Convolution as matrix multiplication

A guide to convolutional arithmetic for deep learning (https://arxiv.org/pdf/1603.07285.pdf)

Convolution involves a multi-dimensional dot product over a large volume

- each location, of each input feature map

Doing this in a loop would be very expensive.

There are many highly efficient libraries for matrix multiplication

- some of which can take advantage of parallelism and GPU's.

Geron equation 13-1

We can turn convolution into matrix multiplication.

For simplicity, we will show this for a single channel, using a (3×3) kernel on a $(4 \times 4 \times 1)$ input volume.

Basically: we flatten out both the kernel matrix W

$$W = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

and the input volume matrix.

Since the input volume is (16×1), we will left multiply by a matrix with number of rows equal to the output volume, and 16 columns.

For simplicity, we do this without padding, so the output volume is (2×2) which flattened is (4×1)

$$
C
= \begin{pmatrix}
w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\
0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \\
0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 \\
0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2}
\end{pmatrix}
$$

Once you understand that the convolution result

- is obtained as CX'_l
- (where X'_l is the flattened inputs to layer l),
- you can imagine an inverse of C to go from the convolution result backwards to X'_l.

That is, we can trace backwards from each activation in a feature map to the inputs that went into its computation.

This will enable us to do back propagation.

In [3]:
```
print("Done")
```

```
Done
```