Padding and Stride

In [1]:

```
from mxnet import nd
from mxnet.gluon import nn

# Takes convolution operation and applies it to X
def comp_conv2d(conv2d, X):
    conv2d.initialize()
    # Add two extra empty dimensions to X
    X = X.reshape((1, 1) + X.shape)
    Y = conv2d(X)
    return Y.reshape(Y.shape[2:])
```

Padding

```
In [2]:  conv2d = nn.Conv2D(1, kernel_size=3, padding=1)
         X = nd.random.uniform(shape=(8, 8))
         # Padding of 1 leaves shape unchanged for 3x3 convolution
         comp_conv2d(conv2d, X).shape

Out[2]:  (8, 8)
```

Strides

```
In [3]:  conv2d = nn.Conv2D(1, kernel_size=3, padding=1, strides=2)
         comp_conv2d(conv2d, X).shape

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

Asymmetric kernels, padding and strides

We can use different strides, different kernel sizes and different padding for height and width. This can be used, e.g. to adjust the size to a desired shape (4:3 to 1:1).

In [4]:
```
# pad only vertically and use different strides on the 8x8 image
conv2d = nn.Conv2D(1, kernel_size=(3, 5), padding=(0, 1), strides=(3, 4))
comp_conv2d(conv2d, X).shape
```

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