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

In [2]:
```
def pool2d(X, pool_size, mode='max'):
    p_h, p_w = pool_size
    Y = torch.zeros((X.shape[0] - p_h + 1, X.shape[1] - p_w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            if mode == 'max':
                Y[i, j] = X[i: i + p_h, j: j + p_w].max()
            elif mode == 'avg':
                Y[i, j] = X[i: i + p_h, j: j + p_w].mean()
    return Y
```

In [3]:
```
X = torch.tensor([[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]])
pool2d(X, (2, 2))
```

Out[3]:
```
tensor([[4., 5.],
        [7., 8.]])
```

In [4]:
```
pool2d(X, (2, 2), 'avg')
```

Out[4]:
```
tensor([[2., 3.],
        [5., 6.]])
```

In [5]:
```
X = torch.arange(16, dtype=torch.float32).reshape((1, 1, 4, 4))
X
```

Out[5]:
```
tensor([[[[ 0.,  1.,  2.,  3.],
          [ 4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11.],
          [12., 13., 14., 15.]]]])
```

In [6]:
```
pool2d = nn.MaxPool2d(3)
pool2d(X)
```

Out[6]:
```
tensor([[[[10.]]]])
```

In [7]:
```
pool2d = nn.MaxPool2d(3, padding=1, stride=2)
pool2d(X)
```

Out[7]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]]]])
```

In [8]:
```
pool2d = nn.MaxPool2d((2, 3), stride=(2, 3), padding=(0, 1))
pool2d(X)
```

Out[8]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]]]])
```

In [9]:
```
X = torch.cat((X, X + 1), 1)
X
```

Out[9]:
```
tensor([[[[ 0.,  1.,  2.,  3.],
          [ 4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11.],
          [12., 13., 14., 15.]],

         [[ 1.,  2.,  3.,  4.],
          [ 5.,  6.,  7.,  8.],
          [ 9., 10., 11., 12.],
          [13., 14., 15., 16.]]]])
```

In [10]:
```
pool2d = nn.MaxPool2d(3, padding=1, stride=2)
pool2d(X)
```

Out[10]:
```
tensor([[[[ 5.,  7.],
          [13., 15.]],

         [[ 6.,  8.],
          [14., 16.]]]])
```

Discussion: 이번 단원에서는 Pooling층이 어떻게 작동하는 것이고, 왜 필요한 것인지에 대해 설명하였다. 풀링은 공간 해상도와 모델의 위치에 대한 민감도를 줄이는 데 사용되며, 출력 차원을 줄여 계산의 효율을 높이는 역할을 한다. Max pooling과 Avg pooling이 있는데, Max는 말 그대로 그 pooling window 중 max값을, avg는 avg값을 출력하는 느낌이다.

나머지 코드는 이러한 pooling 층에서도 앞 단원의 padding과 stride를 설정할 수 있다는 내용에 관한 것이고, 또한 풀링은 다중 채널을 처리할 때 각 채널별로 독립적으로 작동하므로 합성곱 층과는 달리 채널 간에 값을 합치지 않고 각 채널에 대해 동일한 풀링 연산을 적용한다고 한다.

Exercise: Prove that max-pooling cannot be implemented through a convolution alone.

합성곱은 linear한 커널을 입력에 적용하는 것이다. 입력 값들의 가중 합을 계산하는 방식으로 이루어진다. 그러나, linear한 연산은 아무리 합성을 하고 하고 하고 해봐도 linear한 특성을 벗어날 수 없다. 그러나 max pooling은 max()함수를 사용하는 nonlinear operation이고, 따라서 아무리 합성곱만을 가지고 이러한 저러한 조작을 해도 절대 max pooling을 구현할 수 없다.