Convolutional Neural Networks

 Romeo Kienzler

Let’s get started with Convolutional Neural Networks

Image Classification

Boy?

Girl?

Image Credits:
by pagarmidna
www.openclipart.org

 Romeo Kienzler

So the task we are looking at now is image classification.

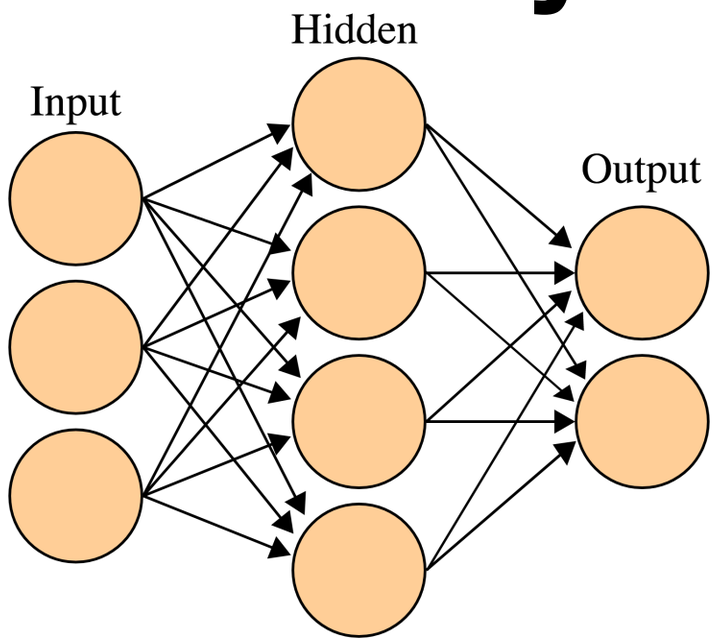
$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ \vdots \\ x_{996} \\ x_{997} \\ x_{998} \\ x_{999} \\ x_{1000} \end{bmatrix}\right) \approx c$$

So an image can be seen as a very large vector of pixel values. Let’s consider a grey scale image first for simplicity. So what we want to try to learn is a hidden function F which maps this vector to a scalar value C which stand for the class. While a scalar might still work for binary classification it is better to use a one hot encoded vector as target.

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ \vdots \\ x_{996} \\ x_{997} \\ x_{998} \\ x_{999} \\ x_{1000} \end{bmatrix}\right) \approx \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

This way we can train the neural network to learn an arbitrary number of classes.

Hidden Layers



By en:User:Cburnett [GFDL (http://www.gnu.org/copyleft/fdl.html)
or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)],
via Wikimedia Commons

So why aren’t we solving this task with an ordinary feed-forward neural network? It turns out, that with an image of only a100 by a100 pixels we would need 10000 weights in the weight matrix. So obviously those can’t be applied to larger images due to computational complexity.

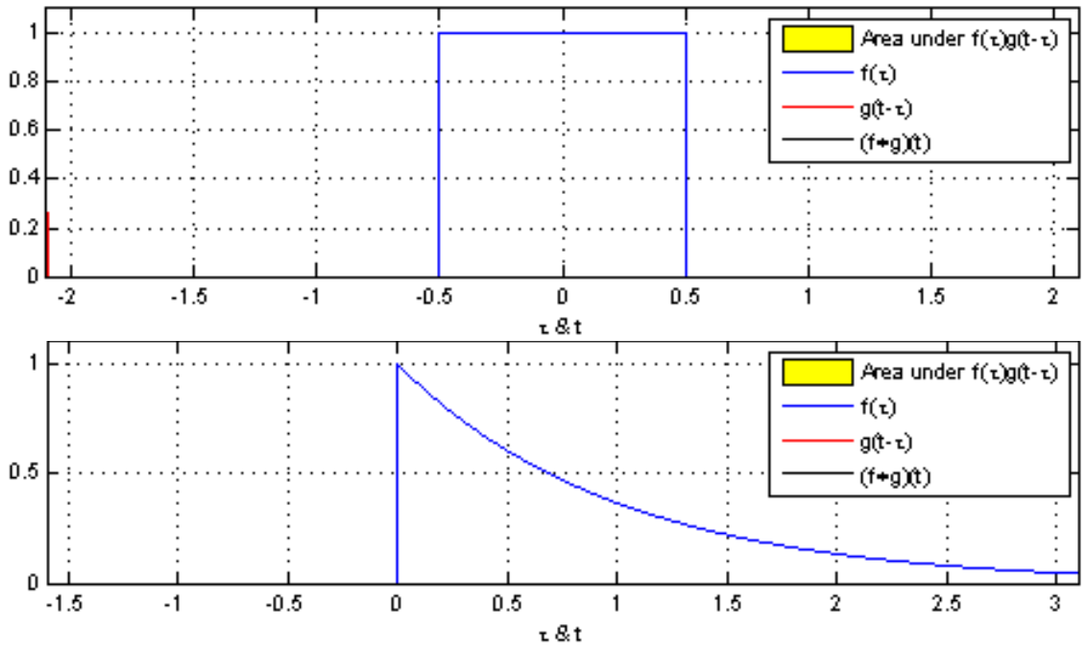


Image Credits:
https://en.wikipedia.org/wiki/Convolution#/media/File:Convolution_of_box_signal_with_itself2.gif
https://en.wikipedia.org/wiki/File:Convolution_of_box_signal_with_itself2.gif

So let’s have a look at those two diagrams. F is the original function whereas g is a function used for convolution. So convolution means sliding over a function using another function as filter. There exists different type of such filters. In this case the are under the two functions is used to create the output of the convolution. Please also note that in this case, although it looks like, its not a continuous sliding, it’s more some sort of stepping over. And the step size is called stride.

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

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

So let’s imagine we have an 5 by 5 pixel monochrome image encoded as 5 by 5 matrix.

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

1	0	1
0	1	0
1	0	1

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

And let’s apply a 3 by 3 convolution matrix using a stride of one. The convolution returns one in case both elements in the matrix are one, zero otherwise.

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

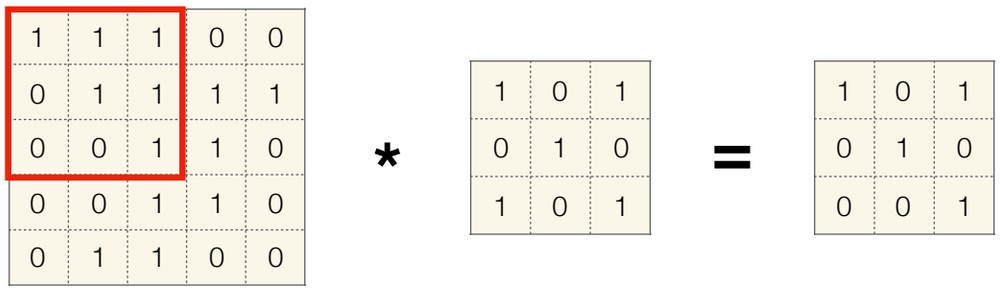
*

1	0	1
0	1	0
1	0	1

=

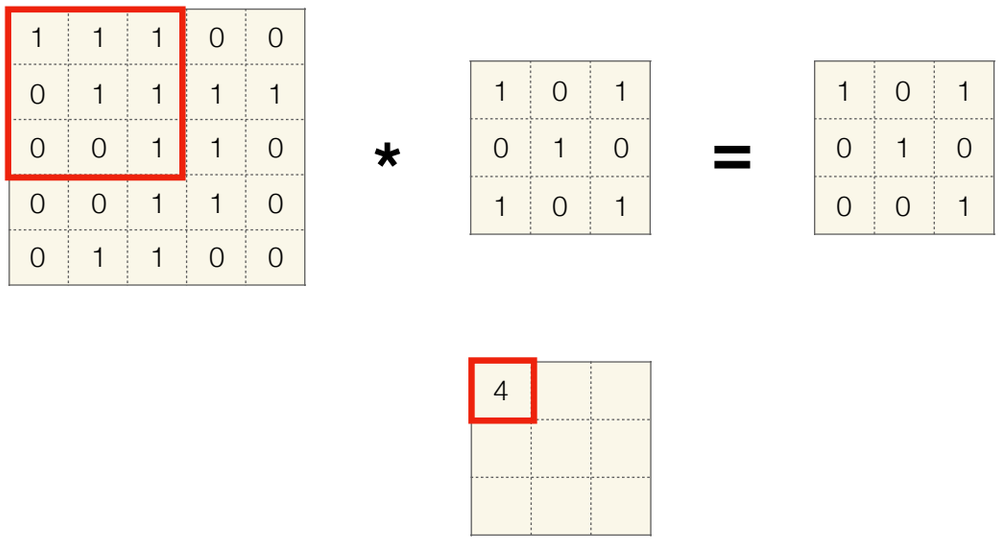
Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

We start with the first step. We convolve the top left part of the input matrix with the convolution matrix and we obtain



Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

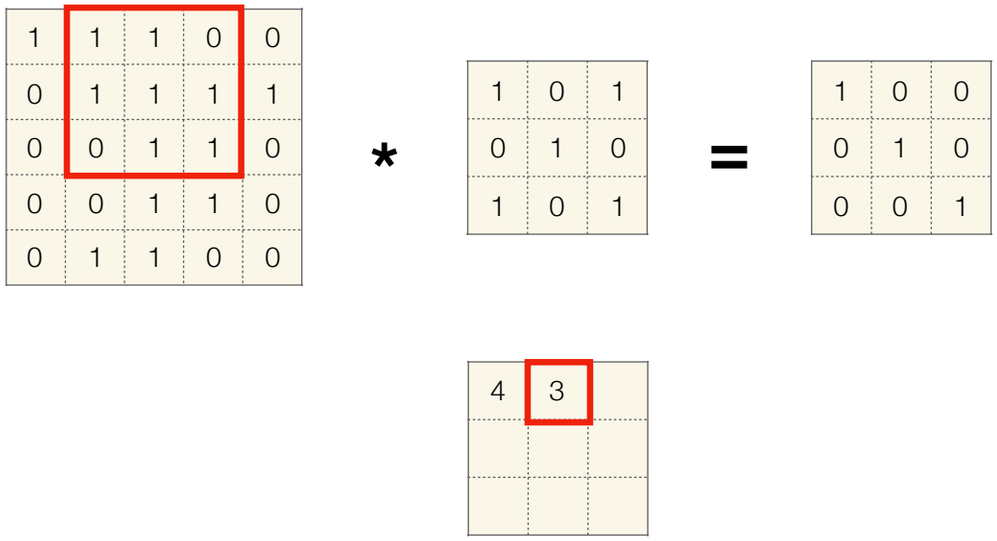
the following result. So the convolution matrix acts as filter, letting only values through at positions where itself has a one. So since the resulting matrix has four elements with one we count the number of ones and put this as a result to the final matrix



Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

 Romeo Kienzler

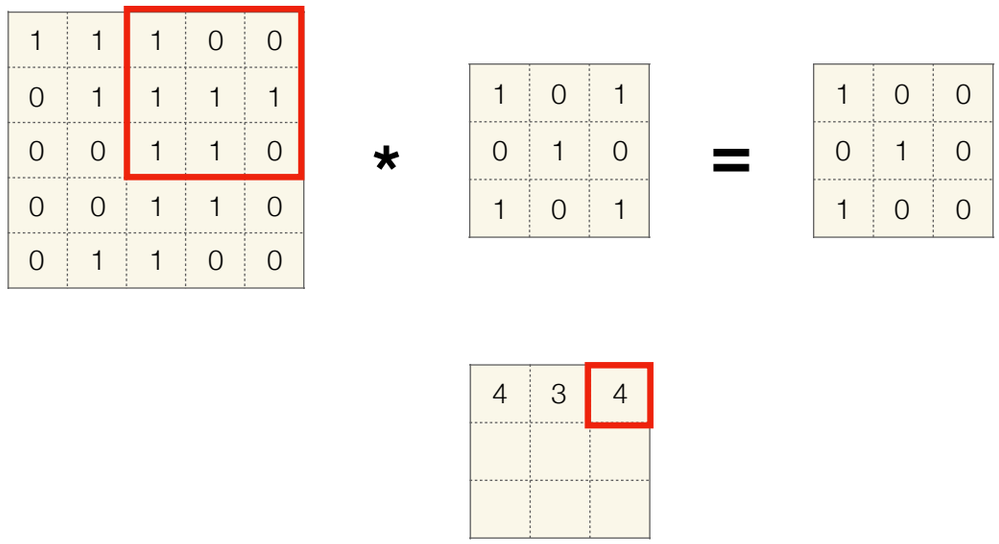
NO TEXT



Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

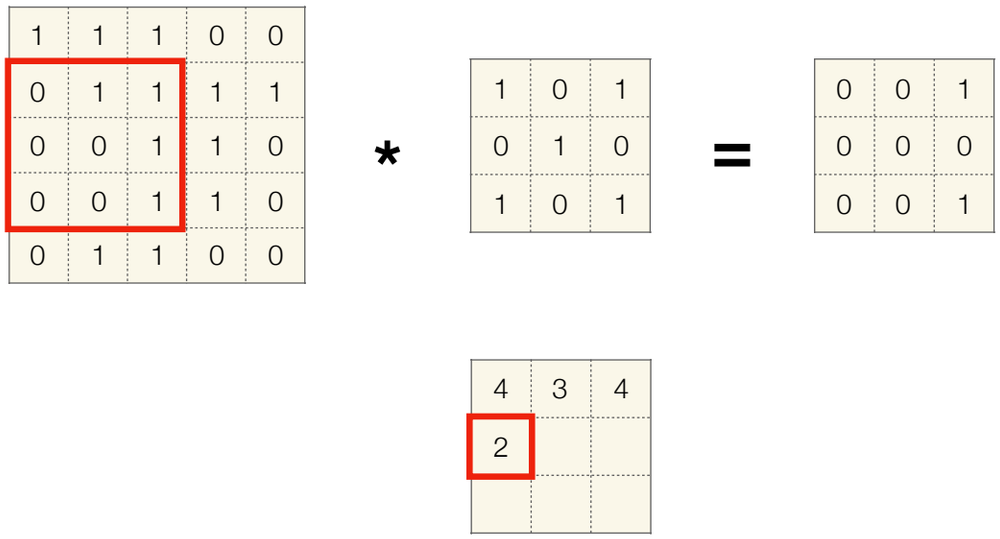
 Romeo Kienzler

Then we do it for the next step. Here we end up with three ones and six zeros.



Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Then we complete the first row



Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Now we start with the second row. Here we obtain two.

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

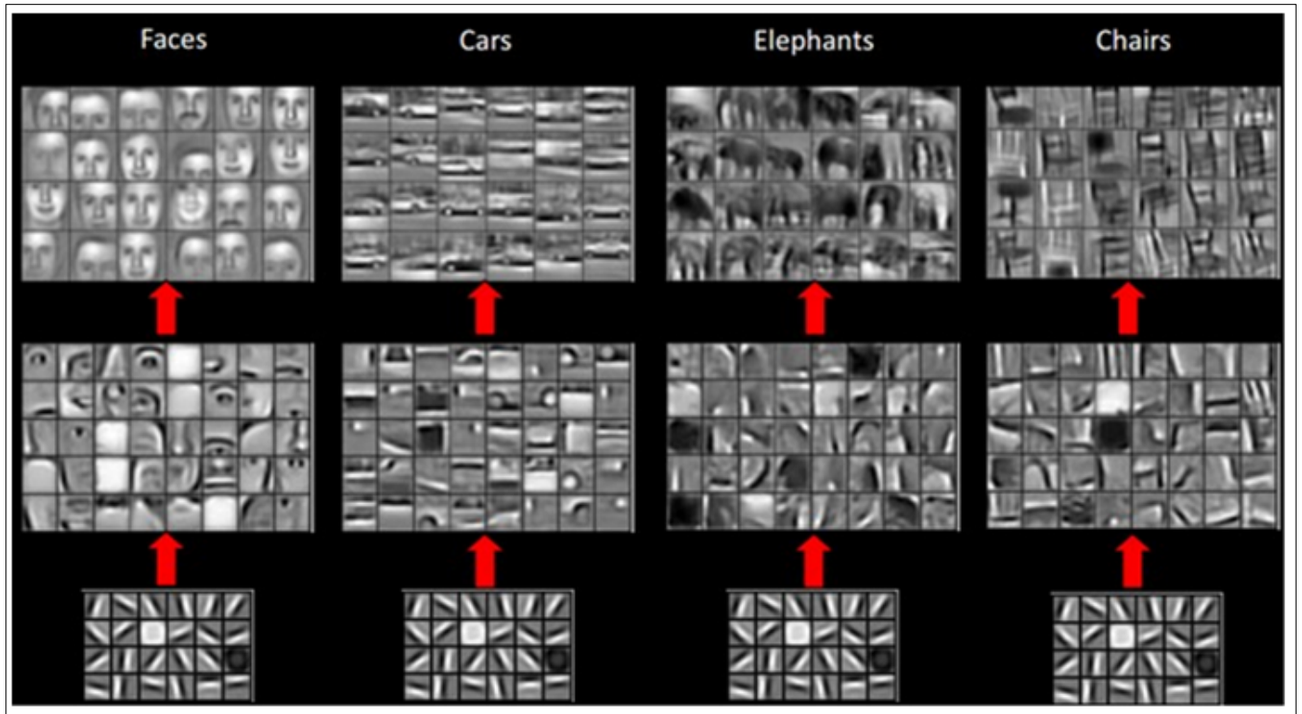
*

1	0	1
0	1	0
1	0	1

4	3	4
2	4	3
2	3	4

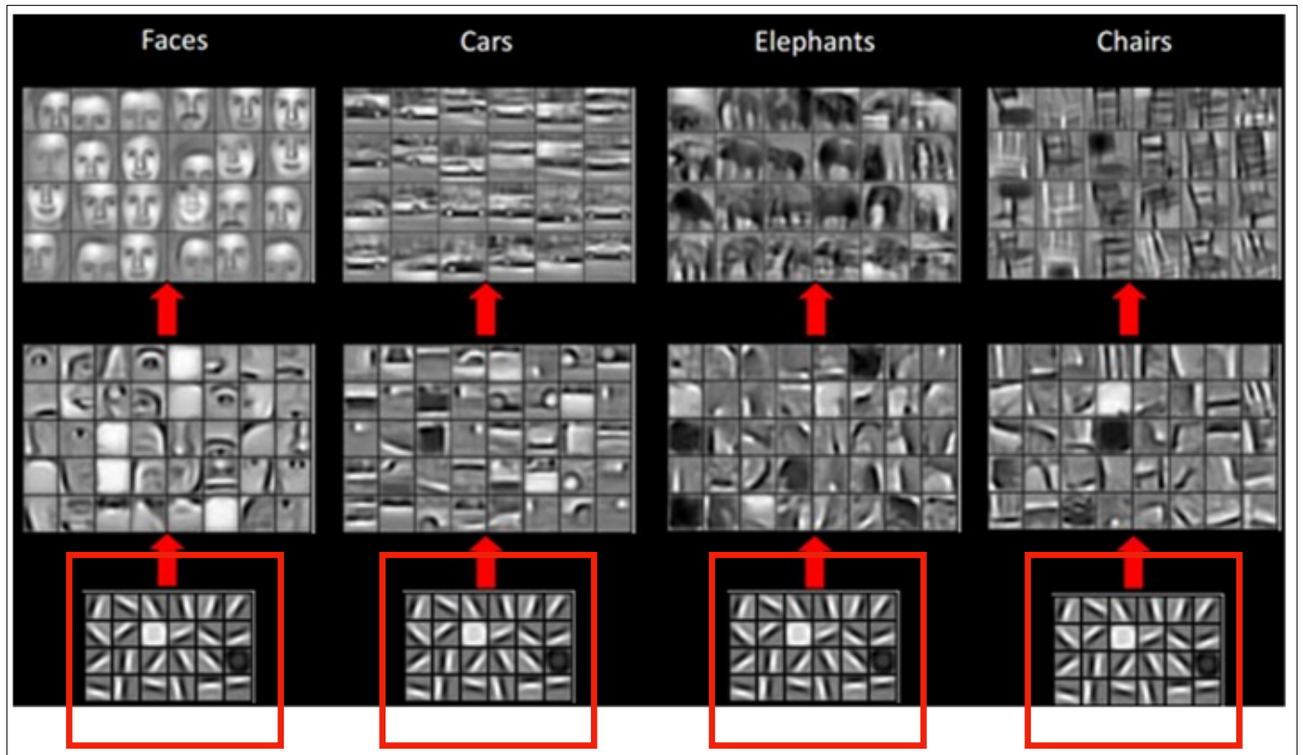
Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

I think you got the pattern, right? :) So this convolution matrix is also called a filter and the values in it are learned from data during neural network training. Note that the filter of course doesn’t contain only zeros and ones as used in this toy example. So we are learning efficient filters.



Source
https://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning/146476

This four way image classification perfectly illustrates this. In the training data set there have been images of faces, cars, elephants and chairs. After training the input layer, some intermediate layer and the output layer have been sampled. Then, for each class twenty four images have been shown to the neural network and the activations have been visualised.



Source
https://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning/146476
Lee et al., Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

This four way image classification perfectly illustrates this. In the training data set there have been images of faces, cars, elephants and chairs. After training the neural network with just one particular class of images, the first layer, the second layer and the third layer have been sampled. Then, for each class twenty four images have been shown to the neural network and the activations have been visualised. So what's important to notice is that the first layer didn't learn anything meaningful. At least nothing meaningful to us. It just learned basic shapes like lines, circles and rectangles. Another interesting thing is that independently of the type of image all those filters look that same. So this is exactly the same what the computer vision guys are doing manually - so we make them redundant. And by the way, the visual cortex in our brain does it the same way, did you know that?

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Another import of layer used in convolutional neural networks for image recognition are pooling layers. The idea is to reduce dimensionality of the intermediate layers without loosing too much spatial information. The process is also called subsampling or downsampling).

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

As in a convolutional layer we are looking at subsets of the original matrix. And from those values we just compute a single one.

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

So if we just take the maximum value - this is called max pooling, we end up with six. There exists other pooling methods like average or sum and so on.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Example inspired by:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Again, I think you got the pattern, right? :)

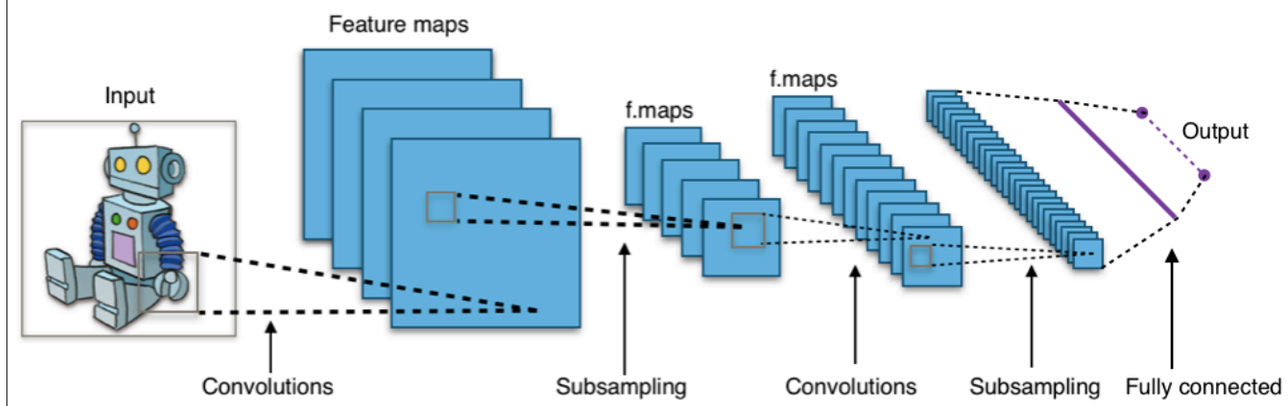


Image Credit:
by Aphex34
Creative Commons Attribution-Share Alike 4.0 International
https://commons.wikimedia.org/wiki/File:Typical_cnn.png

So this brings us to the topology of a Deep Convolutional Network used for image classification. Note that Fully Connected layer is nothing else than an ordinary feed forward neural network layer. So this is just a toy example, Convolutional Neural Networks usually have a large number of Convolutional and Pooling Layer pairs. We will see a practical example later in the course, so we'll skip this for now.

Recurrent Neural Networks

 Romeo Kienzler

Let’s get started with Recurrent Neural Networks.