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

In [2]:
```
def corr2d_multi_in(X, K):
    return sum(d2l.corr2d(x, k) for x, k in zip(X, K))
```

In [3]:
```
X = torch.tensor([[[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]],
                  [[1.0, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]])
K = torch.tensor([[[0.0, 1.0], [2.0, 3.0]], [[1.0, 2.0], [3.0, 4.0]]])

corr2d_multi_in(X, K)
```

Out[3]:
```
tensor([[ 56.,  72.],
        [104., 120.]])
```

In [4]:
```
def corr2d_multi_in_out(X, K):
    return torch.stack([corr2d_multi_in(X, k) for k in K], 0)
```

In [5]:
```
K = torch.stack((K, K + 1, K + 2), 0)
K.shape
```

Out[5]:
```
torch.Size([3, 2, 2, 2])
```

In [6]:
```
corr2d_multi_in_out(X, K)
```

Out[6]:
```
tensor([[[ 56.,  72.],
         [104., 120.]],

        [[ 76., 100.],
         [148., 172.]],

        [[ 96., 128.],
         [192., 224.]]])
```

In [7]:
```
def corr2d_multi_in_out_1x1(X, K):
    c_i, h, w = X.shape
    c_o = K.shape[0]
    X = X.reshape((c_i, h * w))
    K = K.reshape((c_o, c_i))
    Y = torch.matmul(K, X)
    return Y.reshape((c_o, h, w))
```

In [8]:
```
X = torch.normal(0, 1, (3, 3, 3))
K = torch.normal(0, 1, (2, 3, 1, 1))
Y1 = corr2d_multi_in_out_1x1(X, K)
Y2 = corr2d_multi_in_out(X, K)
assert float(torch.abs(Y1 - Y2).sum()) < 1e-6
```

Discussion: 이번 단원에서는, CNN 중 다중 입력 채널 및 다중 출력 채널을 처리하는 방식에 대해서 배웠다. 이번 단원 전까지는 단일 입력 채널에 대해서 다루었지만, 실제로 이미지에서는 여러 입력 채널이 있기 때문에 이를 처리하여야 한다. 합성곱 연산을 할 때에도 마찬가지로, 입력 데이터의 채널의 수 n에 따라 각 채널마다 별도의 n차원 커널이 필요하다는 것이 중요한 내용이라고 생각했다. 이러한 입력 채널과 마찬가지로, 출력 채널 역시 다중 출력 채널이 필요한데, 이미지에는 한 가지 특징이 있는 것이 아니라 많은 특징을 가지고 있기 때문에 다중 출력 채널을 사용하여 여러 가지 특징을 동시에 학습할 수 있도록 해준다. (예를 들어, edge를 감지하는 채널과 shape를 감지하는 채널) 이렇게 출력 채널을 여러 개로 만들기 위해서는, 각 출력 채널마다 별도의 합성곱 커널이 필요하기 때문에 커널의 수가 많아지는 특징이 있다. 추가적으로, 1x1 convolution이라는 개념도 나왔는데, 이는 각 픽셀에서 channel간의 상호작용을 학습하는 데 사용하여. 각 픽셀의 다양한 특징을 통합하는 기능을 한다고 한다. 각 픽셀의 값을 linear combination하여 새로운 값을 계산하는 방식이라고 한다. 다른 기초 전공들 (자구, 알고)에서 많이 다루었던 개념이기도 한 연산의 cost는 O(input size x output size)라고 한다.

Exercise Assume that we have two convolution kernels of size k1 and k2, respectively (with no nonlinearity in between).

Prove that the result of the operation can be expressed by a single convolution. 합성곱 연산은 결합법칙이 성립하기 때문에, 차례로 입력하는 상황인 (X * k1) * k2와 , 한번에 적용하는 상황인 X * (k1 * k2)가 같을 수 밖에 없다. What is the dimensionality of the equivalent single convolution? (k1 + k2 - 1)^2 (k1의 차원이 k1^2이고, k2의 차원이 k2^2라고 가정) Is the converse true, i.e., can you always decompose a convolution into two smaller ones? 이것은 잘 몰라서 인터넷을 검색해 봤는데, 특정한 경우를 제외하고는 불가능하다고 한다. 왜냐하면 큰 커널의 학습 과정에서 얻어진 복잡한 필터링은 더 작은 커널로 나누었을 때 동일한 필터링 결과를 보장할 수 없기 때문이라고 한다.