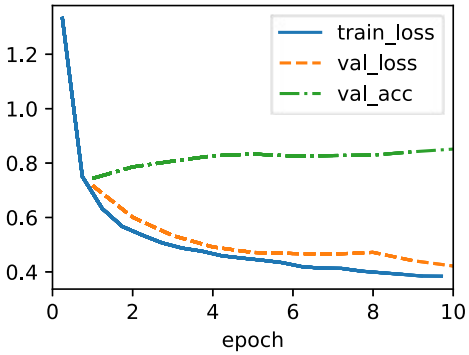
In [1]:
```
import torch
from torch import nn
from d2l import torch as d2l
```

In [2]:
```
class MLPScratch(d2l.Classifier):
    def __init__(self, num_inputs, num_outputs, num_hiddens, lr, sigma=0.01):
        super().__init__()
        self.save_hyperparameters()
        self.W1 = nn.Parameter(torch.randn(num_inputs, num_hiddens) * sigma)
        self.b1 = nn.Parameter(torch.zeros(num_hiddens))
        self.W2 = nn.Parameter(torch.randn(num_hiddens, num_outputs) * sigma)
        self.b2 = nn.Parameter(torch.zeros(num_outputs))
```

In [3]:
```
def relu(X):
    a = torch.zeros_like(X)
    return torch.max(X, a)
```

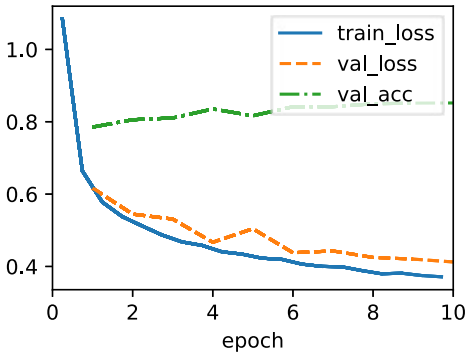
In [4]:
```
@d2l.add_to_class(MLPScratch)
def forward(self, X):
    X = X.reshape((-1, self.num_inputs))
    H = relu(torch.matmul(X, self.W1) + self.b1)
    return torch.matmul(H, self.W2) + self.b2
```

In [5]:
```
model = MLPScratch(num_inputs=784, num_outputs=10, num_hiddens=256, lr=0.1)
data = d2l.FashionMNIST(batch_size=256)
trainer = d2l.Trainer(max_epochs=10)
trainer.fit(model, data)
```



In [6]:
```
class MLP(d2l.Classifier):
    def __init__(self, num_outputs, num_hiddens, lr):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(nn.Flatten(), nn.LazyLinear(num_hiddens),
                                 nn.ReLU(), nn.LazyLinear(num_outputs))
```

In [7]:
```
model = MLP(num_outputs=10, num_hiddens=256, lr=0.1)
trainer.fit(model, data)
```



Discussion: In this chapter, I had the opportunity to actually implement a Multilayer Perceptron and conduct experiments. In the training process before the Concise Implementation, the example on this website showed fluctuations that gradually reduced, but in my case, both the val_loss and val_acc steadily approached the target without any noticeable dips. This made me realize that the intermediate process can vary greatly from person to person and with each run. Once again, I was reminded that this isn't about finding a definitive answer, but rather an artificial intelligence subject.