In [1]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

In [2]:
```
def init_cnn(module):
    """Initialize weights for CNNs."""
    if type(module) == nn.Linear or type(module) == nn.Conv2d:
        nn.init.xavier_uniform_(module.weight)

class LeNet(d2l.Classifier):
    """The LeNet-5 model."""
    def __init__(self, lr=0.1, num_classes=10):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(
            nn.LazyConv2d(6, kernel_size=5, padding=2), nn.Sigmoid(),
            nn.AvgPool2d(kernel_size=2, stride=2),
            nn.LazyConv2d(16, kernel_size=5), nn.Sigmoid(),
            nn.AvgPool2d(kernel_size=2, stride=2),
            nn.Flatten(),
            nn.LazyLinear(120), nn.Sigmoid(),
            nn.LazyLinear(84), nn.Sigmoid(),
            nn.LazyLinear(num_classes))
```

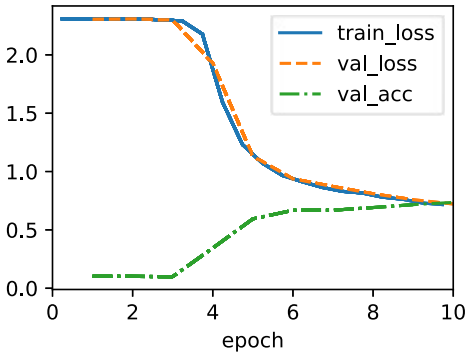
In [3]:
```
@d2l.add_to_class(d2l.Classifier)
def layer_summary(self, X_shape):
    X = torch.randn(*X_shape)
    for layer in self.net:
        X = layer(X)
        print(layer.__class__.__name__, 'output shape:\t', X.shape)

model = LeNet()
model.layer_summary((1, 1, 28, 28))
```

```
Conv2d output shape:     torch.Size([1, 6, 28, 28])
Sigmoid output shape:    torch.Size([1, 6, 28, 28])
AvgPool2d output shape:  torch.Size([1, 6, 14, 14])
Conv2d output shape:     torch.Size([1, 16, 10, 10])
Sigmoid output shape:    torch.Size([1, 16, 10, 10])
AvgPool2d output shape:  torch.Size([1, 16, 5, 5])
Flatten output shape:    torch.Size([1, 400])
Linear output shape:     torch.Size([1, 120])
Sigmoid output shape:    torch.Size([1, 120])
Linear output shape:     torch.Size([1, 84])
Sigmoid output shape:    torch.Size([1, 84])
Linear output shape:     torch.Size([1, 10])
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/torch/nn/modules/lazy.py:181: UserWarning: Lazy modules are a new feature under heavy development so changes to the API or functionality can happen at any moment.
  warnings.warn('Lazy modules are a new feature under heavy development '
```

In [4]:
```
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128)
model = LeNet(lr=0.1)
model.apply_init([next(iter(data.get_dataloader(True)))[0]], init_cnn)
trainer.fit(model, data)
```



Discussion: 이번 단원에서는 LeNet이라는 초기의 합성곱 신경망 구조에 대해 설명하고, 실제 구현을 해보는 단원이다. 이번 7-1~7-5의 복습 및 정리 단원 느낌이고, 수업 시간에 LeNet과 관련되어 어렴풋이 들은 기억만 남아 있는데, 이 부분에 대해 실제로 구현해보고 설명을 보며 조금 더 잘 이해할 수 있게 된 것 같다. LeNet은 크게, 두개의 합성층 곱과 pooling 층으로 이루어진 Convolutional Encoder과, 3개의 완전 연결된 층으로 구성되어있다고 한다. 첫 번째 합성곱 층이 6개의 채널을 생성하고, 시그모이드 사용하고, 두 번째 합성곱이 16개 생성하고, 시그모이드 사용하고, lazy linear로 120 -> 84 -> 19으로 해서 10개의 class로 분류한다고 생각하면 된다. Lenet 학습도 진행하였는데, 위의 코드에서는 10epoch동안 학습하였고 손실 함수로는 교차 엔트로피를 사용하고, mini-batch SGD로 최적화하였다.

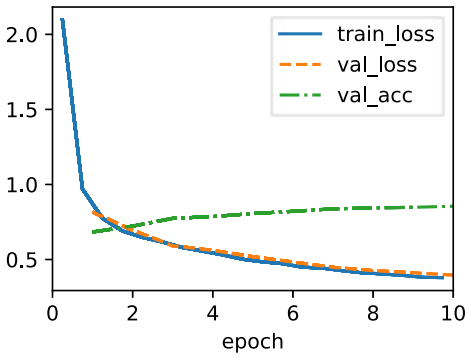
Exercise - Let's modernize LeNet. Implement and test the following changes:

Replace average pooling with max-pooling.

Replace the softmax layer with ReLU.

In [9]:
```
class modernizedLeNet(d2l.Classifier):
    """The LeNet-5 model."""
    def __init__(self, lr=0.1, num_classes=10):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(
            nn.LazyConv2d(6, kernel_size=5, padding=2), nn.Sigmoid(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.LazyConv2d(16, kernel_size=5), nn.Sigmoid(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Flatten(),
            nn.LazyLinear(120), nn.ReLU(),
            nn.LazyLinear(84), nn.ReLU(),
            nn.LazyLinear(num_classes))
```

In [10]:
```
trainer = d2l.Trainer(max_epochs=10, num_gpus=1)
data = d2l.FashionMNIST(batch_size=128)
model = modernizedLeNet(lr=0.1)
model.apply_init([next(iter(data.get_dataloader(True)))[0]], init_cnn)
trainer.fit(model, data)
```



해본 결과, train, val loss가 훨씬 더 적은 epoch부터 줄어든 결과가 나왔다.