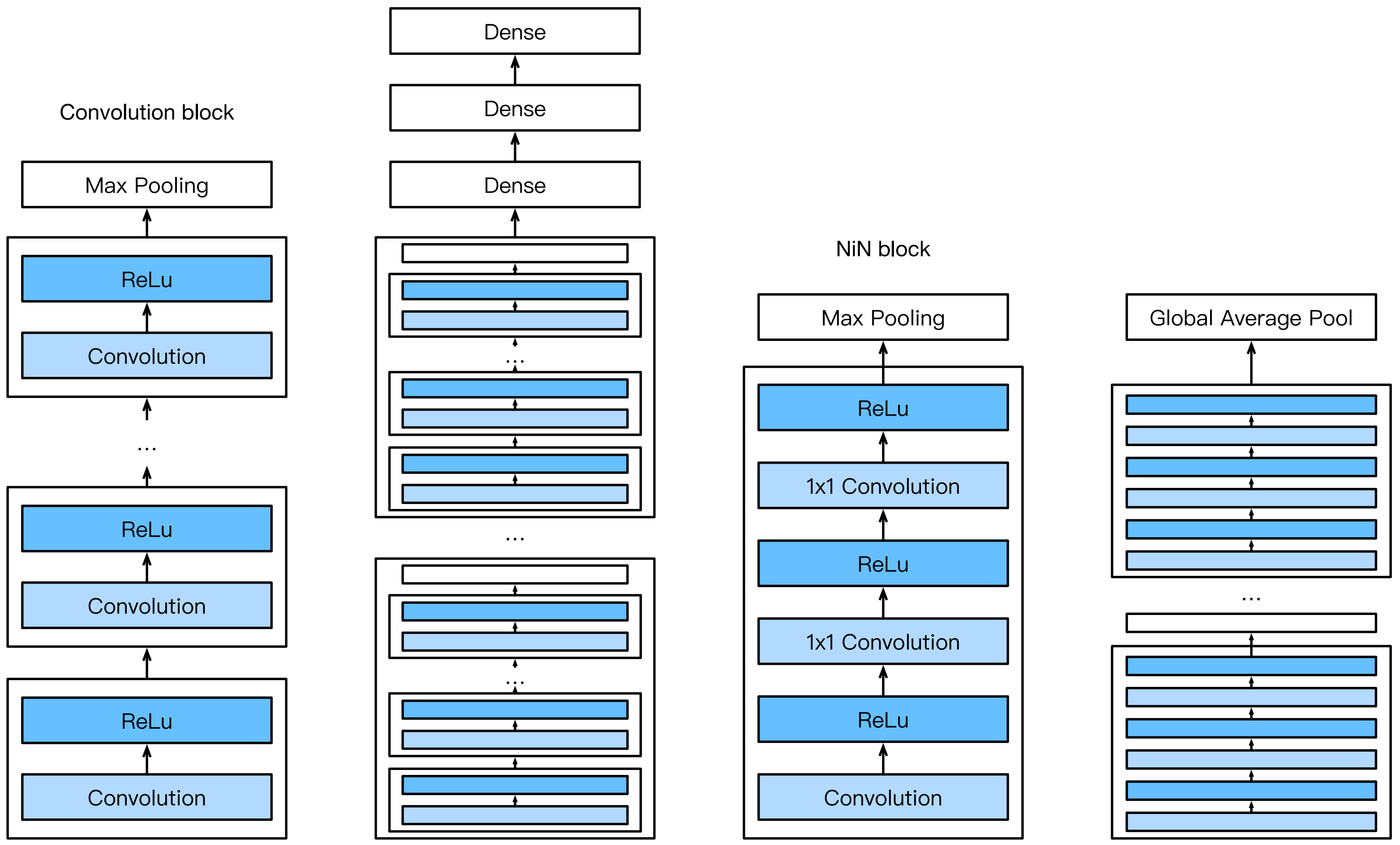
Network in Network (NiN)



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

In [2]:
```
def nin_block(num_channels, kernel_size, strides, padding):
    blk = nn.HybridSequential()
    blk.add(nn.Conv2D(num_channels, kernel_size, strides, padding, activation='rel
u'),
            nn.Conv2D(num_channels, kernel_size=1, activation='relu'),
            nn.Conv2D(num_channels, kernel_size=1, activation='relu'))
    return blk
```

NiN Model

In [3]:
```
net = nn.HybridSequential()
net.add(nin_block(96, kernel_size=11, strides=4, padding=0),
        nn.MaxPool2D(pool_size=3, strides=2),
        nin_block(256, kernel_size=5, strides=1, padding=2),
        nn.MaxPool2D(pool_size=3, strides=2),
        nin_block(384, kernel_size=3, strides=1, padding=1),
        nn.MaxPool2D(pool_size=3, strides=2),
        nn.Dropout(0.5),
        # There are 10 label classes.
        nin_block(10, kernel_size=3, strides=1, padding=1),
        # The global average pooling layer aggregates over all entries
        nn.GlobalAvgPool2D(),
        # Transform the four-dimensional output into two-dimensional output
        nn.Flatten())
```

Output shapes throughout the network

In [4]:
```python
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)
```

```
hybridsequential1 output shape:  (1, 96, 54, 54)
pool0 output shape:      (1, 96, 26, 26)
hybridsequential2 output shape:  (1, 256, 26, 26)
pool1 output shape:      (1, 256, 12, 12)
hybridsequential3 output shape:  (1, 384, 12, 12)
pool2 output shape:      (1, 384, 5, 5)
dropout0 output shape:   (1, 384, 5, 5)
hybridsequential4 output shape:  (1, 10, 5, 5)
pool3 output shape:      (1, 10, 1, 1)
flatten0 output shape:   (1, 10)
```

Training

In [5]:
```
lr, num_epochs, batch_size, ctx = 0.1, 5, 128, d2l.try_gpu()
net.initialize(force_reinit=True, ctx=ctx, init=init.Xavier())
net.hybridize()
trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate': lr})
train_iter, test_iter = d2l.load_data_fashion_mnist(batch_size, resize=224)
d2l.train_ch5(net, train_iter, test_iter, batch_size, trainer, ctx, num_epochs)
```

```
training on gpu(0)
epoch 1, loss 2.2738, train acc 0.166, test acc 0.286, time 23.5 sec
epoch 2, loss 1.5193, train acc 0.438, test acc 0.615, time 22.2 sec
epoch 3, loss 0.7831, train acc 0.702, test acc 0.762, time 22.1 sec
epoch 4, loss 0.6504, train acc 0.755, test acc 0.814, time 22.2 sec
epoch 5, loss 0.5287, train acc 0.803, test acc 0.828, time 22.2 sec
```