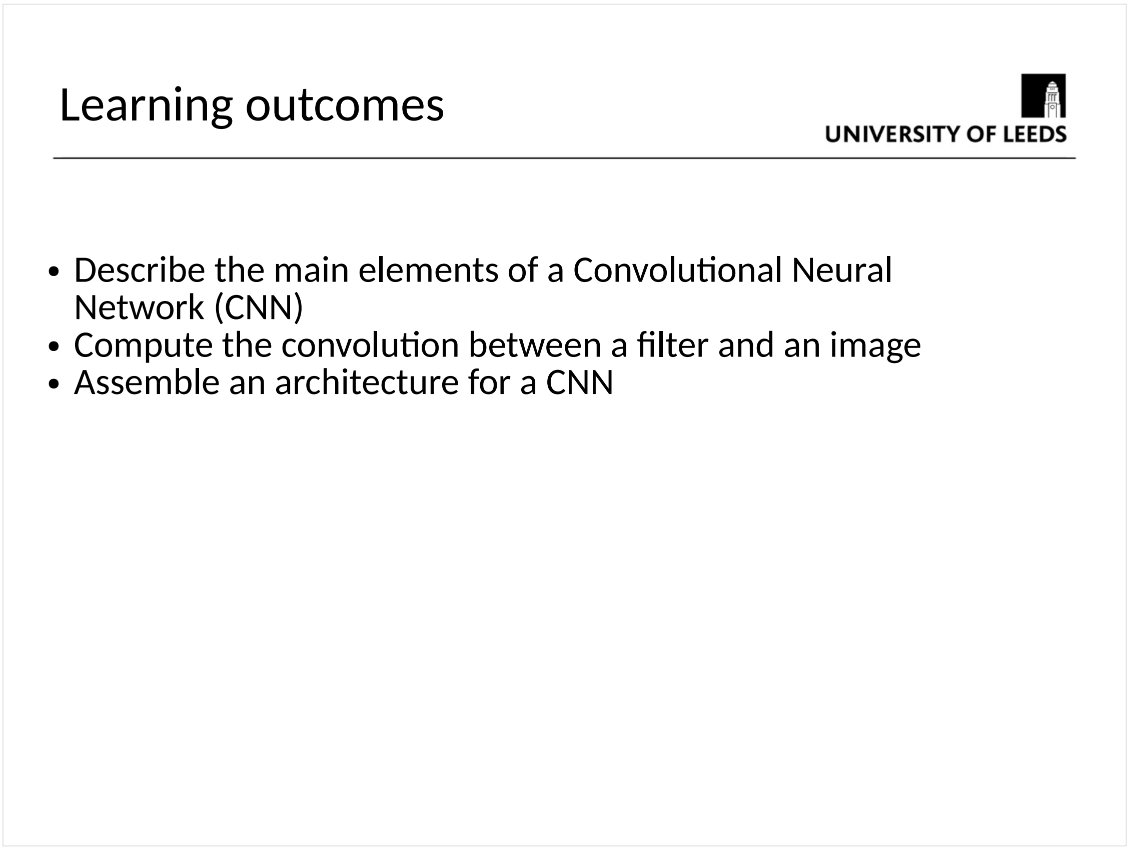
UNIVERSITY OF LEEDS

Class: Machine Learning

Convolutional Neural Networks

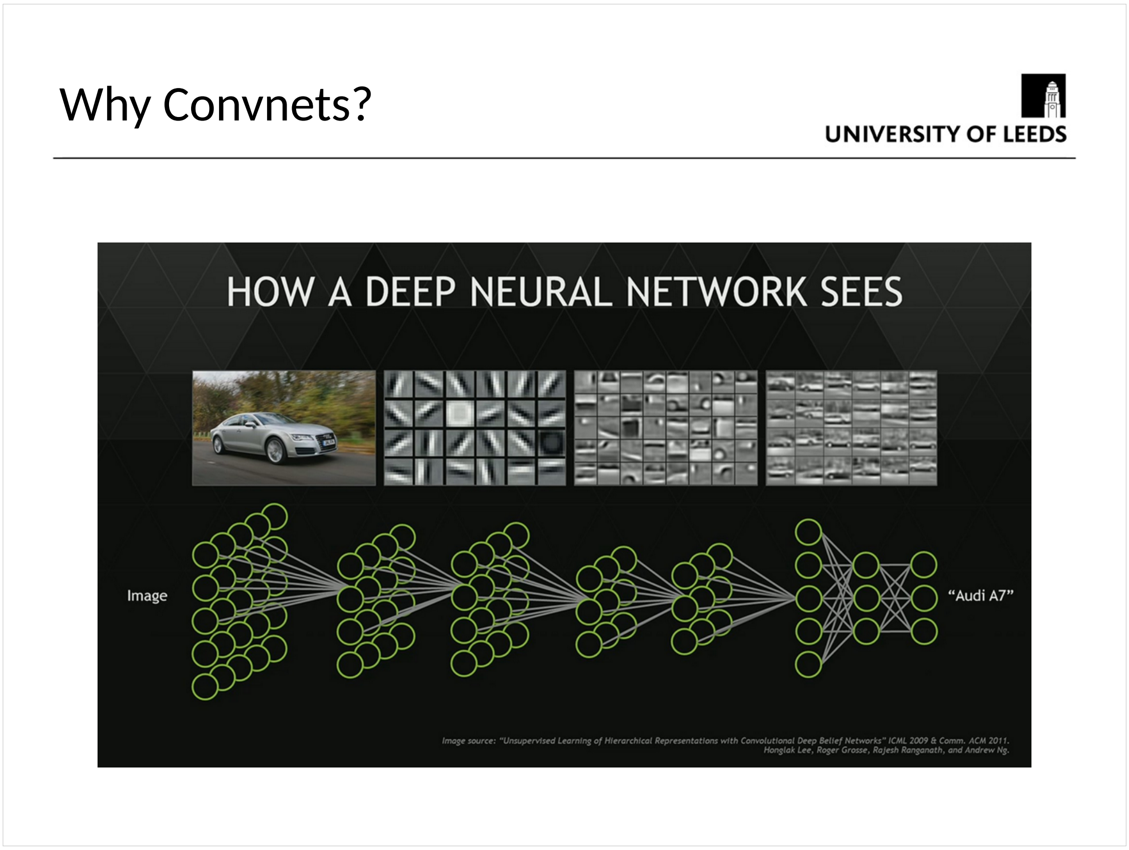
Instructor: Matteo Leonetti



Learning outcomes

UNIVERSITY OF LEEDS

- Describe the main elements of a Convolutional Neural Network (CNN)
- Compute the convolution between a filter and an image
- Assemble an architecture for a CNN



Why Convnets?

UNIVERSITY OF LEEDS

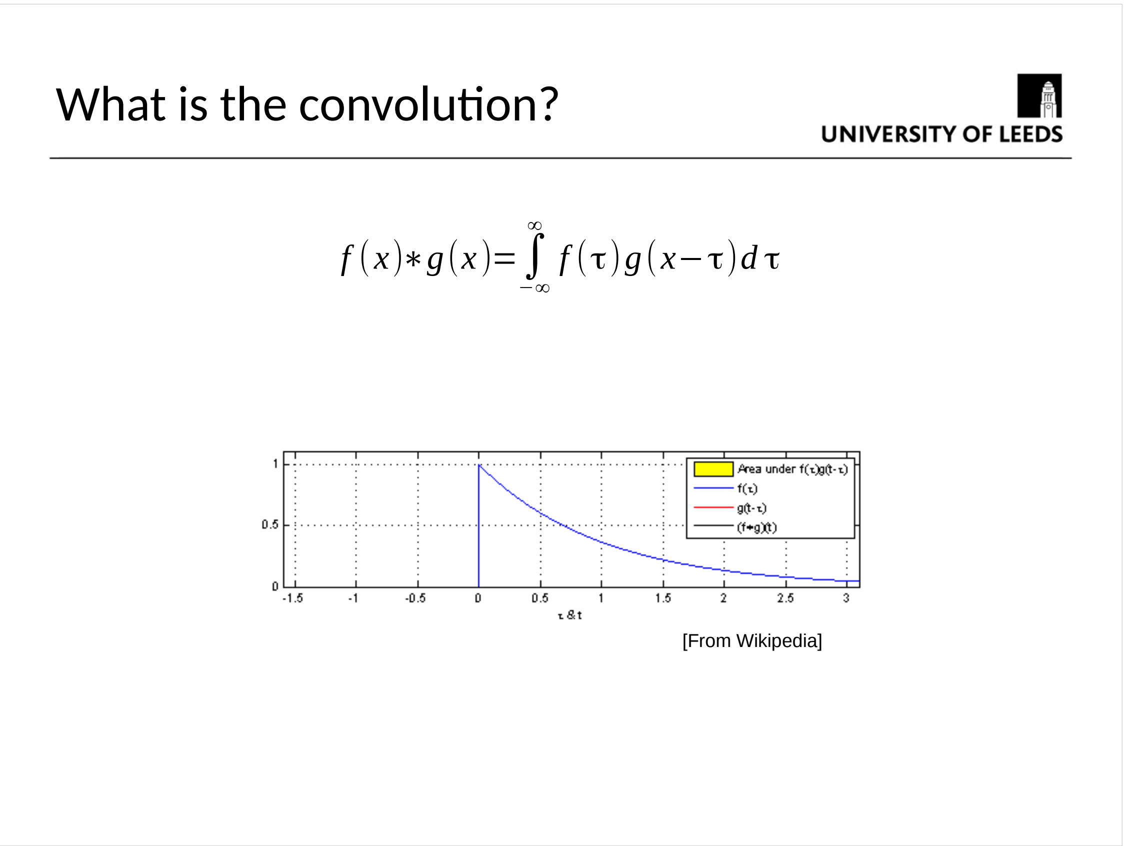
Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011. Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.

Convolutional Neural Networks are used for the classification of images.

There are tons of applications that rely on image classification, from autonomous driving to medical diagnosis.

We could give the image in input to an MLP, but the MLP would discard the spatial information: inputs are just numbers, the network does not know that those numbers are pixel values, and that certain pixels are close to one another.

Spatial proximity, however, is fundamental in image recognition. Convolutional NNs have been developed to address this limitation of the MLP.



What is the convolution?

UNIVERSITY OF LEEDS

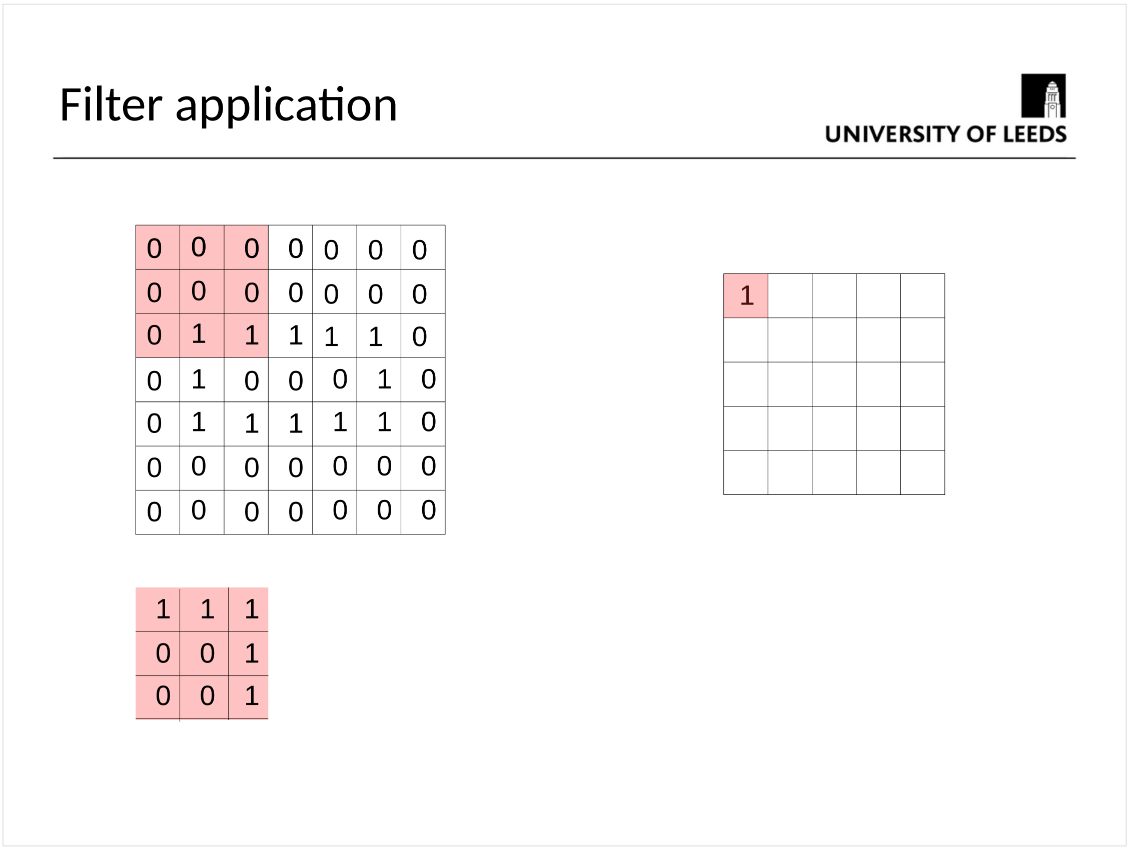
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) g(x-\tau) d\tau$$

[From Wikipedia]

The convolution of two signals is shown here in a gif from wikipedia. One signal, g, is slid over the other, and point by point the two functions are multiplied and integrated.

The important thing to remember for us, is that the convolution has these characteristics: one function is a ***filter***, and is ***slid*** across the other function. The two functions are then ***multiplied*** and integrated.

Convolutional neural networks have this name because their main operation resembles very closely the convolution between two signals. We will see that they are based on ***filters***, which are ***slid*** across the image, and ***multiplied*** by the pixels. The values are the summed (analogously to the integration).



Filter application

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	0	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

1				

1	1	1
0	0	1
0	0	1

The input image in this example is a 7x7 black & white matrix, where each pixel can be either 0 or 1.

The filter is a 3x3 matrix.

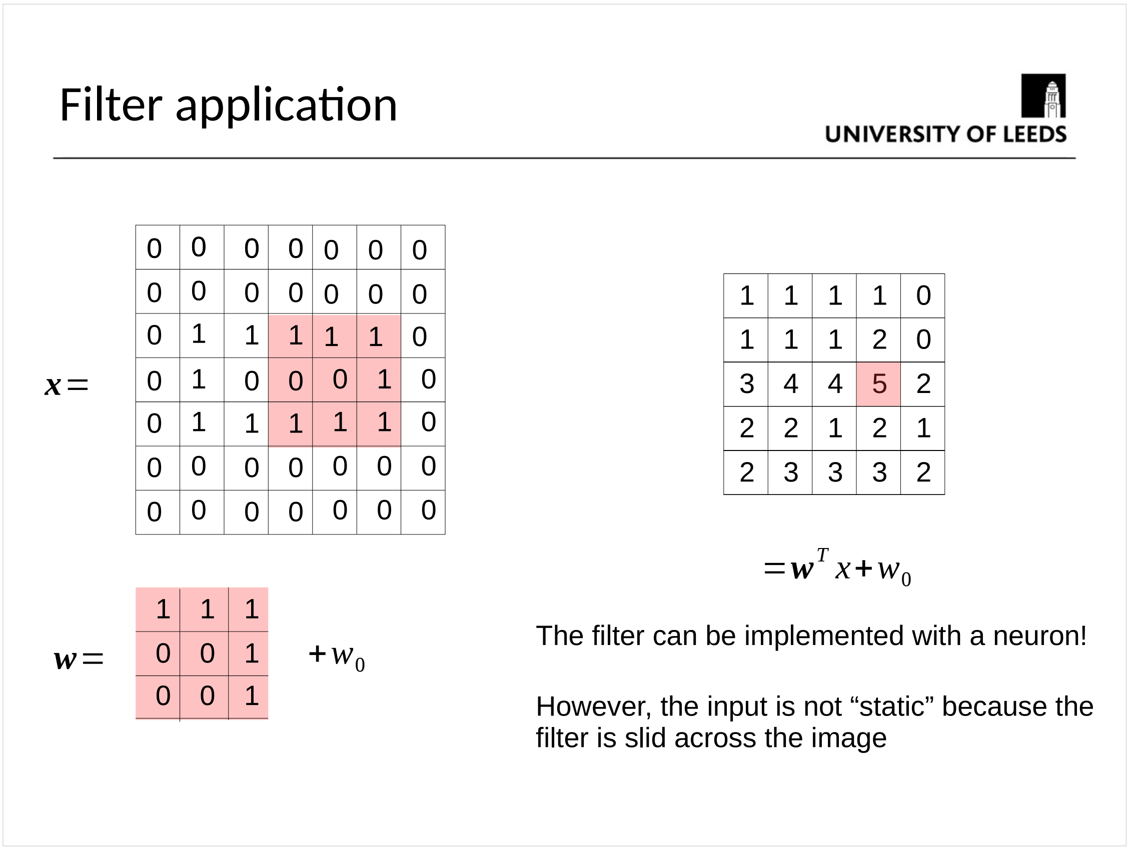
The filter is applied to every 3x3 sub-matrix of the image, each pixel is multiplied by the corresponding pixel of the filter, and all of the values are summed

Filter application

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	0	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

1	1			

1	1	1
0	0	1
0	0	1



Filter application

UNIVERSITY OF LEEDS

$\boldsymbol{x}=$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	0	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$\boldsymbol{w}=$

1	1	1
0	0	1
0	0	1

$+w_0$

1	1	1	1	0
1	1	1	2	0
3	4	4	5	2
2	2	1	2	1
2	3	3	3	2

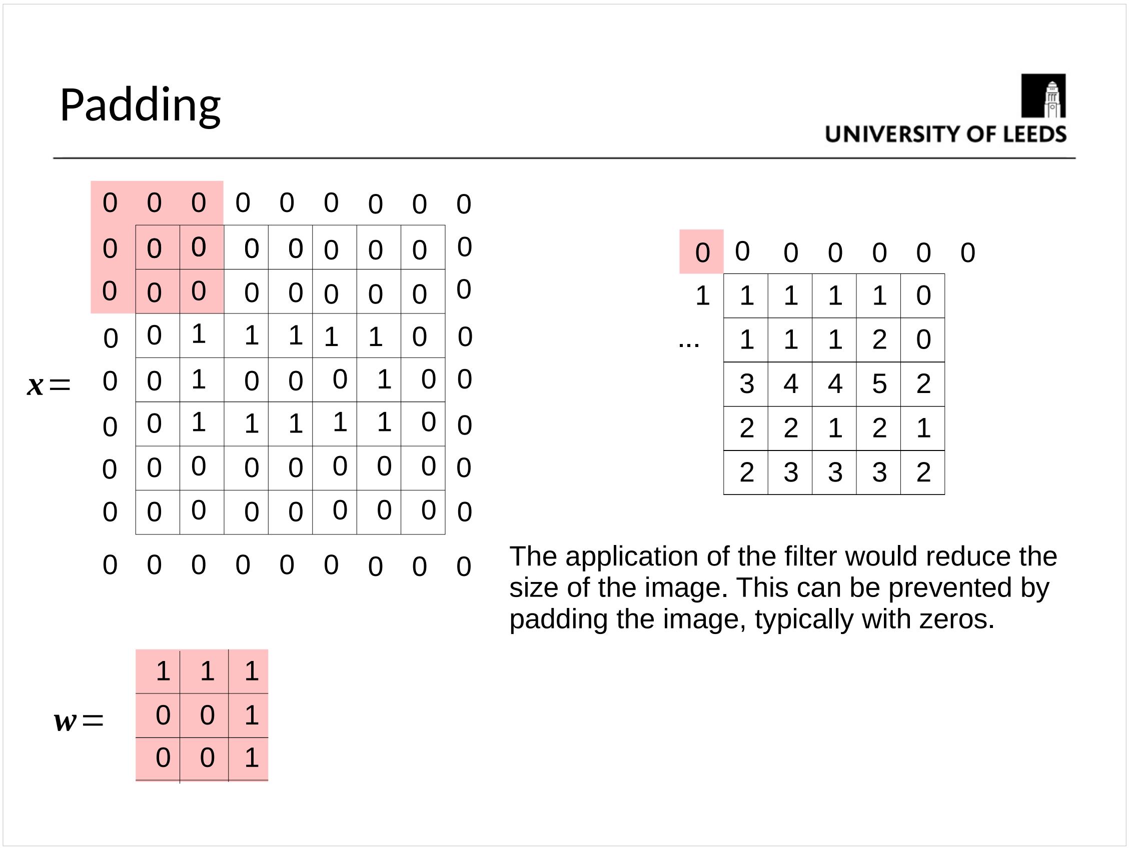
$$=\boldsymbol{w}^T x+w_0$$

The filter can be implemented with a neuron!

However, the input is not “static” because the filter is slid across the image

The operation that the filter implements is the dot product between the pixels of the filter and those of the image, plus a bias value.

We already know a component that implements this operation… the neuron!



Padding

UNIVERSITY OF LEEDS

$x =$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	0	0	0	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$w =$

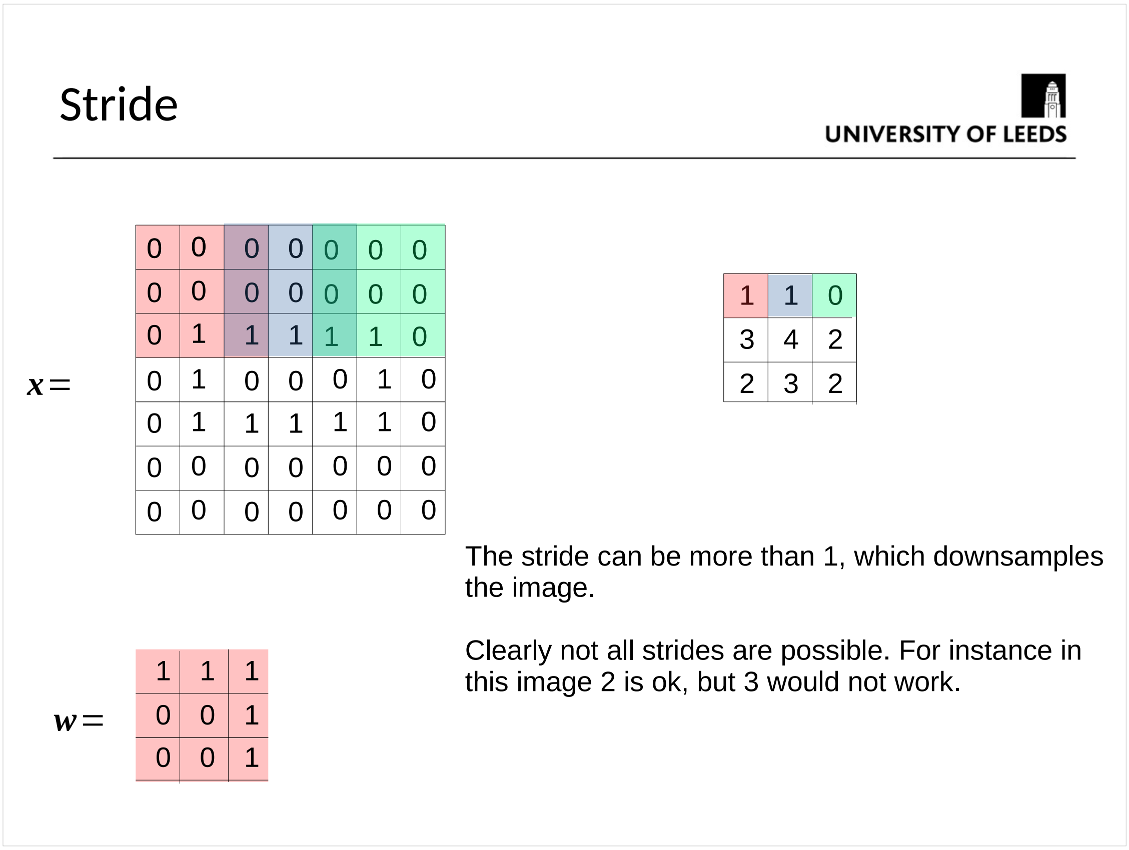
1	1	1
0	0	1
0	0	1

0	0	0	0	0	0	0
1	1	1	1	1	0	
...	1	1	1	2	0	
	3	4	4	5	2	
	2	2	1	2	1	
	2	3	3	3	2	

The application of the filter would reduce the size of the image. This can be prevented by padding the image, typically with zeros.

In order to apply the filter, we need a 3x3 matrix within the image, which leaves out the corners.

This can be avoided by padding the image, typically with zeros.



Stride

UNIVERSITY OF LEEDS

$x =$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	0	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

1	1	0
3	4	2
2	3	2

The stride can be more than 1, which downsamples the image.

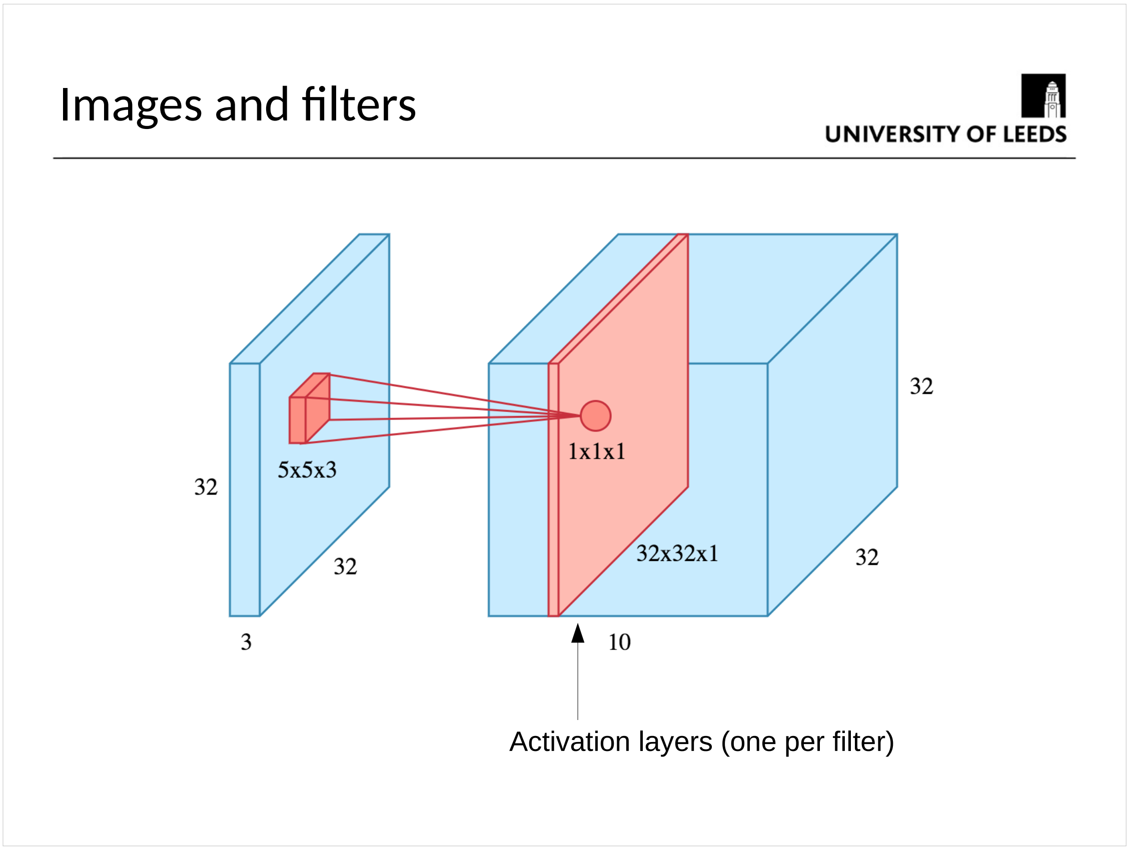
Clearly not all strides are possible. For instance in this image 2 is ok, but 3 would not work.

$w =$

1	1	1
0	0	1
0	0	1

The filter can be shifted by more than one pixel at a time, which is called the *stride*.

A stride larger than 1 effectively downsamples the image.



Images and filters

UNIVERSITY OF LEEDS

Activation layers (one per filter)

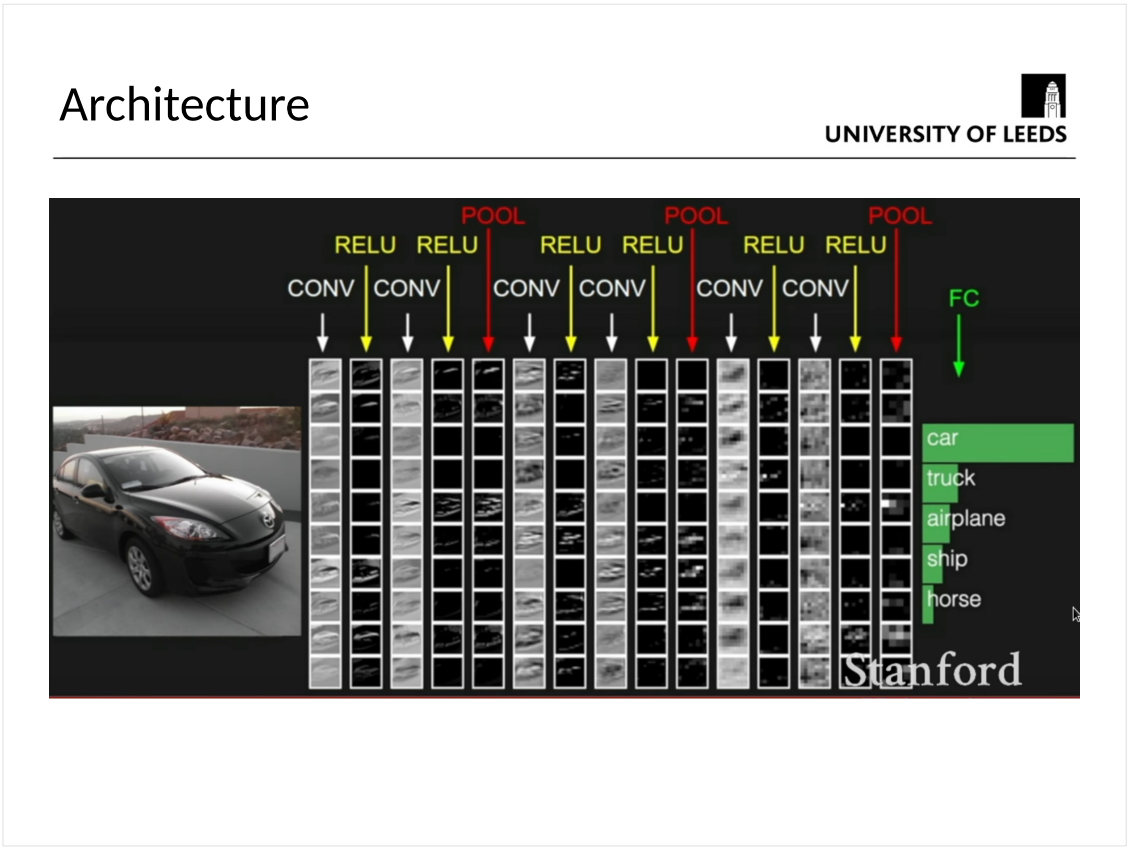
In practice, the input image will have its own size, for instance 32x32x3, where 32x32 is the number of pixels, and depth of 3 is because of the RGB channels.

A filter is a smaller matrix with the same depth as the image, for instance 5x5x3. For greyscale images the depth of both the image and the filters is 1.

In colour images, the filters are applied to all colours at the same time.

Each filter produces an activation layer of depth 1.

A convnet can use many filters, in this example it would have 10, which produce a volume of activation layers of size 32x32x10.

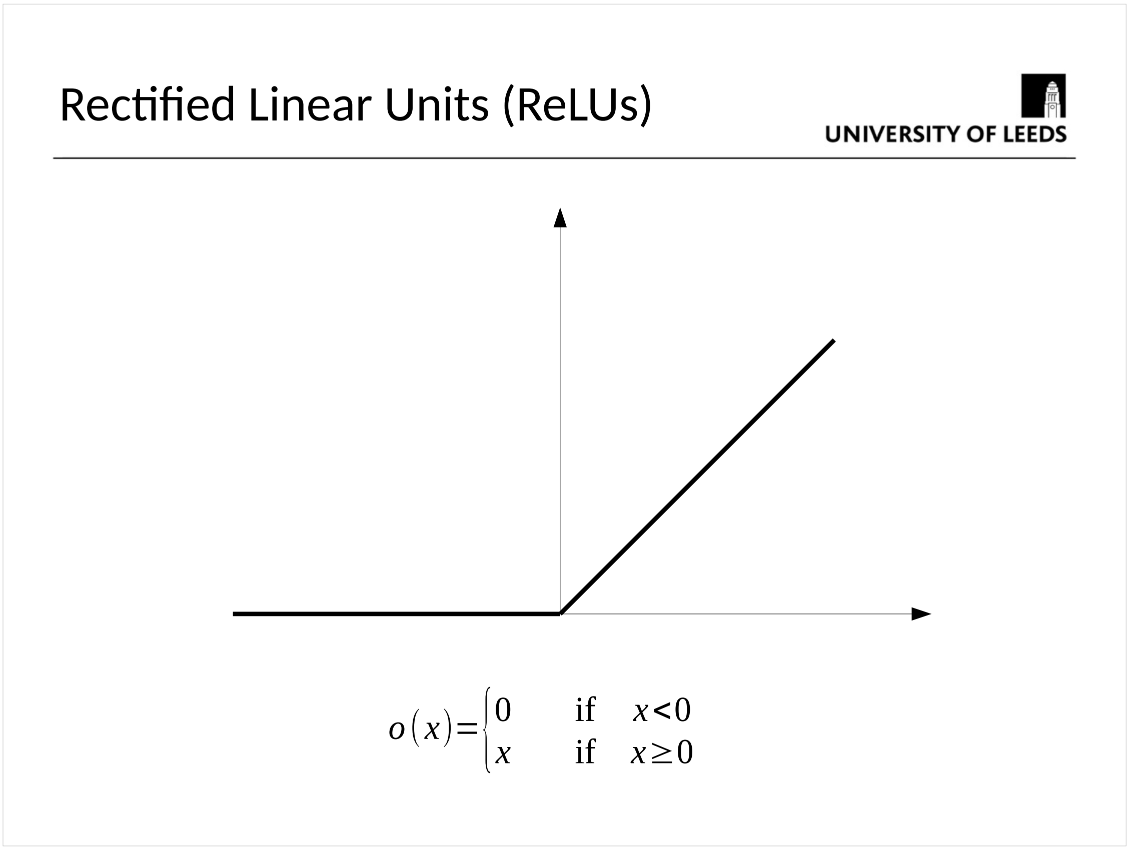


Architecture

UNIVERSITY OF LEEDS

A convnet has many layers, each one is normally either a convolutional layers (like the one we just discussed), a ReLU, or a pooling layer.

We are now going to discuss ReLUs and pooling layers.



Rectified Linear Units (ReLUs)

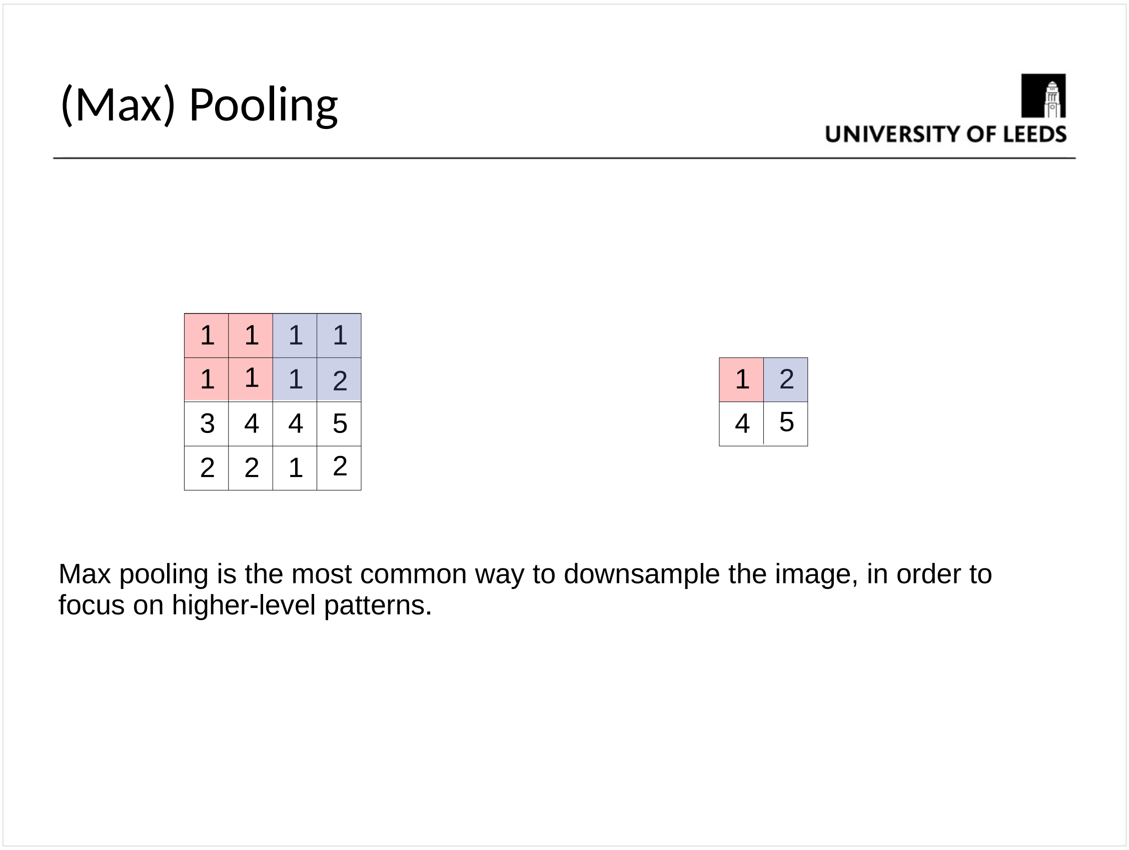
UNIVERSITY OF LEEDS

$$o(x)=\begin{cases}0 & \text{if} \quad x<0\\ x & \text{if} \quad x\geq 0\end{cases}$$

A rectified linear unit simply makes sure that the output of a convolutional layer is still an image, that is, positive.

The learned weights could be negative, and cause the output of the convolution to be negative.

The ReLU layer filters negative values, and forces them to 0, while leaving positive values intact.



(Max) Pooling

UNIVERSITY OF LEEDS

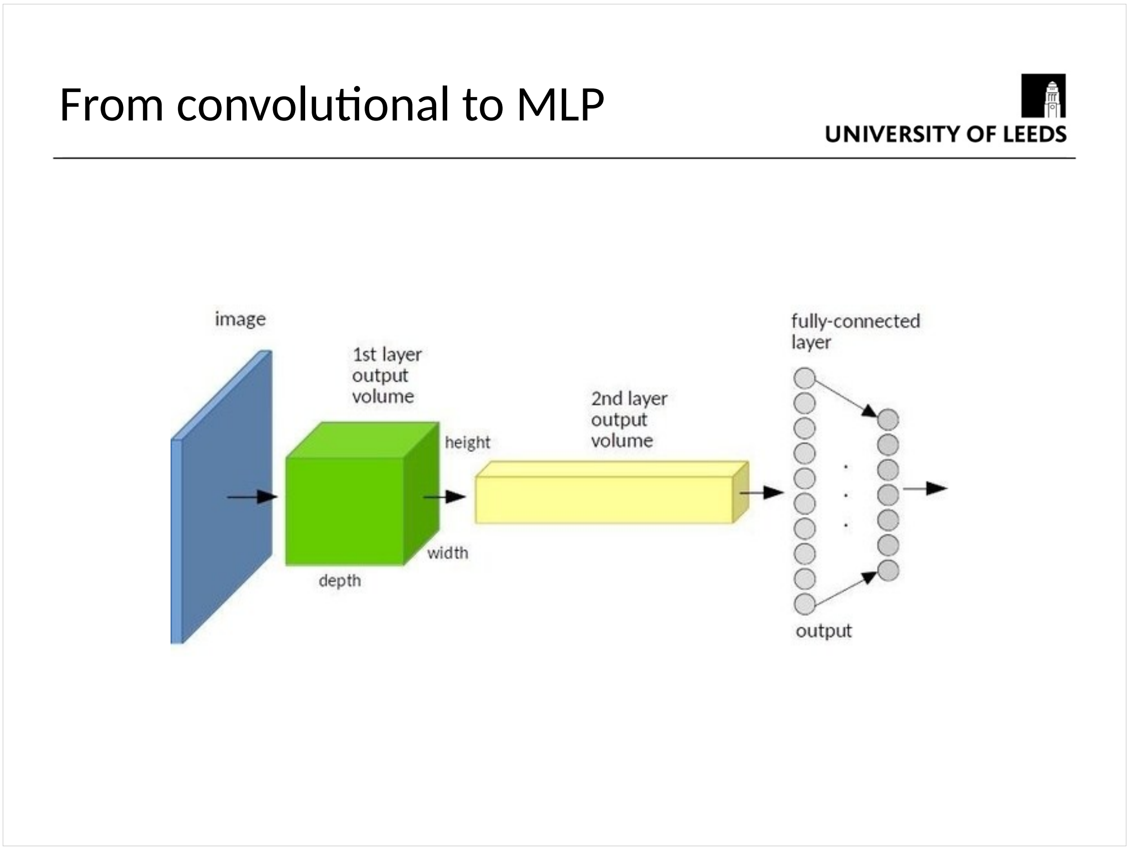
1	1	1	1
1	1	1	2
3	4	4	5
2	2	1	2

1	2
4	5

Max pooling is the most common way to downsample the image, in order to focus on higher-level patterns.

Max pooling is the most common way to downsample images.

Every few convolutional layers we want a pooling layer to reduce the resolution of the patterns the system learns.



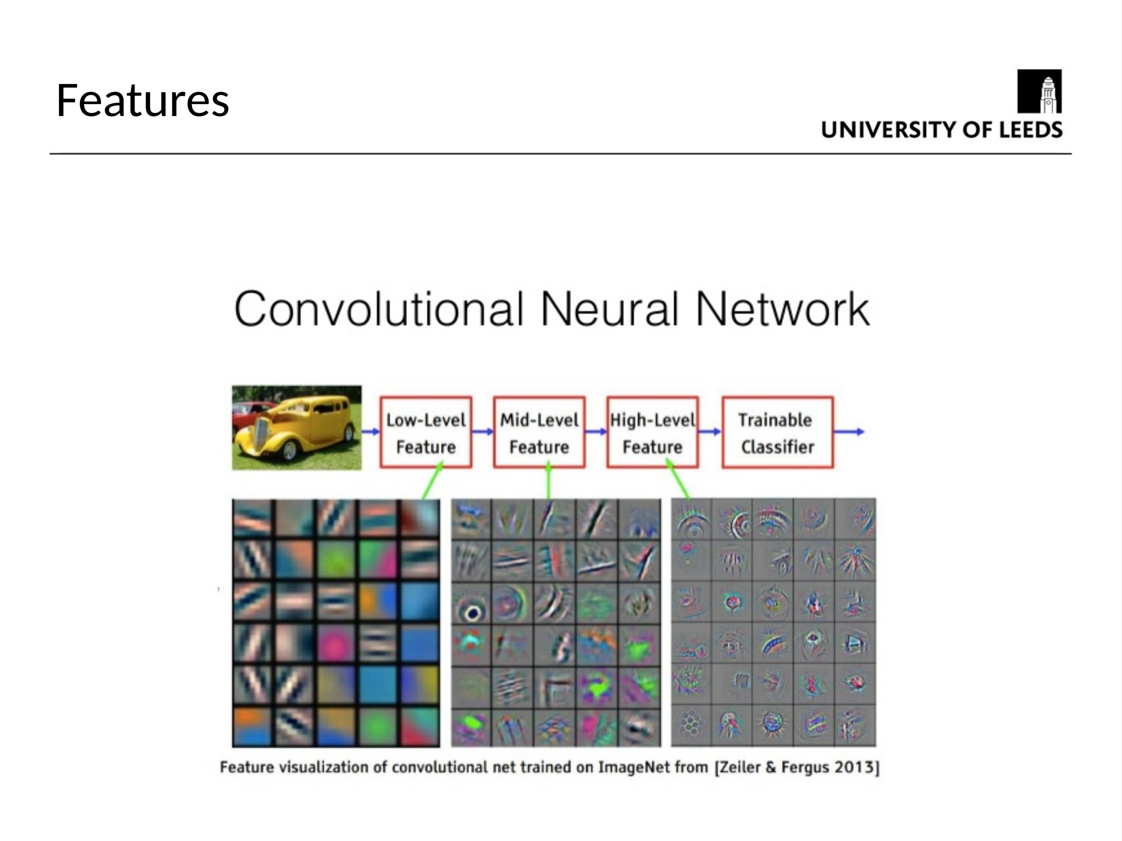
From convolutional to MLP

UNIVERSITY OF LEEDS

Convolutions and pooling for volumes of reducing resolution but increasing depth.

By the end, we get many very small activation layers, which summarise all the spatial information.

Such layers are stretched into a single long vector, and given as input to an MLP for the actual classification.



Features

UNIVERSITY OF LEEDS

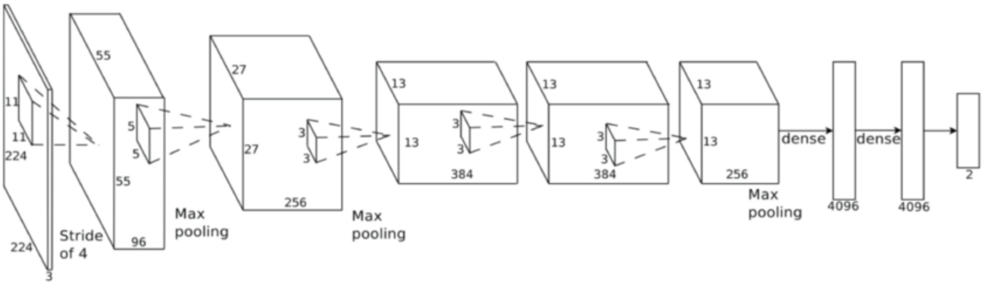
Convolutional Neural Network

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

The convolutions at different layers create an hierarchy of features, which initially recognize simple elements (such as edges and corners) and later focus on more complex shapes.

Architectures

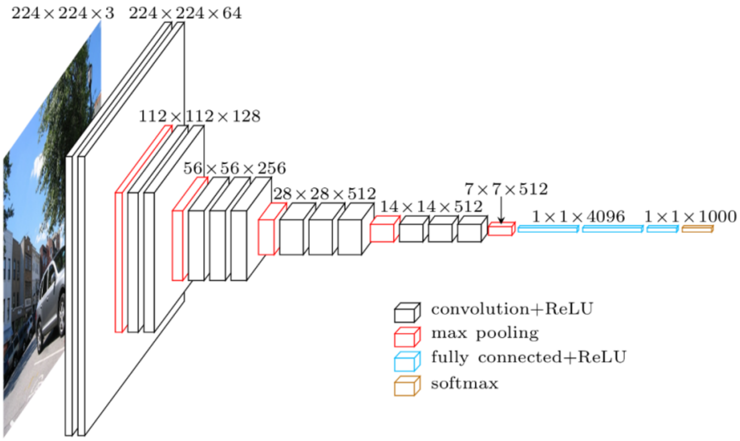
AlexNet Krizhevsky et al. in 2012



60 million parameters

Architectures

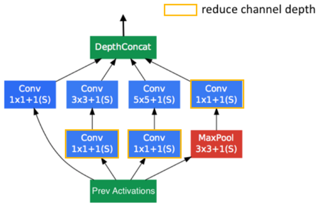
VGG16 Simonyan and Zisserman 2014


$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

138 million parameters

Architectures

Inception (Google) Szegedy, et al 2015



🟦 convolution

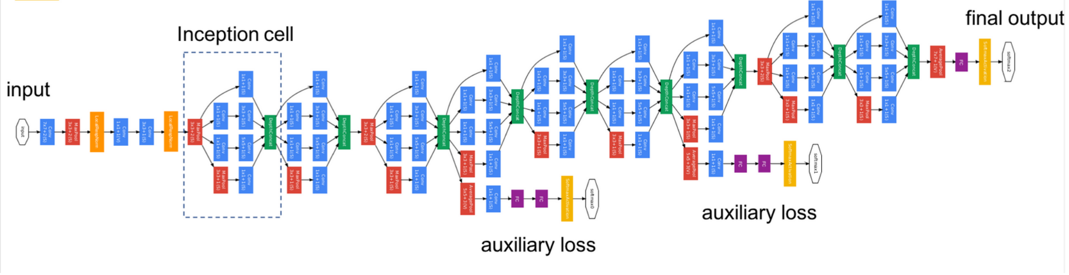
🟥 max pooling

🟩 channel concatenation

🟧 channel-wise normalization

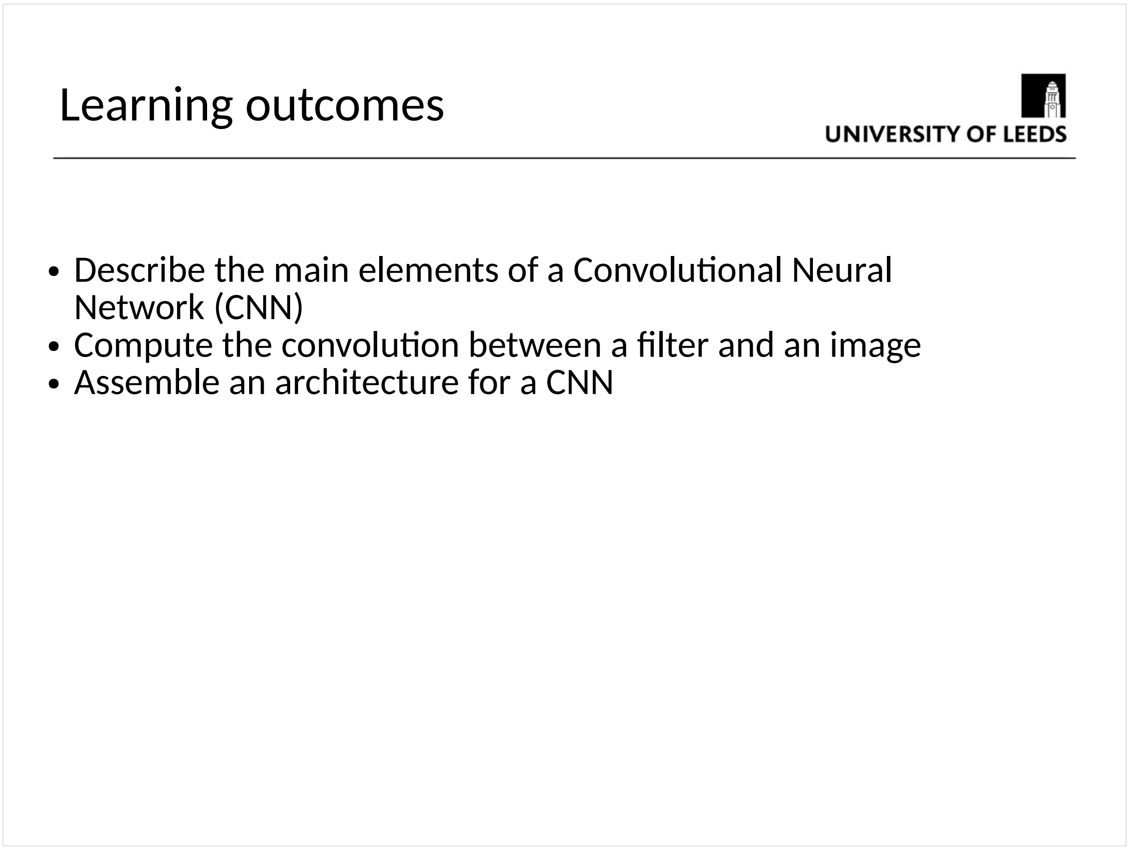
🟪 fully-connected layer

🟨 softmax



5 million parameters, then revised with 23 million parameters

Conclusion



Learning outcomes

UNIVERSITY OF LEEDS

- Describe the main elements of a Convolutional Neural Network (CNN)
- Compute the convolution between a filter and an image
- Assemble an architecture for a CNN