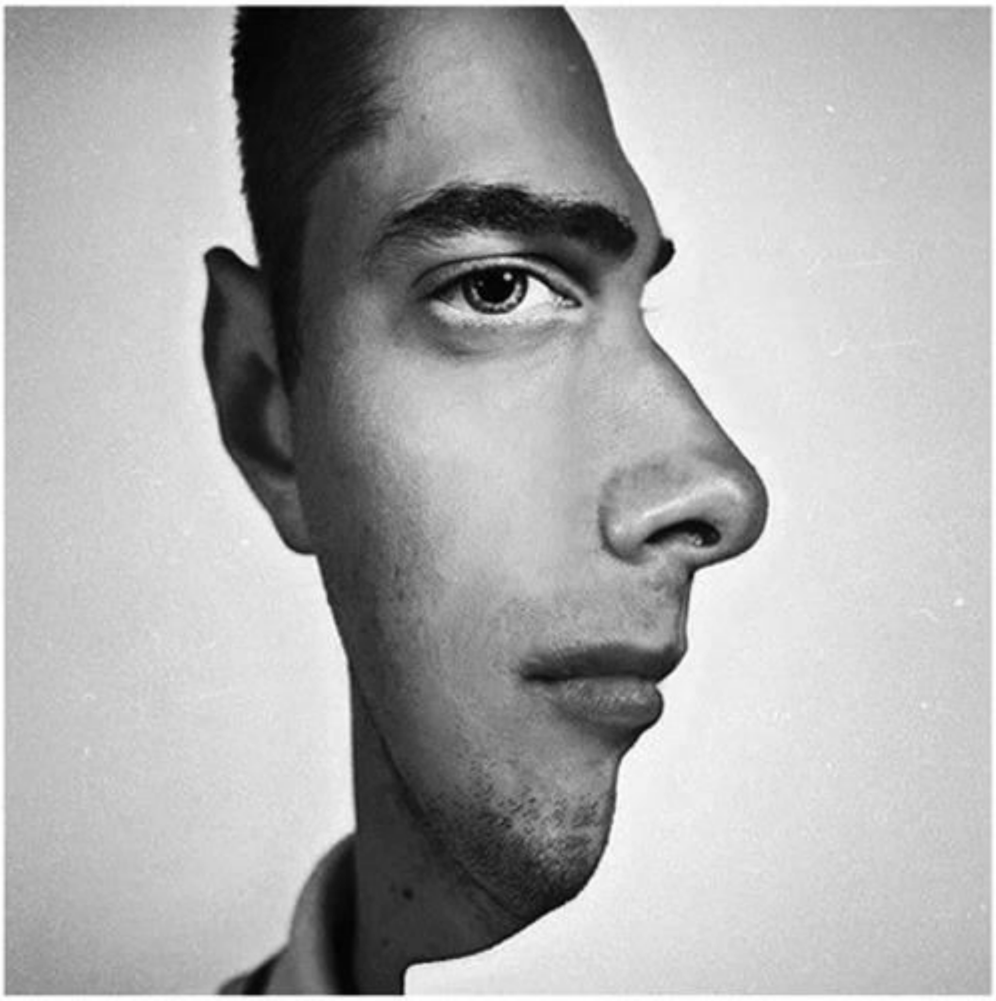
Convolutional Neural Networks

What we will learn in this section:

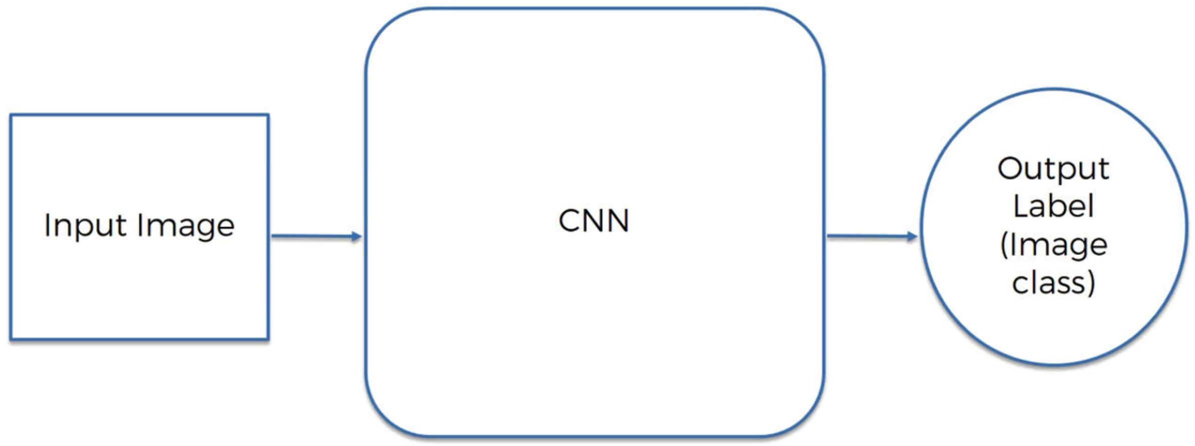
- What are Convolutional Neural Networks?
- Step 1 - Convolution Operation
- Step 1(b) - ReLU Layer
- Step 2 - Pooling
- Step 3 - Flattening
- Step 4 - Full Connection
- Summary

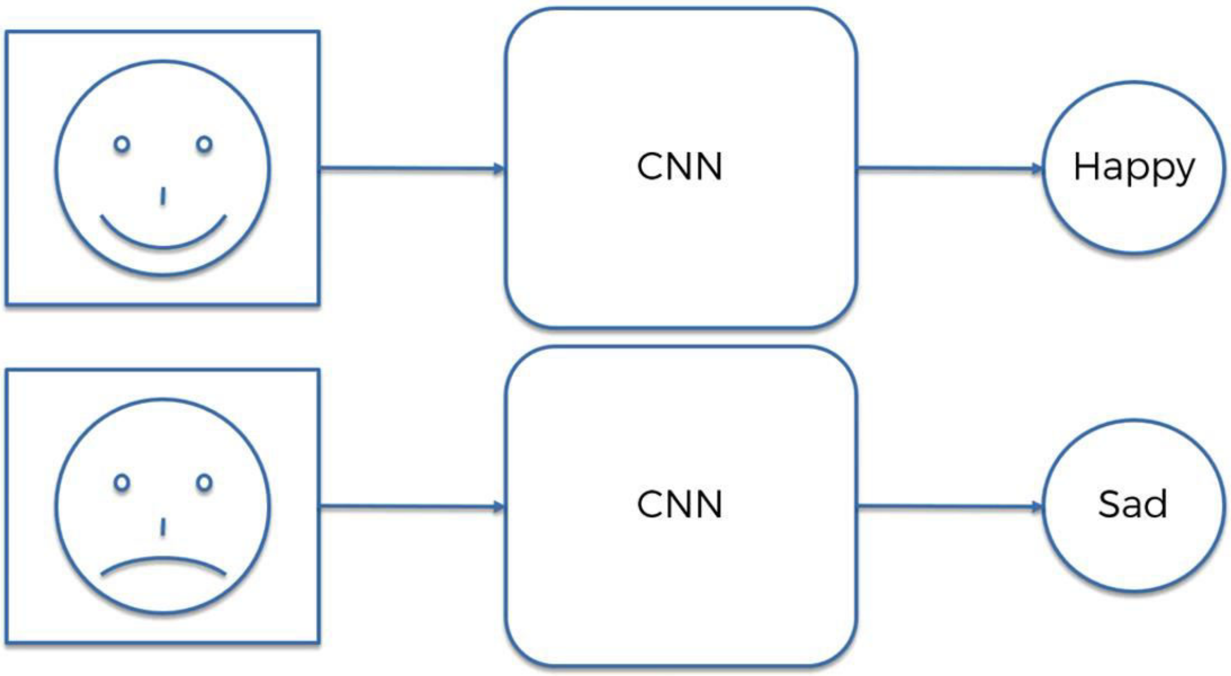
- EXTRA: Softmax & Cross-Entropy

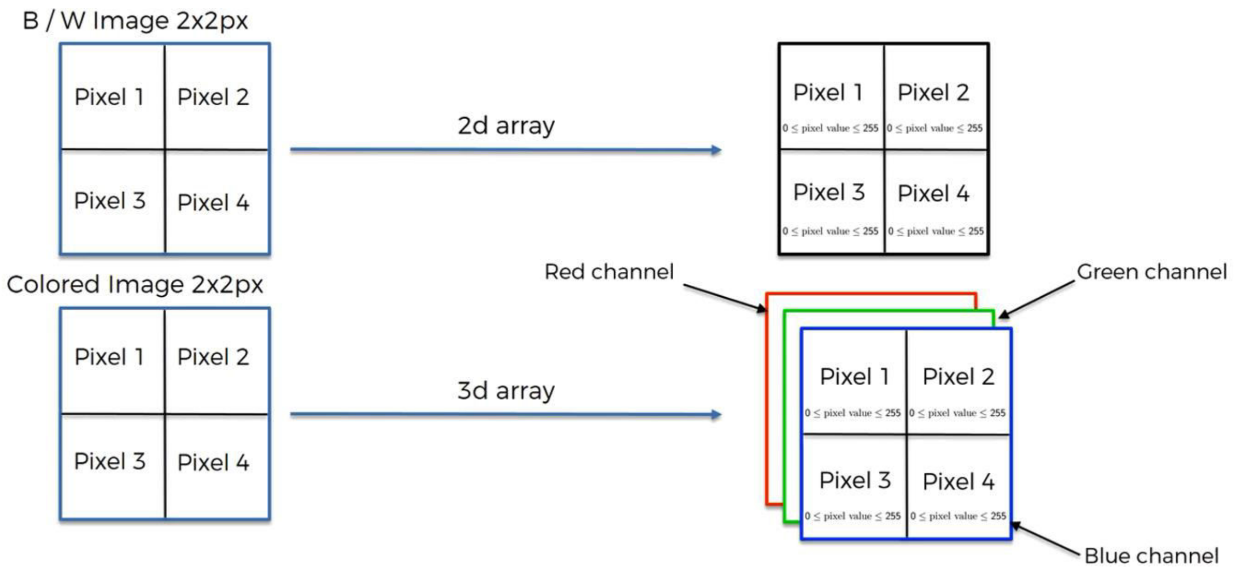


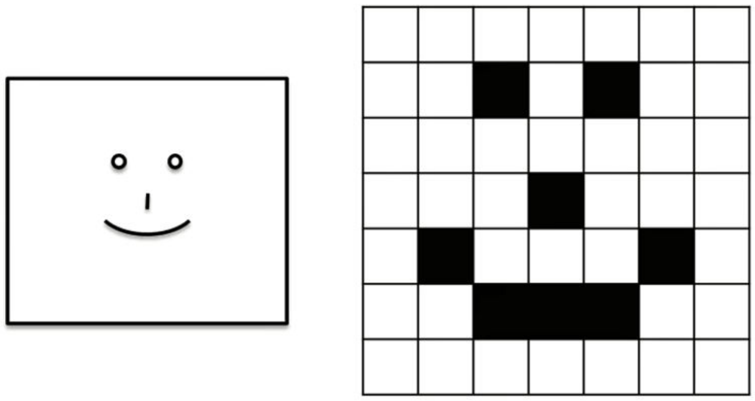
What we see when we see things FEATURES.
and on the basis of features our brain deciding it what it is.

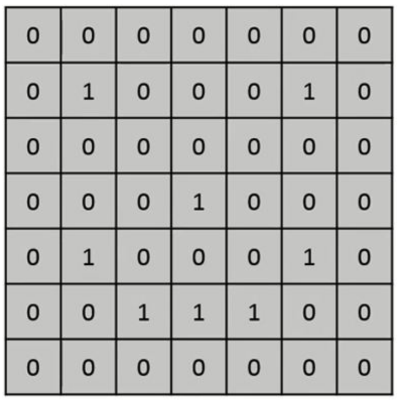
Google
Facebook


Input Image
CNN
Output Label (Image class)




B / W Image 2x2px
Pixel 1
Pixel 2
Pixel 3
Pixel 4
2d array
Pixel 1 $0 \leq$ pixel value ≤ 255
Pixel 2 $0 \leq$ pixel value ≤ 255
Pixel 3 $0 \leq$ pixel value ≤ 255
Pixel 4 $0 \leq$ pixel value ≤ 255
Colored Image 2x2px
Pixel 1
Pixel 2
Pixel 3
Pixel 4
3d array
Red channel
Green channel
Blue channel
Pixel 1 $0 \leq$ pixel value ≤ 255
Pixel 2 $0 \leq$ pixel value ≤ 255
Pixel 3 $0 \leq$ pixel value ≤ 255
Pixel 4 $0 \leq$ pixel value ≤ 255





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

STEP 1: Convolution

↓

STEP 2: Max Pooling

↓

STEP 3: Flattening

↓

STEP 4: Full Connection

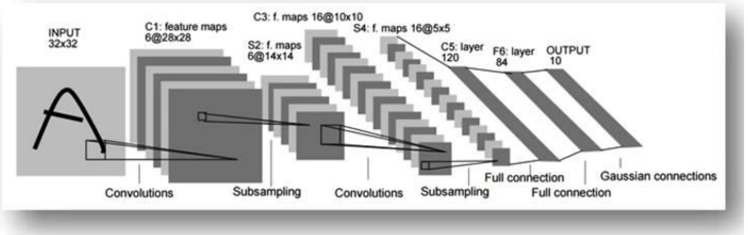
Additional Reading:

Gradient-Based Learning Applied to Document Recognition

By Yann LeCun et al. (1998)

Link:

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf



Step 1 - Convolution

A convolution is combined integration of two function and it shows you how one function modifies the shape of another.

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Additional Reading:

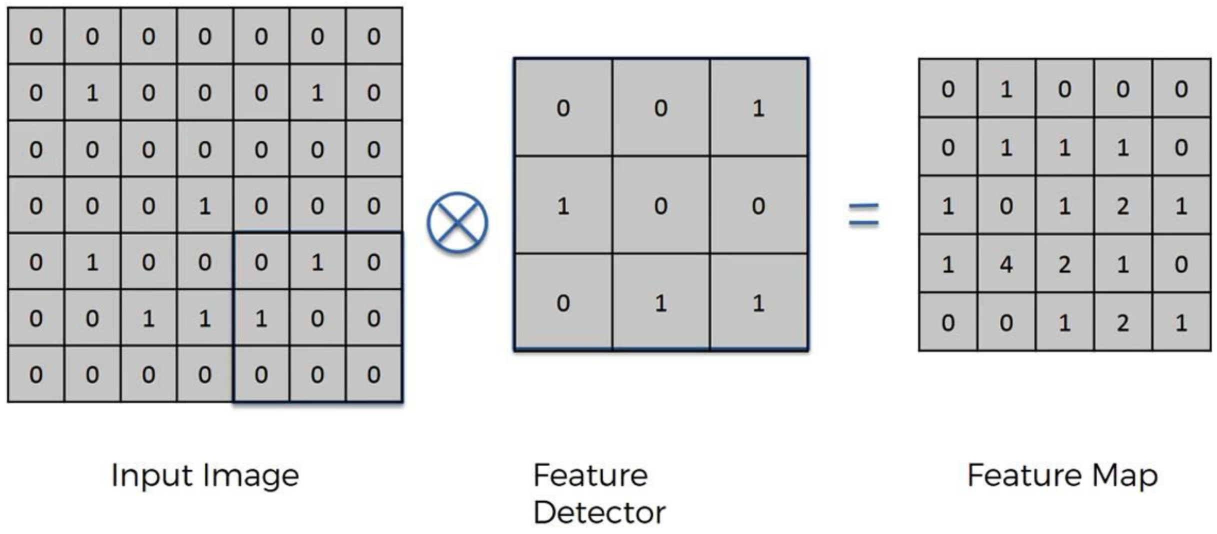
Introduction to Convolutional Neural Networks

By Jianxin Wu (2017)

$$\begin{aligned}\frac{\partial z}{\partial(\operatorname{vec}(\boldsymbol{y})^T)}(F^T \otimes I) &= \left((F \otimes I)\frac{\partial z}{\partial \operatorname{vec}(\boldsymbol{y})}\right)^T \\ &= \left((F \otimes I)\operatorname{vec}\left(\frac{\partial z}{\partial Y}\right)\right)^T \\ &= \operatorname{vec}\left(I\frac{\partial z}{\partial Y}F^T\right)^T \\ &= \operatorname{vec}\left(\frac{\partial z}{\partial Y}F^T\right)^T,\end{aligned}$$

Link:

http://cs.nju.edu.cn/wujx/paper/CNN.pdf



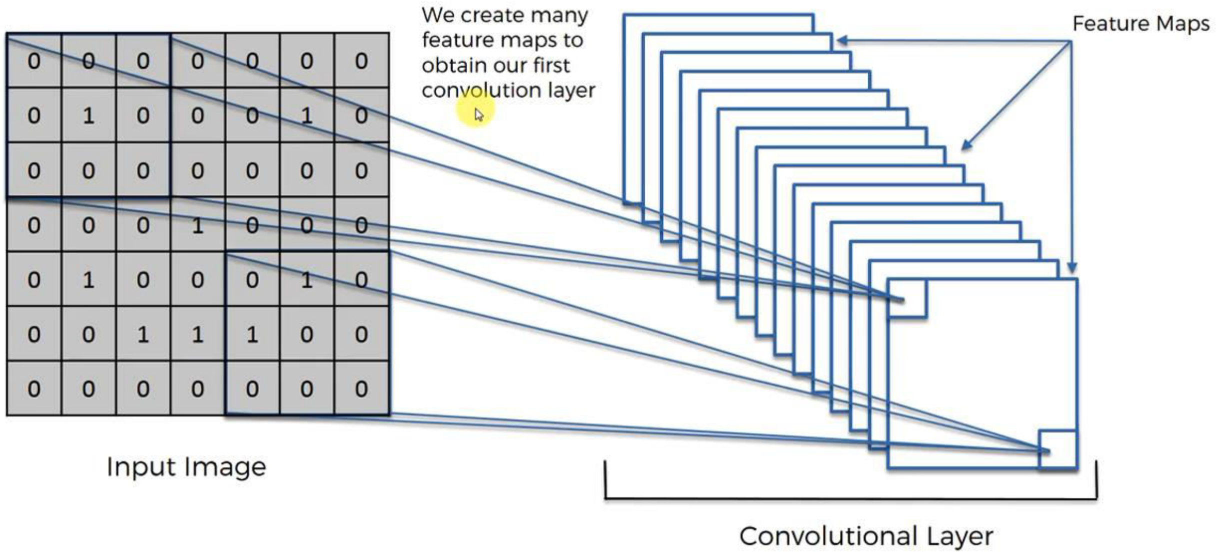
Here you can see feature detector. Feature detector is a 3x3 matrix. Does it have to be 3x3 no it doesn't. AlexNet uses 7x7 one more Imagenet uses 5x5 feature detector.
Feature detector also called as Kernel and Filter.
Steps which we are moving our filter is called strides. Here strides is 1 you can use it as per your need conventionally people uses 2 for that

Feature Map can also be called as convoled map or activation map.

If you increase your stride size your feature will decrease as per the stride no.

Why we need to reduce the size of the image. why we have to reduce the size of the matrix, answer is simple we are reducing it so that it will be easy to process and it will be faster.
So the question here is are we loosing information when we applying feature detector. Yes some information we are loosing of course because we have less value inside our resulting matrix but at the same time purpose of the feature detector is to detect certain features certain parts of the image that are integral. If you think it in this way the feature detector has the certain pattern on it. The highest no in your feature map is when that pattern matches up. Infact, the highest no you get when feature matches exactly.

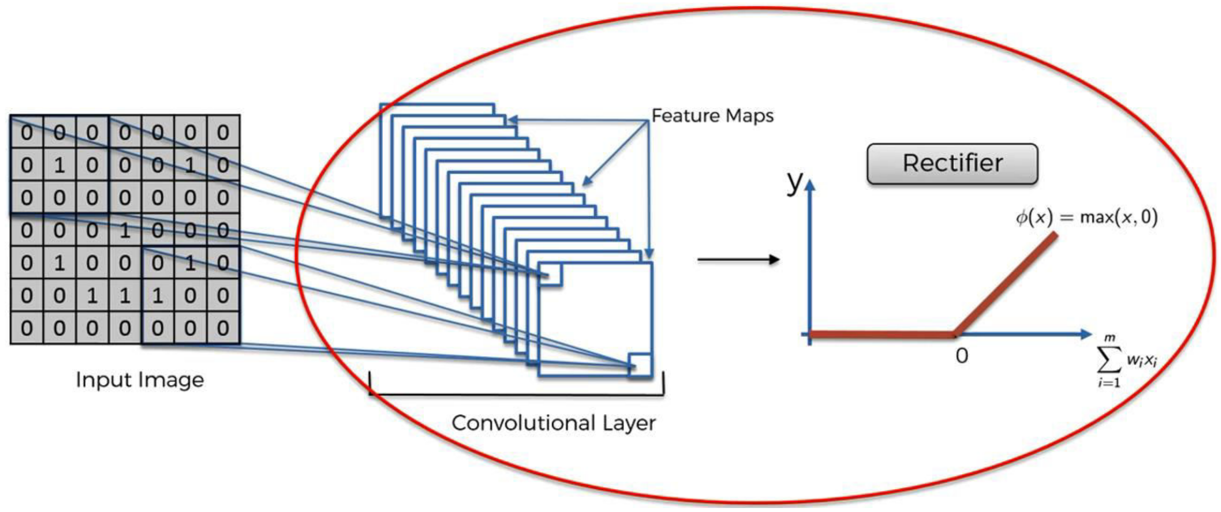
And as we discuss in the beginning itself, it's the feature how we see things, how we identify things.


We create many feature maps to obtain our first convolution layer
Feature Maps
Input Image
Convolutional Layer

This is our input image, we create a feature map, let's say the front one we just create, but we create multiple feature maps
because we use different filters. And that's another way we preserve lots of the information. We Basically the network decides through it's training. Through it's training network decides which features are important for certain types or certain categories and it looks for them and therefore we used to have multiple different filters.

Step 1(B) – ReLU Layer

ReLU = Rectified Linear Unit



Here we have our Input image , we have our convolution layer and on top of that we are going to apply our filter.
The reason why we are applying rectifiers because we want to increase non linearity in our image/input/connvolution neural network. And Rectifier acts as filter which breaks up the linearity. The reason why we want to increase non linearity in our CNN network is because images itself are highly non linear, if pick any of the image and try to check the linearity or i can say similarity between the features , pixels you won't get much linearity there that's why we try to achieve the non linearity in the CNN layer as well. When we are running the feature detector or convolution we take risk that we might create some thing linear in between the different layers of convolution layer and therefore we need to break up the linearity.

Let's Have a example

Here we have one image the real image. When we apply feature detector to this image, we get something like this.

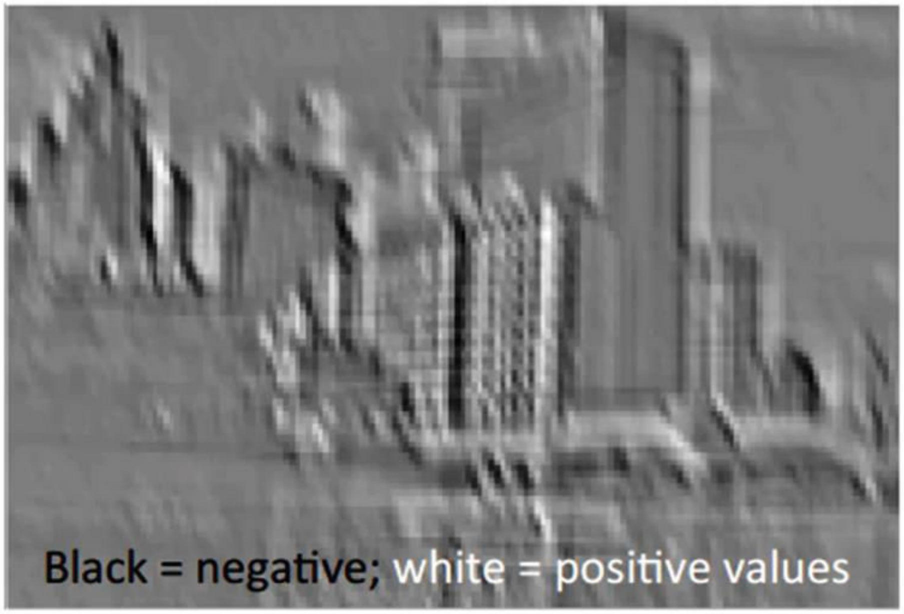


Image Source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Here if we apply ReLU on above image what it will do, it will try to remove all the negatives, so at the end you will only have non negatives values.

It's pretty hard to see what is the benefit of breaking up the linearity. It's heavily a mathematical concept.

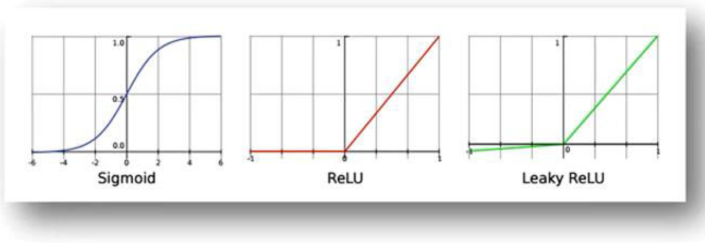
Additional Reading:

Understanding Convolutional Neural Networks with A Mathematical Model

By C.-C. Jay Kuo (2016)

Link:

https://arxiv.org/pdf/1609.04112.pdf



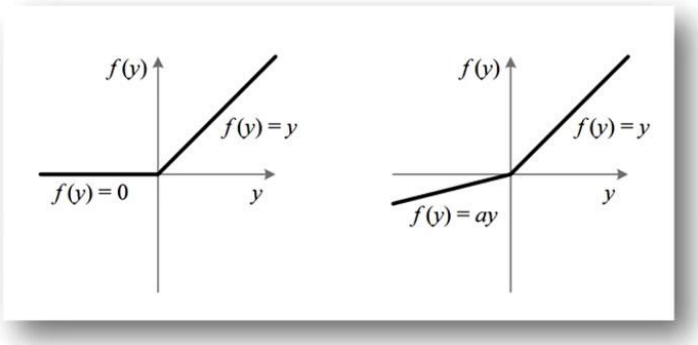
Additional Reading:

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

By Kaiming He et al. (2015)

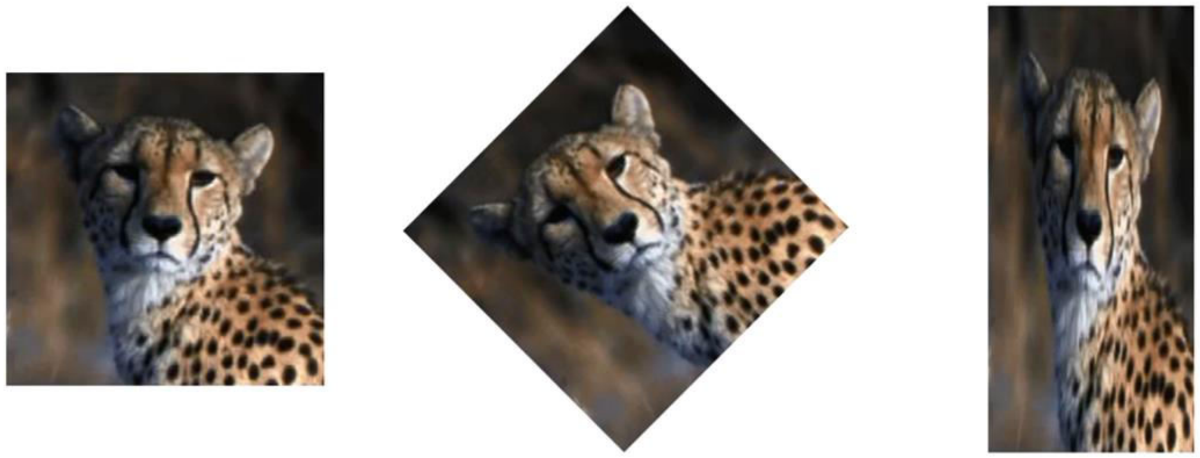
Link:

https://arxiv.org/pdf/1502.01852.pdf



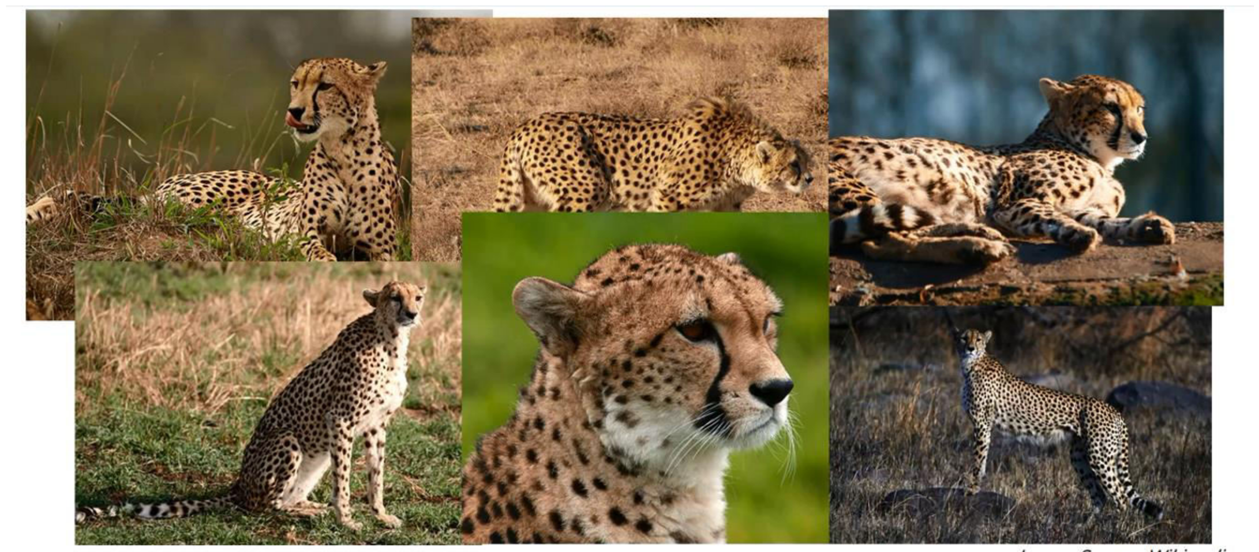
Step 2 – Max Pooling

What is pooling and why need to do that



Let's have a look at all these three images. In these three images we have a cheetah . In these three images we have a cheetah, in fact it is the same exact cheetah, on the first image the image is positioned properly and that she is looking straight at you on the second image it's a bit rotated and the third image a bit squashed.

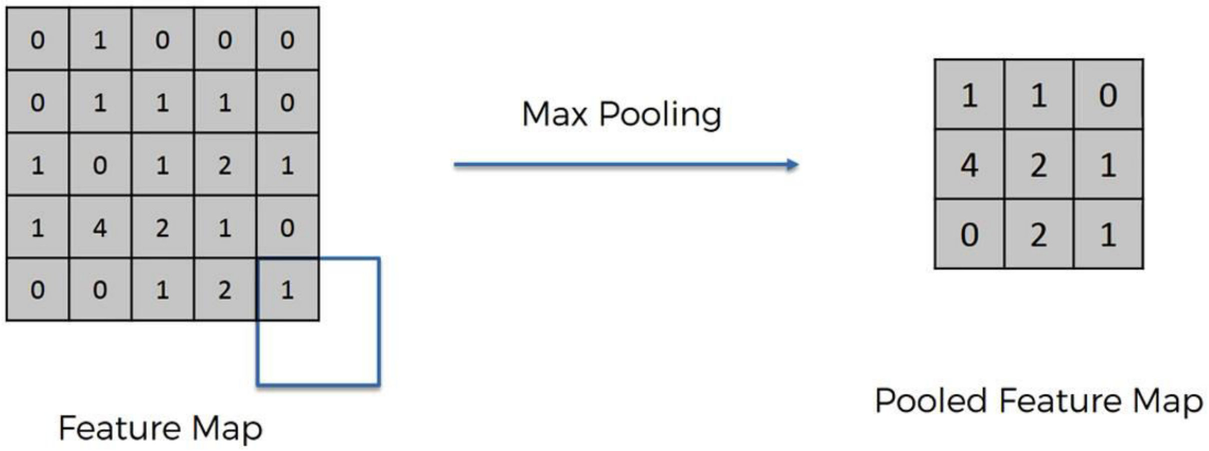
And the thing here is that we want the neural network to be able to recognize the cheetah in every single one of the images.
In fact this is just one cheetah, what if we have lots of different cheetah.



and we want the neural network to recognize all of these cheetah as cheetah and how can it do that if they're all looking in different directions they're all in different parts of the image like their faces are positioned in different parts of the image somebody is on the right hand side somebody in the left corner or somebody in the middle.
They're all a bit different and the texture is a little bit different. The lighting is a bit different. There's lots of little differences and so if the neural network looks for exactly a certain feature
for instance a distinctive feature of the cheetah is the tears that are on its face going from the eyes or the sheer. The shadows that look like tears the texture of the pattern that is going from its
eyes down it's on the sides of its nose and looks like tears that's a distinctive feature of the Cheetah.

But if it's looking for that feature which it learned from certain cheetahs in an exact location or
an exact shape or form or texture it will never find these other cheetah.
So we have to make sure that our neural network has a property called spatial invariance meaning that it doesn't care where the features are again, not so much as itch which part of the image because we're we've kind of taken that into consideration with our map we are poor with our convolutional there
but it doesn't have to care if the features are a bit tilted if the features are a bit different in
texture if the features are a bit closer of features or a bit further apart relative to relative to
each other. So if the feature itself is a bit distorted our neural network has to have some level of

flexibility
to be able to still find that feature. And that is what pooling is all about.



Here's our feature map so we've already done our convolution and we've completed that part and now we're working with the convolutional there.
Now we're going to apply pooling. So how does it work.
We're going to be applying max pooling.
There's several different types of pooling mean pooling, Max pooling, some pooling.
But for now we're just applying Max pooling so we take a box of two by two pixels like that and again it doesn't have to be two by two you can choose any size of box and you place it in the top left hand corner and you find the maximum value in that box
and then you record only that value and you disregard the other three. So in your box you have four values you just disregard three, you only keep one the maximum which is
one in this case. Then you move your box to the right by stride you select the stride once again.
So here we select the stride of two and that's what you normally select you can say like the stride of one you can select. So there are overlapping boxes you can select any kind of stride that you like even three if you want
but we're selecting a stride of two here and that's what is commonly used. And then you repeat the process you record that maxim. Here if you cross over and doesn't
matter you just keep continue doing what you're doing.
So you still record the maximum here the maximum is four.
Here are the maximums to here the maximum is 10 and then so on.
So as you can see a few things happened.
First of all we still were able to preserve the features right.
The maximum numbers they represent because we know how the conclusion layer works.
We know that the maximal or the large numbers in your feature map they represent where you actually found the closest similarity to a feature.
But by then pooling these features we are first of all getting rid of 75% percent of the information that is not the feature which is which is not the important things that we're looking out for because

we're just really disregarding three pixels out of four.
So we're only getting 25% percent.
And then also because we are taking the maximum of the pixels that or the values that we have
we are therefore accounting for any distortion.
So for instance two images in which, for example the cheater's tears on the eyes are in one image there a bit to the left or a bit rotated to the left and another one there a bit.
And are how they're supposed to be or how we like if you take one as the basis and another one there are bits rotate to the left.
The pulled feature will be exactly the same.
So you can see here if we are talking about the cheetah's tears then let's say this is the four and
this is where it was here then if it was a bit rotated.
Then when we are doing the pooling we're still going to get the same pool feature map and that's kind of the principle behind it.

And in addition to all of that we are reducing the size so there's another benefit.
So we've got we're preserving the features we're introducing spatial invariants we're reducing the size by 75% percent which is huge which is really going to help us in terms of processing.
And moreover another benefit of pooling is we are reducing the number of parameters so we're reducing
again by 75% percent or reducing number of parameters that are going to go into our final Layers of the neural network and therefore we're preventing overfitting.

It is a very important benefit of pooling that we're removing information and that is a good thing.
That is a good thing because that way our model won't be able to over fit onto that information because especially that information is not well and remember like at the very start we're talking about
even for human as humans it's important to see exactly the features rather than all this other noise that is coming into our eyes.
Well same thing for neural networks they by disregarding the unnecessary non-important information we're helping with preventing of overfitting.

And the question here is of course why Max pooling right there's lots of different types of pooling
and why a stride of two why a size of two by two pixels lots of all these things.

Additional Reading:

Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition

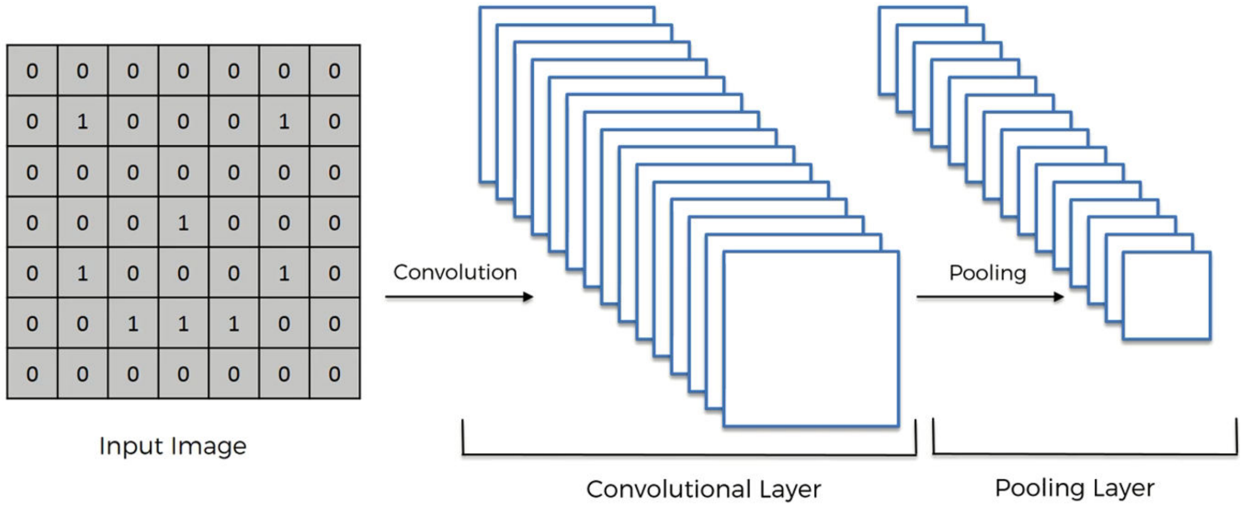
By Dominik Scherer et al. (2010)

Link:

http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

And on that note I'd like to introduce you to this lovely research paper called evaluation of pooling operations in convolutional architectures for object recognition by Dominic Scherrer from University of Bonn.
There is the link and the beauty about this paper is that it's very simple very straightforward
So if you've never read a research paper before what you'd like to give it a go.

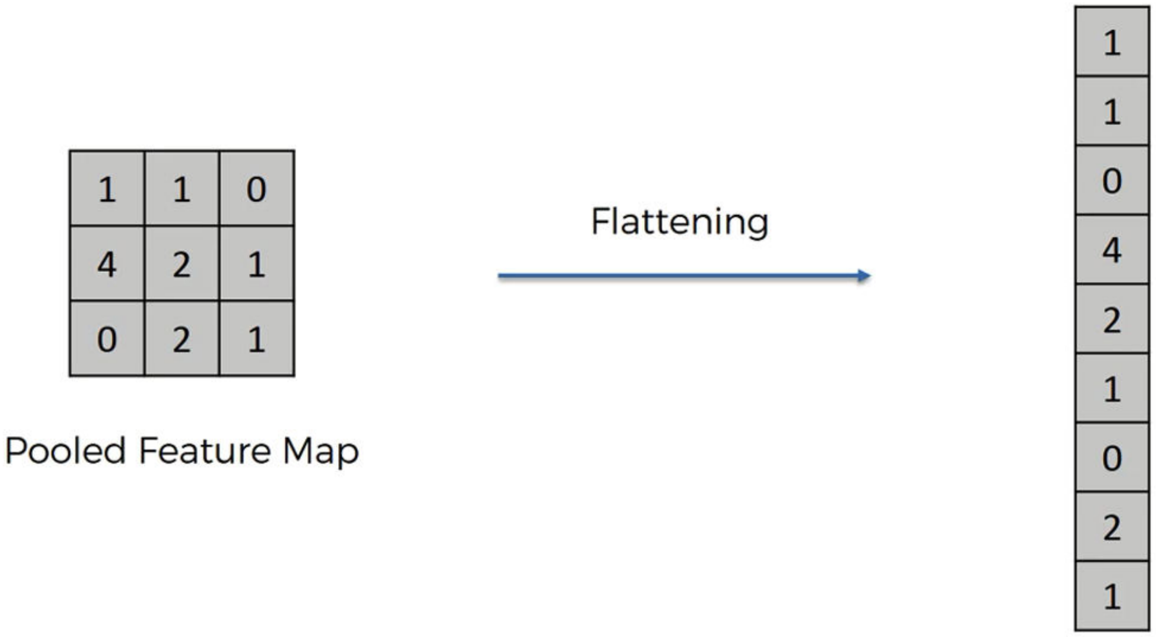
And one thing that you do need to know about this paper they talk about a concept called subsampling which is subsampling is basically average pooling.

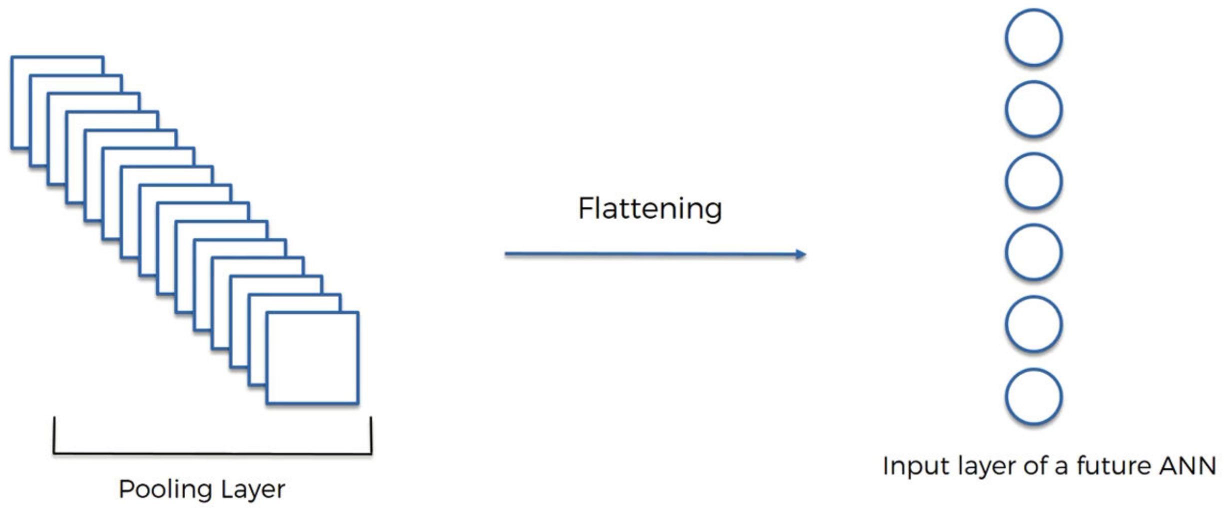


0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Step 3 – Flattening

All right so we so far we've got the pulled layer pulled feature map and that is after we apply the convolution operation to our image and then we apply pooling to the result of the convolution which the convolved image. And so what are we going to do if this pulled feature map. Well we're going to take it and we're going to flatten it into a column. So basically just take the numbers row by row and put them into this one long column. And the reason for that is because we want to later input this into an artificial neural network for further processing.



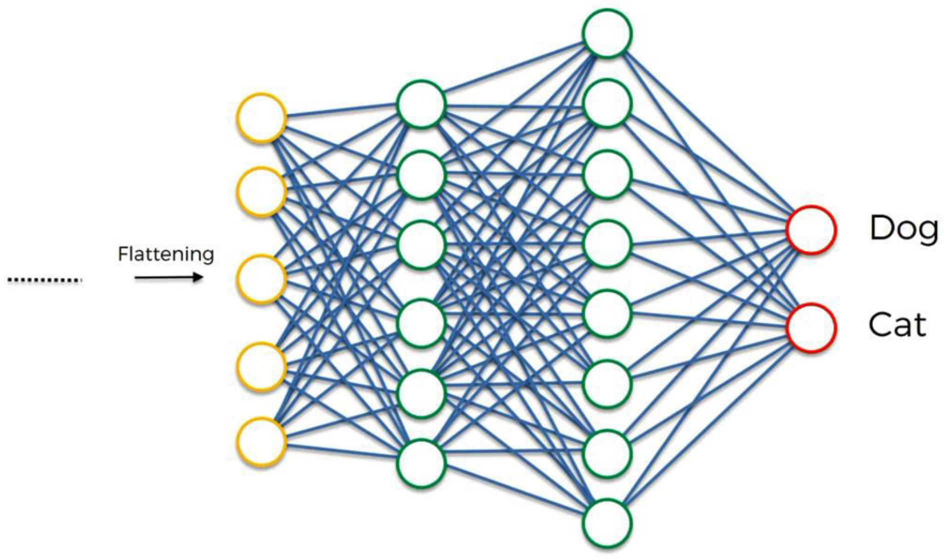


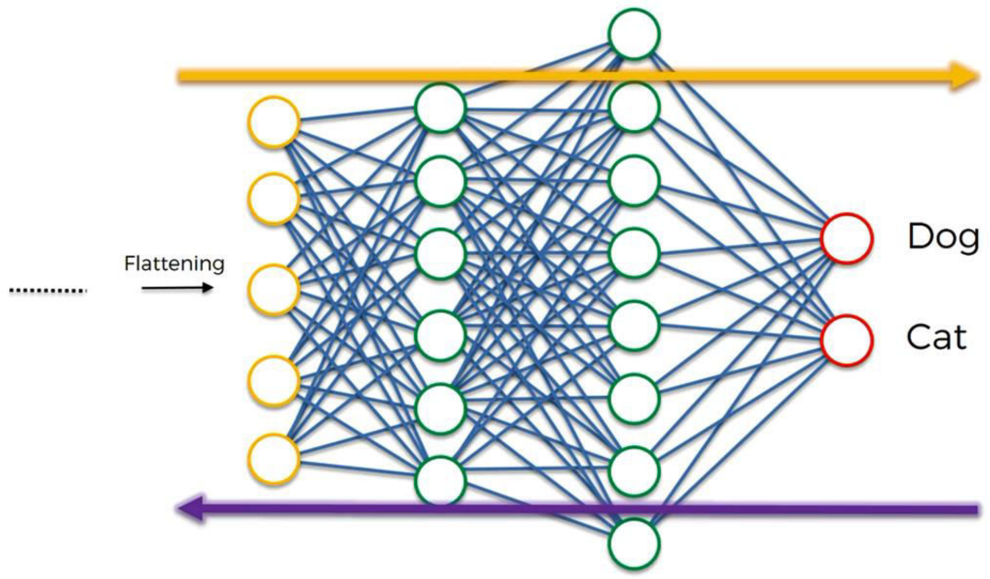
So this is what it looks like when you have many pooling layers or you have the pulling levers with

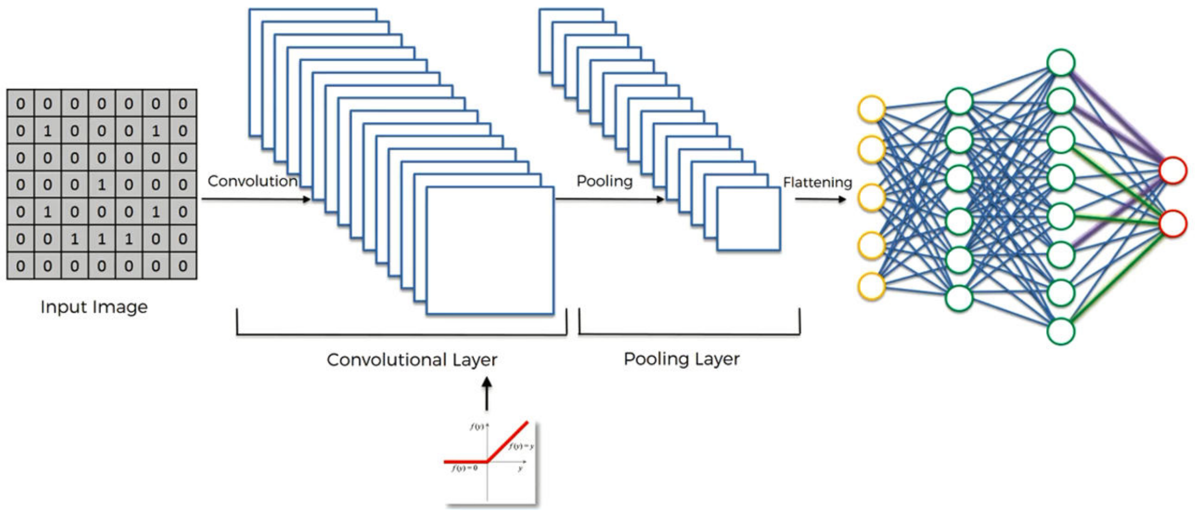
many puled feature maps and then you flatten them so you put them into this one long column sequentially

one after the other and you get one huge vector of inputs for an artificial neural network.

Step 4 – Full Connection



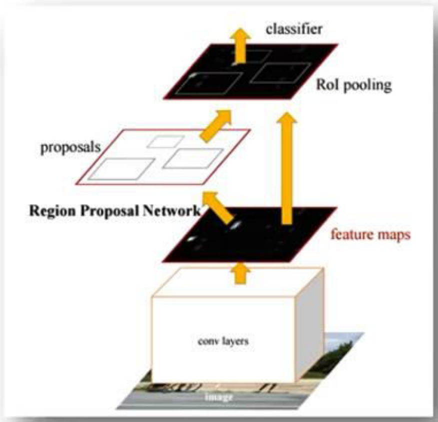

Flattening
Dog
Cat


Input Image
Convolution
Convolutional Layer
Pooling
Pooling Layer
Flattening

Additional Reading:

The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)

Adit Deshpande (2016)



Link:

https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html