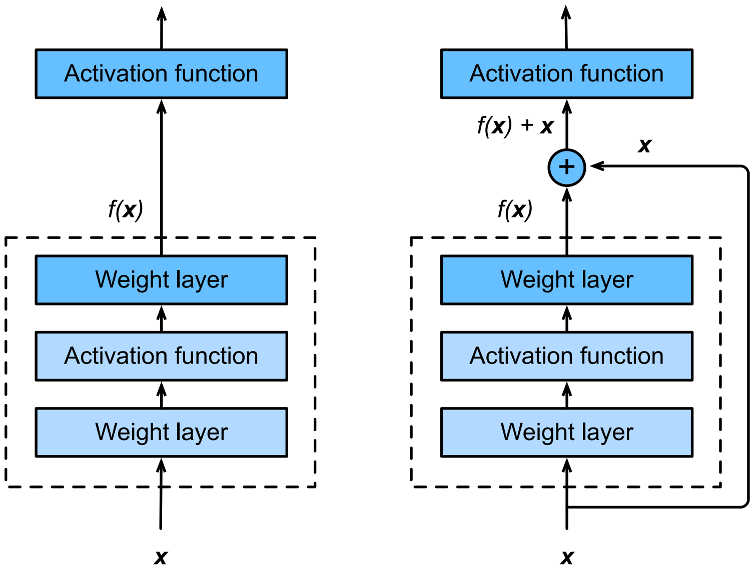
Residual Networks (ResNet)

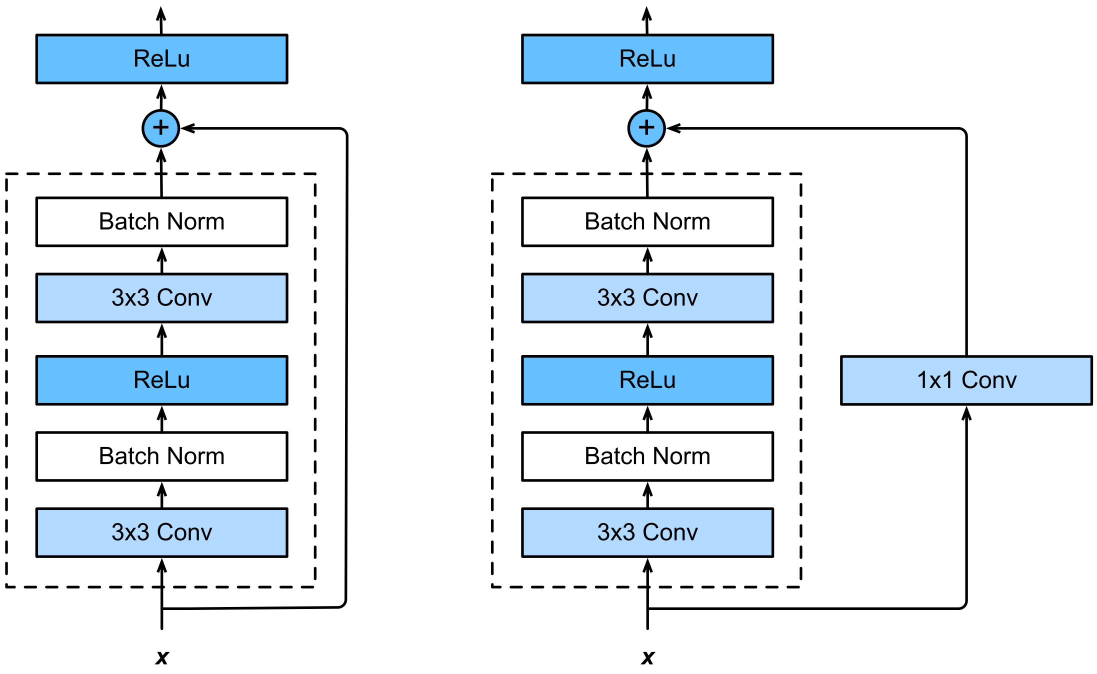


In [1]:
```
import d2l
from mxnet import gluon, init, nd
from mxnet.gluon import nn
```

In [2]:
```
class Residual(nn.Block): # This class is part of the d2l package
    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=str
ides)
        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 forward(self, X):
        Y = nd.relu(self.bn1(self.conv1(X)))
        Y = self.bn2(self.conv2(Y))
        if self.conv3:
            X = self.conv3(X)
        return nd.relu(Y + X)
```

Networks


ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x
ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
1x1 Conv
x

In [3]:
```
blk = Residual(3)
blk.initialize()
X = nd.random.uniform(shape=(4, 3, 6, 6))
blk(X).shape
```

Out[3]: (4, 3, 6, 6)

We also have the option to halve the output height and width while increasing the number of output channels.

In [4]:
```
blk = Residual(6, use_1x1conv=True, strides=2)
blk.initialize()
blk(X).shape
```

Out[4]: (4, 6, 3, 3)

ResNet Model Stage 1

ResNet and GoogLeNet are quite similar on the initial layers.

In [5]:
```
net = nn.Sequential()
net.add(nn.Conv2D(64, kernel_size=7, strides=2, padding=3),
        nn.BatchNorm(), nn.Activation('relu'),
        nn.MaxPool2D(pool_size=3, strides=2, padding=1))
```

We also need a ResNet block.

In [6]:
```
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(Residual(num_channels, use_1x1conv=True, strides=2))
        else:
            blk.add(Residual(num_channels))
    return blk
```

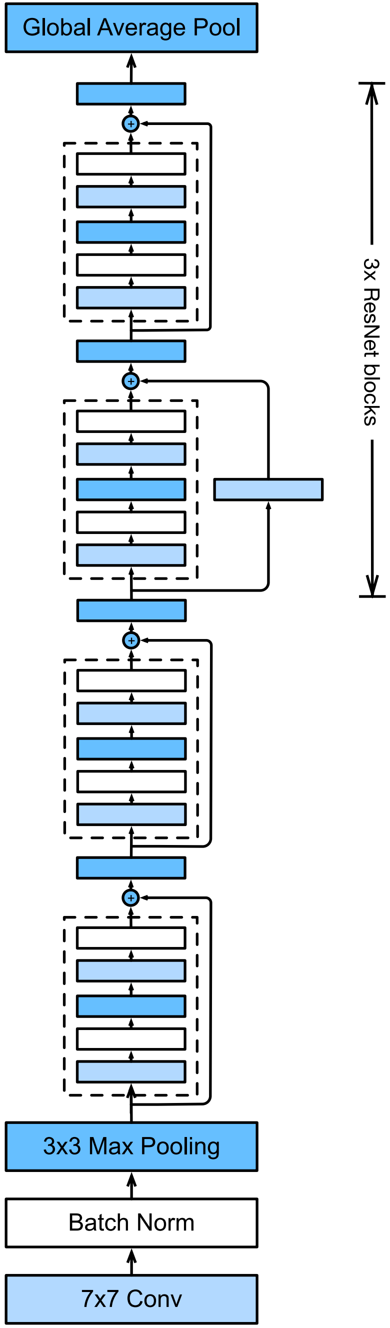
Then, we add all the residual blocks to ResNet. Here, two residual blocks are used for each module.

In [7]:
```
net.add(resnet_block(64, 2, first_block=True),
        resnet_block(128, 2),
        resnet_block(256, 2),
        resnet_block(512, 2))
```

Finally, just like GoogLeNet, we add a global average pooling layer, followed by the fully connected layer output.

In [8]:
```
net.add(nn.GlobalAvgPool2D(), nn.Dense(10))
```

Full ResNet-18


Global Average Pool
3x ResNet blocks
3x3 Max Pooling
Batch Norm
7x7 Conv

In [9]:

```
X = nd.random.uniform(shape=(1, 1, 224, 224))
net.initialize()
for layer in net:
    X = layer(X)
    print(layer.name, 'output shape:\t', X.shape)
```

```
conv5 output shape:      (1, 64, 112, 112)
batchnorm4 output shape:         (1, 64, 112, 112)
relu0 output shape:      (1, 64, 112, 112)
pool0 output shape:      (1, 64, 56, 56)
sequential1 output shape:        (1, 64, 56, 56)
sequential2 output shape:        (1, 128, 28, 28)
sequential3 output shape:        (1, 256, 14, 14)
sequential4 output shape:        (1, 512, 7, 7)
pool1 output shape:      (1, 512, 1, 1)
dense0 output shape:     (1, 10)
```

Data Acquisition and Training

We train ResNet on the Fashion-MNIST data set, just like before. The only thing that has changed is the learning rate that decreased again, due to the more complex architecture.

In [10]:

```
lr, num_epochs, batch_size, ctx = 0.05, 5, 256, d2l.try_gpu()
net.initialize(force_reinit=True, ctx=ctx, init=init.Xavier())
trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate': lr})
train_iter, test_iter = d2l.load_data_fashion_mnist(batch_size, resize=96)
d2l.train_ch5(net, train_iter, test_iter, batch_size, trainer, ctx, num_epochs)
```

```
training on gpu(0)
epoch 1, loss 0.4824, train acc 0.830, test acc 0.887, time 15.5 sec
epoch 2, loss 0.2572, train acc 0.906, test acc 0.903, time 13.8 sec
epoch 3, loss 0.1976, train acc 0.927, test acc 0.904, time 13.8 sec
epoch 4, loss 0.1481, train acc 0.947, test acc 0.915, time 13.8 sec
epoch 5, loss 0.1134, train acc 0.960, test acc 0.917, time 13.8 sec
```