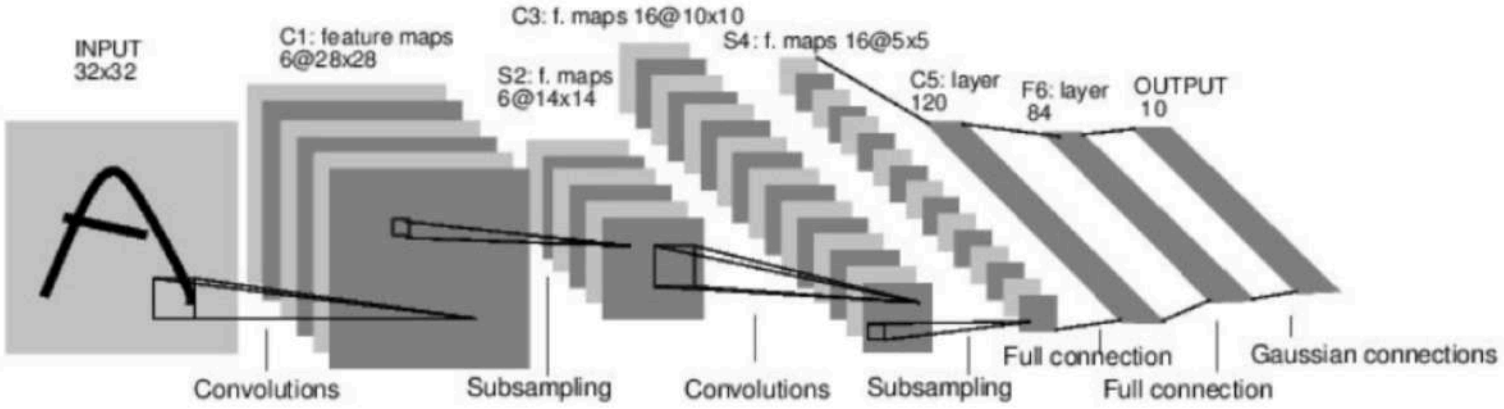
CNN for Image Classification

(Using Tensor-flow & Keras)

-Deepak Kumar Pradhan
Research Engineer | Educator
www.deepakpradhan.in

CNNs / ConvNets



Where do they exist in real world ?

Where do they exist in real world ?

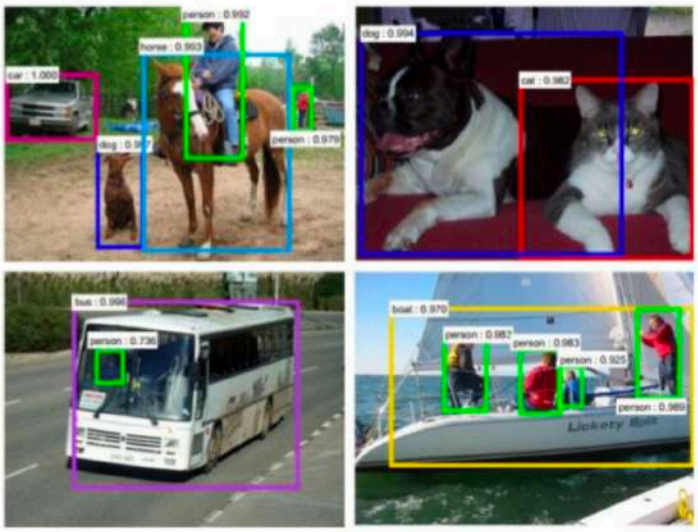
Classification

Retrieval

[Krizhevsky 2012]

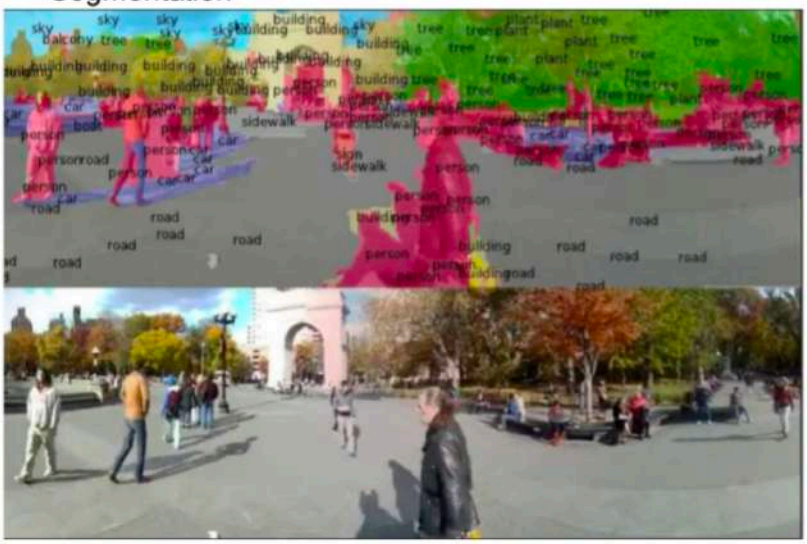
Where do they exist in real world ?

Detection



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

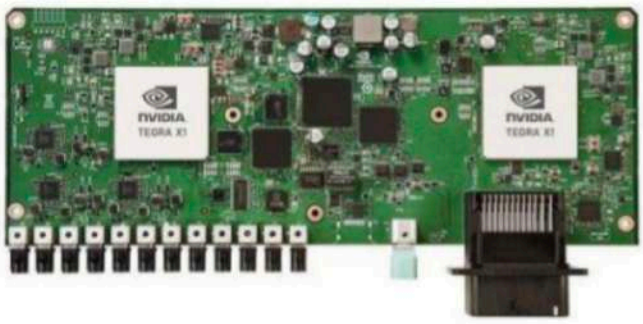
Segmentation



[Farabet et al., 2012]

Where do they exist in real world ?

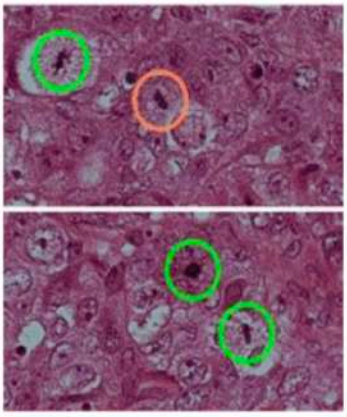
self-driving cars



NVIDIA Tegra X1

Where do they exist in real world ?

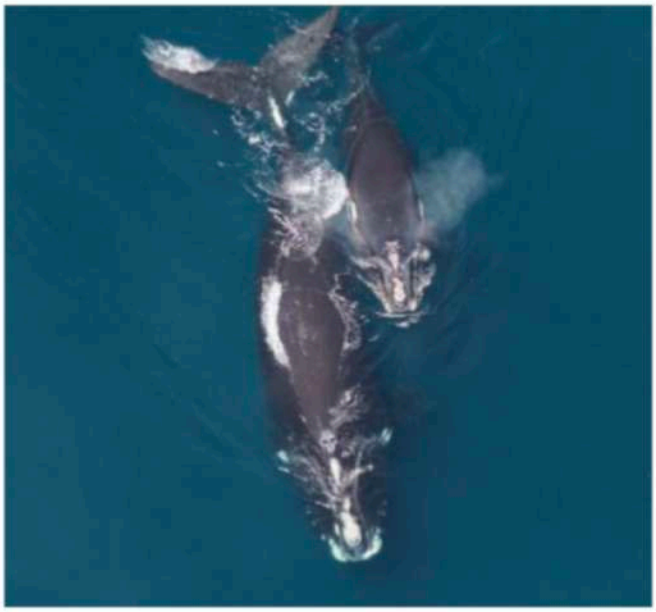
[Toshev, Szegedy 2014]



[Ciresan et al. 2013]

[Sermanet et al. 2011]
[Ciresan et al.]

Where do they exist in real world ?



Whale recognition, Kaggle Challenge

Mnih and Hinton, 2010

Where do they exist in real world ?

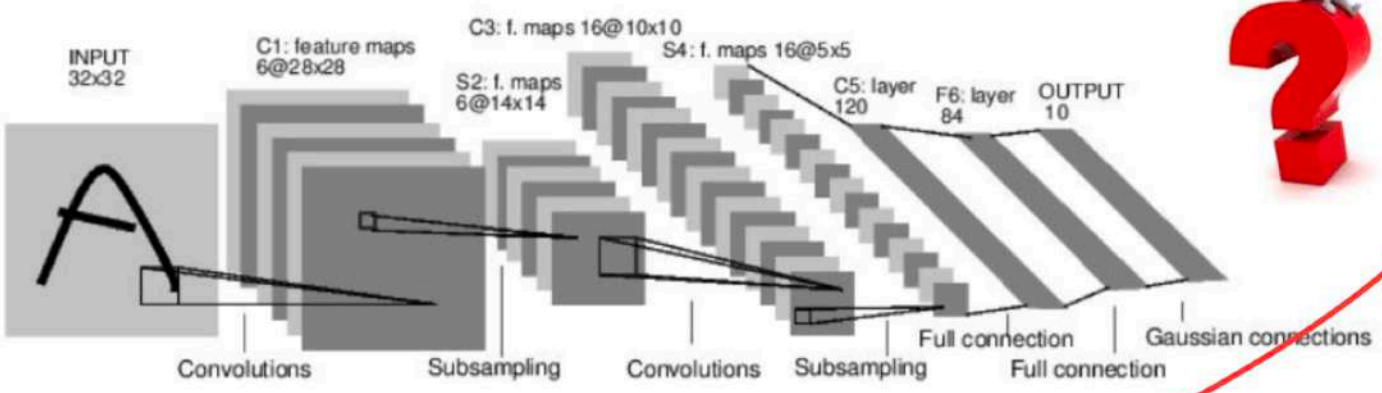
Image Captioning

[Vinyals et al., 2015]

Where do they exist in real world ?

reddit.com/r/deepdream

So, Convolution Nets...



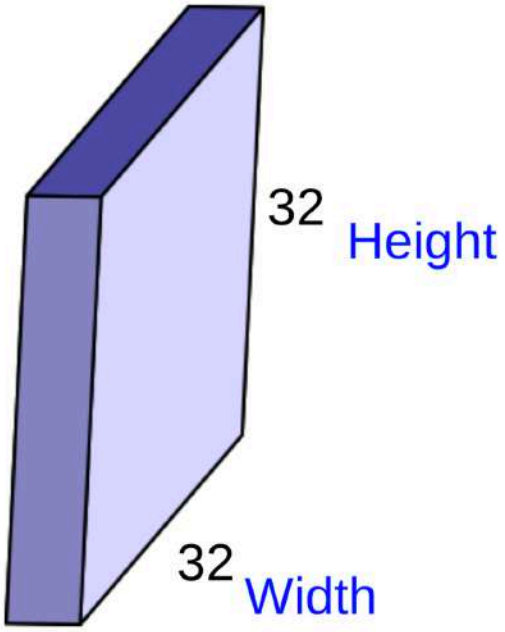
How do they work !

CNNs / ConvNets

- CNNs are
  - Similar to Ordinary neural networks.
  - Same concepts of weights and biases.
  - Neurons receive inputs and preform dot product, then followed by a non linearity.
- **So what does change ?**
  - Explicit assumption that the **inputs are IMAGES.**
- **Advantage ?**
  - Vast reduction in the amount of parameters in the network.

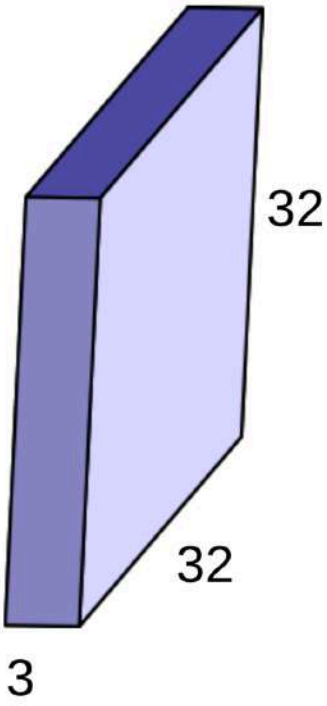
Convolution Layer

32 x 32 x 3 Image

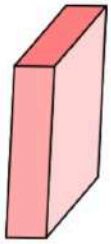


Convolution Layer

32 x 32 x 3 Image



5 x 5 x 3 Filter



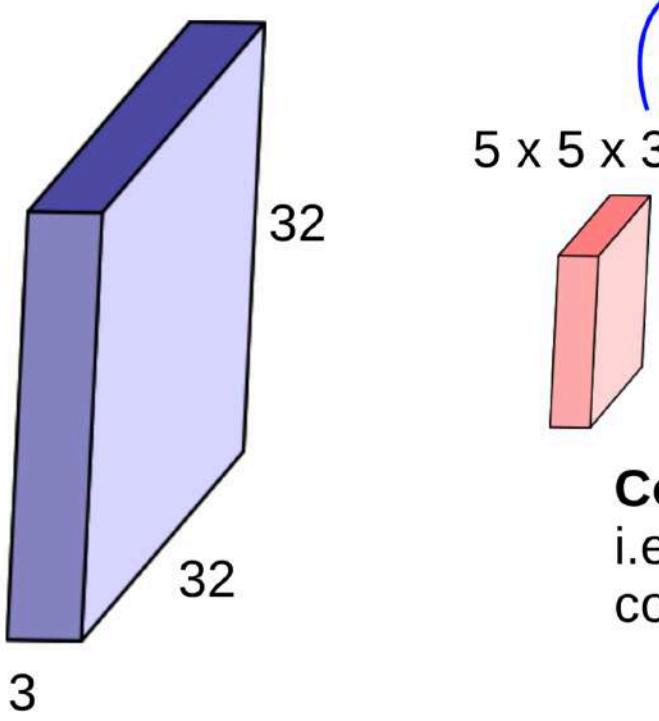
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

32 x 32 x 3 Image

Filters always extend the full depth of the input volume

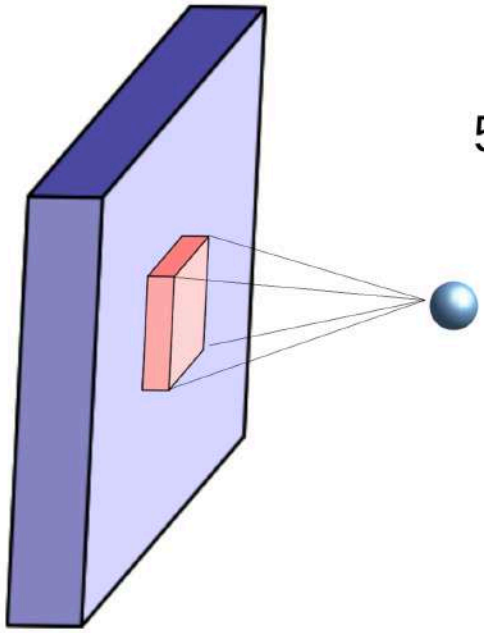
5 x 5 x 3 Filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

32 x 32 x 3 Image

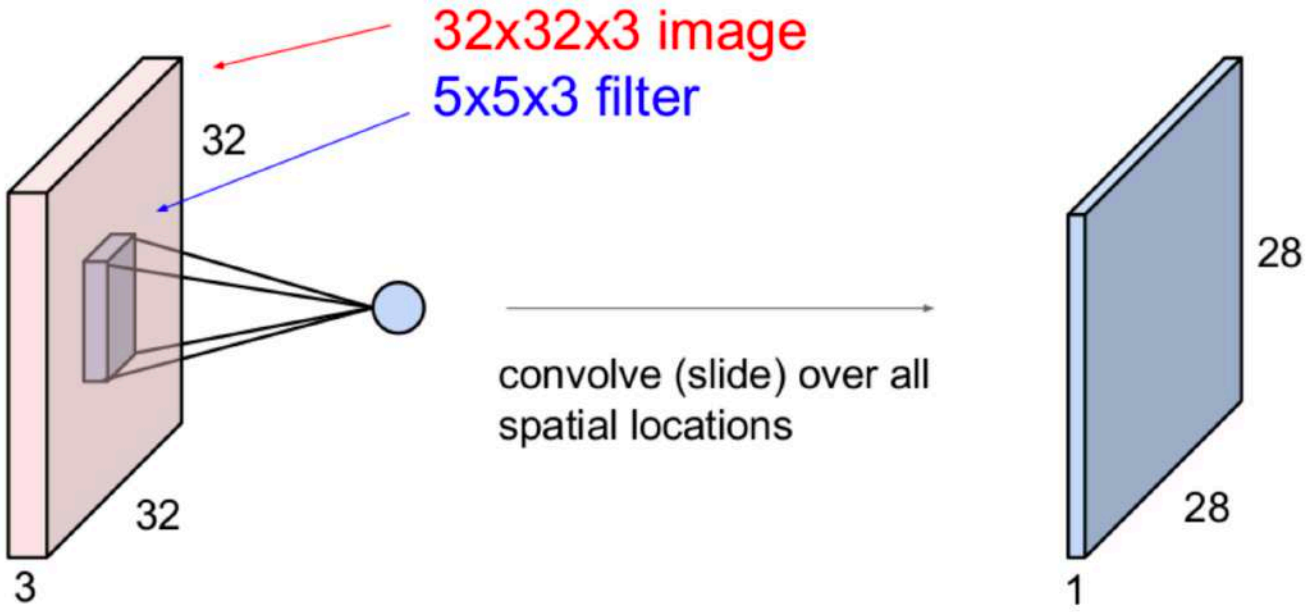


5 x 5 x 3 Filter

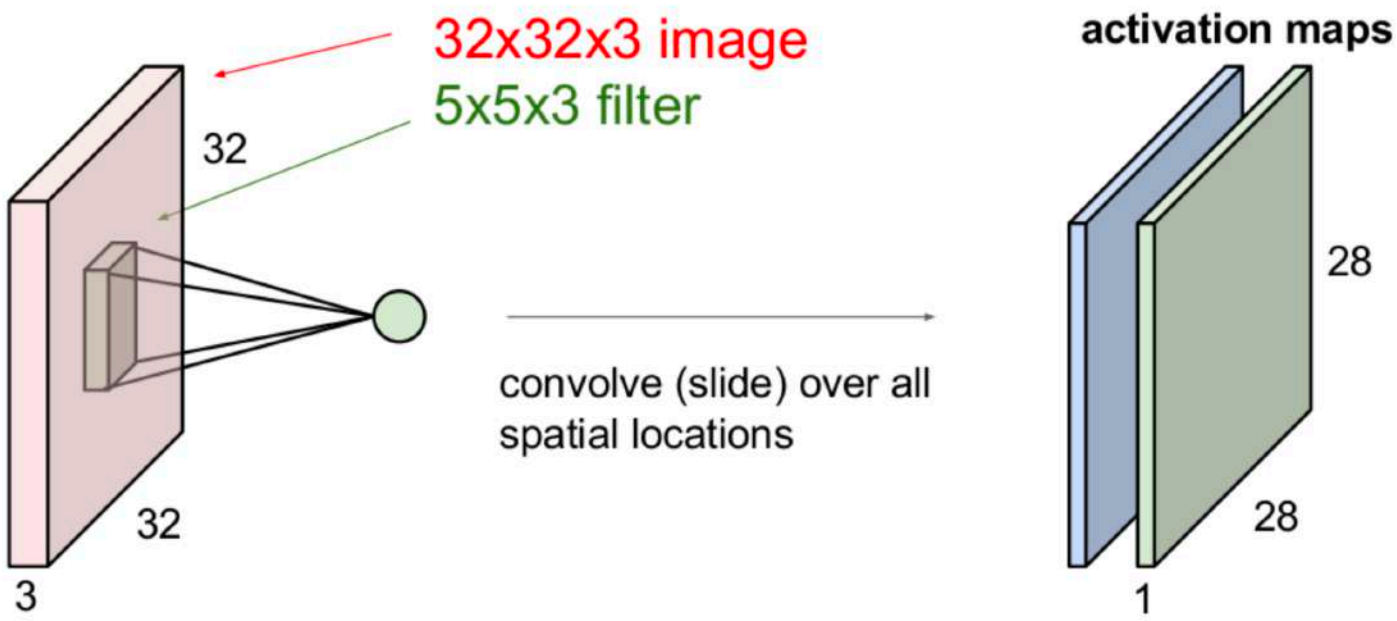
1 number: $w^T x + b$

The result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

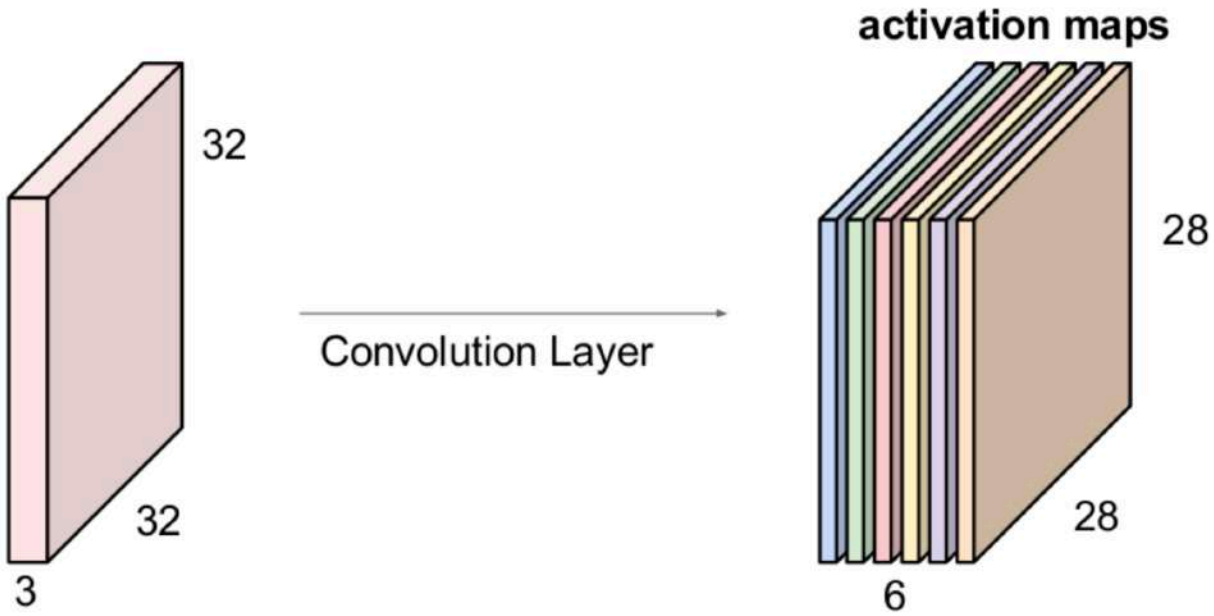
Convolution Layer



Convolution Layer

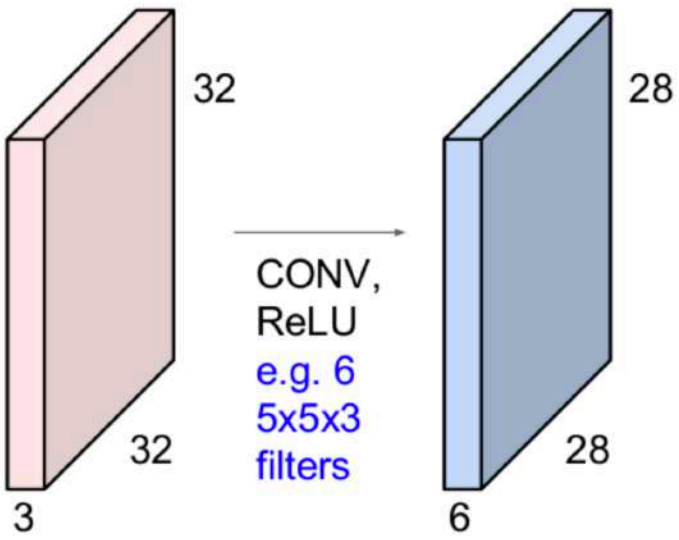


Convolution Layer

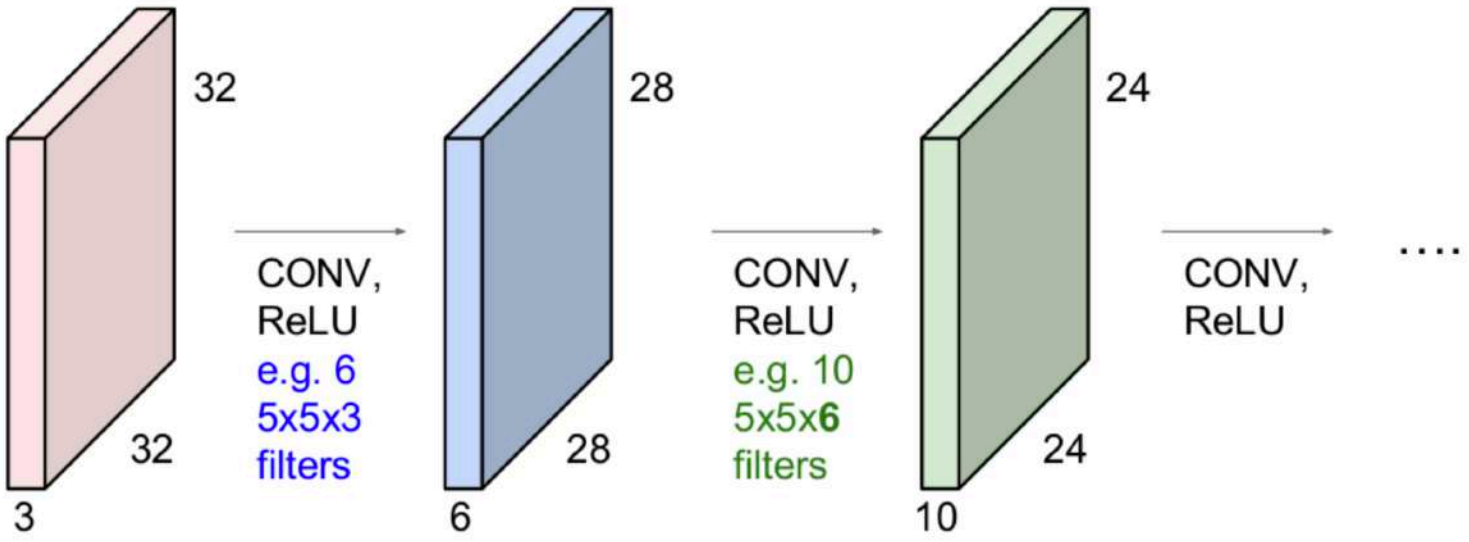


We stack these up to get a “new image” of size 28x28x6!

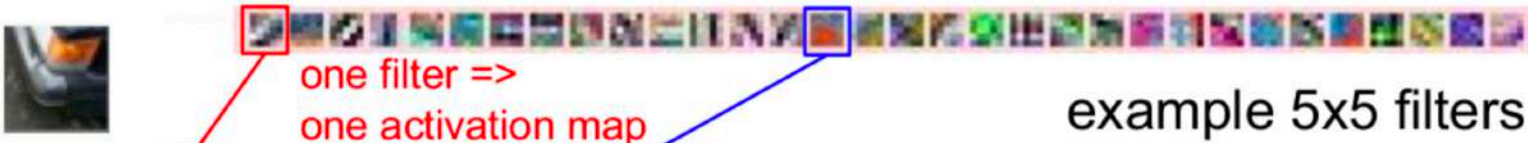
Convolution Layer



Convolution Layer

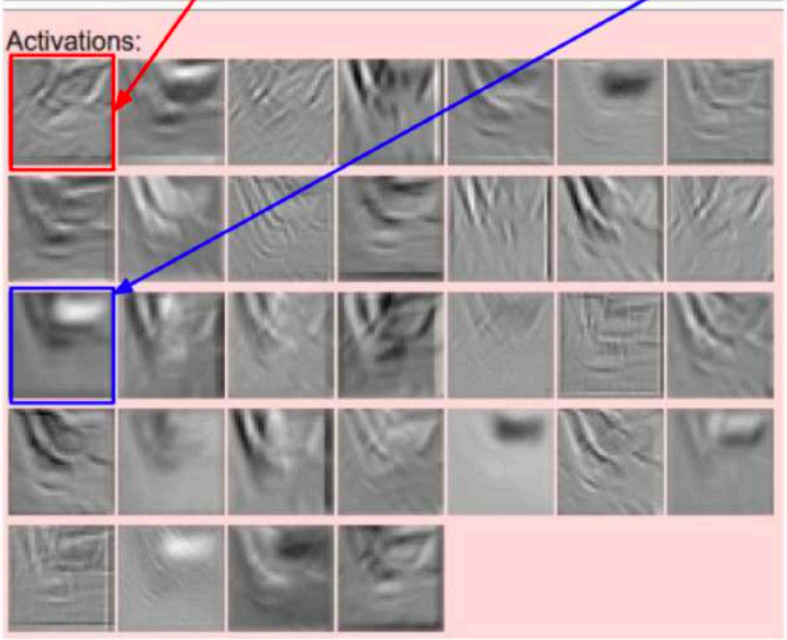


Convolution Layer



one filter =>
one activation map

example 5x5 filters
(32 total)

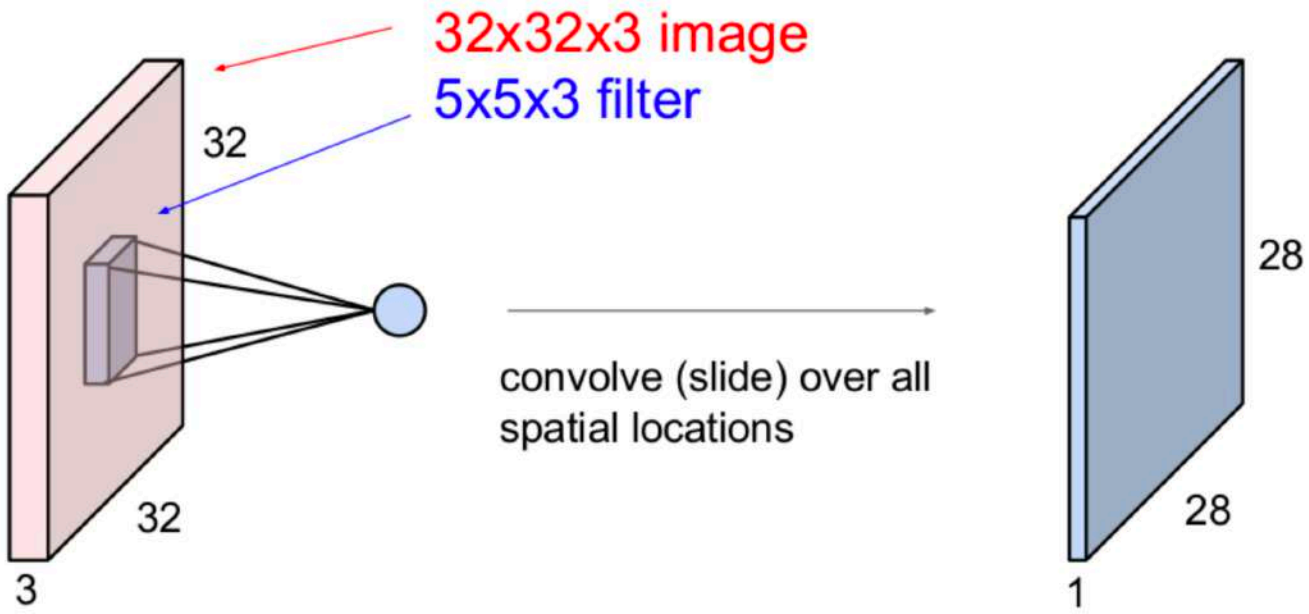


We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

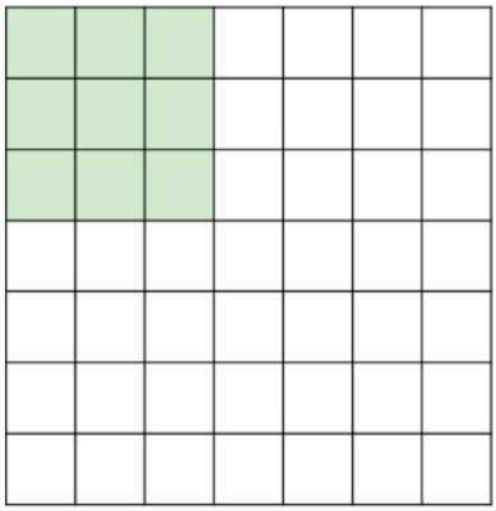
A closer look at spatial dimenssions



A closer look at spatial dimenssions

7

with stride 1



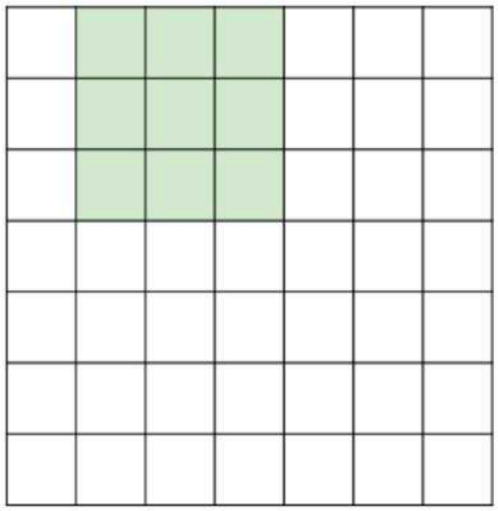
7x7 input
(spatially)
assume 3x3 filter

7

A closer look at spatial dimenssions

7

with stride 1



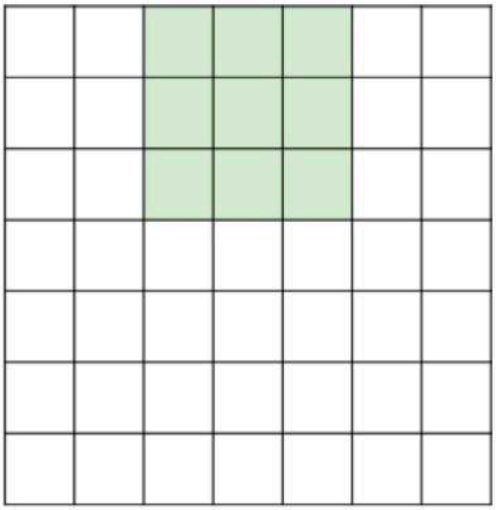
7x7 input
(spatially)
assume 3x3 filter

7

A closer look at spatial dimenssions

7

with stride 1



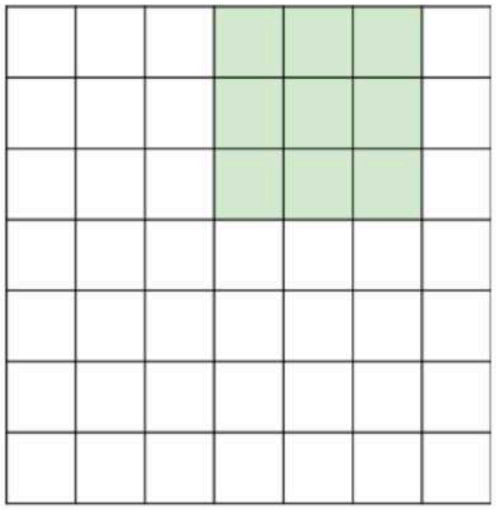
7x7 input
(spatially)
assume 3x3 filter

7

A closer look at spatial dimenssions

7

with stride 1



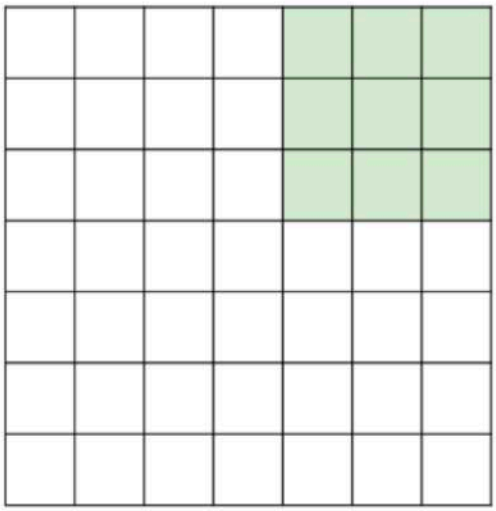
7x7 input
(spatially)
assume 3x3 filter

7

A closer look at spatial dimenssions

7

with stride 1



7x7 input
(spatially)
assume 3x3 filter

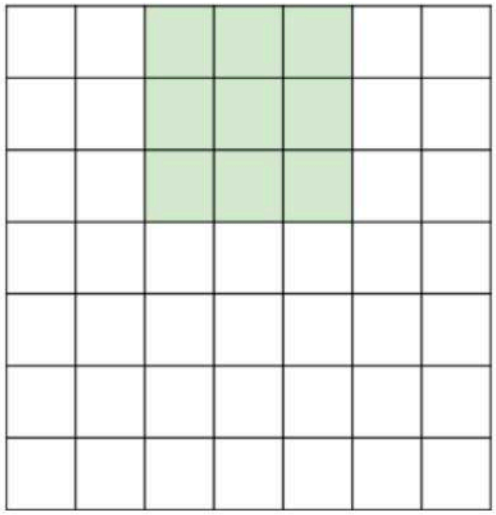
7

=> 5x5 output

A closer look at spatial dimenssions

7

with stride 2



7x7 input
(spatially)
assume 3x3 filter

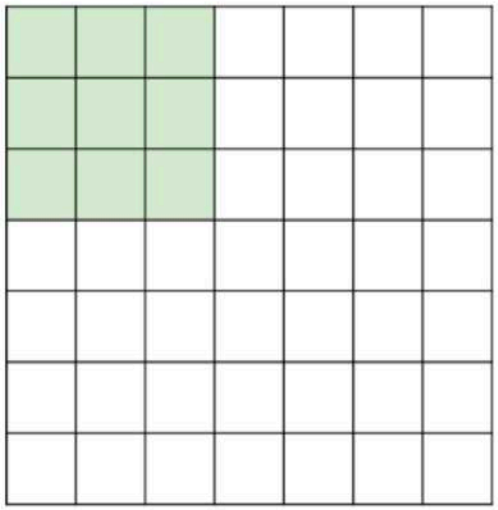
7

=> 3x3 output

A closer look at spatial dimenssions

7

with stride 3 **?**



7x7 input
(spatially)
assume 3x3 filter

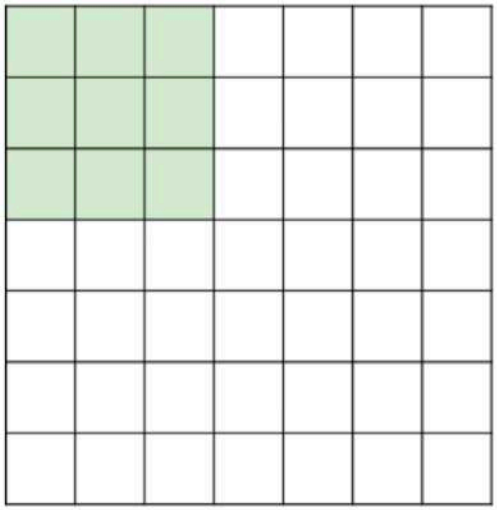
7

Output !!

A closer look at spatial dimenssions

with stride 3 **?**

7



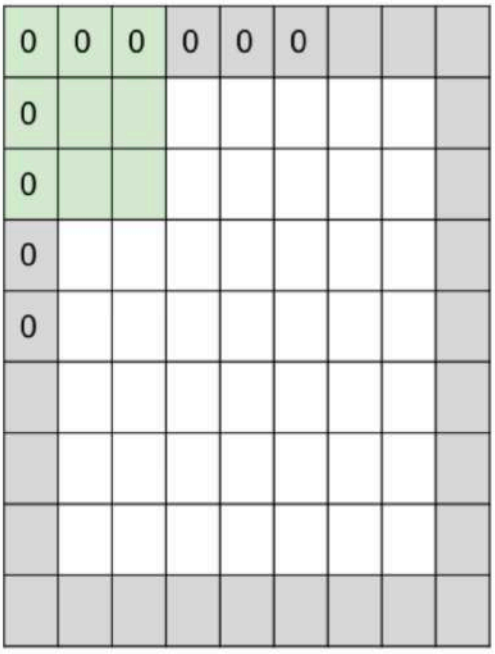
7x7 input
(spatially)
assume 3x3 filter

7

=> Doesn't FIT !!

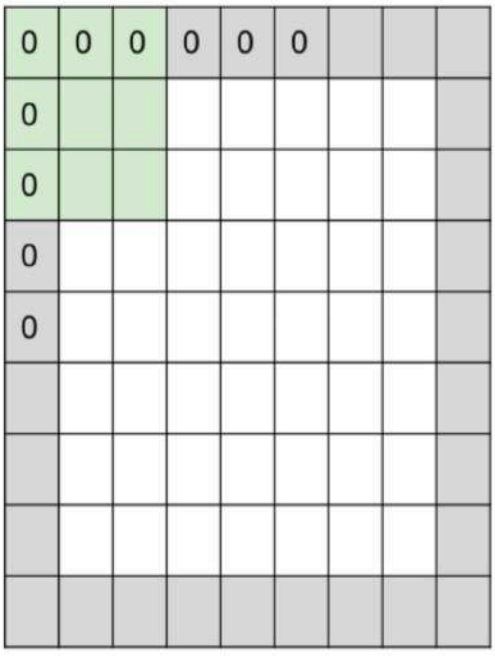
We can not apply 3x3 filter on 7x7 input with stride 3.

In Practice: zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

In Practice: zero pad the border

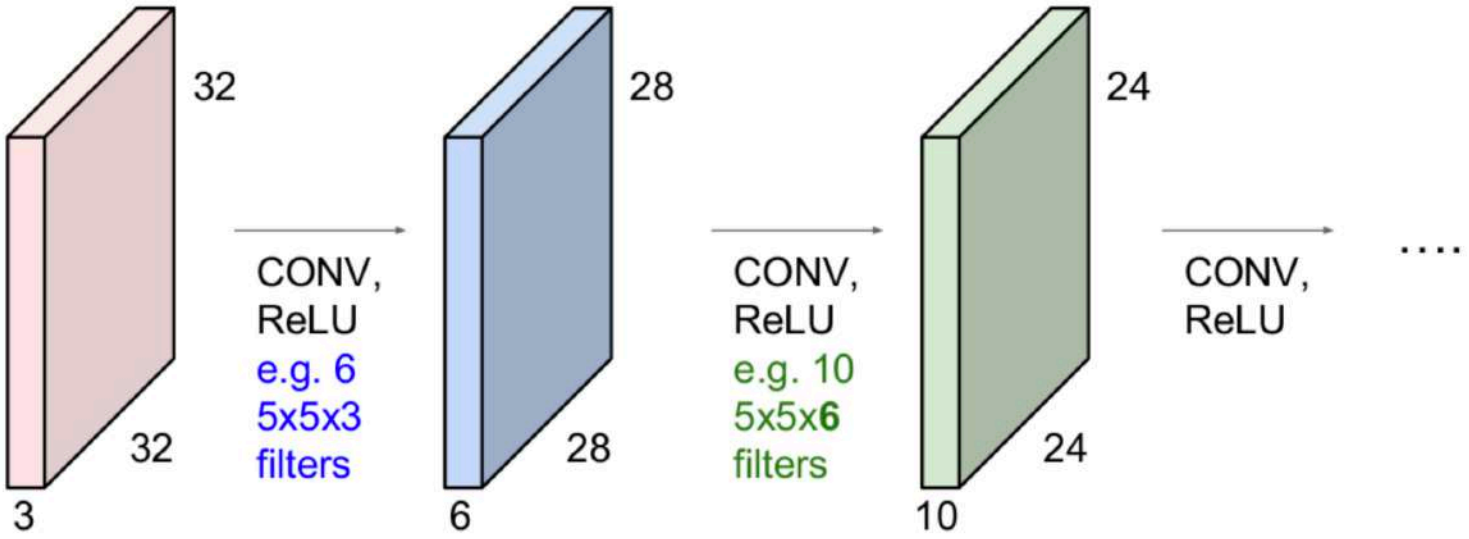


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Remember & Notice: Shrinking Volume

32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

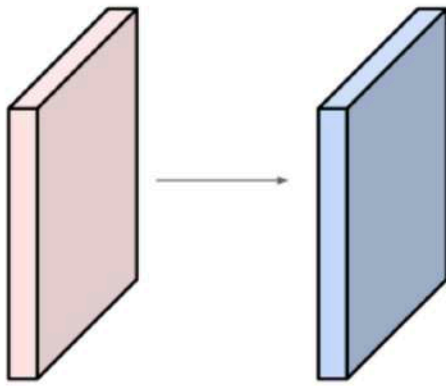


In Practice: zero pad the border

Examples time:

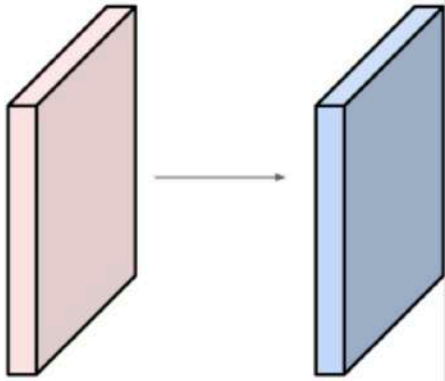
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?



In Practice: zero pad the border

Examples time:

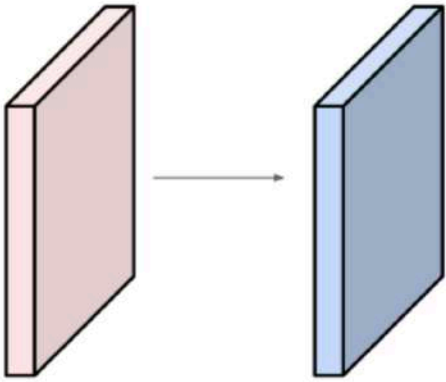


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

In Practice: zero pad the border

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

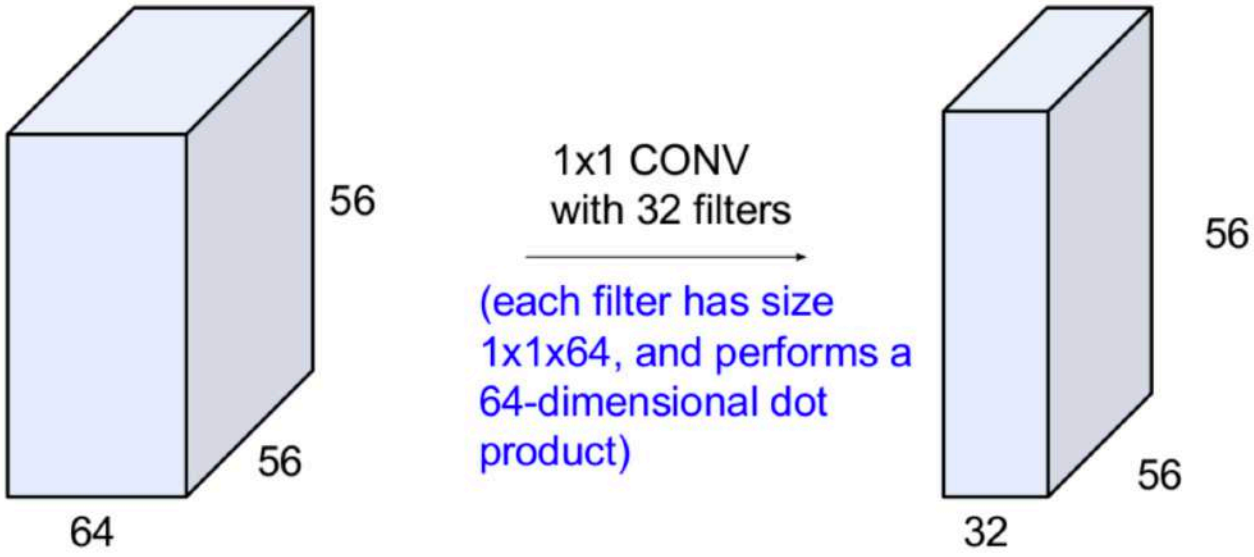
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Summary: Convolution Layer

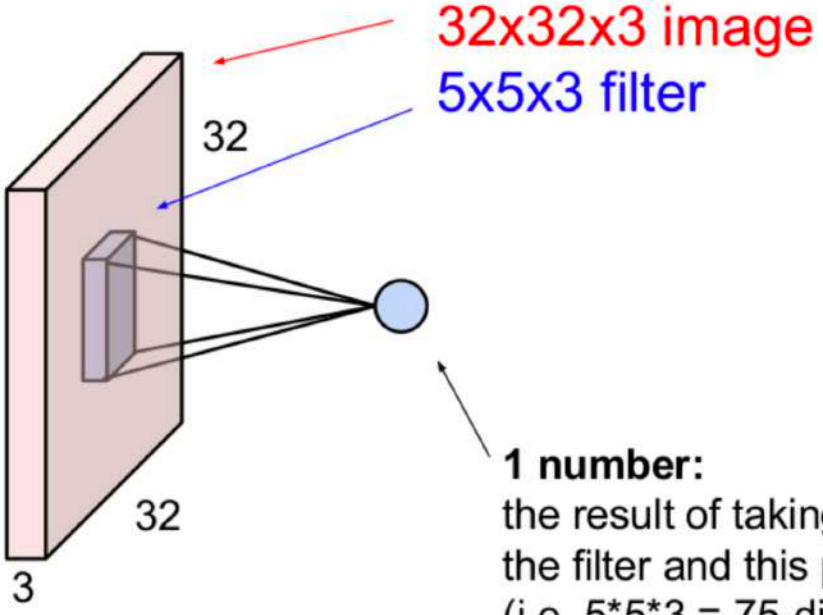
- Accepts a volume of size $W_1 * H_1 * D_1$
- Require four hyperparameters:
  - Number of filters **K**
  - The spatial extent **F**
  - The stride **S**
  - The amount of zero padding **P**
- Produce a volume of size $\mathbf{W_2 * H_2 * D_2}$, where
  - $\mathbf{W_2 = (W_1 - F + 2P)/S + 1}$
  - $\mathbf{W_2 = (W_1 - F + 2P)/S + 1}$
  - $\mathbf{D_2 = K}$
- With parameter sharing, introduces $\mathbf{F * F * D_1}$ weights per filter, for total of $\mathbf{(F * F * D_1) * K}$ weights and **K** biases.

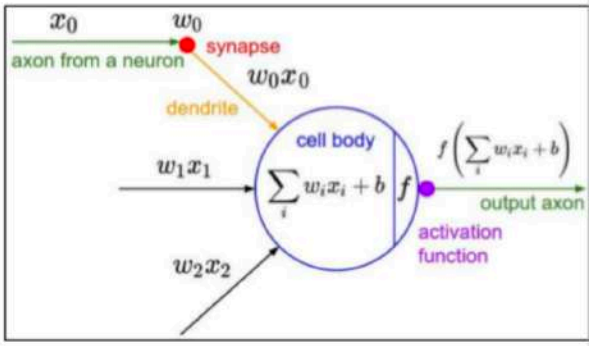
An analogy

(btw, 1x1 convolution layers make perfect sense)



Brain View of CONV Layer

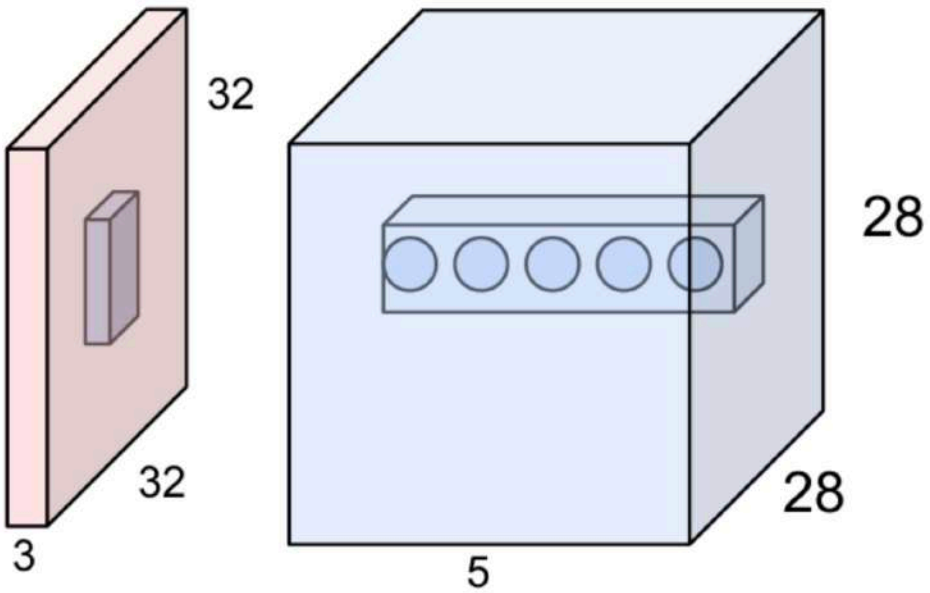




It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

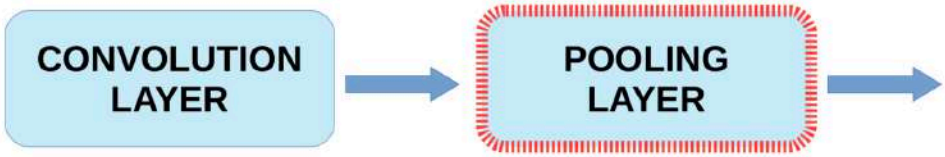
Brain View of CONV Layer



- E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)
- There will be 5 different neurons all looking at the same region in the input volume

Two more layers to go

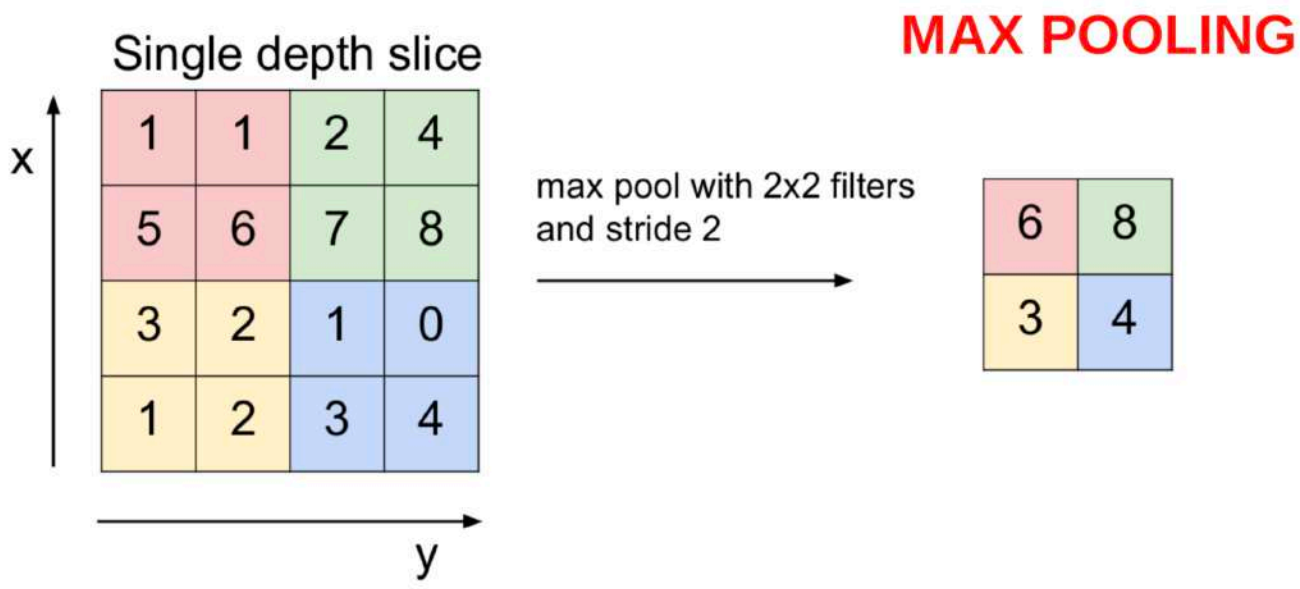
Two more layers to go



Two more layers to go

Pooling Layer

- Makes the representation smaller and more managable
- Operates over each avctivation map indiendently

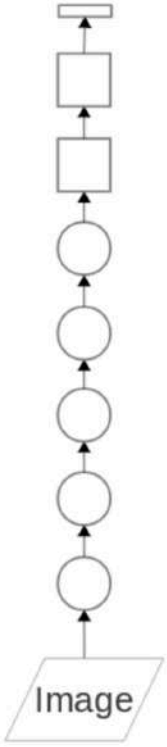


Summary: POOLING Layer

- Accepts a volume of size $\mathbf{W_1}$*$\mathbf{H_1}$*$\mathbf{D_1}$
- Requires three hyperparameter :
  - Their spetial extent **F**,
  - The stride **S**
- Produces a volume of size W_2*H_2*D_2
  - $\mathbf{W_2 = (W_1\text{-}F)/S+1}$
  - $\mathbf{H_2 = (H_1\text{-}F)/S+1}$
  - $\mathbf{D_2 = D_1}$
- Introduces ZERO parameters since it computes a fixed function of the input
- Note: Its not common to use zero padding for polling layer.

Case study: AlexNet

[Krizhevsky et al. 2012]



- **Deep** : 7 Hidden Layer
- Entirely Supervised Learning.
- 5 CONV LAYER + 2 FC LAYER
- 650,000 Neurons
- 60,000,000 parameters
- 630,000,000 connections
- Final feature layer: 4096-dimensional

Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

Thank You

-Deepak Kumar Pradhan
Research Engineer | Educator
www.deepakpradhan.in