AlexNet

Use it on scaled-up 224x224 dimensional Fashion-MNIST with 10 class output.

In [1]:

```
import d2l
from mxnet import autograd, gluon, init, nd
from mxnet.gluon import data as gdata, nn
from mxnet.gluon import loss as gloss
import os
import sys
import time
```

In [2]:

```
net = nn.Sequential()
# Use a larger 11 x 11 window to capture objects. Stride of 4 to reduce size.
# The number of output channels is much larger than that in LeNet.
net.add(nn.Conv2D(96, kernel_size=11, strides=4, activation='relu'),
        nn.MaxPool2D(pool_size=3, strides=2),
        # Make the convolution window smaller, set padding to 2 for consistent hei
ght/width
        nn.Conv2D(256, kernel_size=5, padding=2, activation='relu'),
        nn.MaxPool2D(pool_size=3, strides=2),
        # Use three successive convolutional layers and a smaller convolution wind
ow.
        nn.Conv2D(384, kernel_size=3, padding=1, activation='relu'),
        nn.Conv2D(384, kernel_size=3, padding=1, activation='relu'),
        nn.Conv2D(256, kernel_size=3, padding=1, activation='relu'),
        # Reduce dimensionality
        nn.MaxPool2D(pool_size=3, strides=2),
        # Expensive dense layer
        nn.Dense(4096, activation="relu"), nn.Dropout(0.5),
        nn.Dense(4096, activation="relu"), nn.Dropout(0.5),
        # Output layer. Since we are using Fashion-MNIST, n = 10
        nn.Dense(10))
```

We construct a single-channel data instance with both height and width of 224 to observe the output shape of each layer. It matches our diagram above.

In [3]:
```
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)
```

```
conv0 output shape:      (1, 96, 54, 54)
pool0 output shape:      (1, 96, 26, 26)
conv1 output shape:      (1, 256, 26, 26)
pool1 output shape:      (1, 256, 12, 12)
conv2 output shape:      (1, 384, 12, 12)
conv3 output shape:      (1, 384, 12, 12)
conv4 output shape:      (1, 256, 12, 12)
pool2 output shape:      (1, 256, 5, 5)
dense0 output shape:     (1, 4096)
dropout0 output shape:   (1, 4096)
dense1 output shape:     (1, 4096)
dropout1 output shape:   (1, 4096)
dense2 output shape:     (1, 10)
```

Reading Data

In [4]:

```
def load_data_fashion_mnist(batch_size, resize=None, root=os.path.join(
        '~', '.mxnet', 'datasets', 'fashion-mnist')):
    root = os.path.expanduser(root)  # Expand the user path '~'.
    transformer = []
    if resize:
        transformer += [gdata.vision.transforms.Resize(resize)]
    transformer += [gdata.vision.transforms.ToTensor()]
    transformer = gdata.vision.transforms.Compose(transformer)
    mnist_train = gdata.vision.FashionMNIST(root=root, train=True)
    mnist_test = gdata.vision.FashionMNIST(root=root, train=False)
    num_workers = 0 if sys.platform.startswith('win32') else 4
    train_iter = gdata.DataLoader(
        mnist_train.transform_first(transformer), batch_size, shuffle=True,
        num_workers=num_workers)
    test_iter = gdata.DataLoader(
        mnist_test.transform_first(transformer), batch_size, shuffle=False,
        num_workers=num_workers)
    return train_iter, test_iter

batch_size = 128
train_iter, test_iter = load_data_fashion_mnist(batch_size, resize=224)
```

```
Downloading /home/ubuntu/.mxnet/datasets/fashion-mnist/train-images-idx3-ubyt
e.gz from https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/dat
aset/fashion-mnist/train-images-idx3-ubyte.gz...
Downloading /home/ubuntu/.mxnet/datasets/fashion-mnist/train-labels-idx1-ubyt
e.gz from https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/dat
aset/fashion-mnist/train-labels-idx1-ubyte.gz...
Downloading /home/ubuntu/.mxnet/datasets/fashion-mnist/t10k-images-idx3-ubyte.
gz from https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/datas
et/fashion-mnist/t10k-images-idx3-ubyte.gz...
Downloading /home/ubuntu/.mxnet/datasets/fashion-mnist/t10k-labels-idx1-ubyte.
gz from https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/datas
et/fashion-mnist/t10k-labels-idx1-ubyte.gz...
```

Training

In [5]:
```
def evaluate_accuracy(data_iter, net, ctx):
    acc_sum, n = nd.array([0], ctx=ctx), 0
    for X, y in data_iter:
        # If ctx is the GPU, copy the data to the GPU.
        X, y = X.as_in_context(ctx), y.as_in_context(ctx).astype('float32')
        acc_sum += (net(X).argmax(axis=1) == y).sum()
        n += y.size
    return acc_sum.asscalar() / n

def train(net, train_iter, test_iter, batch_size, trainer, ctx,
          num_epochs):
    print('training on', ctx)
    loss = gloss.SoftmaxCrossEntropyLoss()
    for epoch in range(num_epochs):
        train_l_sum, train_acc_sum, n, start = 0.0, 0.0, 0, time.time()
        for X, y in train_iter:
            X, y = X.as_in_context(ctx), y.as_in_context(ctx)
            with autograd.record():
                y_hat = net(X)
                l = loss(y_hat, y).sum()
            l.backward()
            trainer.step(batch_size)
            y = y.astype('float32')
            train_l_sum += l.asscalar()
            train_acc_sum += (y_hat.argmax(axis=1) == y).sum().asscalar()
            n += y.size
        test_acc = evaluate_accuracy(test_iter, net, ctx)
        print('epoch %d, loss %.4f, train acc %.3f, test acc %.3f, '
              'time %.1f sec'
              % (epoch + 1, train_l_sum / n, train_acc_sum / n, test_acc,
                 time.time() - start))
```

Compared to LeNet the main change is the smaller learning rate and much slower progress due to much larger images.

In [6]:
```
lr, num_epochs, ctx = 0.01, 5, d2l.try_gpu()
net.initialize(force_reinit=True, ctx=ctx, init=init.Xavier())
trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate': lr})
train(net, train_iter, test_iter, batch_size, trainer, ctx, num_epochs)
```

```
training on gpu(0)
epoch 1, loss 1.3052, train acc 0.509, test acc 0.745, time 18.9 sec
epoch 2, loss 0.6477, train acc 0.756, test acc 0.815, time 22.5 sec
epoch 3, loss 0.5331, train acc 0.803, test acc 0.831, time 16.6 sec
epoch 4, loss 0.4651, train acc 0.829, test acc 0.855, time 16.6 sec
epoch 5, loss 0.4276, train acc 0.844, test acc 0.867, time 16.6 sec
```