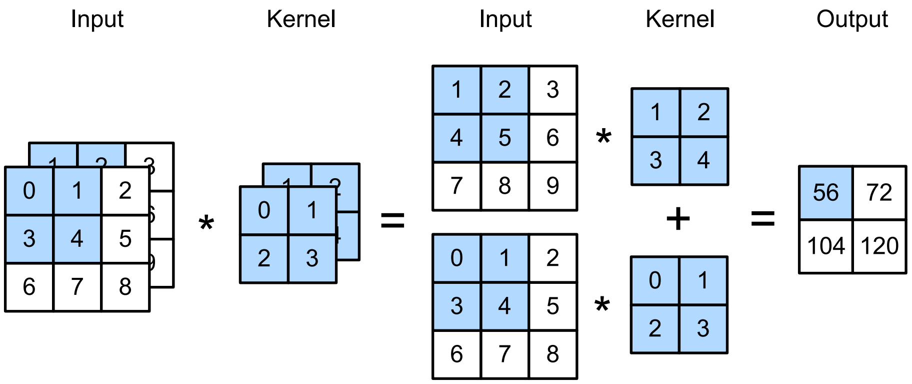
Multiple Input and Output Channels

Multiple Input Channels



In [1]:

```
import d2l
from mxnet import nd

def corr2d_multi_in(X, K):
    # First, traverse along the 0th dimension (channel dimension) of X and K.
    # Then, add them together by using *
    return nd.add_n(*[d2l.corr2d(x, k) for x, k in zip(X, K)])
```

We can construct the input array `X` and the kernel array `K` of the above diagram to validate the output of the cross-correlation operation.

In [2]:
```
X = nd.array([[[0, 1, 2], [3, 4, 5], [6, 7, 8]],
              [[1, 2, 3], [4, 5, 6], [7, 8, 9]]])
K = nd.array([[[0, 1], [2, 3]], [[1, 2], [3, 4]]])

corr2d_multi_in(X, K)
```

Out[2]:
```
[[ 56.  72.]
 [104. 120.]]
<NDArray 2x2 @cpu(0)>
```

Multiple Output Channels

For multiple output channels we simply generate multiple outputs and then stack them together.

In [3]:
```
def corr2d_multi_in_out(X, K):
    # Traverse along the 0th dimension of K, and each time, perform cross-correlat
ion
    # operations with input X. All of the results are merged together using the st
ack function.
    return nd.stack(*[corr2d_multi_in(X, k) for k in K])
```

We construct a convolution kernel with 3 output channels by concatenating the kernel array `K` with `K+1` (plus one for each element in `K`) and `K+2`.

In [4]:
```
K = nd.stack(K, K + 1, K + 2)
K.shape
```

Out[4]:
```
(3, 2, 2, 2)
```

We can have multiple input and output channels.

In [5]:
```
print(X.shape)
print(K.shape)
print(corr2d_multi_in_out(X, K))
```

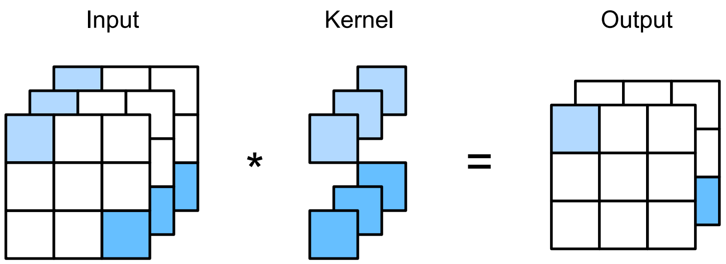
```
(2, 3, 3)
(3, 2, 2, 2)

[[[ 56.  72.]
  [104. 120.]]

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

 [[ 96. 128.]
  [192. 224.]]]
<NDArray 3x2x2 @cpu(0)>
```

1×1 Convolutions



In [6]:
```
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 = nd.dot(K, X)  # Matrix multiplication in the fully connected layer.
    return Y.reshape((c_o, h, w))
```

This is equivalent to cross-correlation with an appropriately narrow 1×1 kernel.

In [7]:

```
X = nd.random.uniform(shape=(3, 3, 3))
K = nd.random.uniform(shape=(2, 3, 1, 1))

Y1 = corr2d_multi_in_out_1x1(X, K)
Y2 = corr2d_multi_in_out(X, K)

(Y1 - Y2).norm().asscalar() < 1e-6
```

Out[7]:

```
True
```