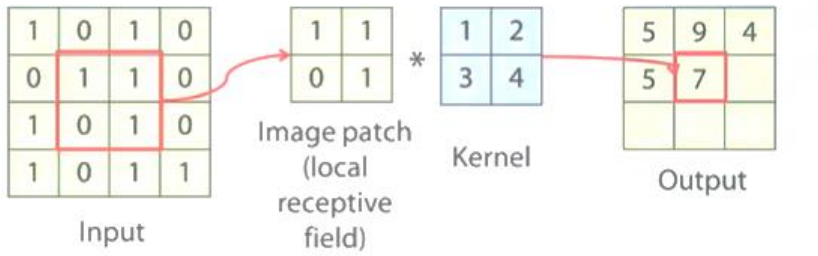
CONVOLUTION NEURAL NETWORK Part II: (Lecture 7)

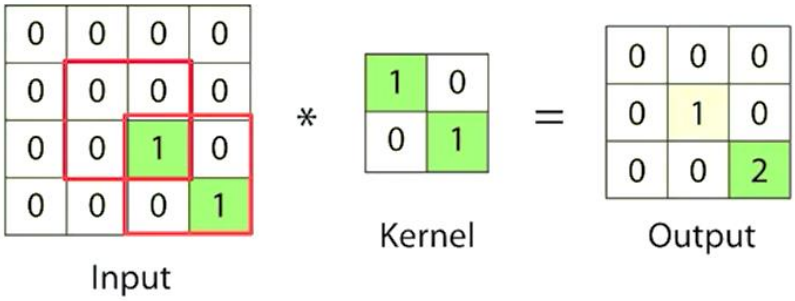
Recap:

What is convolution?

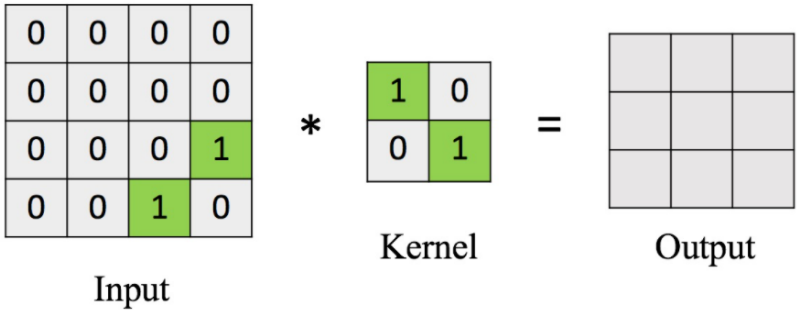
Convolution is a dot product of a kernel (or filter) and a patch of an image (local receptive field) of the same size

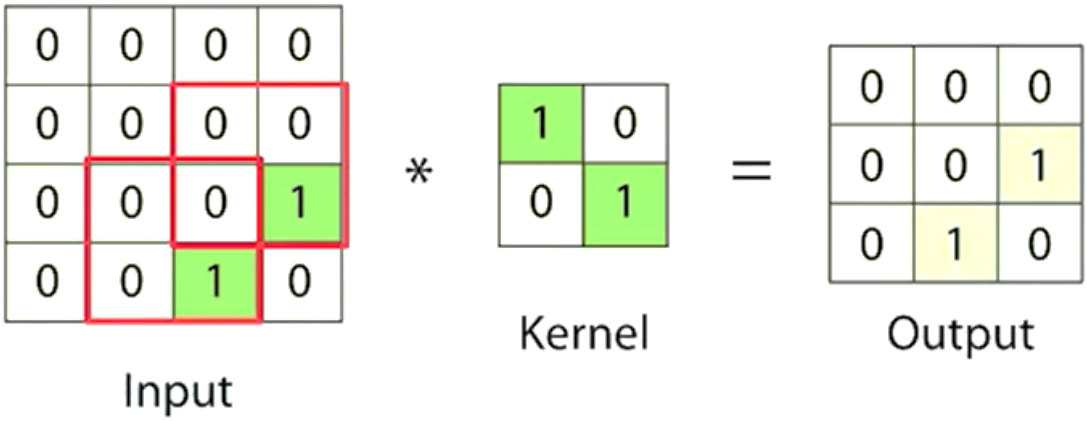


Convolution is like correlation:

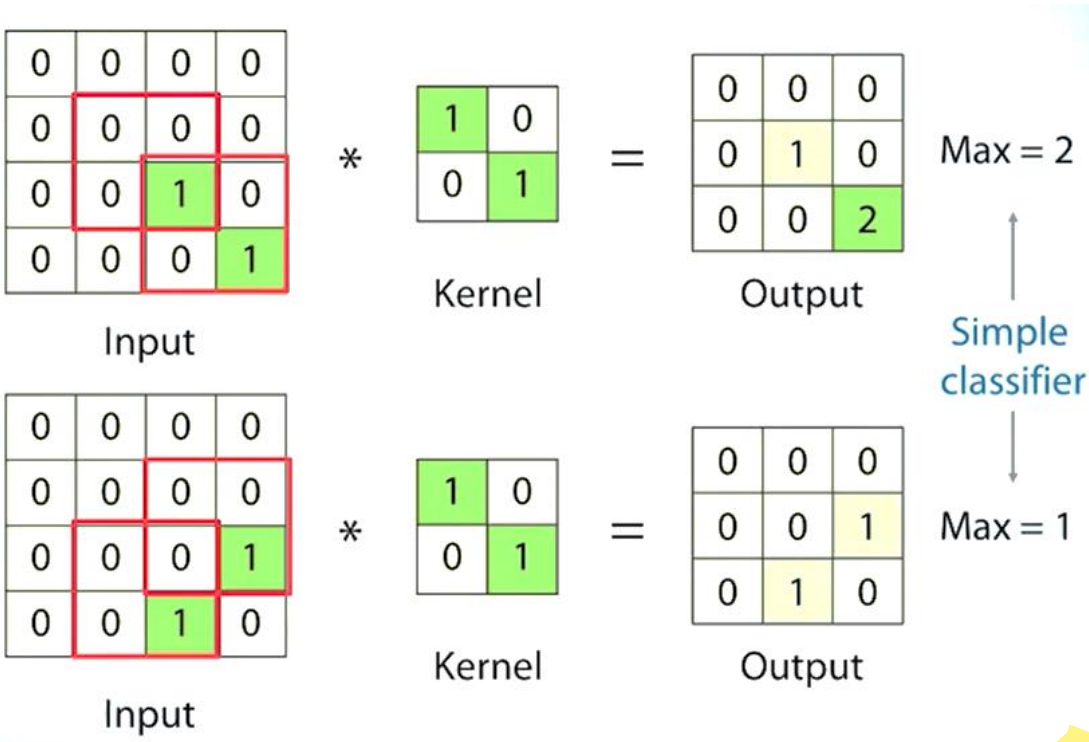


What is the sum of convolution outputs for the following image? We use 2x2 convolution with the step of sliding window of 1 (stride = 1).

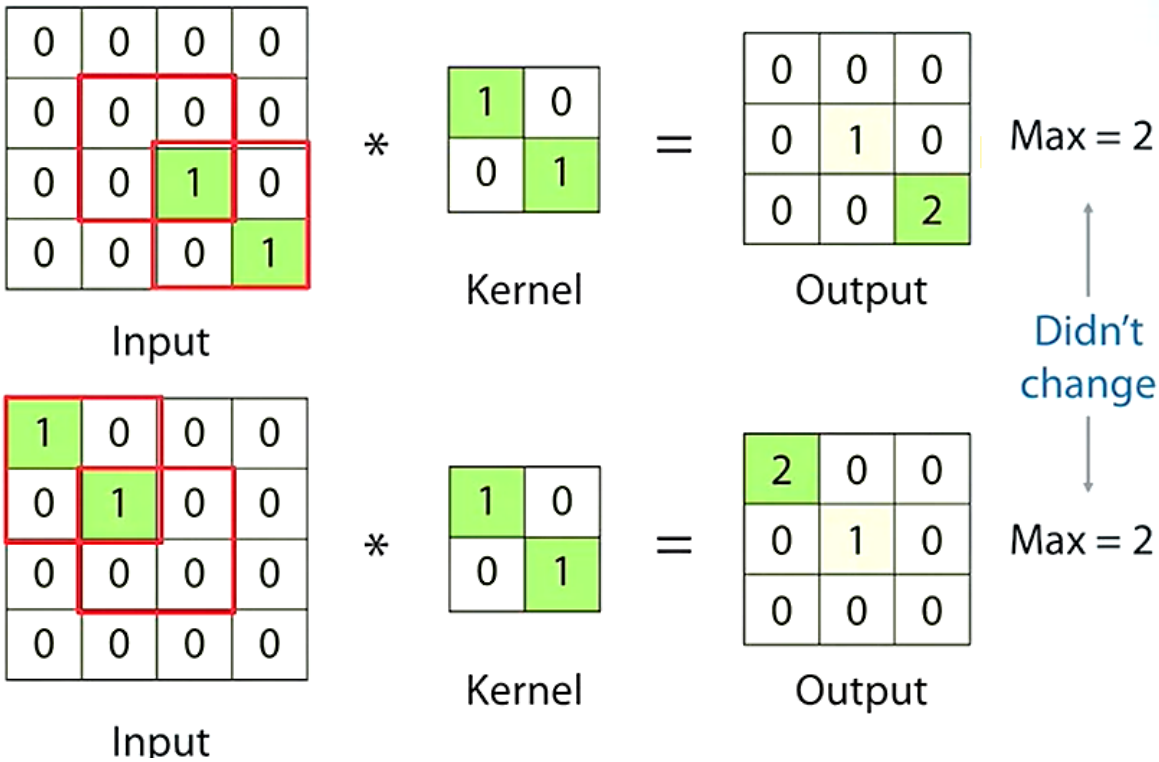




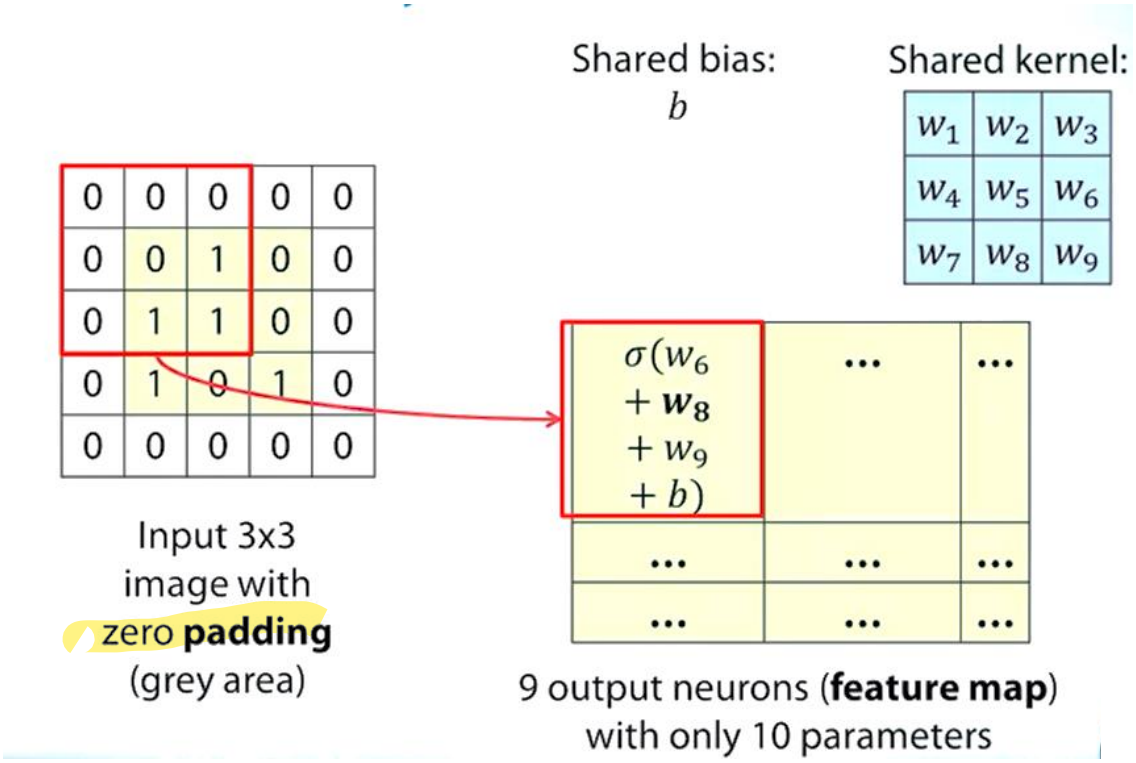
What do we observe?

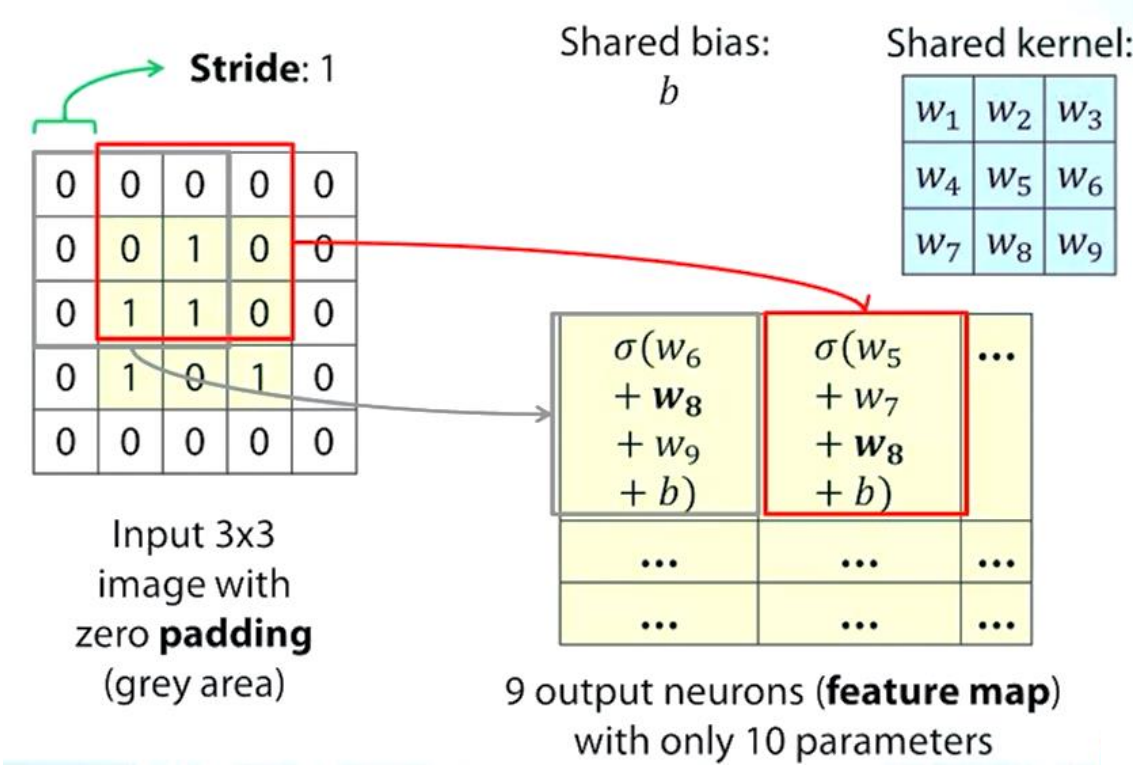


CONVOLUTION IS TRANSLATION EQVIVARIANT:



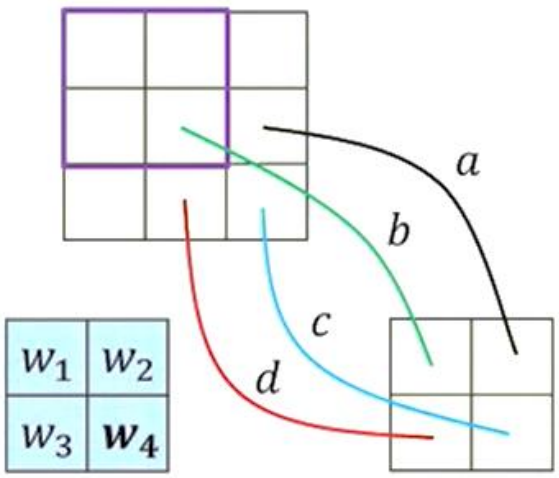
HOW DOES CONVOLUTION WORKS IN NEURAL NETWORK?





HOW DOES BACKPROPAGATION WORKS FOR CNN?

Gradients are first calculated as if the kernel weights were not shared



$$a = a - \gamma \frac{\partial L}{\partial a} \qquad b = b - \gamma \frac{\partial L}{\partial b}$$

$$c = c - \gamma \frac{\partial L}{\partial c} \qquad d = d - \gamma \frac{\partial L}{\partial d}$$

$$w_4 = w_4 - \gamma \left(\frac{\partial L}{\partial a} + \frac{\partial L}{\partial b} + \frac{\partial L}{\partial c} + \frac{\partial L}{\partial d} \right)$$

So, the gradients of the same shared weights are summed up

CONVOLUTION VS FULLY CONNECTED LAYER

In convolutional layer the same kernel is used for every output neuron, this way we share parameters of the network and train a better model

QUESTION:

Suppose we have a 300x300 input and 300x300 output. How many parameters do you need to train a convolutional layer with kernel size 5x5?

26

But

How much do we need in a fully connected layer?

8.1×10^9

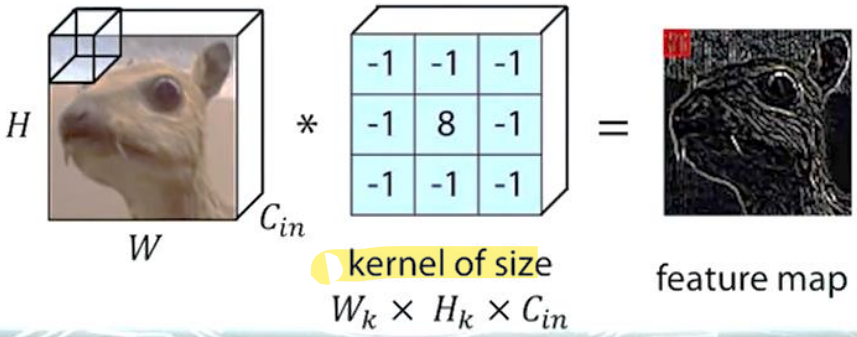
So,

Convolutional layer can be viewed as a special case of a fully connected layer when all the weights outside the local receptive field of each neuron equal 0 and kernel parameters are shared between neurons.

SO How It Works Altogether?

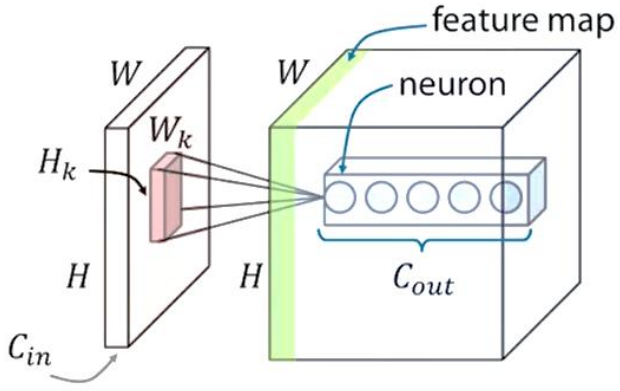
Let's say we have a color image as an input, which is $W \times H \times C_{in}$ **tensor** (multidimensional array), where

- W – is an image width,
- H – is an image height,
- C_{in} – is a number of input channels (e.g. 3 RGB channels).



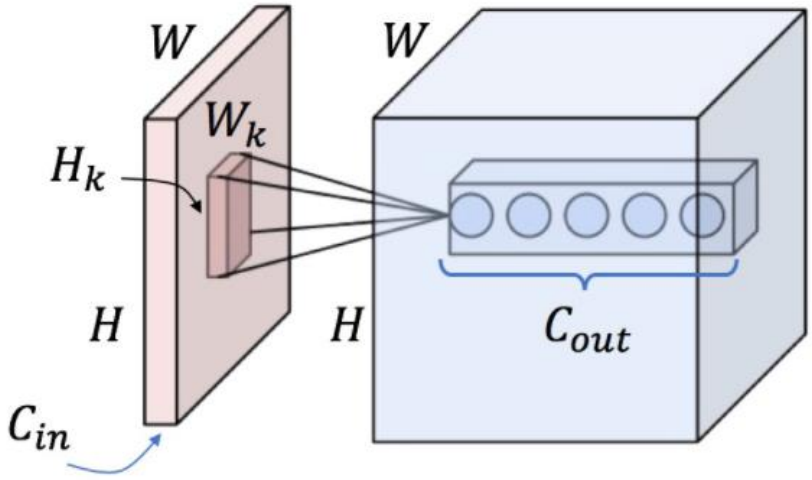
One Kernel is not enough!

- We want to train C_{out} kernels of size $W_k \times H_k \times C_{in}$.
- Having a stride of 1 and enough zero padding we can have $W \times H \times C_{out}$ output neurons.

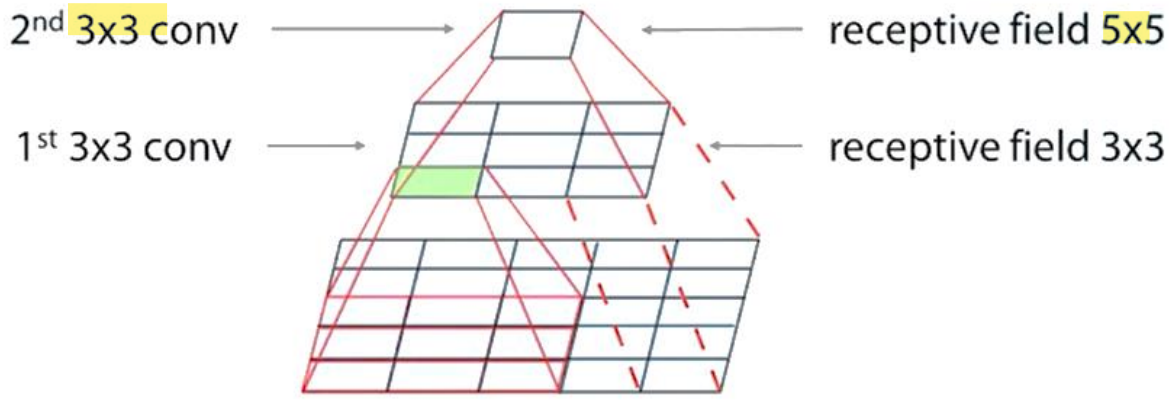


QUESTION:

How many parameters do we need to train this convolutional layer (C_{out}C_{out} output channels)?



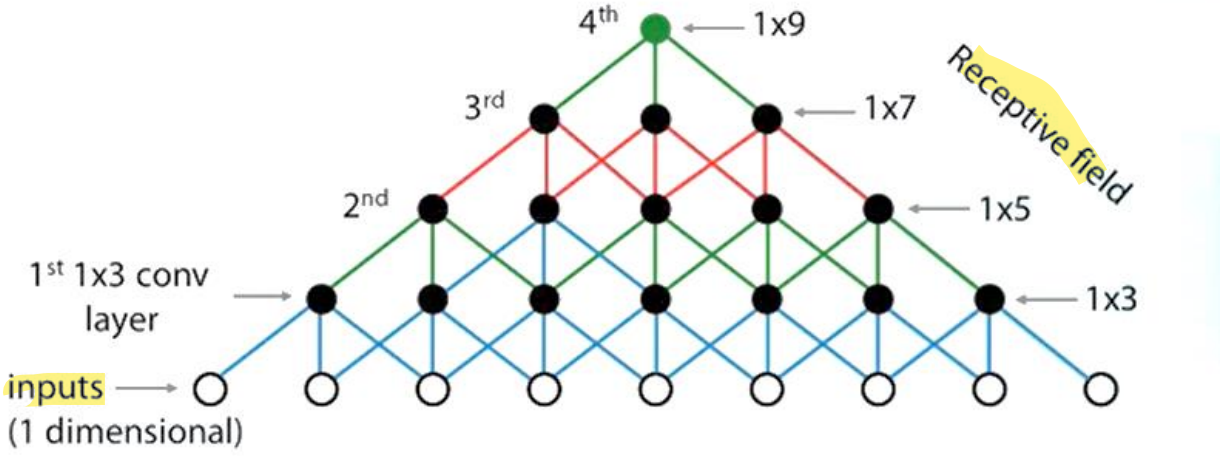
Using $(W_k * H_k * C_{in} + 1) * C_{out}$ parameters.



QUESTION:

What is the receptive field for the 4th convolutional layer (all layers have 3x3 kernels and stride 1)?

What is Receptive Field After n-Convolutional Layers:

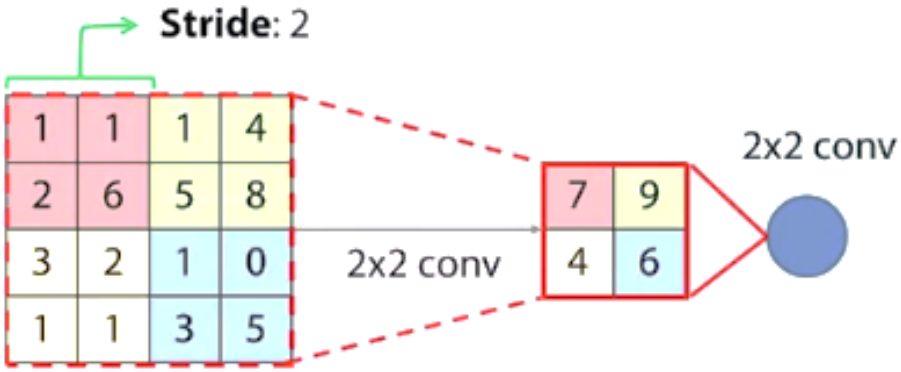


- If we stack N convolutional layers with the same kernel size 3x3 the receptive field on the N-th layer will be
  2N + 1 x 2N +1

- It looks like we need to stack a lot of convolutional layers

Solution:

- We need to grow receptive field faster!
- We can increase a **stride** in our convolutional layers to reduce the output dimensions



What is a Receptive Field?

- *The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)*.

- A receptive field of a feature can be described by its **center location** and its **size**.

- However, not all pixels in a receptive field is equally important to its corresponding CNN's feature.

- Within a receptive field, the closer a pixel to the center of the field, the more it contributes to the calculation of the output feature.

- Which means that a feature does not only look at a particular region (i.e. its receptive field) in the input image, but also focus exponentially more to the middle of that region.

Assignment:

Instructions

*Warning! This assignment requires **several hours** of computation, because it contains training of a real world neural network architecture on a CPU, which we think is essential for learning deep learning. We recommend to use offline instructions at the bottom to setup your own environment (Docker or Anaconda). If it's not an option, you can proceed with Coursera Notebooks, but do expect kernel interruptions. In the meantime we're adding model checkpointing to our tasks, this should help to continue from where you left off. We think the time to setup your own environment is worth the comfort throughout the course.*

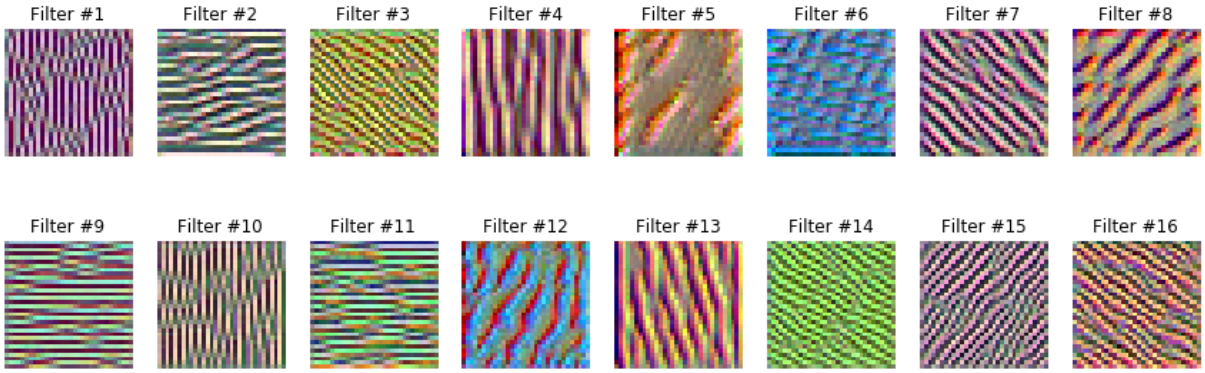
In this task you will build your first Convolutional Neural Network!

More specifically you will do the following:

1. Define your first CNN architecture for CIFAR-10 dataset, which contains 32x32 colour images from 10 classes: **airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck**:

2. Train it from scratch, observing the decrease of the loss for several hours :)

3. Visualise learnt filters like this:



You will see ## GRADED PART, DO NOT CHANGE! in some cells that are meant for grading. You must run them to submit your answers to graded assignment parts. You can send partial answers to see your progress on the task.

Good luck!