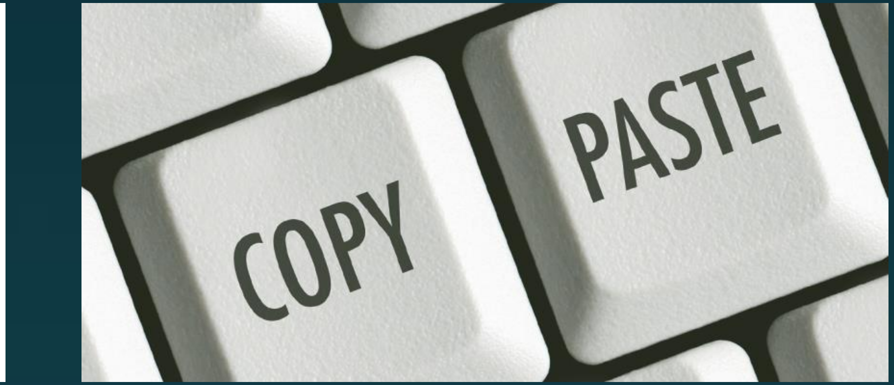
Zero-to-Hero Deep Learning in Python

Intro to Neural Nets and CNNs

25/01/2020
Samuel White
info@osgdigitallabs.com

Your Top Friends

Google



Also good places:

Medium

medium.com

towards data science

towardsdatascience.com

MACHINE LEARNING MASTERY

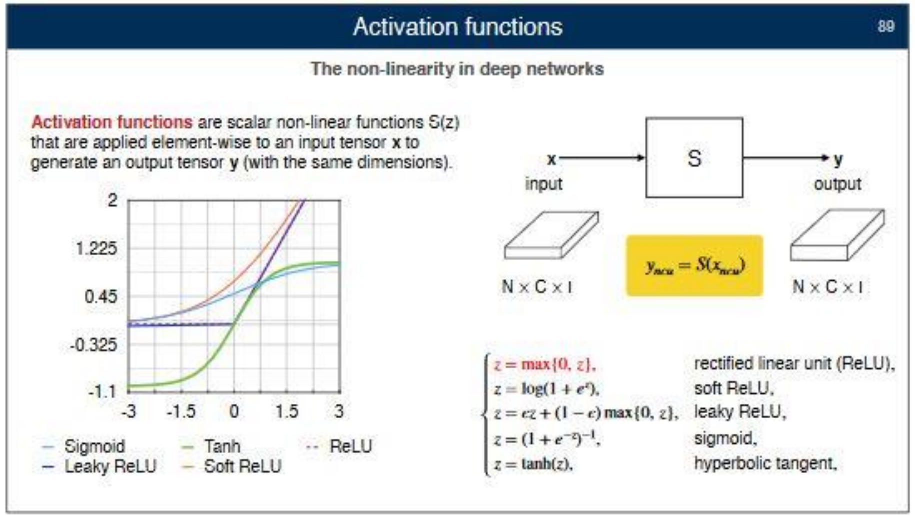
machinelearningmastery.com/

K Keras

keras.io/

stackoverflow.com/

Neural Nets – Horrendous Maths



Activation functions

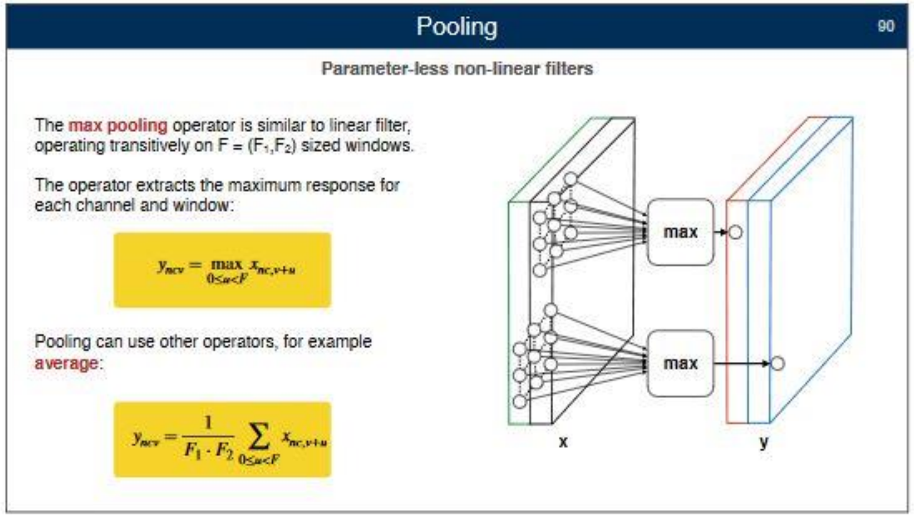
89

The non-linearity in deep networks

Activation functions are scalar non-linear functions S(z) that are applied element-wise to an input tensor **x** to generate an output tensor **y** (with the same dimensions).

$$y_{ncu} = S(x_{ncu})$$

$$\begin{cases} z = \max\{0, z\}, & \text{rectified linear unit (ReLU),} \\ z = \log(1 + e^z), & \text{soft ReLU,} \\ z = \epsilon z + (1-\epsilon)\max\{0, z\}, & \text{leaky ReLU,} \\ z = (1 + e^{-z})^{-1}, & \text{sigmoid,} \\ z = \tanh(z), & \text{hyperbolic tangent,} \end{cases}$$



Pooling

90

Parameter-less non-linear filters

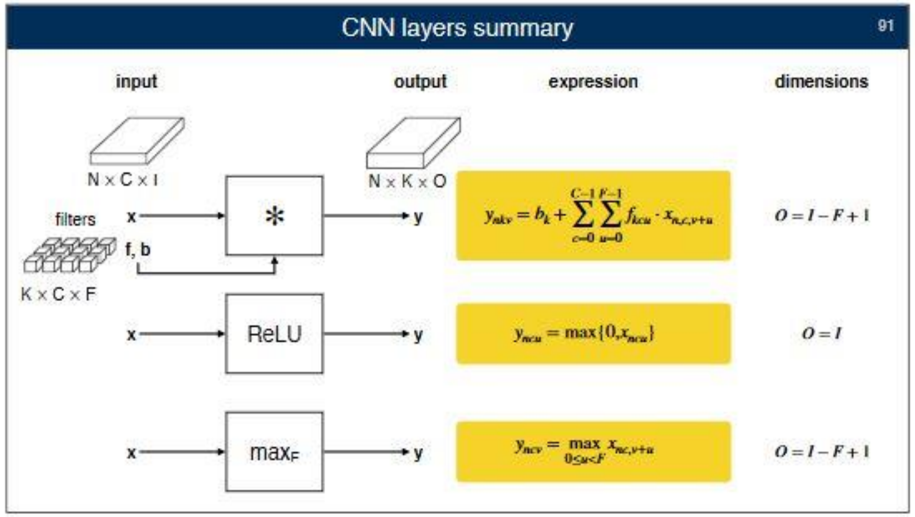
The max pooling operator is similar to linear filter, operating transitively on F = (F₁,F₂) sized windows.

The operator extracts the maximum response for each channel and window:

$$y_{ncv} = \max_{0 \le u < F} x_{nc,v+u}$$

Pooling can use other operators, for example average:

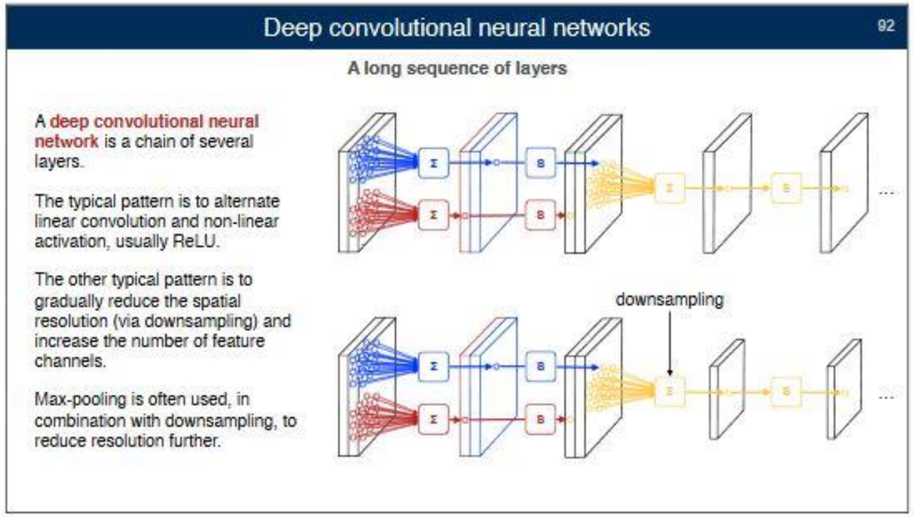
$$y_{ncv} = \frac{1}{F_1 \cdot F_2} \sum_{0 \le u < F} x_{nc,v+u}$$



CNN layers summary

91

expression	dimensions
$y_{nkv} = b_k + \sum_{c=0}^{C-1} \sum_{u=0}^{F-1} f_{kcu} \cdot x_{n,c,v+u}$	$O = I - F + 1$
$y_{ncu} = \max\{0, x_{ncu}\}$	$O = I$
$y_{ncv} = \max_{0 \le u < F} x_{nc,v+u}$	$O = I - F + 1$



Deep convolutional neural networks

92

A long sequence of layers

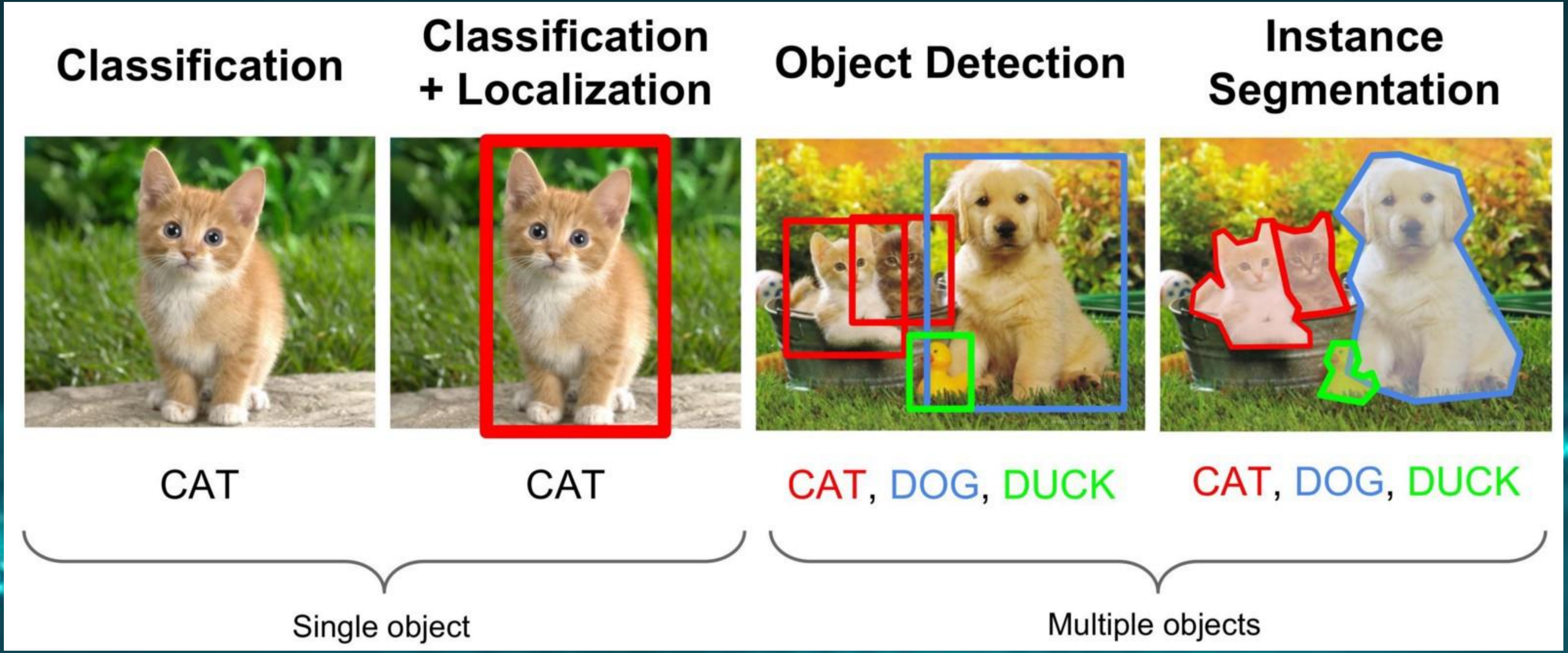
A deep convolutional neural network is a chain of several layers.

The typical pattern is to alternate linear convolution and non-linear activation, usually ReLU.

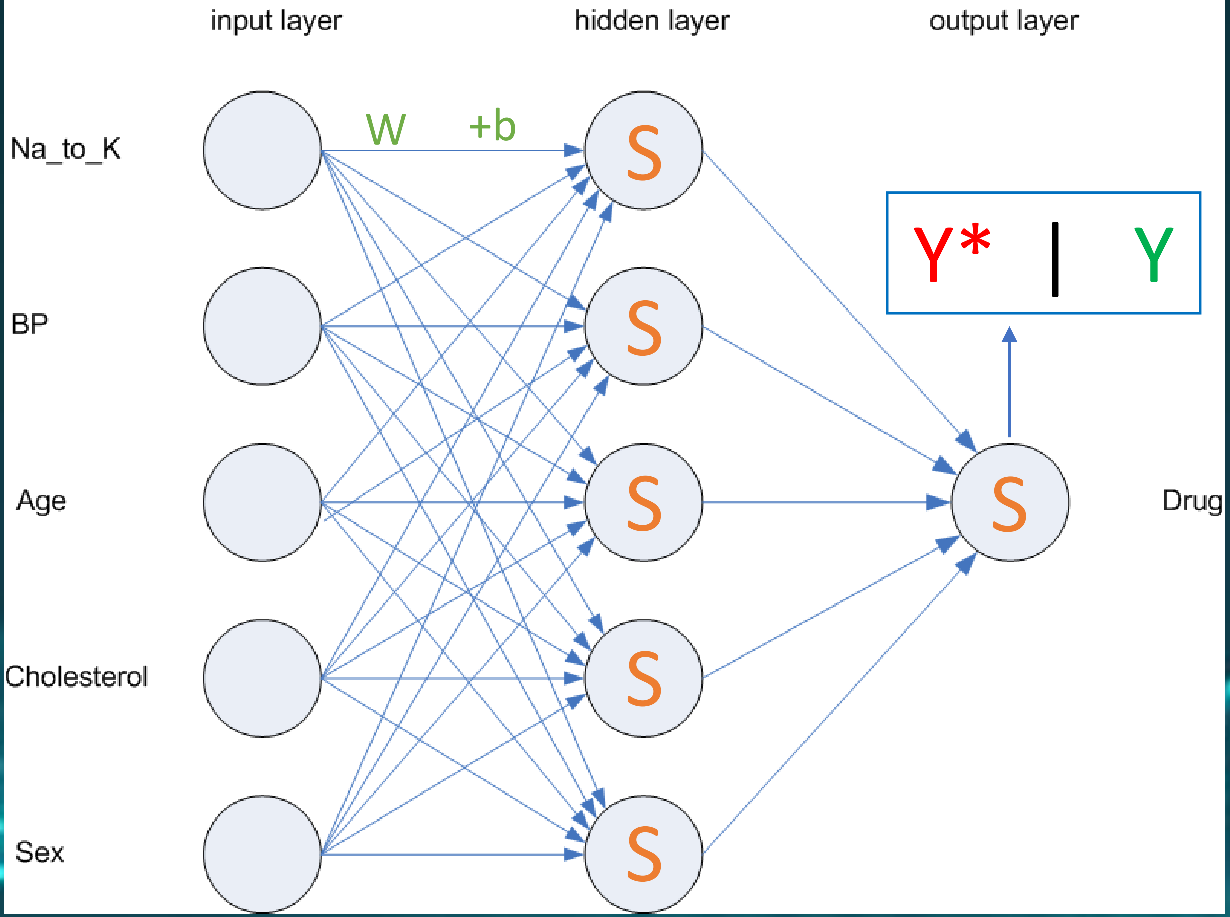
The other typical pattern is to gradually reduce the spatial resolution (via downsampling) and increase the number of feature channels.

Max-pooling is often used, in combination with downsampling, to reduce resolution further.

Neural Nets – Cool Stuff & Cute Cats



Neural Nets – The Cookbook



An NN (in simple terms) comprises of 3 things:

- *Weights*
- *Biases*
- *Activation functions*

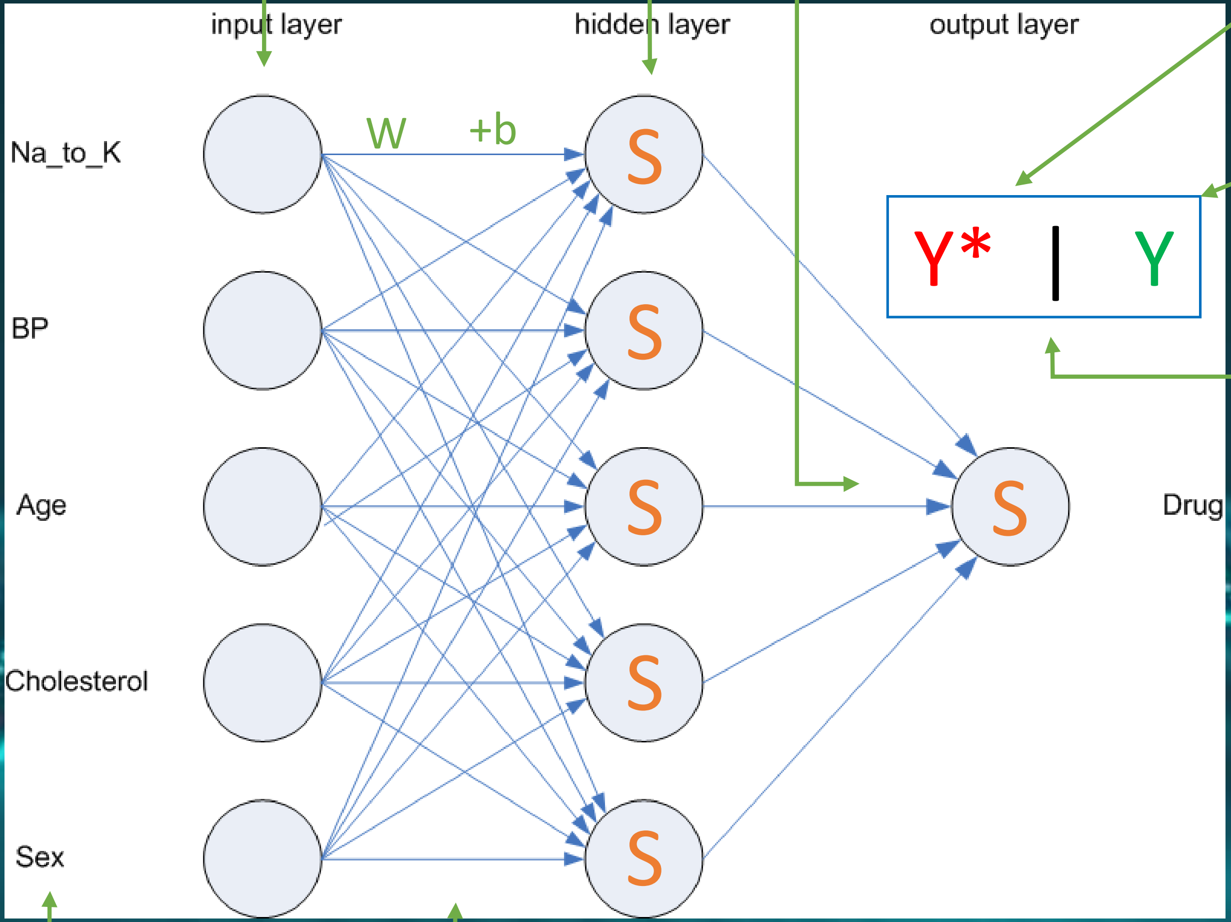
But uses / generates 5 other important things to train:

- *A set of input data*
- *An output layer or vector (Y* in the image)*
- *Some true labels for data (Y in this image)*
- *A loss function (some measure of difference between what the model thinks: Y* and what the answer should be: Y)*
- *A learning function, that updates the model weights and biases based upon the result of the loss function*

Neural Nets – The Cookbook

Layers:
The entire structure of a neural net is based around multiplying an input (say "Age") by a weight (W), adding a little bit to shift the data (+b), and then doing some transform of the data (the orange "S"s) eg. say that if the result was less than zero, then just make it zero instead.

We'll explain this more on the next slide.



Inputs:
The vector of data that you want your model to predict on

Outputs: what the model thinks the answer is

Labels: what the true answer is (in supervised learning)

Loss function: some measure of how different Y* is from Y. It factors into how much the learning function adjusts the weights in the model.

Learning function: uses the results from the loss to adjust the weights and biases that make up a model.

Neural Nets – The Formula

output = F (sum (input*weight) + bias)

This is either the final output, or is the output from one layer node into the next one

The F is the *activation function,* it simply changes the result from the brackets in some predefined way.

Relu is a function, for instance, that simply sets the result to zero if it is negative.

Lots of activation functions exist, and they help to introduce non-linearity into the model.

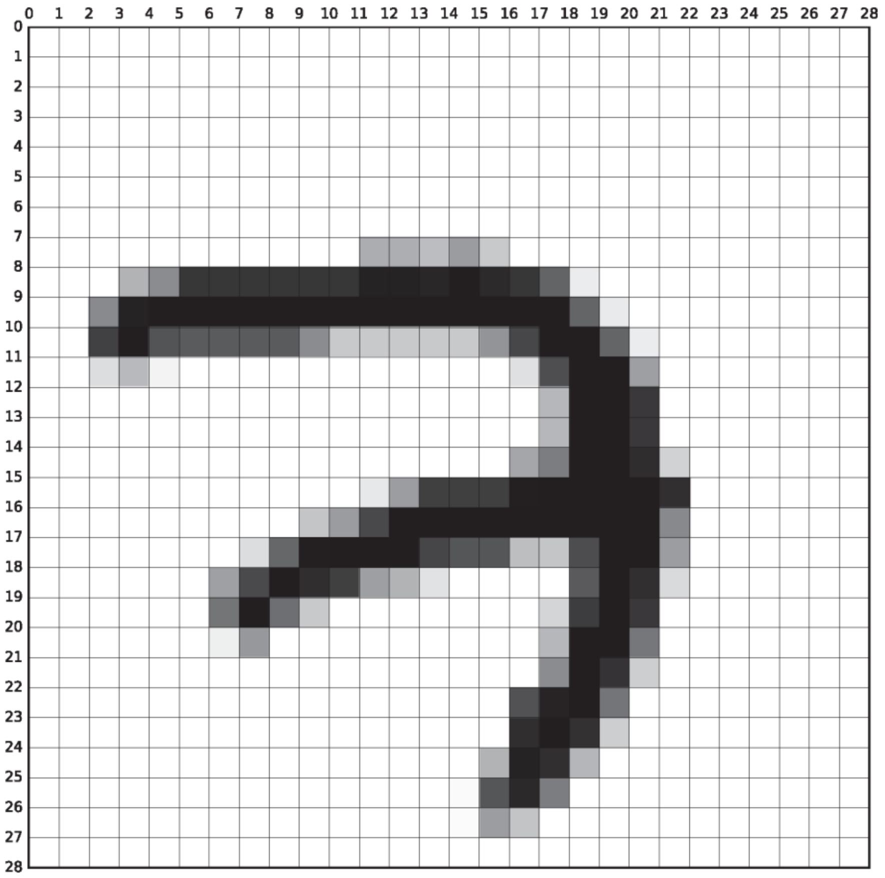
Either the original input of your data, or the output from the last layer.

For each of the nodes in the layer you're in:

- Multiply the outputs from the last layer with unique weights.
- Sum the result, and add (or subtract) a bias.

The aim of the model is to learn the weights and biases that make it minimise the the difference between its output value and the true value

Neural Nets – Not Good For Images



(**a**) MNIST sample belonging to the digit '7'.

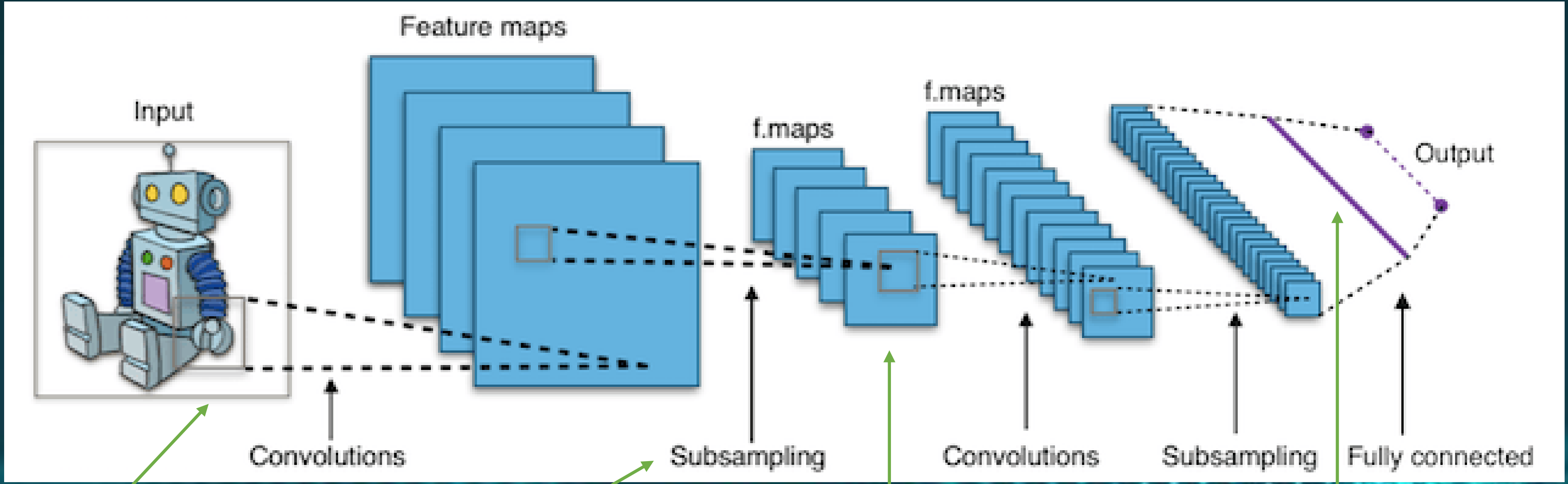
This here is an image from the MNIST hand-written digit dataset. It's a 28 by 28 pixel image in greyscale, supposedly of a 7…

The way we look at it, we can see that it has curves, lines, intersections etc. and can therefore decide what digit it is.
But in a NN we turn this image into just one big vector, so pixel 29 in this image is near pixel 28, but they're actually completely opposite sides of the image!
In real life pixel 29 is next to pixel 1, 2, 30, 57 and 58 – so we haven't preserved this ***locality*** at all.

CNNs aim to overcome this locality issue using filters…

Convolutional Neural Nets – The Difference

As the feature maps are local – work only on a small grid – then they solve the problem of treating all distances and pixels equally.



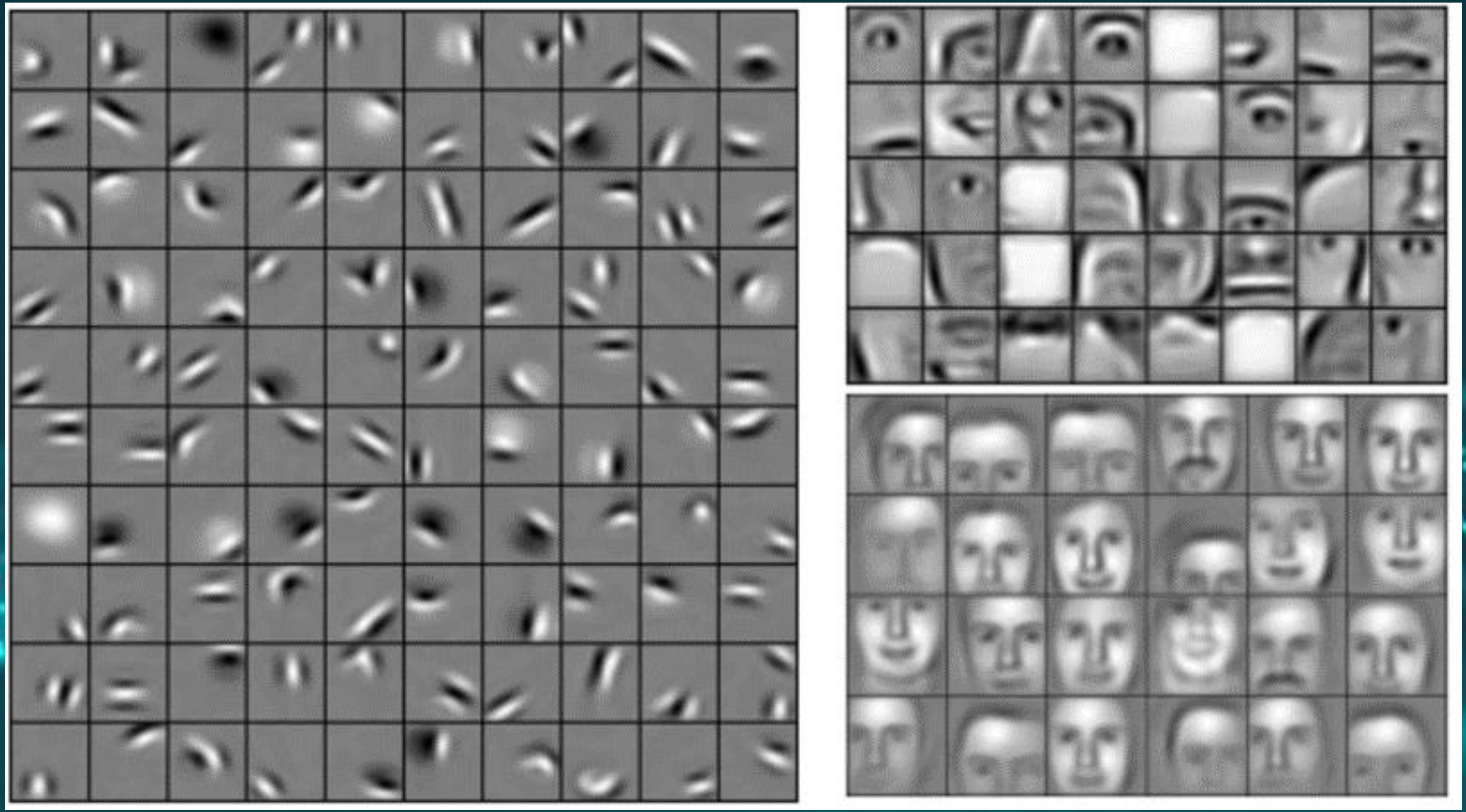
Passes a filter over the image to try and learn features, such as curves or lines

Sub-sampling /pooling reduces the dimensionality of the data, and allows for some translational invariance in the image.

Further convolutions will try and produce more high-level features (eg. the connection of curves into eyes etc.)

The final small feature maps are then compressed into a long vector (a representation of the image) which is passed into a fully connected layer. This fully connected layer does the thinking about what is important for discrimination of the different digits.

CNNs – What is it learning?



Google Colab workbook:

https://tinyurl.com/osgd-nn-cnn