Convolution Neural Networks

Lesson 2

Tiago Cabo

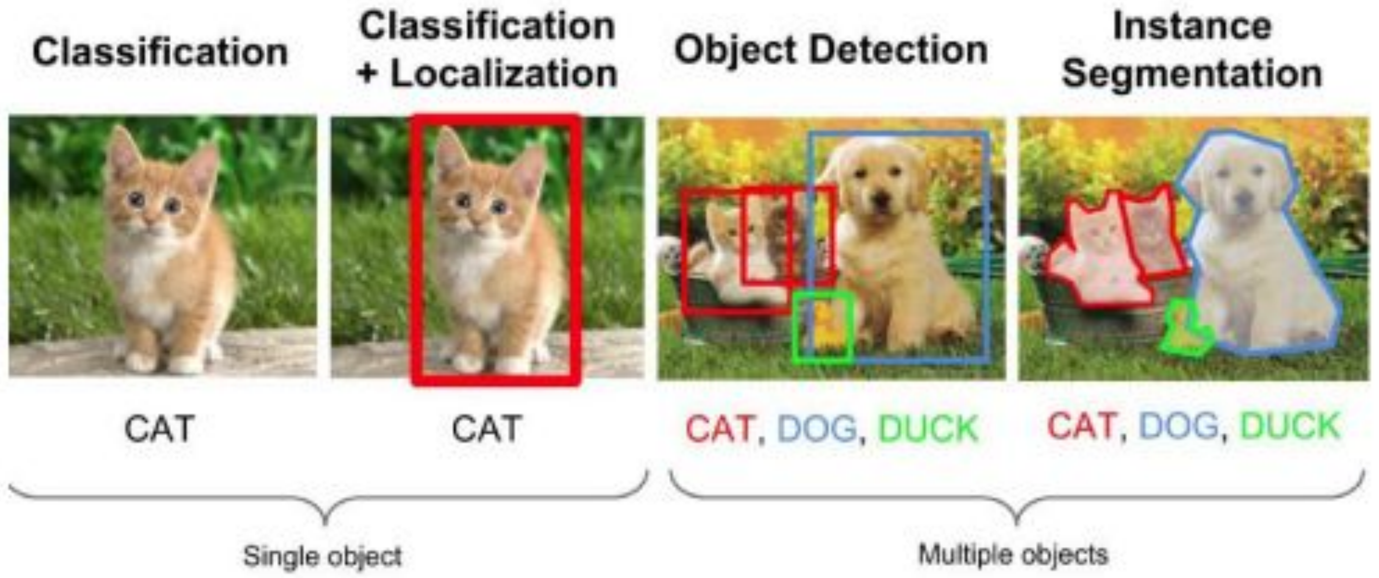
Lesson Plan

- What are CNNs?
- Why CNNs are such big thing now?
- Understand main parameters and consequences
- Visualize layers
- Be able to apply trained CNNs

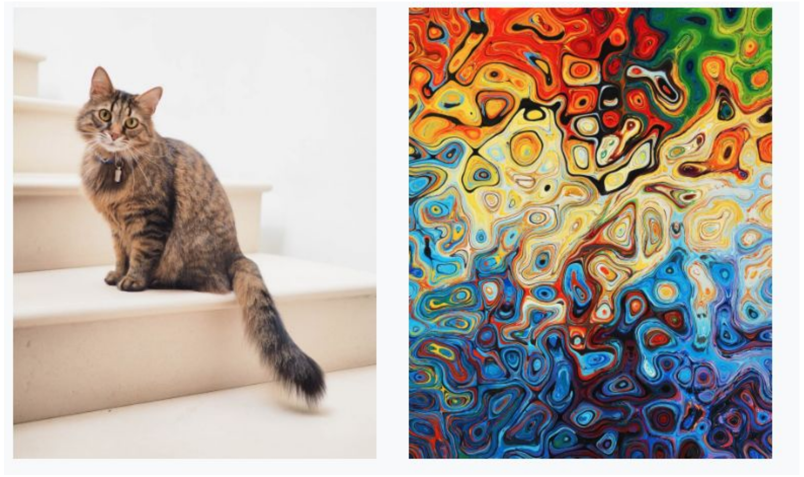
Lesson goals

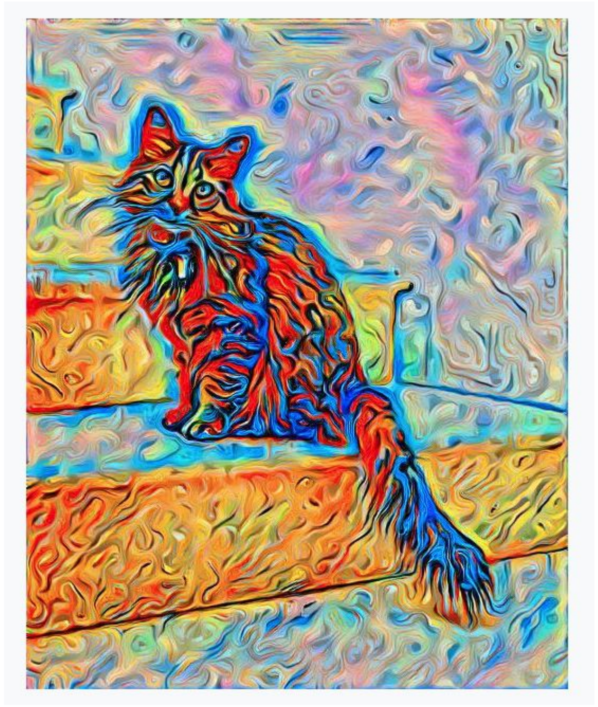
- Be able to explain what CNN are
- Be able to understand main building blocks and apply a simple CNNs

CNNs Aplications

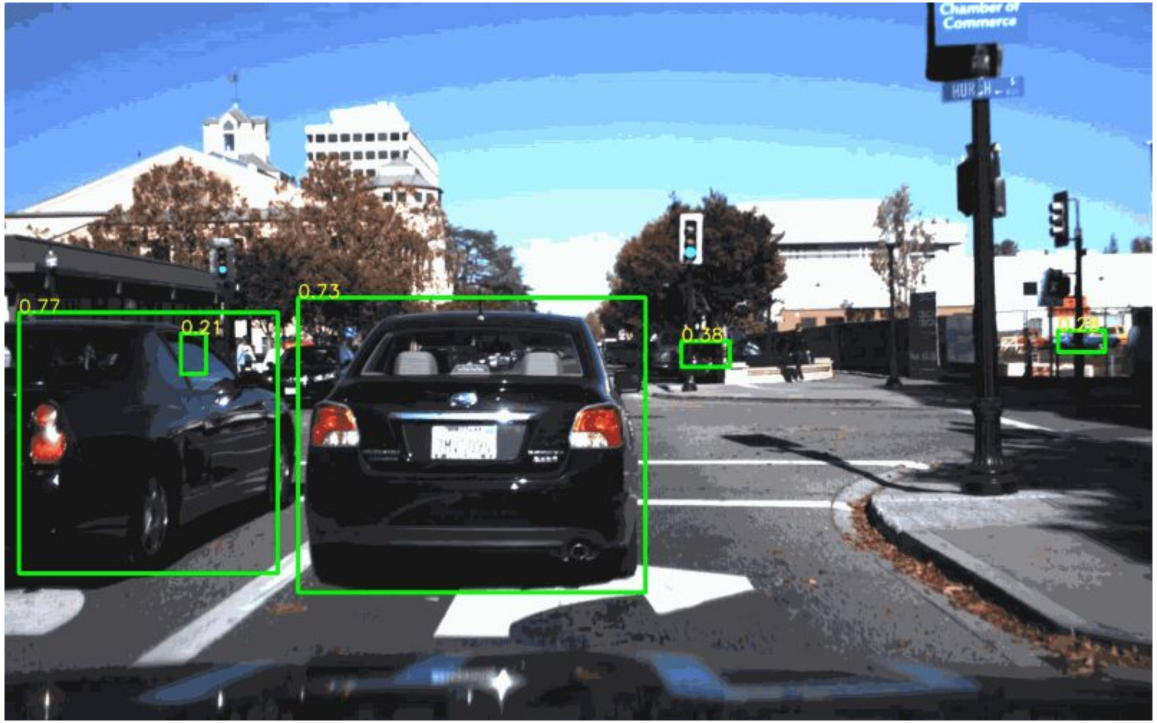


Style Transfer





Self-driving cars



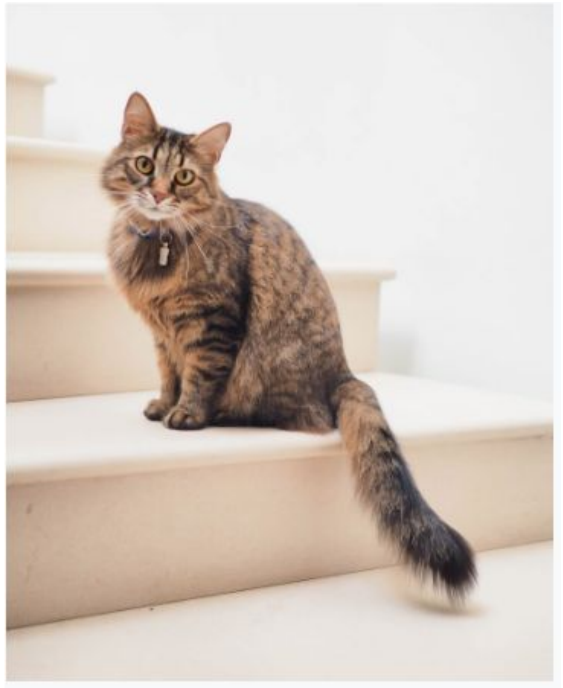
Why not DNN (MLP)?

Let’s consider this cat. Can someone tell me what are the features in our very small dataset?

The features are each individual pixel. So for example in 1MB image you may can easily end up with 3B parameters.

This leads to :

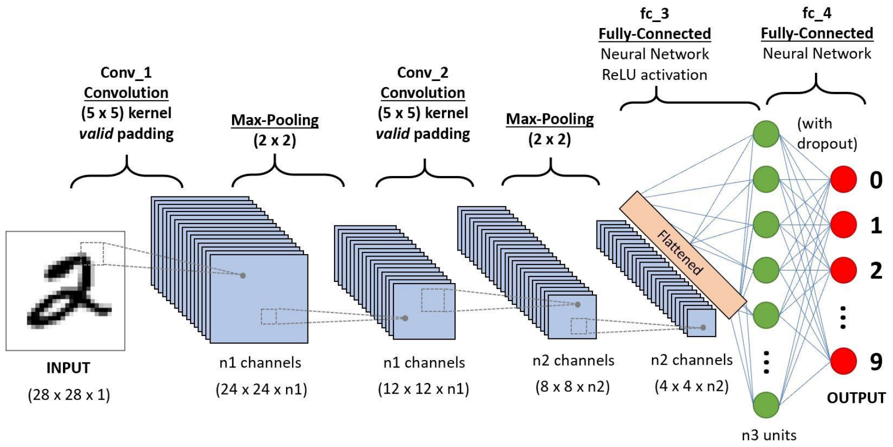
- Memory issues
- Easy to overfit
- Impossible to obtain local information about the picture. For example which part of the image was more important to distinguish the cat from a dog for example.



What are CNNs?

The roots date back to 1950 when David Hubel and Torsten Wiesel found that the brain contains specific neurons that are more aware of specific feature in the visual space.

This lead to the invention of the neocognitron by Kunihiko Fukushima in 1980, that developed the bases of convolutional Neural networks. Developed concepts like filters.



LeNet-5

After several innovation in the 1980s and 1990s, Yann LeCun used for the first time back-propagation to learn the convolution kernel coefficients directly from the image. This was the first time that the learning process was fully automatic.

LeCun is currently Lead Researcher at Facebook (or Meta :))

Video time :)

https://www.reddit.com/r/MachineLearning/comments/kuc6tz/d_a_demo_from_1993_of_32yearold_yann_lecun/

CNNs advantages

Parameter sharing

- Features of the image can be shared across the network

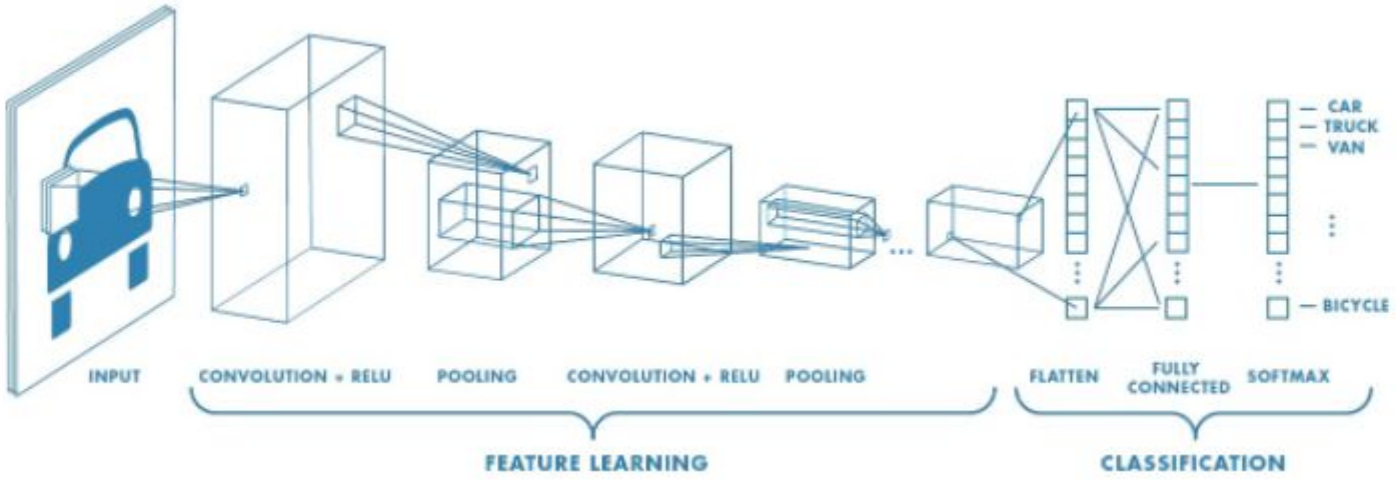
Sparsity of connections

- Each output only depends on parts on parts of the features

Translation Invariance

- This means that if we switch the inputs of the network wor, for example, rotate the image, we should get close results

CNN architecture



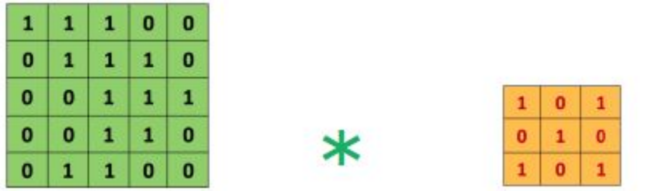
The magic of CNN happen on the feature engineering part. The classification part is a normal NN.

https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

CNN layers

- Convolution
- Pooling
- Relu
- Fully connected DNN

Convolution layer



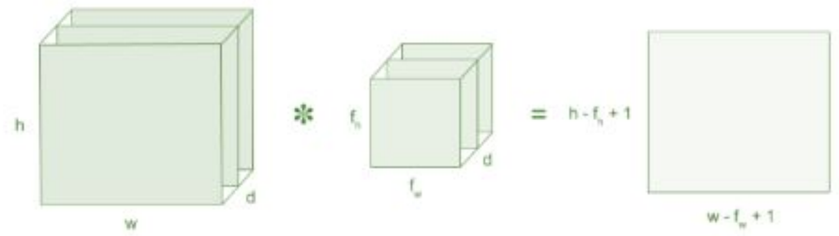
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

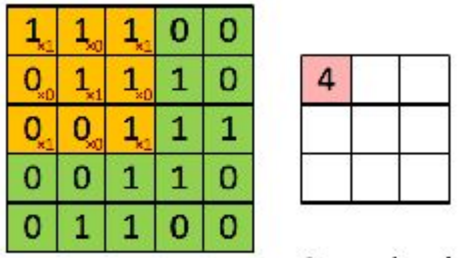
$*$

1	0	1
0	1	0
1	0	1

5 x 5 – Image Matrix **3 x 3 – Filter Matrix**

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**





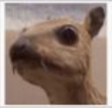
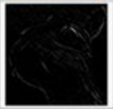
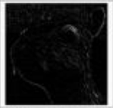
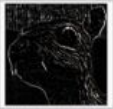
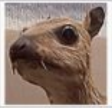
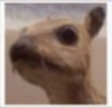
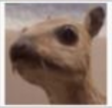
Image

Convolved Feature

Different filters

Let's see this in action:

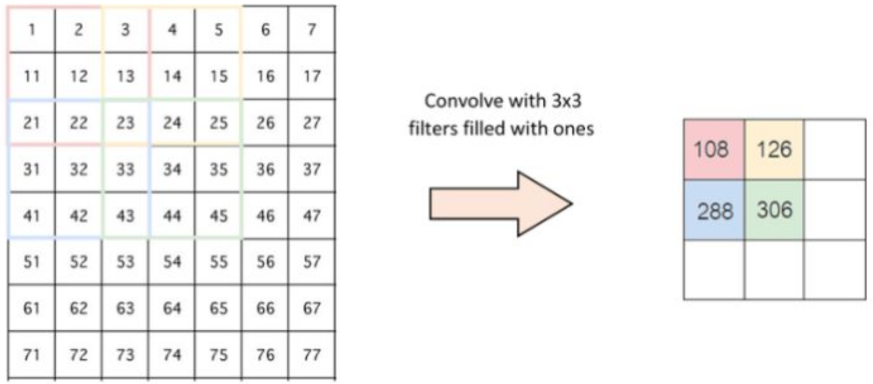
- Load a cat image
- Shape the image to be black and white
- Apply different filter with tensorflow

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Stride

Stride is the number of pixels that we shift when applying the filter.

Let’s use the same notebook and experiment with different strides to see the impact.

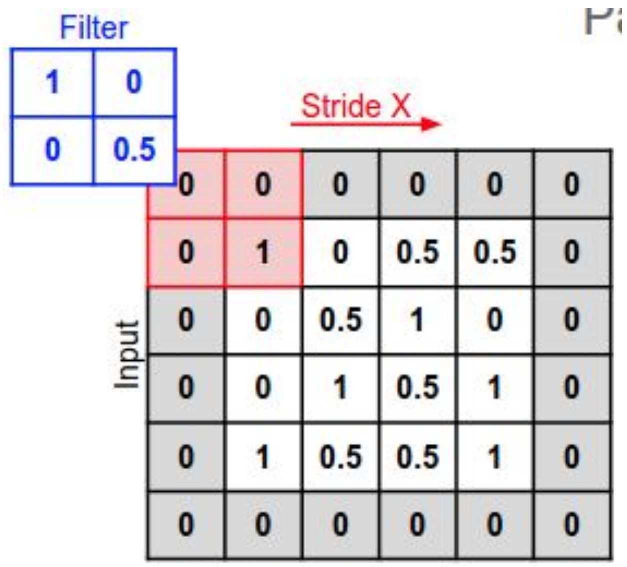


Padding

Sometimes the shape of the filter that not match the image input.

So we have two solutions:

- Pad the image with zeros
- Drop the part of the image where the filter does not fit. This is called valid padding.

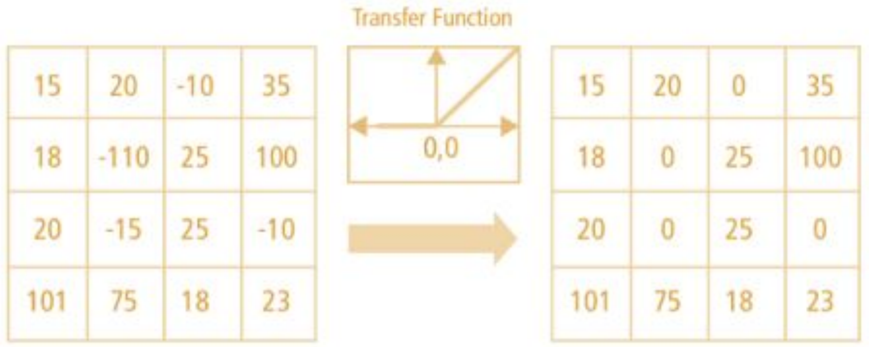


ReLu

ReLU's purpose is to introduce non-linearity in our ConvNet.

Since, the real world data would want our ConvNet to learn would be non-negative linear values.

Let's apply in our example.



15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

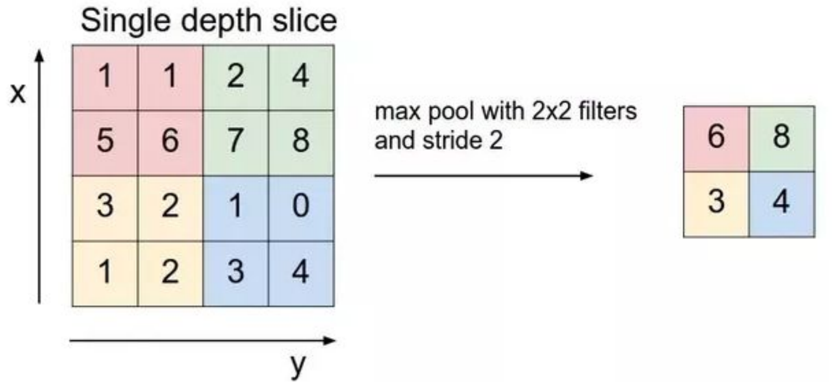
15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

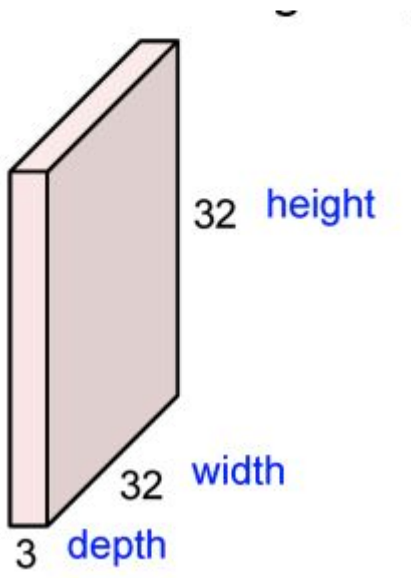
Let's experiment with different values.



Let’s talk about shapes

When working with images, the most common type of images are RGB. This means we have 3 channels, one for each color range for each channel.

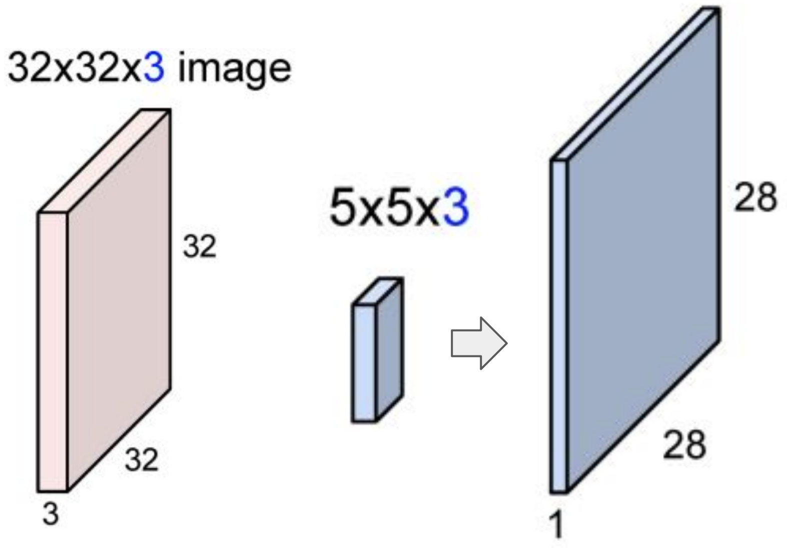
So the shape of our data is (32, 32, 3)



Applying filters

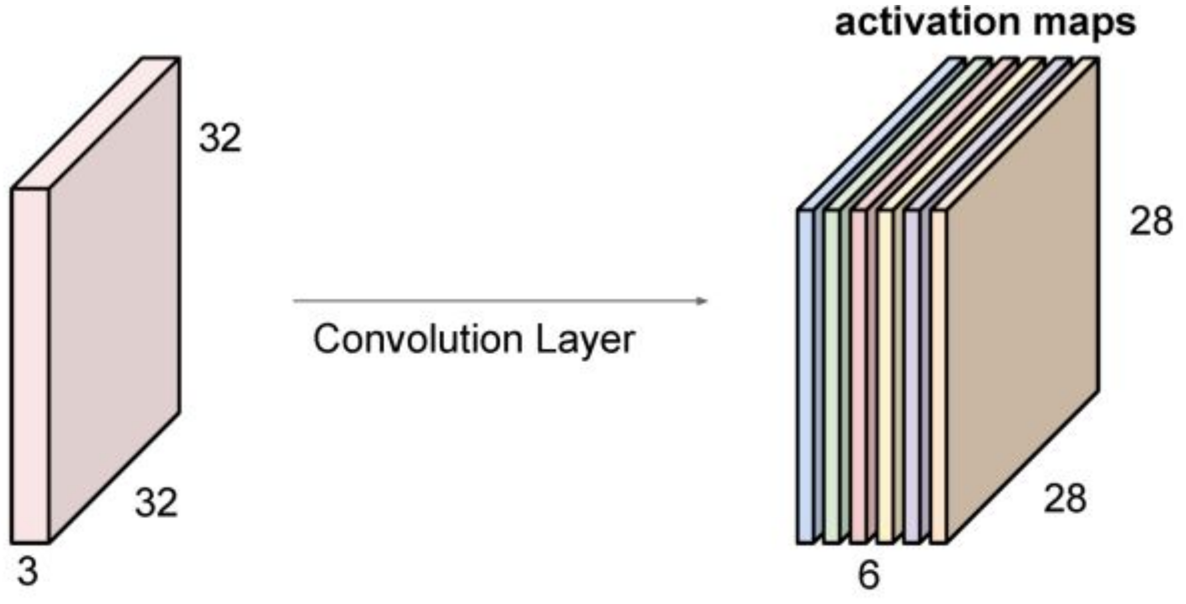
This will lead to an image with the following shape (28, 28, 1).

This means that for a single data point we are calculation a 5*5*3 + bias = 76 dimension dot product



What happens if we had more filters?

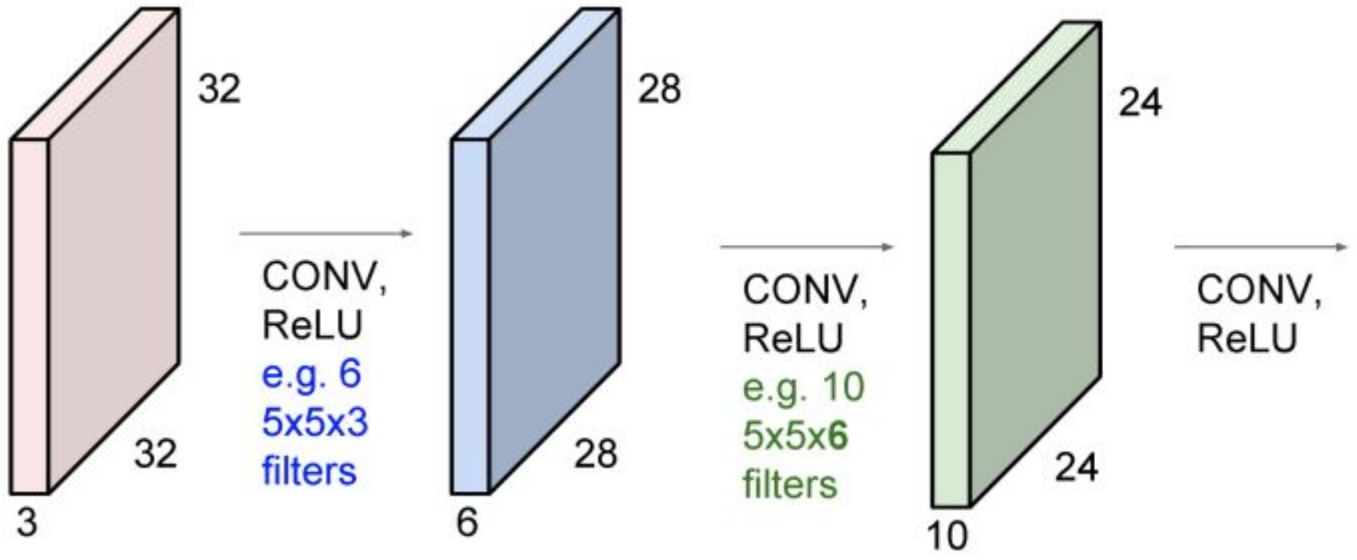
If we had 6 filters?



We stack these up to get a "new image" of size 28x28x6!

Real CNNs

Real world CNN are a concatenation of different convolutions and activation functions

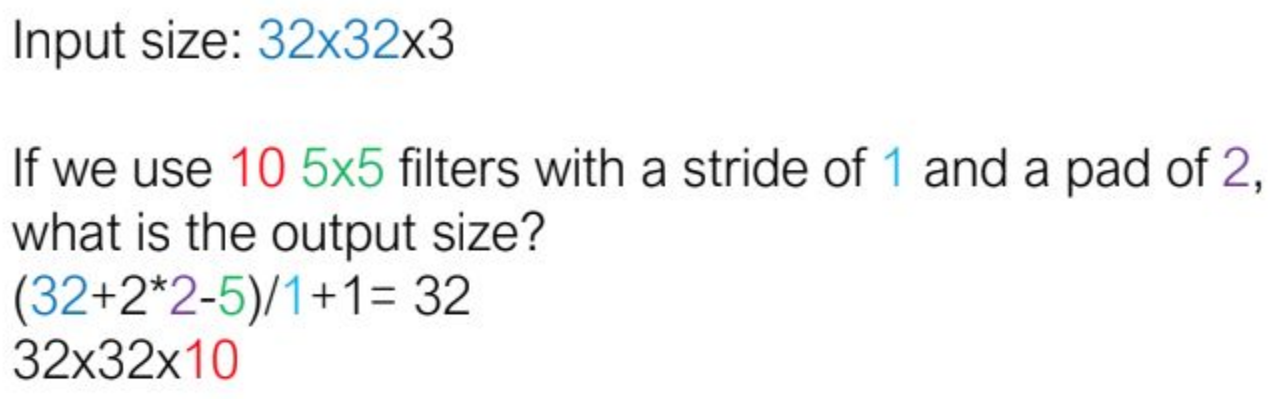


Exercise

Input size: 32x32x3

If we use 10 5x5 filters with a stride = 1 and padding = 2, what is the resulting shape?

Solution



Input size: 32x32x3

If we use 10 5x5 filters with a stride of 1 and a pad of 2, what is the output size?

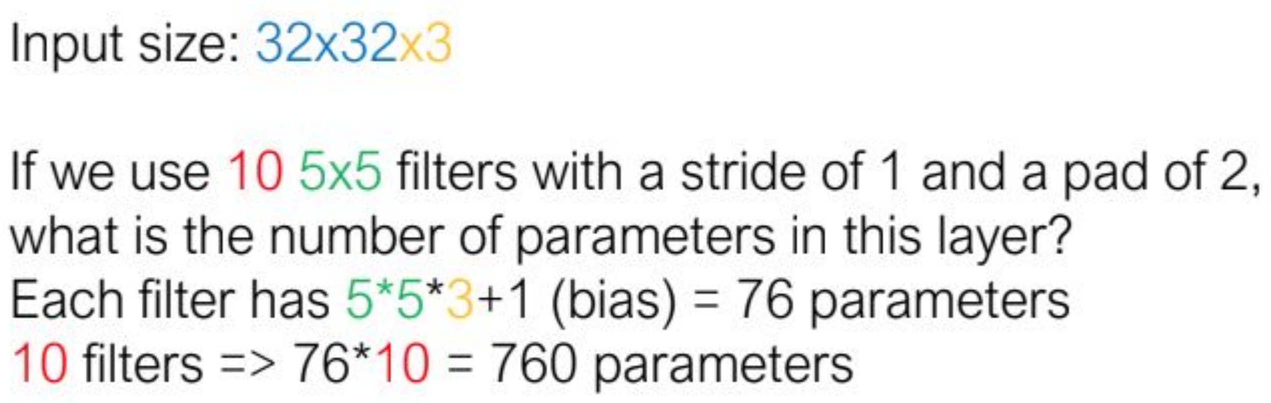
(32+2*2-5)/1+1= 32

32x32x10

Exercise

What is the number of parameters?

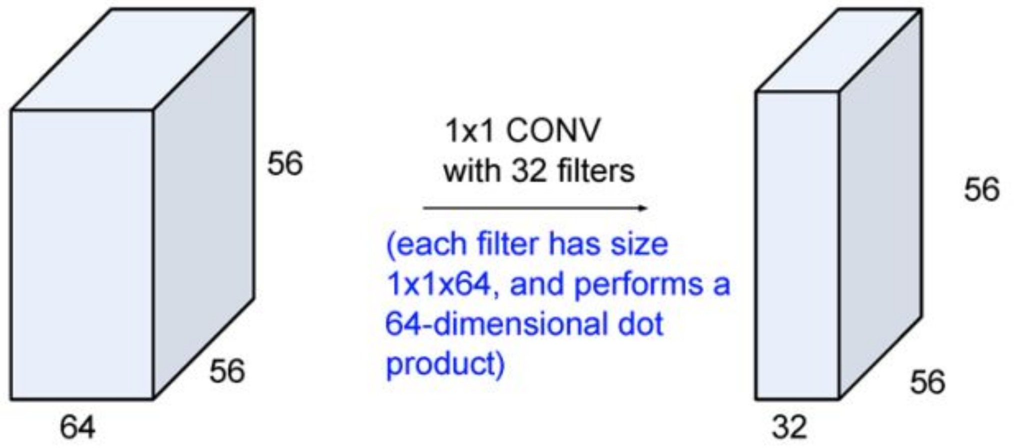
Solutions



Input size: 32x32x3

If we use 10 5x5 filters with a stride of 1 and a pad of 2, what is the number of parameters in this layer?
Each filter has 5*5*3+1 (bias) = 76 parameters
10 filters => 76*10 = 760 parameters

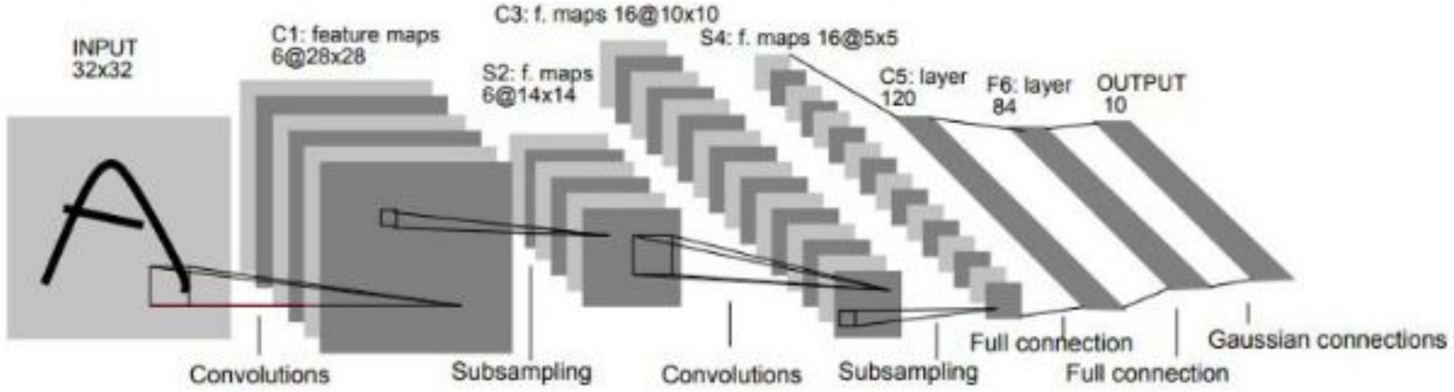
1x1 Conv layers


56
56
64
1x1 CONV
with 32 filters
(each filter has size 1x1x64, and performs a 64-dimensional dot product)
56
56
32

https://machinelearningmastery.com/introduction-to-1x1-convolutions-to-reduce-the-complexity-of-convolutional-neural-networks/

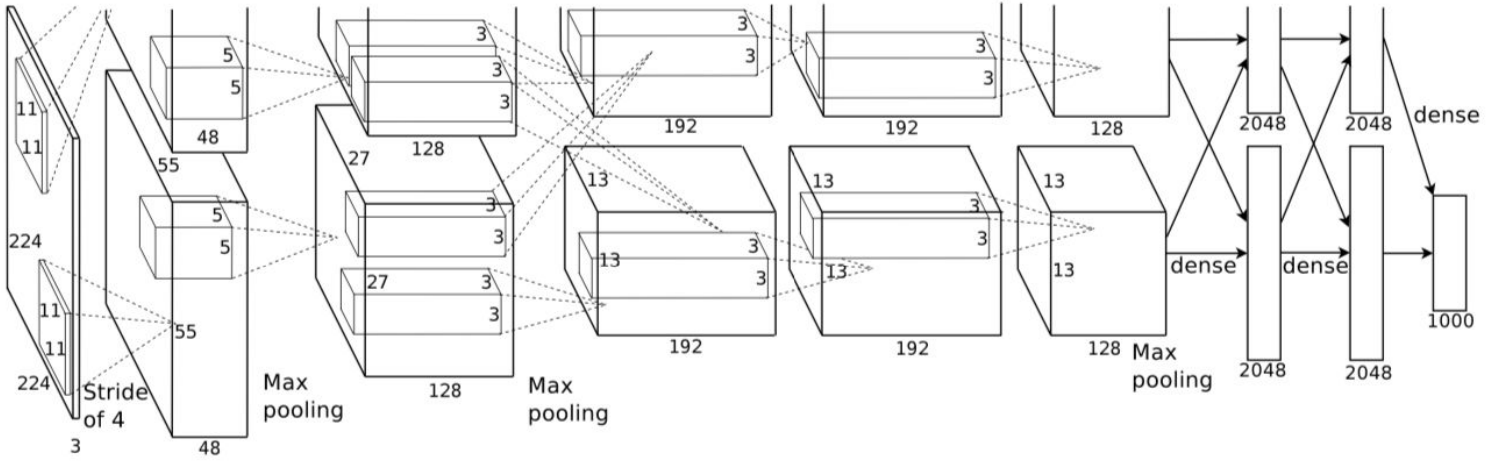
Pre - trained Networks

Pretrained arquitectures - LeNet5



https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/

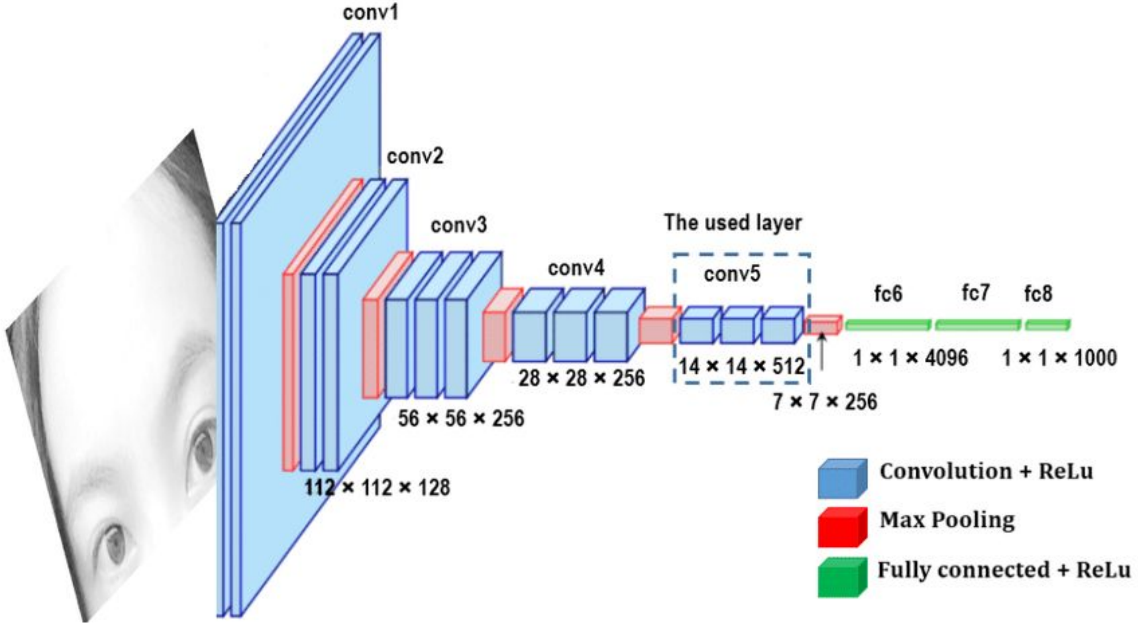
Pretrained arquitectures - AlexNet



https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951

https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

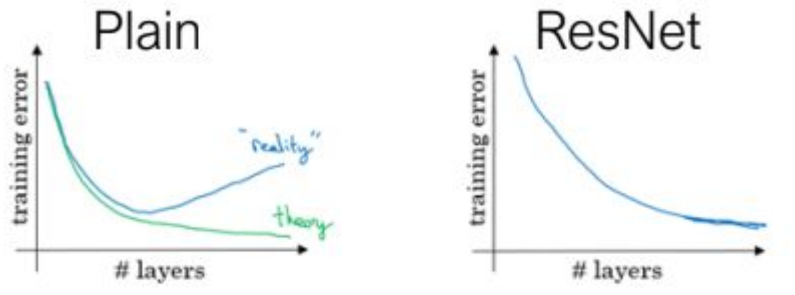
Pretrained arquitectures - VGG



https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c

Pretrained arquitectures - Resnet

Resnet is a 50 layers CNN very popular that achieved awesome results due to a breakthrough in the network that enables better training over an increase number of layers.
In Resnet network there are also skip connections that fixed the issues of vanishing gradients.



https://viso.ai/deep-learning/resnet-residual-neural-network/

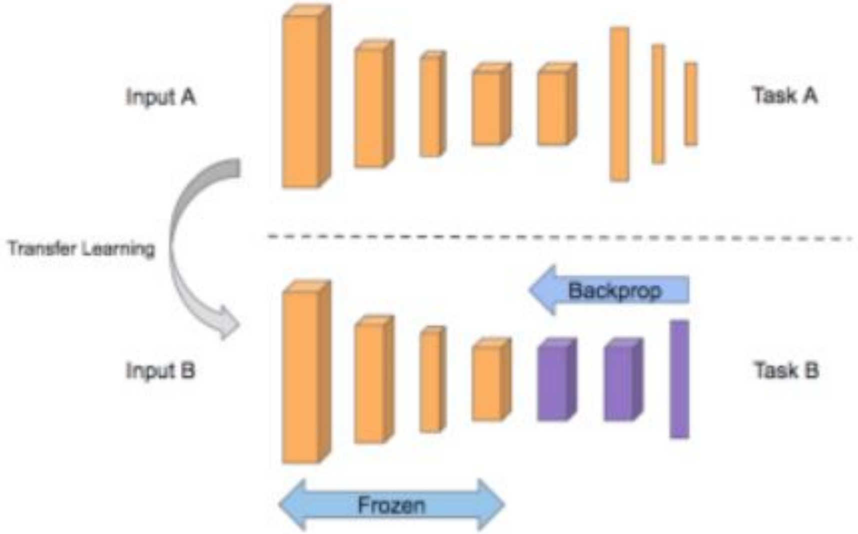
Exercise

- Load fashion MNIST
- Define CNN
- Calculate predict
- Visualize features

Transfer learning

Transfer learning

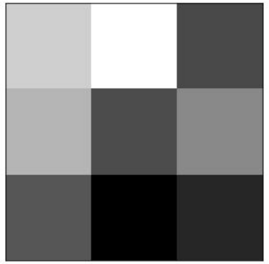
Transfer learning consists of the act of using a pre-trained network and restrained with new data. This is very important, because deep learning depends a lot on hardware to be trained. So most companies cannot afford to train a huge network from scratch, so what most do is to reuse models like ResNet-50 and fine tuned with your specific data.

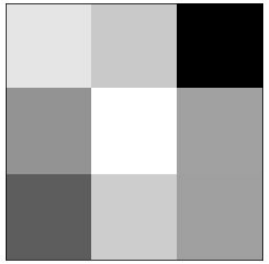


How to visualize filters

CNN uses learned filters to convolve the feature maps from the previous layer. Filters are two- dimensional weights and these weights have a spatial relationship with each other. The steps:

- Iterate through all the layers of the model using model.layers
- If the layer is a convolutional layer, then extract the weights and bias values using get_weights() for that layer.
- Normalize the weights for the filters between 0 and 1
- Plot the filters for each of the convolutional layers and all the channels. For Color image, you will have three channels for RGB. For a grayscale image, the number of channels will be 1





How to visualize activation layer

Feature maps are generated by applying Filters or Feature detectors to the input image or the feature map output of the prior layers. Feature map visualization will provide insight into the internal representations for specific input for each of the Convolutional layers in the model. The steps are:

1. Define a new model, visualization_model that will take an image as the input. The output of the model will be feature maps, which are an intermediate representation for all layers after the first layer. This is based on the model we have used for training.
2. Load the input image for which we want to view the Feature map to understand which features were prominent to classify the image.
3. Convert the image to NumPy array
4. Normalize the array by rescaling it
5. Run the input image through the visualization model to obtain all
6. intermediate representations for the input image.
7. Create the plot for all of the convolutional layers and the max pool layers but not for the fully connected layer. For plotting the Feature maps, retrieve the layer name for each of the layers in the model.

Exercise

- Load pretrained VGG16
- Load bird.img
- Calculate predict
- Visualize features