية كافية، قد نفكر في الضبط الدقيق لـ BERT لتطبيقات المعالجة اللغوية الطبيعية.

فيما يلي، نقوم بتعميم مجموعة فرعية من تطبيقات المعالجة اللغوية الطبيعية كمستوى التسلسل sequence-level ومستوى الرمز token-level. على مستوى التسلسل، نقدم كيفية تحويل تمثيل BERT لإدخال النص إلى تسمية الإخراج في تصنيف نص واحد وتصنيف أزواج النص أو الانحدار. على مستوى الرمز، سنقدم بإيجاز تطبيقات جديدة مثل وضع علامات على النص والإجابة على الأسئلة وإلقاء الضوء على كيفية تمثيل BERT لمدخلاتهم وتحويلها إلى تسميات إخراج. أثناء الضبط الدقيق fine-tuning، فإن "الحد الأدنى من التغييرات الهيكلية" التي تتطلبها BERT عبر التطبيقات المختلفة هي الطبقات الإضافية المتصلة بالكامل. أثناء التعلم الخاضع للإشراف لتطبيق downstream، يتم التعرف على معلمات الطبقات الإضافية من نقطة الصفر بينما يتم ضبط جميع المعلمات في نموذج BERT الذي تم اختباره مسبقًا.

16.6.1. تصنيف نص واحد Single Text Classification

يأخذ تصنيف النص الفردي Single text classification تسلسلًا نصيًا واحدًا كمدخلات ومخرجات نتيجة التصنيف الخاصة به. إلى جانب تحليل المشاعر الذي درسناه في هذا الفصل، فإن مجموعة المقبولية اللغوية Corpus of Linguistic Acceptability (CoLA) هي أيضًا مجموعة بيانات لتصنيف النص الفردي، لتحديد ما إذا كانت جملة معينة مقبولة نحويًا أم لا (Warstadt et al.، 2019). على سبيل المثال، "I should study." مقبول ولكن "I should studying." ليس مقبول.



الشكل 16.6.1 الضبط الدقيق لـ BERT لتطبيقات تصنيف النص الفردي، مثل تحليل المشاعر واختبار القبول اللغوي. افترض أن إدخال نص واحد يحتوي على ستة رموز.

يصف القسم 15.8 تمثيل المدخلات لـ BERT. يمثل تسلسل إدخال BERT بشكل لا لبس فيه كلاً من النص الفردي وأزواج النص، حيث يتم استخدام رمز التصنيف الخاص "<cls>" لتصنيف التسلسل ويمثل رمز التصنيف الخاص "<sep>" نهاية النص الفردي أو يفصل بين زوج من النص. كما هو مبينفي الشكل 16.6.1،في تطبيقات تصنيف النص الفردي، يشفر تمثيل BERT لرمز التصنيف الخاص "<cls>" معلومات تسلسل نص الإدخال بأكمله. كتمثيل للنص الفردي للإدخال، سيتم إدخالهفي MLP صغير يتكون من طبقات متصلة (كثيفة) بالكامل لإخراج توزيع جميع قيم التسميات المنفصلة.

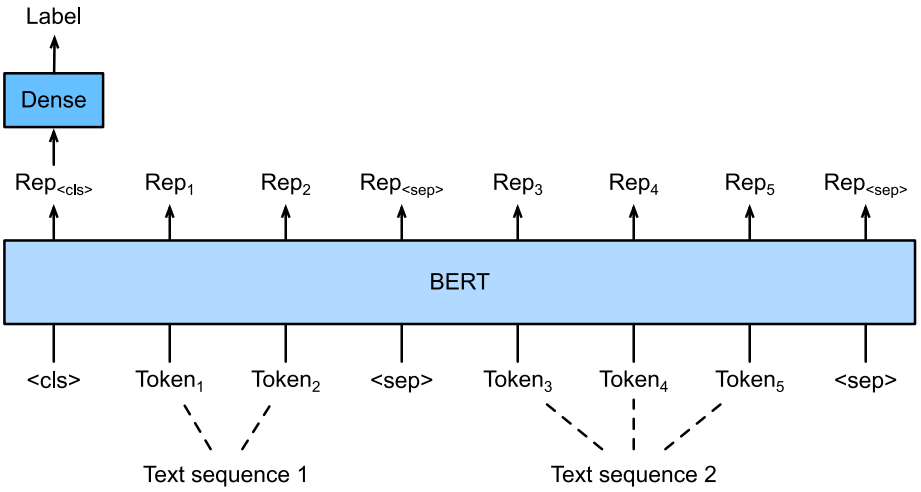
16.6.2. تصنيف زوج النص أو الانحدار Text Pair Classification or Regression

لقد قمنا أيضًا بفحص الاستدلال اللغوي الطبيعي natural language inferenceفي هذا الفصل. إنه ينتمي إلى تصنيف أزواج النص، وهو نوع من التطبيقات التي تصنف زوجًا من النص.

بأخذ زوج من النص كمدخل ولكن إخراج قيمة مستمرة، فإن التشابه النصي الدلالي semantic textual similarity هو مهمة انحدار زوج نصي شائع. تقيس هذه المهمة التشابه الدلالي للجمل. على سبيل المثال،في مجموعة بيانات مقياس التشابه النصي الدلالي Semantic Textual Similarity Benchmark dataset، تكون درجة التشابه لزوج من الجمل مقياسًا ترتيبيًا يتراوح من 0 (لا يوجد تداخلفي المعنى) إلى 5 (يعني التكافؤ) (Cer et al.، 2017). الهدف هو توقع هذه الدرجات. تتضمن الأمثلة من مجموعة بيانات مقياس التشابه النصي الدلالي (الجملة 1، الجملة 2، درجة التشابه):

- “A plane is taking off.”, “An air plane is taking off.”, 5.000;
- “A woman is eating something.”, “A woman is eating meat.”, 3.000;
- “A woman is dancing.”, “A man is talking.”, 0.000.

وبالمقارنة مع تصنيف النص الفرديفي الشكل 16.6.1، فإن الضبط الدقيق لـ BERT لتصنيف أزواج النصفي الشكل 16.6.2 يختلففي تمثيل الإدخال. بالنسبة لمهام انحدار أزواج النص مثل التشابه النصي الدلالي، يمكن تطبيق تغييرات عادية مثل إخراج قيمة تسمية مستمرة واستخدام متوسط الخسارة التربيعية: فهي شائعة للانحدار.



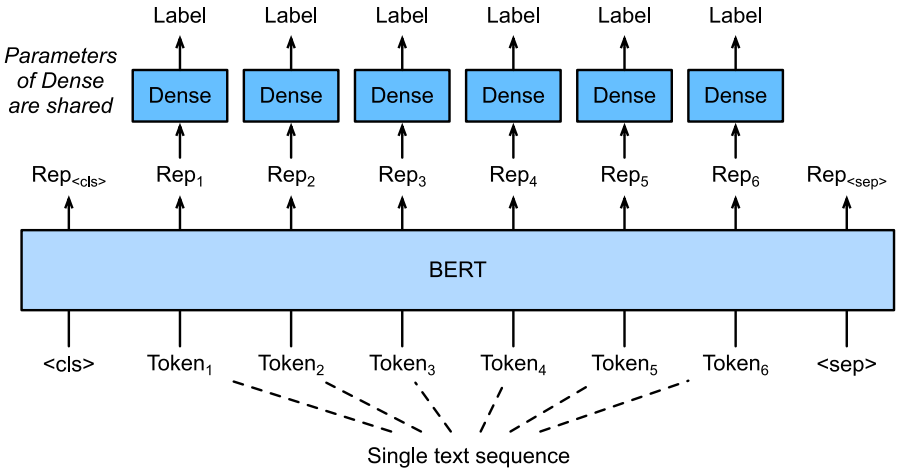
الشكل 16.6.2 الضبط الدقيق لـ BERT لتصنيف أزواج النص أو تطبيقات الانحدار، مثل استدلال اللغة الطبيعية والتشابه النصي الدلالي. افترض أن زوج إدخال النص يحتوي على رمزين وثلاثة رموز.

16.6.3. وضع علامات على النص Text Tagging

دعونا الآن نفكرفي المهام على مستوى الرمز، مثل وضع العلامات النصية text tagging، حيث يتم تعيين تسمية لكل رمز. من بين مهام وضع العلامات على النص، تخصص علامات جزء من الكلام لكل كلمة علامة جزء من الكلام (على سبيل المثال، الصفة adjective والمُحدد determiner) وفقًا لدور الكلمةفي الجملة. على سبيل المثال، وفقًا لمجموعة علامات Penn Treebank II، يجب وضع علامة على الجملة "John Smith 's car is new" على أنها

“NNP (noun, proper singular) NNP POS (possessive ending) NN (noun, singular or mass) VB (verb, base form) JJ (adjective)”

يوضح الشكل 16.6.3 الضبط الدقيق لـ BERT لتطبيقات تعليم النص. بالمقارنة مع الشكل 16.6.1، يكمن التمييز الوحيدفي أنهفي وضع علامات على النص، يتم تغذية تمثيل BERT لكل رمز لنص الإدخالفي نفس الطبقات الإضافية المتصلة بالكامل لإخراج تسمية الرمز، مثل جزء_ علامة الكلام part-of-speech tag.



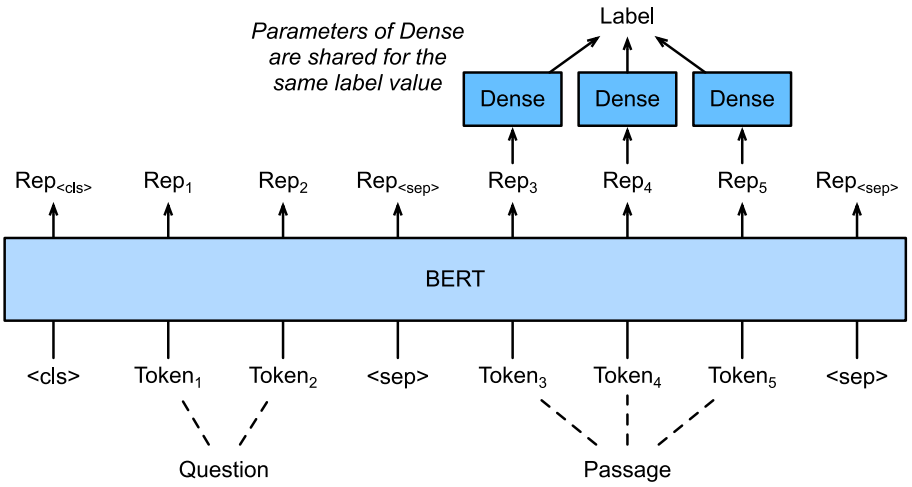
الشكل 16.6.3 الضبط الدقيق BERT لتطبيقات وضع علامات على النص، مثل وضع علامات على جزء من الكلام. افترض أن إدخال نص واحد يحتوي على ستة رموز.

16.6.4. إجابة السؤال Question Answering

كتطبيق آخر على مستوى الرموز token-level application، تعكس الإجابة على الأسئلة قدرات فهم القراءة. على سبيل المثال، تتكون مجموعة بيانات الإجابة على سؤال ستانفورد Stanford Question Answering Dataset (SQuAD v1.1) من قراءة مقاطع وأسئلة، حيث تكون الإجابة على كل سؤال مجرد جزء من النص (امتداد النص text span) من المقطع الذي يدور حوله السؤال (Rajpurkar et al.، 2016). للتوضيح، ضع في اعتبارك مقطعًا "Some experts report that a mask's efficacy is inconclusive. However, mask makers insist that their products, such as N95 respirator masks, can guard against the virus." والسؤال هو "Who say that N95 respirator masks can guard against the virus?". يجب أن تكون الإجابة هي امتداد النص "mask makers" في الفقرة. وبالتالي، فإن الهدف في الإصدار 1.1 من SQuAD هو التنبؤ ببداية ونهاية امتداد النص في المقطع بالنظر إلى زوج من الأسئلة والمقطع passage.

للضبط الدقيق لـ BERT للإجابة على الأسئلة، يتم تعبئة السؤال والمقطع كتسلسل نصي أول وثاني، على التوالي، في إدخال BERT. للتنبؤ بموضع بداية امتداد النص، فإن نفس الطبقة الإضافية المتصلة بالكامل ستحول تمثيل BERT لأي رمز من المقطع بالموضع i إلى درجة قياسية s_i. يتم تحويل هذه الدرجات من جميع الرموز للمقطع بشكل إضافي بواسطة عملية softmax إلى توزيع احتمالي، بحيث يتم تعيين احتمال p_i أن يكون كل موضع رمزي i في المقطع هو بداية امتداد النص. إن توقع نهاية امتداد النص هو نفسه كما هو مذكور أعلاه، باستثناء أن

المعلماتفي الطبقة الإضافية المتصلة بالكامل مستقلة عن تلك الخاصة بالتنبؤ بالبدء. عند التنبؤ بالنهاية، يتم تحويل أي رمز للمقطع بواسطة نفس الطبقة المتصلة بالكامل إلى درجة قياسية. يوضح الشكل 16.6.4 الضبط الدقيق BERT للإجابة على الأسئلة.



الشكل 16.6.4 الضبط الدقيق لـ BERT للإجابة على الأسئلة. افترض أن زوج إدخال نص يحتوي على اثنين وثلاثة من الرموز.

للإجابة على الأسئلة، يكون الهدف التدريبي للتعلم الخاضع للإشراف مباشرًا مثل تعظيم احتمالية لوغاريتم لبداية الحقيقة الواقعية ومواقف النهاية. عند توقع الامتداد span، يمكننا حساب الدرجة $s_i + e_j$ لامتداد صالح من موضع i إلى موضع $j (i \leq j)$ ، وإخراج الامتداد بأعلى درجة.

16.6.5. الملخص

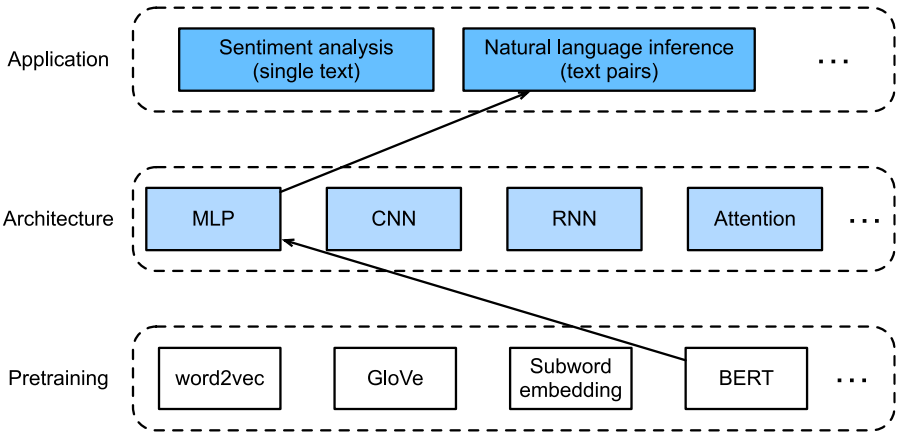
- تتطلب BERT تغييرات طفيفةفي البُنية (طبقات إضافية متصلة بالكامل) لتطبيقات معالجة اللغة الطبيعية على مستوى التسلسل ومستوى الرمز، مثل تصنيف النص الفردي (على سبيل المثال، تحليل المشاعر واختبار القبول اللغوي)، تصنيف أزواج النص أو الانحدار (على سبيل المثال، استدلال اللغة الطبيعية والتشابه الدلالي النصي)، ووضع علامات على النص (على سبيل المثال، وضع علامات على جزء من الكلام)، والإجابة على الأسئلة.
- أثناء التعلم الخاضع للإشراف لتطبيق downstream، يتم التعرف على معلمات الطبقات الإضافية من نقطة الصفر بينما يتم الضبط الدقيق لجميع المعلماتفي نموذج BERT الذي تم تدريبه مسبقًا.

16.6.6. التمارين

1. دعونا نصمم خوارزمية محرك بحث لمقالات الأخبار. عندما يتلقى النظام استعلامًا query (على سبيل المثال، "oil industry during the coronavirus outbreak")، يجب أن يعرض قائمة مصنفة بالمقالات الإخبارية الأكثر صلة بالاستعلام. افترض أن لدينا مجموعة ضخمة من المقالات الإخبارية وعددًا كبيرًا من الاستفسارات. لتبسيط المشكلة، افترض أنه تم تصنيف المقالة الأكثر صلة بكل استعلام. كيف يمكننا تطبيق أخذ العينات السلبية negative sampling (انظر القسم 15.2.1) وBERT في تصميم الخوارزمية؟
2. كيف يمكننا الاستفادة من BERT في تدريب نماذج اللغة؟
3. هل يمكننا الاستفادة من BERT في الترجمة الآلية؟

16.7. استدلال اللغة الطبيعية: الضبط الدقيق لـ BERT Natural Language Inference: Fine-Tuning BERT

في الأقسام السابقة من هذا الفصل، قمنا بتصميم بُنية قائمة على الانتباه attention-based architecture (في القسم 16.5) لمهمة استدلال اللغة الطبيعية في مجموعة بيانات SNLI (كما هو موضح في القسم 16.4). الآن نعيد النظر في هذه المهمة عن طريق الضبط الدقيق fine-tuning لـ BERT. كما تمت مناقشته في القسم 16.6، يعتبر الاستدلال اللغوي الطبيعي مشكلة تصنيف زوجي على مستوى التسلسل، ولا يتطلب الضبط الدقيق BERT سوى معمارية إضافية قائمة على MLP، كما هو موضح في الشكل 16.7.1.



الشكل 16.7.1 يغذي هذا القسم BERT المُدرب مسبقًا إلى بُنية قائمة على MLP لاستدلال اللغة الطبيعية.

في هذا القسم، سننزل نسخة صغيرة مدربة مسبقًا من BERT، ثم نضبطها لاستدلال اللغة الطبيعية على مجموعة بيانات SNLI.

```
import json
import multiprocessing
import os
from mxnet import gluon, np, npx
from mxnet.gluon import nn
from d2l import mxnet as d2l

npx.set_np()
```

16.7.1. تحميل BERT مدربة مسبقاً Loading Pretrained BERT

لقد أوضحنا كيفية تدريب BERT مسبقًا على مجموعة بيانات WikiText-2 في القسمين 15.9 والقسم 15.10 (لاحظ أن نموذج BERT الأصلي مُدرب مسبقًا على مجموعة أكبر بكثير). كما تمت مناقشته في القسم 15.10، يحتوي نموذج BERT الأصلي على مئات الملايين من المعلمات. فيما يلي، نقدم نسختين من BERT المدرب مسبقًا: "bert.base" بحجم نموذج BERT الأساسي الذي يتطلب الكثير من الموارد الحسابية لضبطه ، بينما "bert.small" هو نسخة صغيرة لتسهيل العرض.

```
d2l.DATA_HUB['bert.base'] = (d2l.DATA_URL +
'bert.base.zip',

'7b3820b35da691042e5d34c0971ac3edbd80d3f4')
d2l.DATA_HUB['bert.small'] = (d2l.DATA_URL +
'bert.small.zip',

'a4e718a47137ccd1809c9107ab4f5edd317bae2c')
```

يحتوي أي من نموذج BERT الذي تم تدريبه مسبقًا على ملف "vocab.json" الذي يحدد مجموعة المفردات وملف "prerained.params" للمعلمات التي تم تدريبها مسبقًا. نقوم بتنفيذ دالة `load_pretrained_model` التالية لتحميل معلمات BERT سابقة التدريب.

```
def load_pretrained_model(pretrained_model, num_hiddens,
ffn_num_hiddens,
                          num_heads, num_blks, dropout,
max_len, devices):
    data_dir = d2l.download_extract(pretrained_model)
    # Define an empty vocabulary to load the predefined
vocabulary
    vocab = d2l.Vocab()
```

```
    vocab.idx_to_token =
json.load(open(os.path.join(data_dir, 'vocab.json')))
    vocab.token_to_idx = {token: idx for idx, token in
enumerate(
        vocab.idx_to_token)}
    bert = d2l.BERTModel(len(vocab), num_hiddens,
ffn_num_hiddens, num_heads,
                         num_blks, dropout, max_len)
    # Load pretrained BERT parameters
    bert.load_parameters(os.path.join(data_dir,
'pretrained.params'),
                         ctx=devices)
    return bert, vocab
```

لتسهيل العرض التوضيحي على معظم الأجهزة، سنقوم بتحميل الإصدار الصغير ("bert.small") وضبطه بدقةفي هذا القسم.في التمرين، سنوضح كيفية الضبط الدقيق لـ "bert.base" الأكبر بكثير لتحسين دقة الاختبار بشكل كبير.

```
devices = d2l.try_all_gpus()
bert, vocab = load_pretrained_model(
    'bert.small', num_hiddens=256, ffn_num_hiddens=512,
num_heads=4,
    num_blks=2, dropout=0.1, max_len=512,
devices=devices)
```

```
Downloading ../data/bert.small.zip from http://d2l-
data.s3-accelerate.amazonaws.com/bert.small.zip...
```

16.7.2. مجموعة البيانات لضبط BERT

لاستنتاج اللغة الطبيعية لمهمة المصبفي مجموعة بيانات SNLI، نحدد فئة مجموعة البيانات المخصصة `SNLIBERTDataset`.في كل مثال، تشكل المقدمة premise والفرضية hypothesis زوجًا من تسلسل النص ويتم تعبئتهمافي تسلسل إدخال BERT كما هو موضح في الشكل 16.6.2. تذكر القسم 15.8.4 أن معرفات المقطع segment IDs تستخدم للتمييز بين المقدمة والفرضيةفي تسلسل إدخال BERT. مع الحد الأقصى المحدد مسبقًا لطول تسلسل إدخال (`max_len`) BERT، يستمر إزالة آخر رمز أطول من زوج نص الإدخال حتى يتم استيفاء `max_len`. لتسريع إنشاء مجموعة بيانات SNLI لضبط BERT، نستخدم 4 عمليات عاملة لإنشاء أمثلة للتدريب أو الاختبار بشكل متوازٍ.

```
class SNLIBERTDataset(gluon.data.Dataset):
    def __init__(self, dataset, max_len, vocab=None):
        all_premise_hypothesis_tokens = [[
```

```
            p_tokens, h_tokens] for p_tokens, h_tokens
in zip(
            *[d2l.tokenize([s.lower() for s in
sentences])
              for sentences in dataset[:2]])]

        self.labels = np.array(dataset[2])
        self.vocab = vocab
        self.max_len = max_len
        (self.all_token_ids, self.all_segments,
         self.valid_lens) =
self._preprocess(all_premise_hypothesis_tokens)
        print('read ' + str(len(self.all_token_ids)) + '
examples')

    def _preprocess(self,
all_premise_hypothesis_tokens):
        pool = multiprocessing.Pool(4)  # Use 4 worker
processes
        out = pool.map(self._mp_worker,
all_premise_hypothesis_tokens)
        all_token_ids = [
            token_ids for token_ids, segments, valid_len
in out]
        all_segments = [segments for token_ids,
segments, valid_len in out]
        valid_lens = [valid_len for token_ids, segments,
valid_len in out]
        return (np.array(all_token_ids, dtype='int32'),
                np.array(all_segments, dtype='int32'),
                np.array(valid_lens))

    def _mp_worker(self, premise_hypothesis_tokens):
        p_tokens, h_tokens = premise_hypothesis_tokens
        self._truncate_pair_of_tokens(p_tokens,
h_tokens)
        tokens, segments =
d2l.get_tokens_and_segments(p_tokens, h_tokens)
        token_ids = self.vocab[tokens] +
[self.vocab['<pad>']] \
                               * (self.max_len -
len(tokens))
```

```
        segments = segments + [0] * (self.max_len -
len(segments))
        valid_len = len(tokens)
        return token_ids, segments, valid_len

    def _truncate_pair_of_tokens(self, p_tokens,
h_tokens):
        # Reserve slots for '<CLS>', '<SEP>', and
'<SEP>' tokens for the BERT
        # input
        while len(p_tokens) + len(h_tokens) >
self.max_len - 3:
            if len(p_tokens) > len(h_tokens):
                p_tokens.pop()
            else:
                h_tokens.pop()

    def __getitem__(self, idx):
        return (self.all_token_ids[idx],
self.all_segments[idx],
                self.valid_lens[idx]), self.labels[idx]

    def __len__(self):
        return len(self.all_token_ids)
```

بعد تنزيل مجموعة بيانات SNLI، نقوم بإنشاء أمثلة للتدريب والاختبار عن طريق إنشاء مثيل لفئة `SNLIBERTDataset`. ستتم قراءة مثل هذه الأمثلةفي الدفعات الصغيرة minibatches أثناء التدريب واختبار الاستدلال اللغوي الطبيعي.

```
# Reduce `batch_size` if there is an out of memory
error. In the original BERT
# model, `max_len` = 512
batch_size, max_len, num_workers = 512, 128,
d2l.get_dataloader_workers()
data_dir = d2l.download_extract('SNLI')
train_set = SNLIBERTDataset(d2l.read_snli(data_dir,
True), max_len, vocab)
test_set = SNLIBERTDataset(d2l.read_snli(data_dir,
False), max_len, vocab)
train_iter = gluon.data.DataLoader(train_set,
batch_size, shuffle=True,
```

```
num_workers=num_workers)
test_iter = gluon.data.DataLoader(test_set, batch_size,

num_workers=num_workers)
```

```
read 549367 examples
read 9824 examples
```

16.7.3. الضبط الدقيق لـ BERT Fine-Tuning BERT

كما يشير الشكل 16.6.2، فإن الضبط الدقيق لـ BERT لاستدلال اللغة الطبيعية يتطلب فقط MLP إضافيًا يتكون من طبقتين متصلتين بالكامل (انظر `self.hidden` و`self.output` في فئة `BERTClassifier` التالية). يحول MLP تمثيل BERT للرمز "<cls>"، الذي يشفر المعلومات الخاصة بكل من المقدمة والفرضية، إلى ثلاث مخرجات لاستدلال اللغة الطبيعية: الاستنتاج entailment والتناقض contradiction والحيادية neutral.

```
class BERTClassifier(nn.Block):
    def __init__(self, bert):
        super(BERTClassifier, self).__init__()
        self.encoder = bert.encoder
        self.hidden = bert.hidden
        self.output = nn.Dense(3)

    def forward(self, inputs):
        tokens_X, segments_X, valid_lens_x = inputs
        encoded_X = self.encoder(tokens_X, segments_X,
valid_lens_x)
        return self.output(self.hidden(encoded_X[:, 0,
:]))
```

فيما يلي، يتم تغذية نموذج BERT مدرب مسبقاًفي شبكة مثيل `BERTClassifier` لتطبيق المصب.في التطبيقات الشائعة للضبط الدقيق لـ BERT، سيتم تعلم معلمات طبقة الإخراج لـ MLP الإضافي (`net.output`) فقط من البداية. سيتم ضبط بدقة جميع معلمات مشفر BERT (net.encoder) والطبقة المخفية لـ MLP الإضافي (`net.hidden`).

```
net = BERTClassifier(bert)
net.output.initialize(ctx=devices)
```

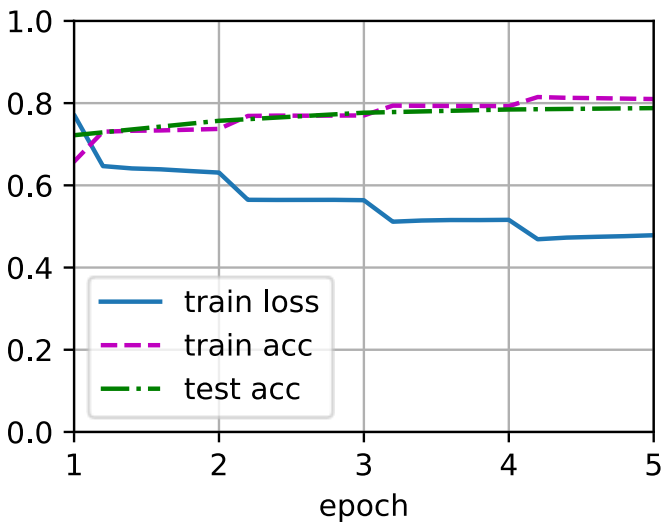
تذكر أنهفي القسم 15.8، تحتوي كل من فئة `MaskLM` وفئة `NextSentencePred` على معلماتفي MLPs المستخدمة. تعد هذه المعلمات جزءًا من تلك الموجودةفي نموذج BERT المُدرب مسبقًا، وبالتالي فهي جزء من المعلماتفي `net`. ومع ذلك، فإن هذه المعلمات

مخصصة فقط لحساب خسارة نمذجة اللغة المقنعة وخسارة توقع الجملة التالية أثناء التدريب المسبق. لا ترتبط دالتا الخسارة هاتان بالضبط الدقيق لتطبيقات downstream، وبالتالي لا يتم تحديث (توقفstaled) معلمات MLPs المستخدمةفي `MaskLM` و `NextSentencePred` عند الضبط الدقيق لـ BERT.

للسماح بالمعلمات ذات الانحدارات القديمة stale gradients، يتم تعيين العلامة `ignore_stale_grad=True` في دالة الخطوة `d2l.train_batch_ch13`. نستخدم هذه الدالة لتدريب وتقييم شبكة النموذج باستخدام مجموعة التدريب (`train_iter`) ومجموعة الاختبار (`test_iter`) لـ SNLI. نظرًا لمحدودية الموارد الحسابية، يمكن تحسين دقة التدريب والاختبار: نترك مناقشاتهافي التدريبات.

```
lr, num_epochs = 1e-4, 5
trainer = gluon.Trainer(net.collect_params(), 'adam',
{'learning_rate': lr})
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss,
trainer, num_epochs, devices,
               d2l.split_batch_multi_inputs)
```

```
loss 0.479, train acc 0.810, test acc 0.788
6798.7 examples/sec on [gpu(0), gpu(1)]
```



16.7.4. الملخص

- يمكننا الضبط الدقيق لنموذج BERT المدرب مسبقًا لتطبيقات downstream، مثل استدلال اللغة الطبيعية على مجموعة بيانات SNLI.
- أثناء الضبط الدقيق، يصبح نموذج BERT جزءًا من نموذج تطبيق downstream. لن يتم تحديث المعلمات التي تتعلق فقط بفقدان التدريب المسبق أثناء الضبط الدقيق.

16.7.5. التمارين

1. قم بالضبط الدقيق لنموذج BERT الأكبر حجمًا والمدرب مسبقًا والذي يكون بحجم نموذج BERT الأساسي الأصلي إذا كان المورد الحسابي الخاص بك يسمح بذلك. قم بتعيين الوسيطاتفي دالة `load_pretrained_model` على النحو التالي: استبدال 'bert.small' بـ 'bert.base' ، وزيادة قيم `num_hiddens=256` ، و `ffn_num_hiddens=512` ، و `num_heads=4` ، و `num_blks=2` إلى 768 ، 3072 ، 12 ، و 12 ، على التوالي. من خلال زيادة فترات الضبط الدقيق (وربما ضبط المعلمات الفائقة الأخرى)، هل يمكنك الحصول على دقة اختبار أعلى من 0.86؟
2. كيف يتم اقتطاع truncate زوج من المتتاليات حسب نسبة طولها؟ قارن طريقة اقتطاع الزوج pair truncation method هذه والطريقة المستخدمةفي فئة `SNLIBERTDataset`. ما هي مزاياها وعيوبها؟

Dive into Deep Learning

Part 3

Scalability, Efficiency, and Applications

ASTON ZHANG, ZACHARY C. LIPTON, MU LI,
AND ALEXANDER J.SMOLA

Translated Into Arabic by
Dr. Alaa Taima