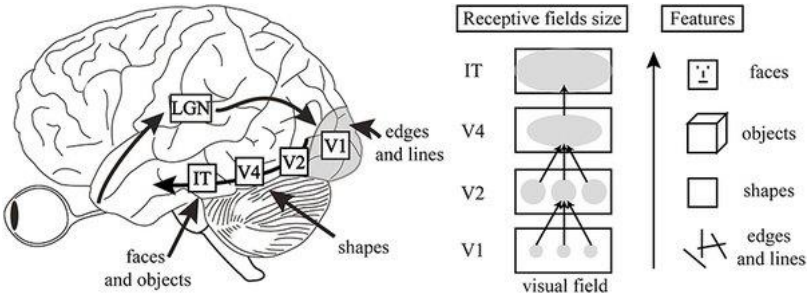
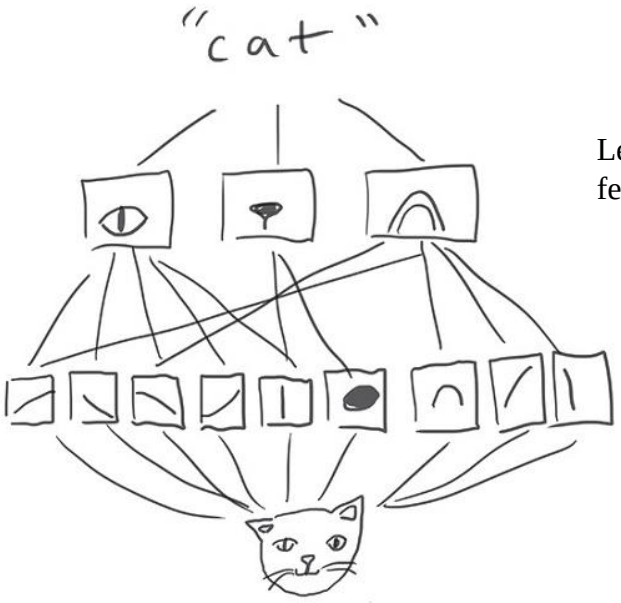
How CNNs work?

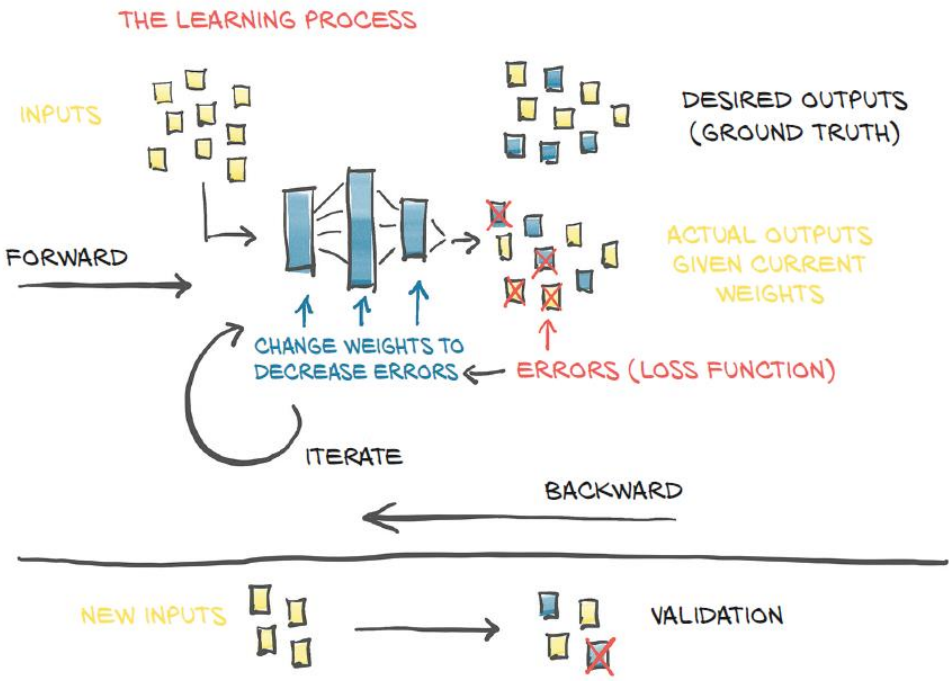


The steps of human vision and perception skills



Learn larger patterns made of the features of the first layers

Learn small local patterns such as edges in firstly layers



Convolutional Neural Networks have the following layers:

- Convolutional
- ReLU Layer
- Pooling
- Fully Connected Layer

Convolutional Layer

Convolution layers learn local patterns, i.e. in the case of images, patterns found in small 2D windows of the inputs.

Convolutional neural network (especially due to Conv. Layer) has two interesting properties:

The patterns they learn are *translation-invariant*, i.e. after learning a certain pattern in the bottom right corner of a picture, a convnet is able to recognize it anywhere, e.g. in the top left corner. This makes convnets very data-efficient when processing images: they need less training samples to learn representations that have generalization power.

They can learn *spatial hierarchies of patterns* (figure 5.2). A first convolution layer will learn small local patterns such as edges, but a second convolution layer will learn larger patterns made of the features of the first layers. And so on. This allows convnets to efficiently learn increasingly complex and abstract visual concepts

Convolutions operate over 3D tensors, called "feature maps", with two spatial axes ("height" and "width") as well as a "depth" axis (also called the "channels" axis). For a RGB image, the dimension of the "depth" axis would be 3, since the image has 3 color channels, red, green, and blue. For a black and white picture, like our MNIST digits, the depth is just 1 (levels of gray). The convolution operation extracts patches from its input feature map, and applies a same transformation to all of these patches, producing an *output feature map*. This output feature map is still a 3D tensor: it still has a width and a height. Its depth can be arbitrary, since the output depth is a parameter of the layer, and the different channels in that depth axis no longer stand for specific colors like in an RGB input, rather they stand for what we call *filters*. Filters encode specific aspects of the input data: at a high level, a single filter could be encoding the concept "presence of a face in the input", for instance.

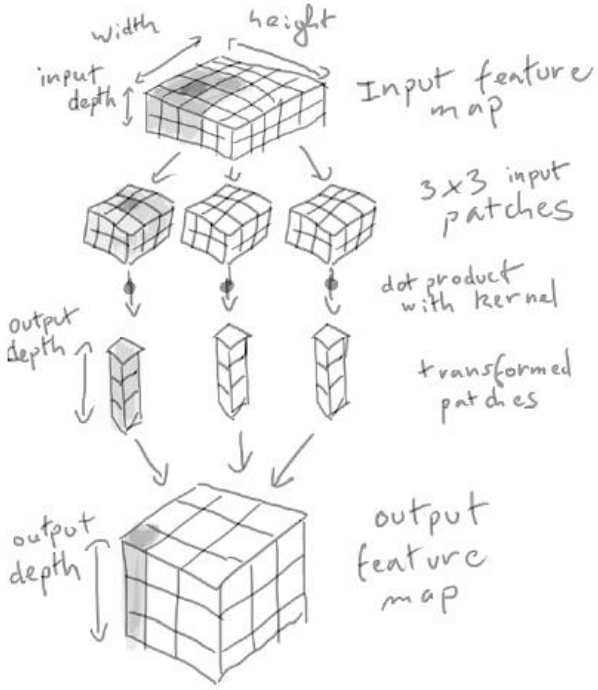
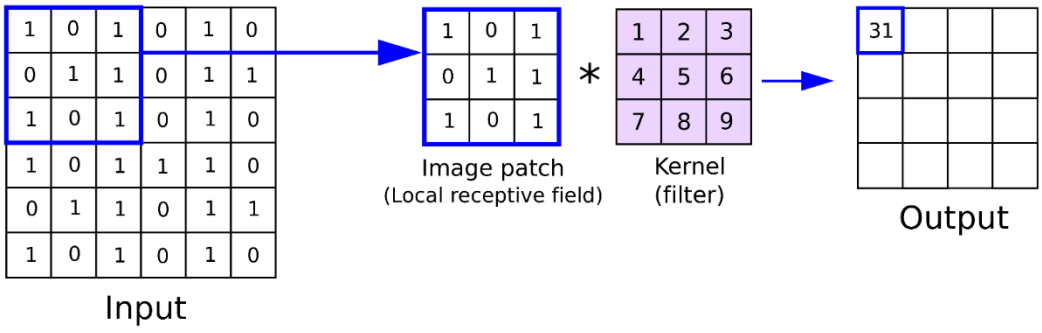
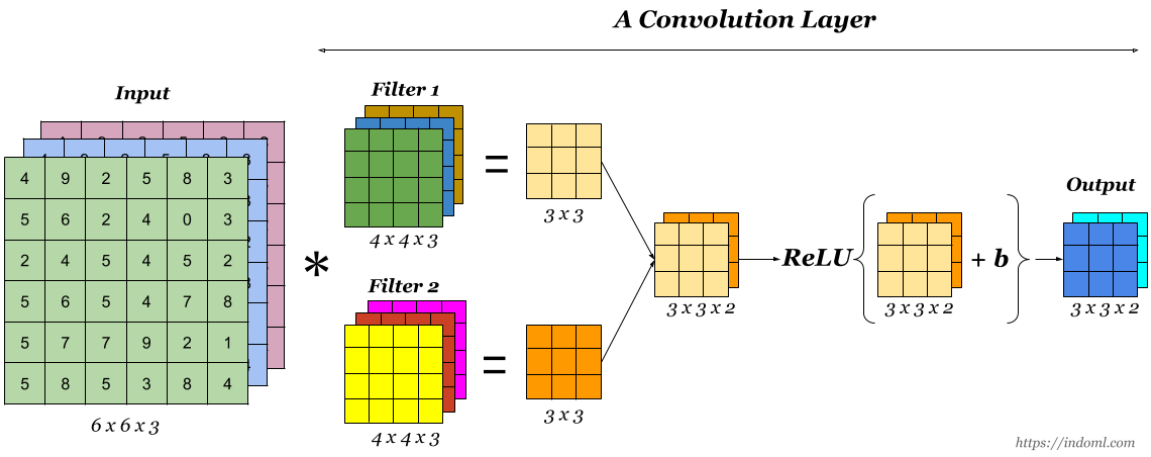


Figure 5.4 How convolution works

Note that the output width and height may differ from the input width and height. They may differ for two reasons:

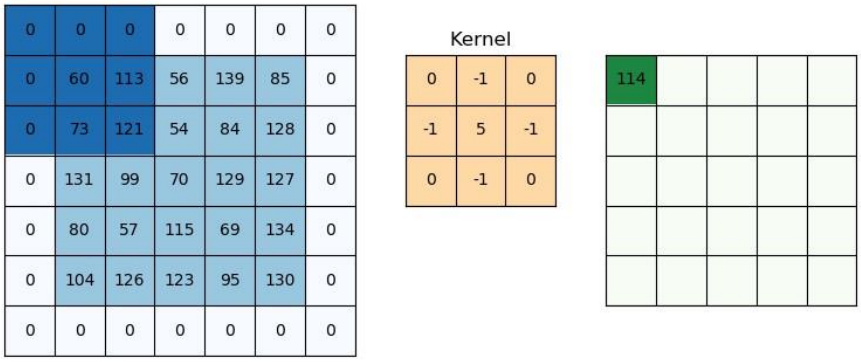
- Border effects, which can be countered by padding the input feature map.
- The use of "strides", which we will define in a second.





What is padding?

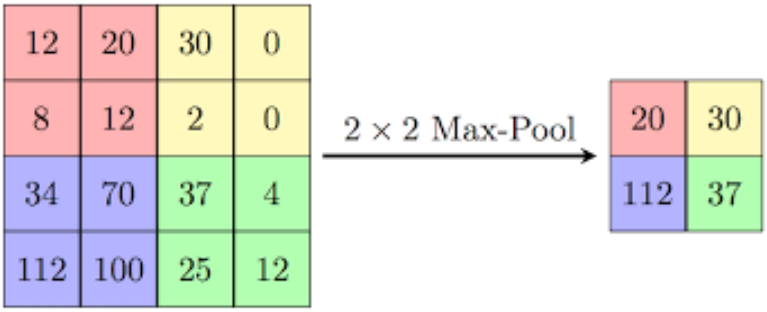
In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images.



The Max Pooling Operation

That's the role of max pooling: to aggressively downsample feature maps, much like strided convolutions.

For instance, before the first MaxPooling2D layers, the feature map is 26x26, but the max pooling operation halves it to 13x13.



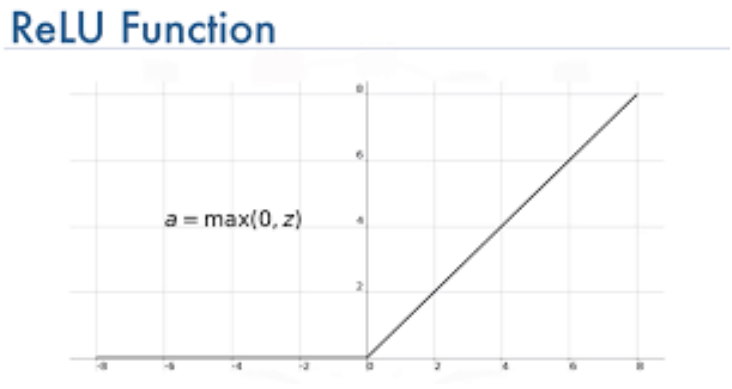
Why do we downsample feature maps in such a way? Why not remove the max pooling layers and keep fairly large feature maps all the way up? Let’s take a look at this

In short, the reason to use downsampling is simply to reduce the number of feature map coefficients to process, as well as to induce spatial filter hierarchies by making successive convolution layers look at increasingly large windows (in terms of the fraction of the original input they cover).

Note that max pooling is not the only way one can achieve such downsampling. As you already know, you could also use strides in the previous convolution layer. And you could also use average pooling instead of max pooling, where each local input patch is transformed by taking the average value of each channel over the patch, rather than the max. However, max pooling tends to work better than these alternative solutions. In a nutshell, the reason for this is that features tend to encode the spatial "presence" of some pattern or concept over the different tiles of the feature map (hence the term "feature map"), and it is more informative to look at the *maximal presence* of different features than at their *average presence*.

ReLU Layer

As a consequence, the usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network. If the CNN scales in size, the computational cost of adding extra ReLUs increases linearly.



ReLU is an activation funtion, An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.