VIRGINIA TECH

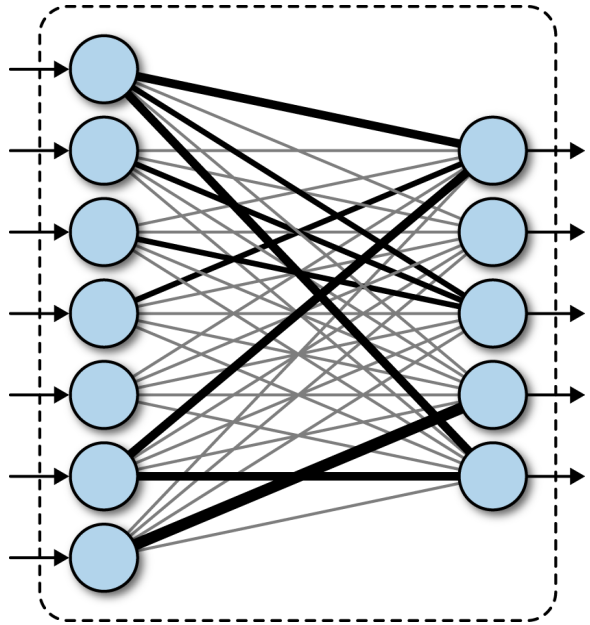
Convolutional Neural Networks

Lifu Huang

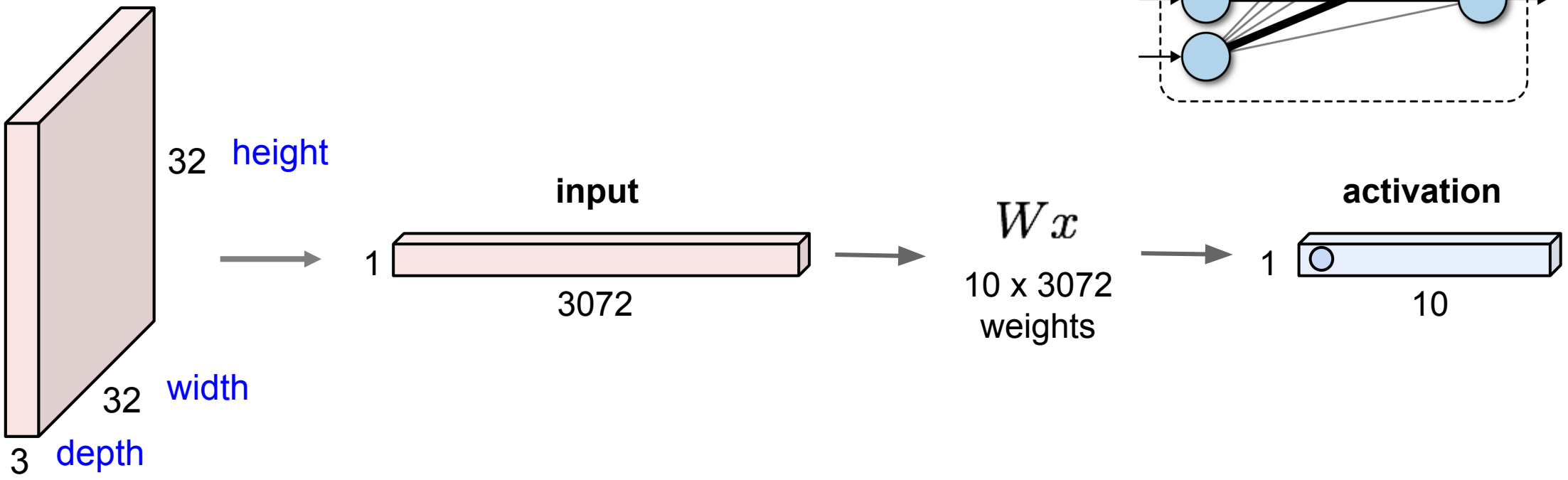
lifuh@vt.edu

Torgersen Hall, Suite 3160E

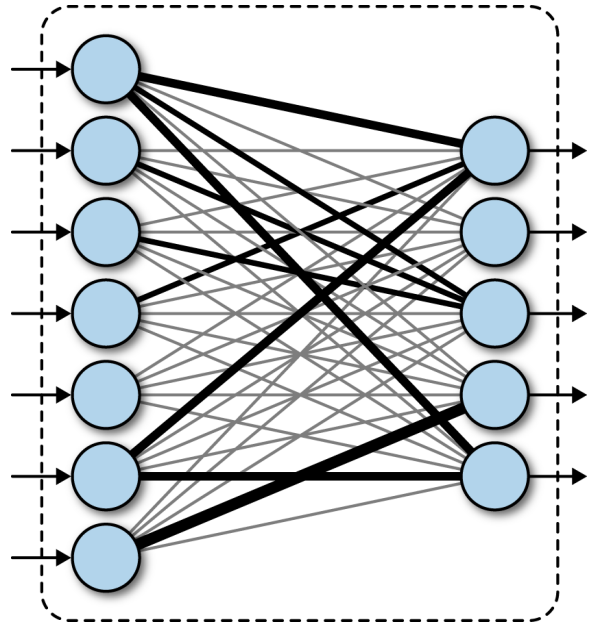
Fully Connected Layer



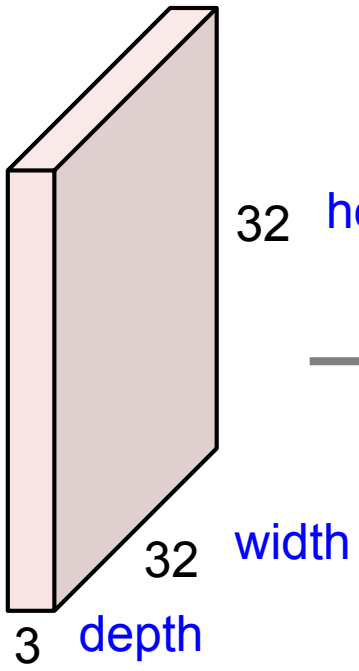
32x32x3 image -> stretch to 3072 x 1

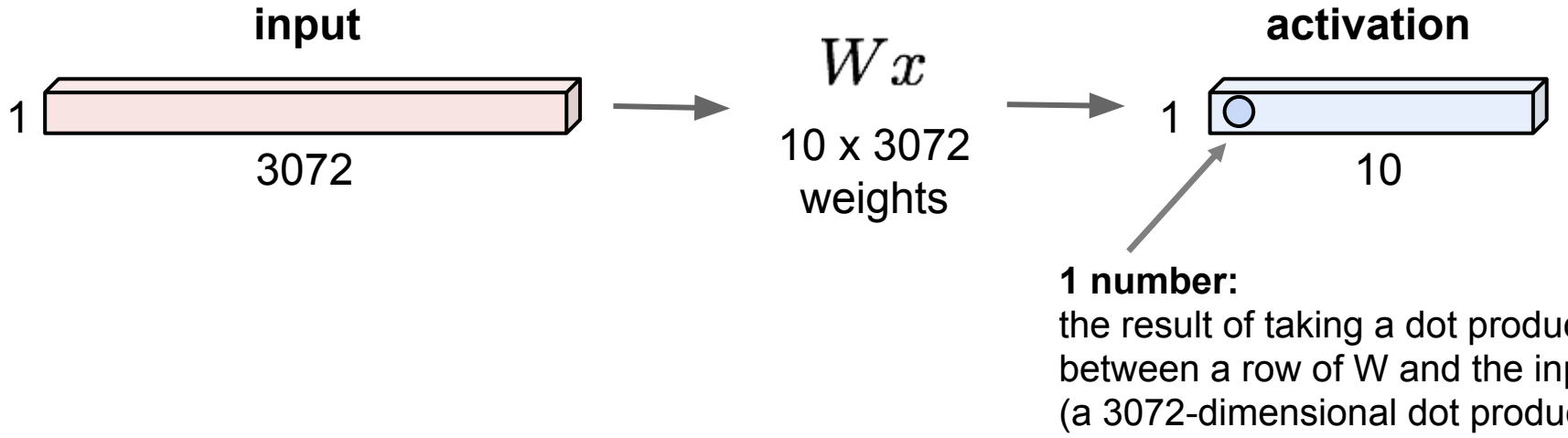


Fully Connected Layer



32x32x3 image -> stretch to 3072 x 1

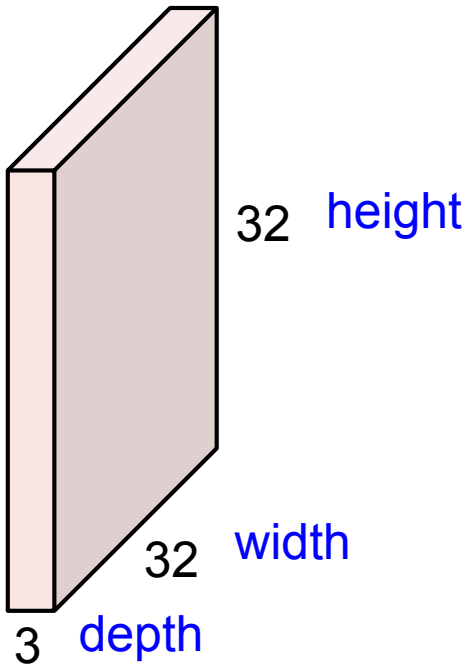




1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

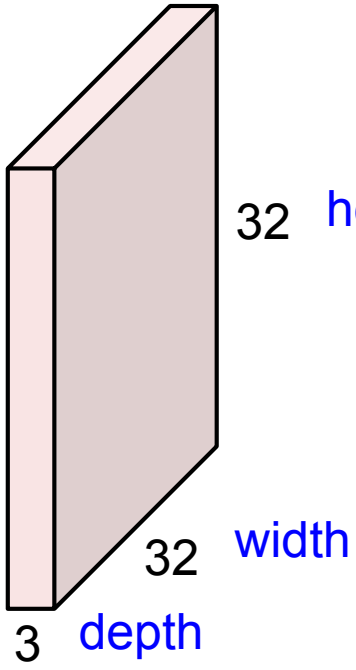
Convolution Layer

32x32x3 image -> preserve spatial structure

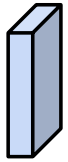


Convolution Layer

32x32x3 image



5x5x3 filter

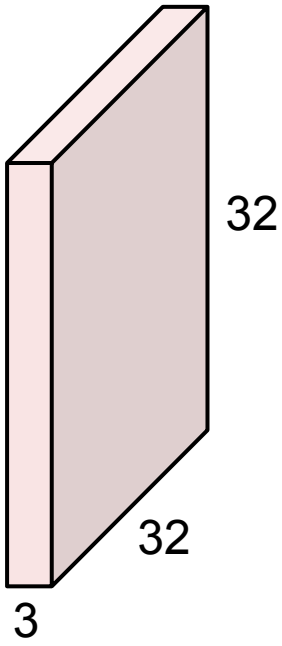


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

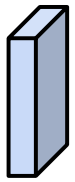
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

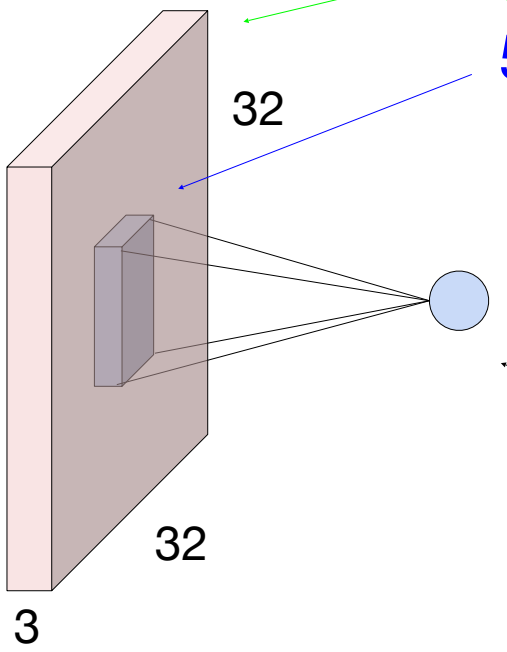


5x5x3 filter



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

Convolution Layer



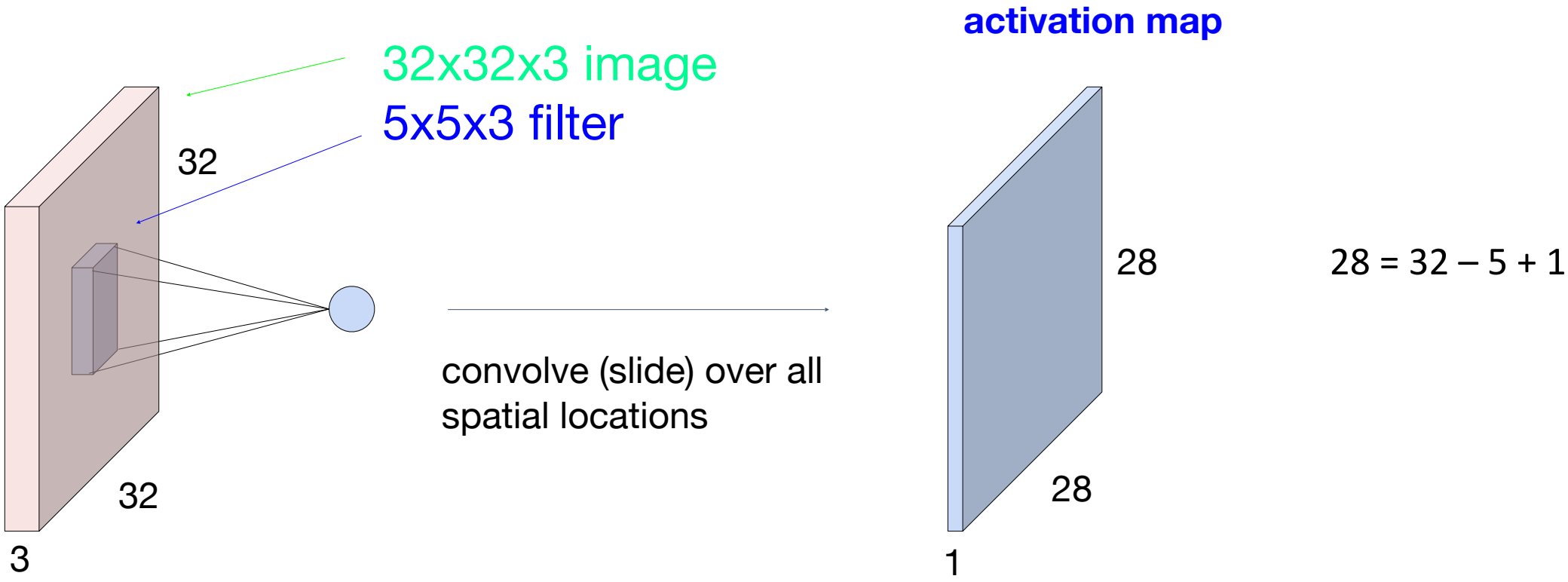
32x32x3 image

5x5x3 filter w

1 number:
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)

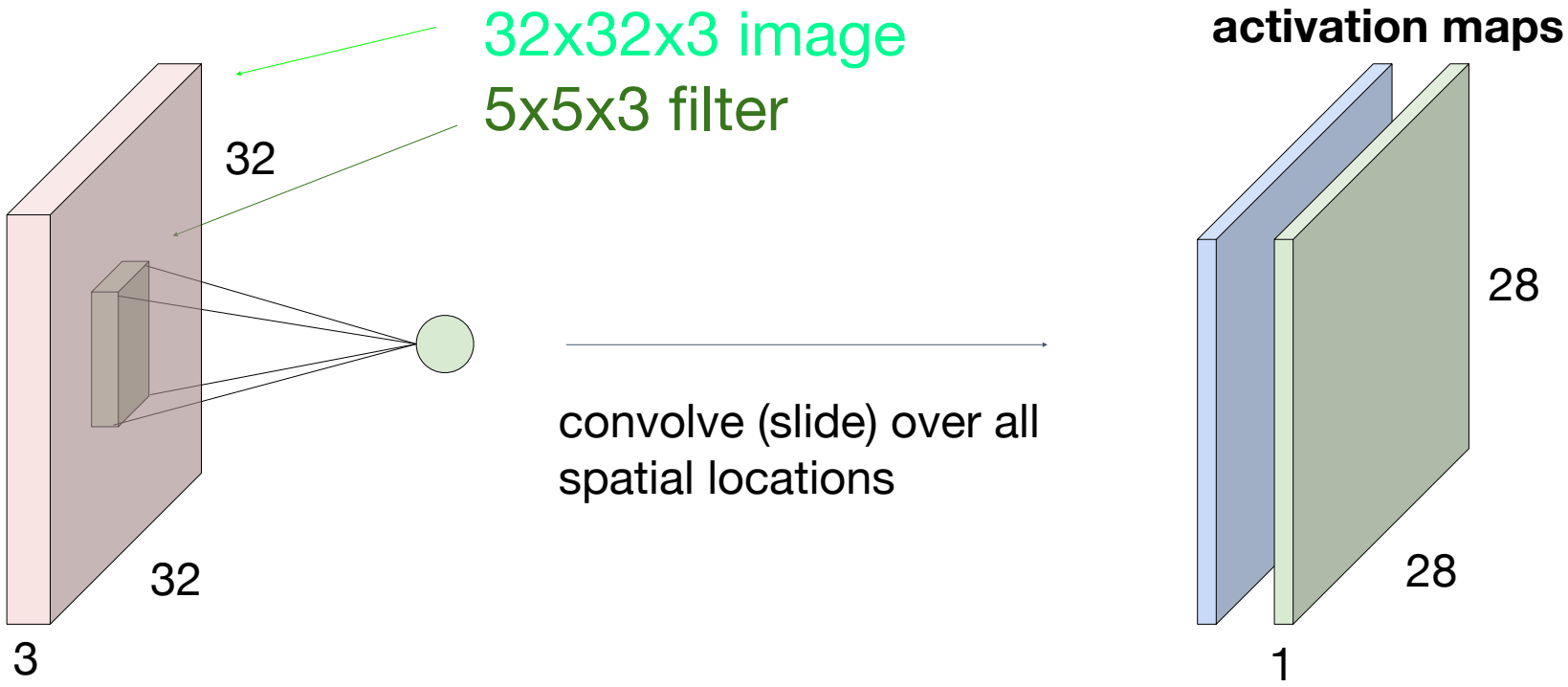
$$w^T x + b$$

Convolution Layer



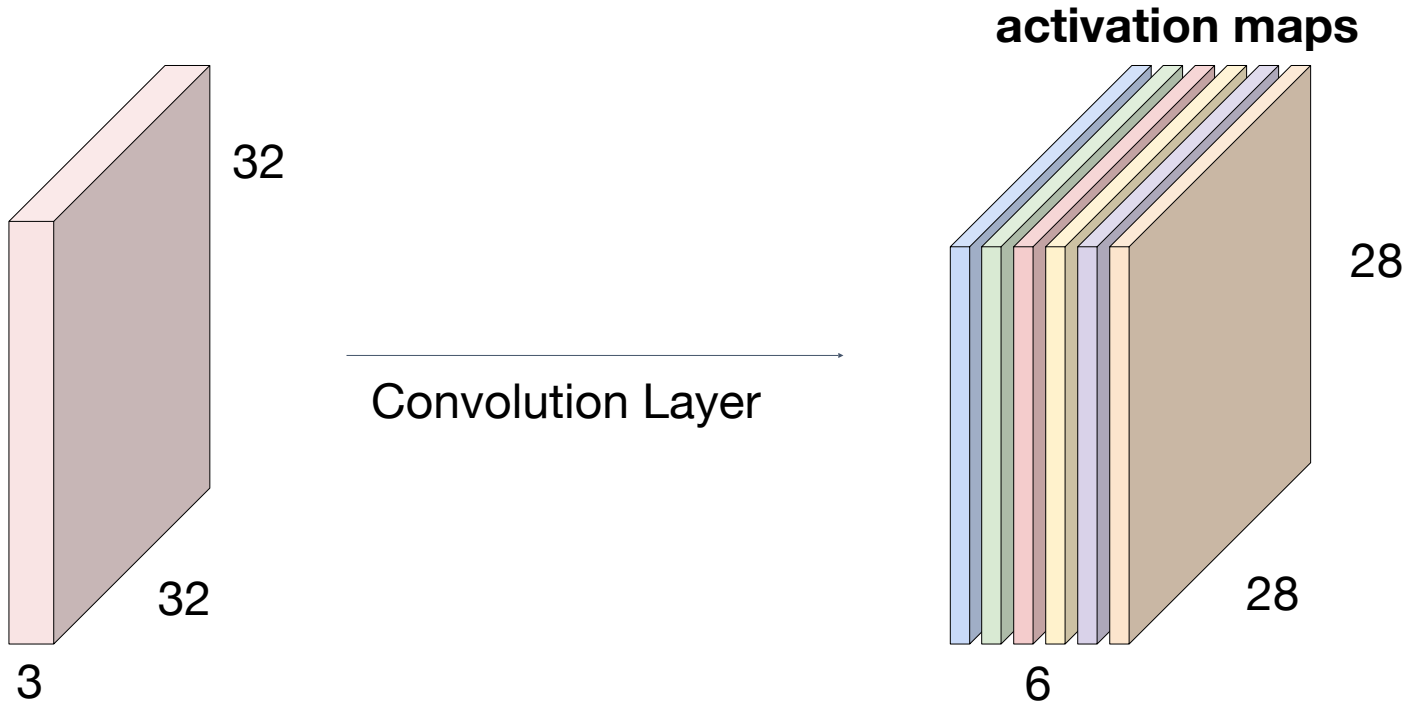
Convolution Layer

consider a second, green filter



Convolution Layer

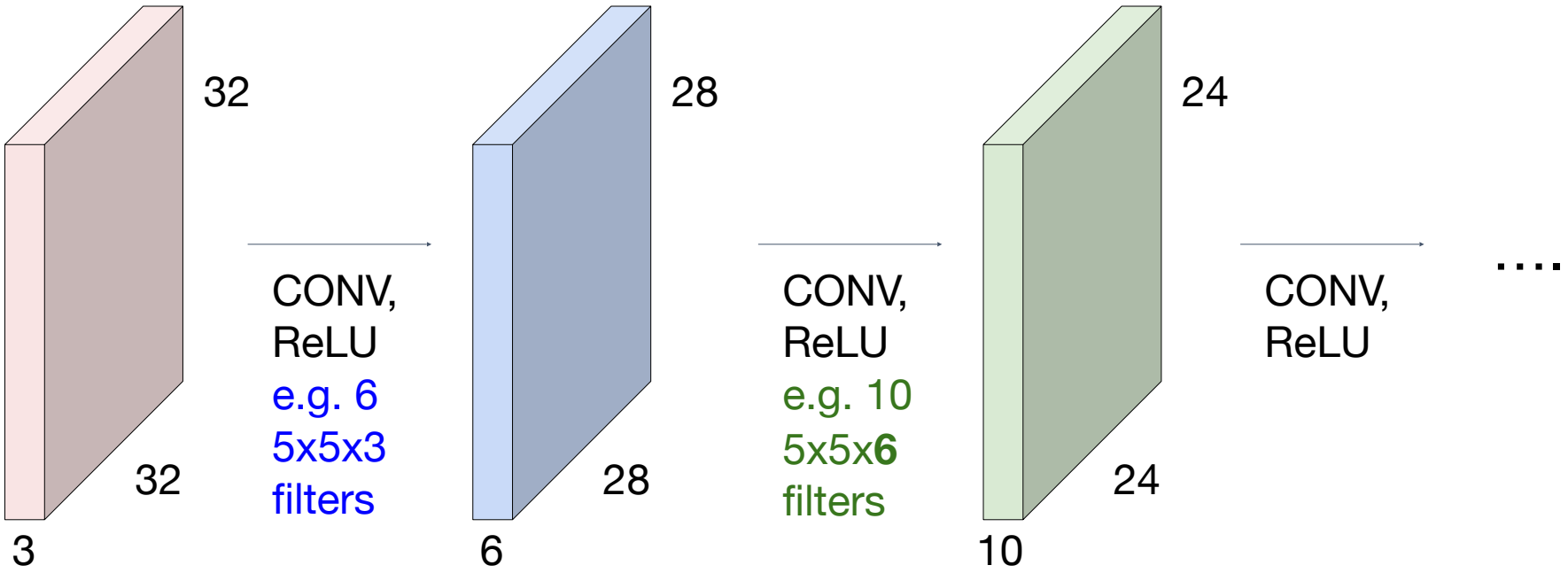
For example, if we had 6 5×5 filters, we will get 6 separate activation maps:



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

Convolutional Neural Networks

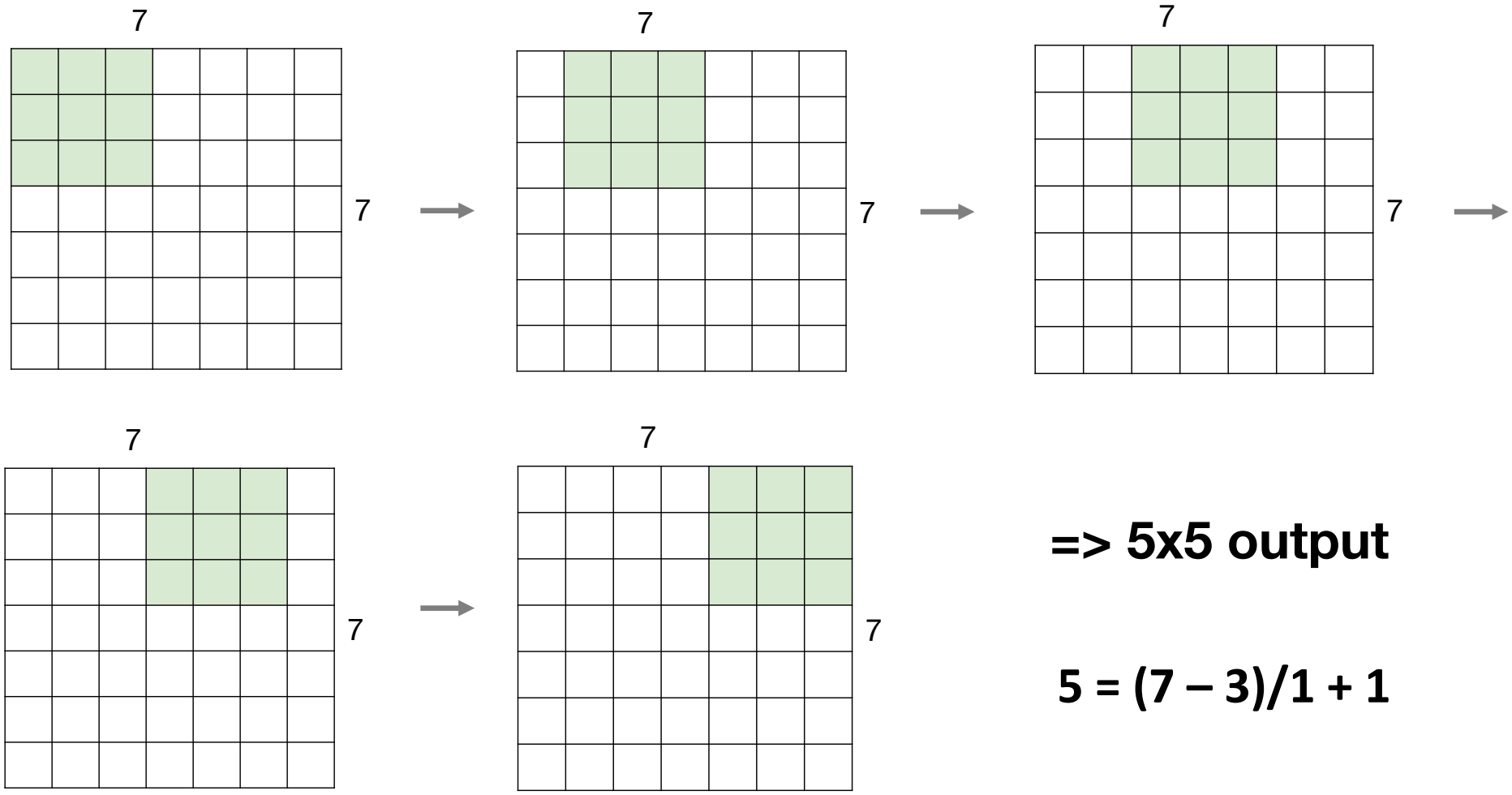
- ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



A Closer Look at Spatial Dimensions

A Closer Look at Spatial Dimensions:

- 7×7 input (spatially), assume 3×3 filter

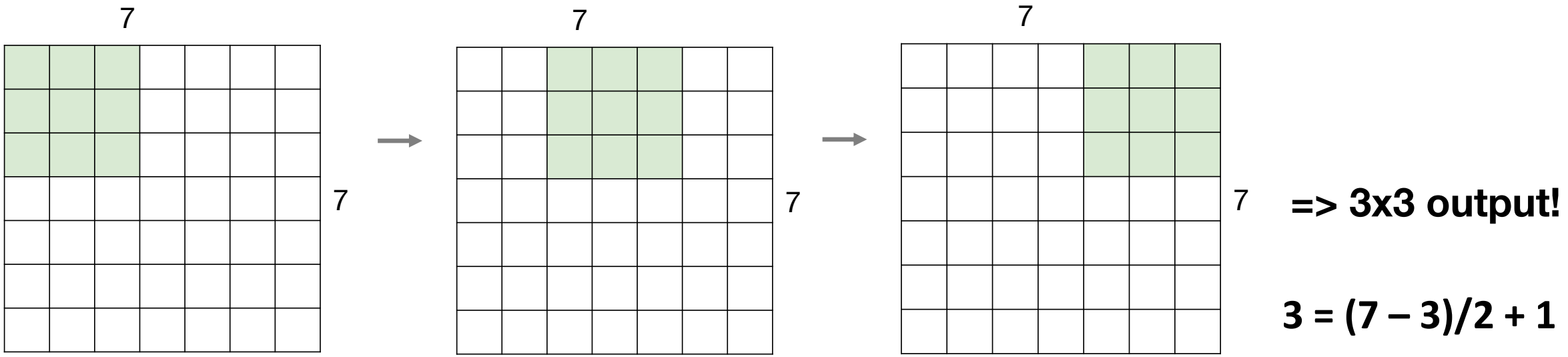


=> 5x5 output

5 = (7 – 3)/1 + 1

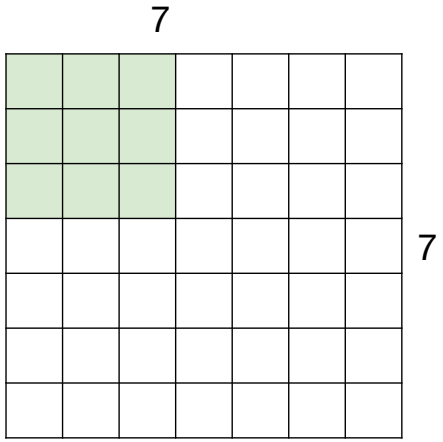
A Closer Look at Spatial Dimensions

- 7×7 input (spatially), assume 3×3 filter, apply with stride 2

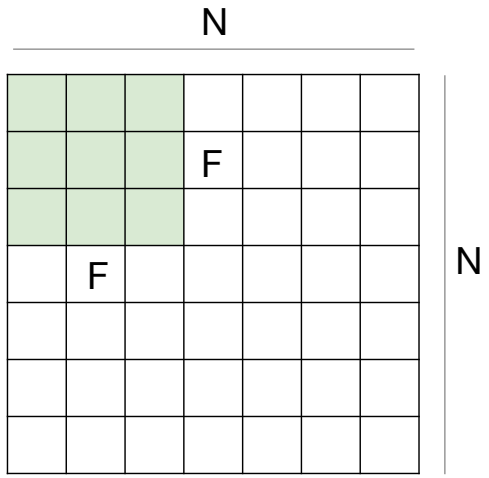


A Closer Look at Spatial Dimensions

- 7×7 input (spatially), assume 3×3 filter, apply with stride 3?



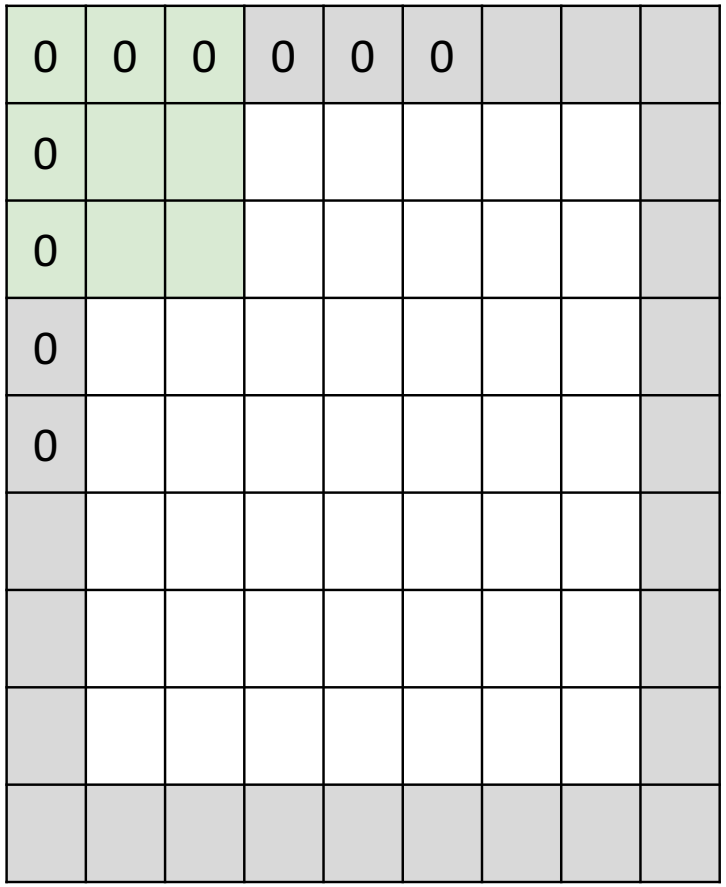
Doesn't fit!!
cannot apply 3×3 filter on 7×7 input with stride 3



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

In Practice: Common to Zero Pad the Border

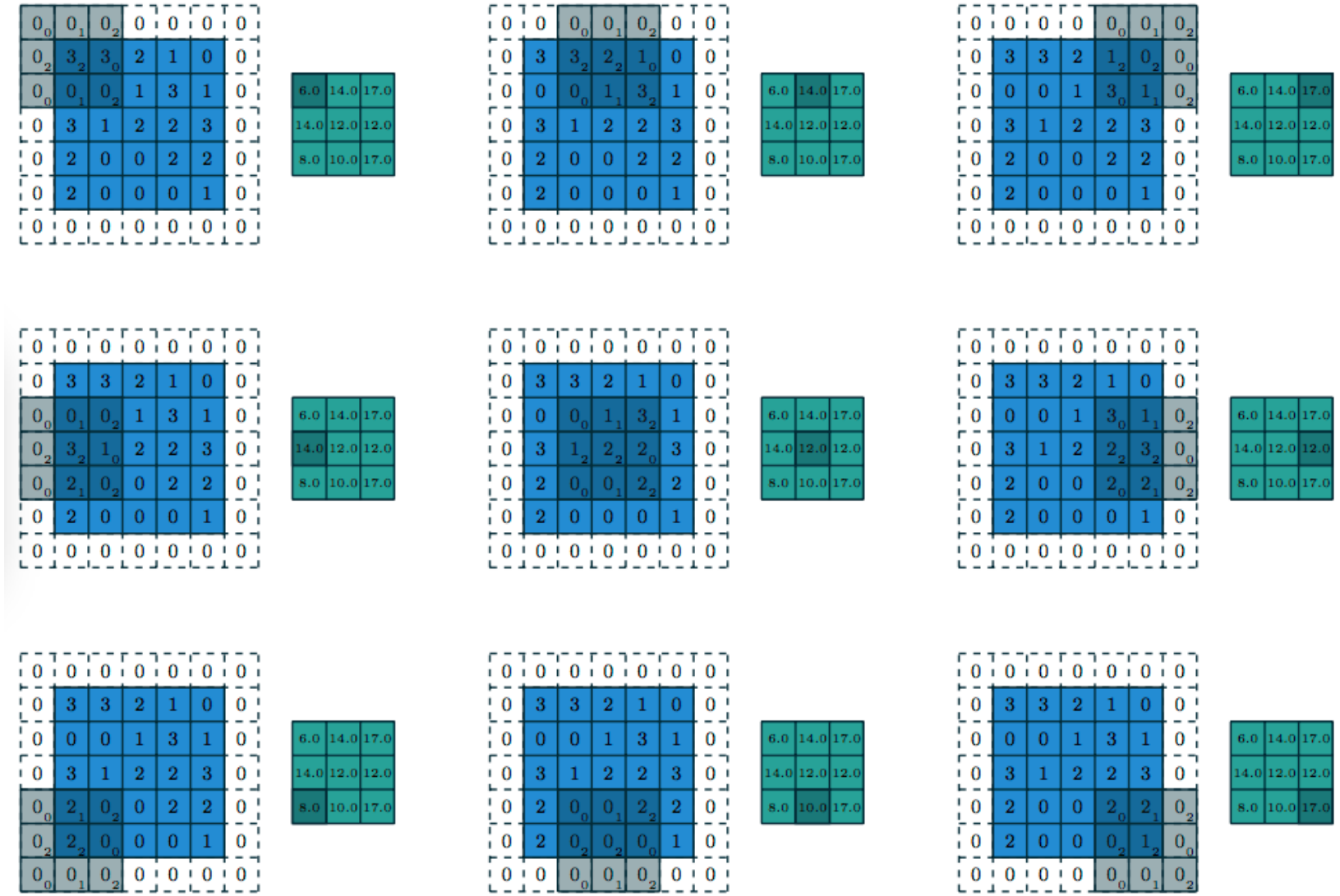


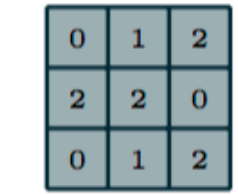
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output! (N - F) / stride + 1

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

Computing the output values of a 2D discrete convolution





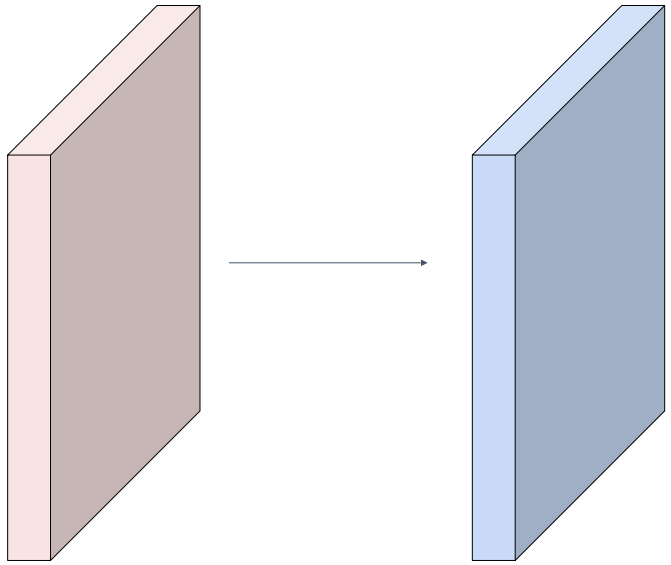
$$f = Wx$$

Image credit: Vincent Dumoulin and Francesco Visin

Examples

(N + 2*P - F) / stride + 1

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



Output volume size: ?

(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Summary

Summary. To summarize, the Conv Layer:

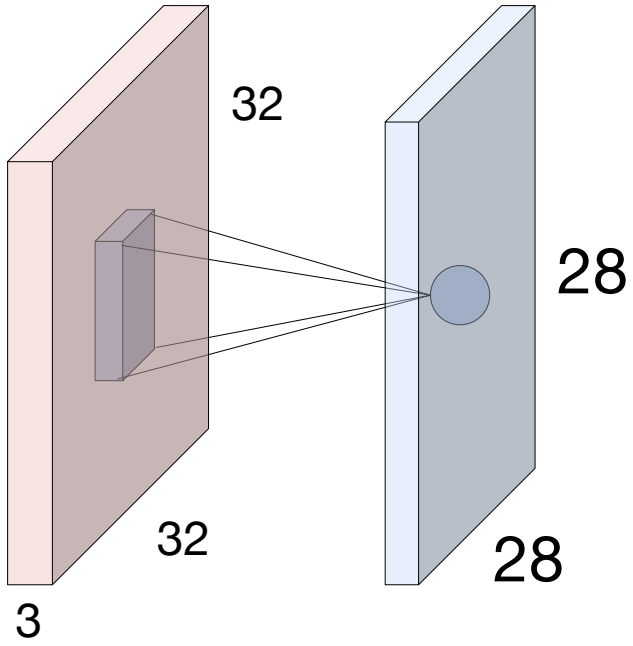
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

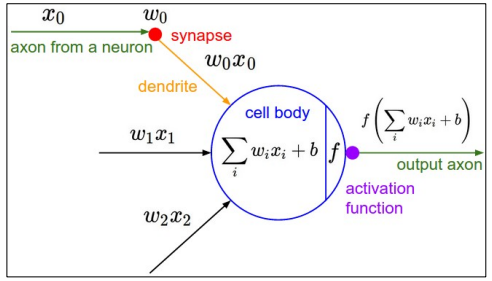
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

The Brain/Neuron View of CONV Layer





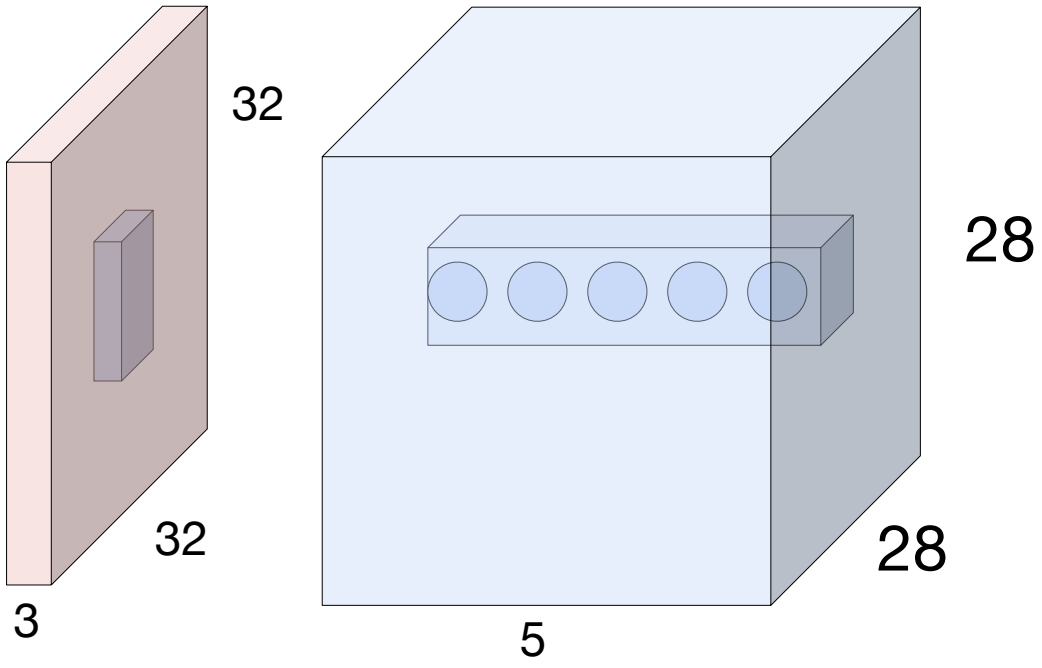
Neuron

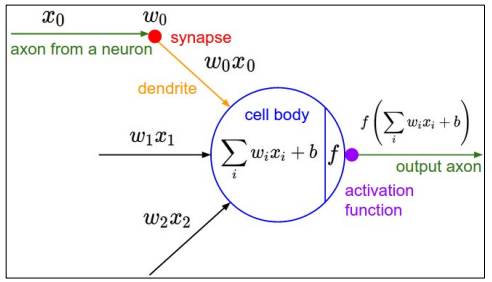
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The Brain/Neuron View of CONV Layer

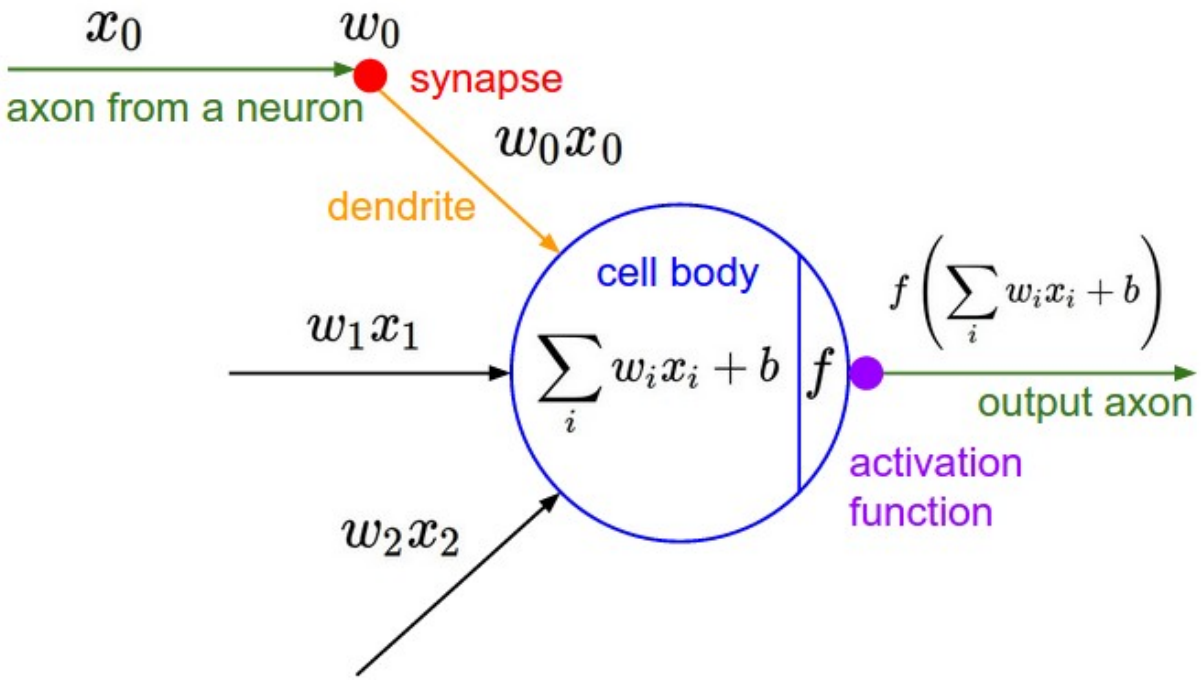




E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

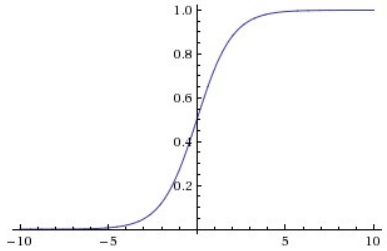
There will be 5 different neurons all looking at the same region in the input volume

Activation Functions

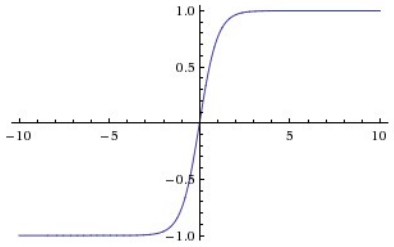


Sigmoid

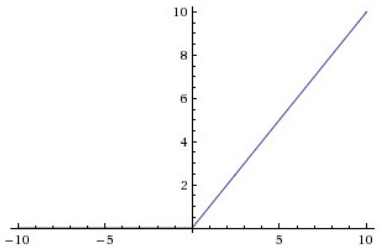
$$\sigma(x) = 1/(1 + e^{-x})$$



tanh tanh(x)

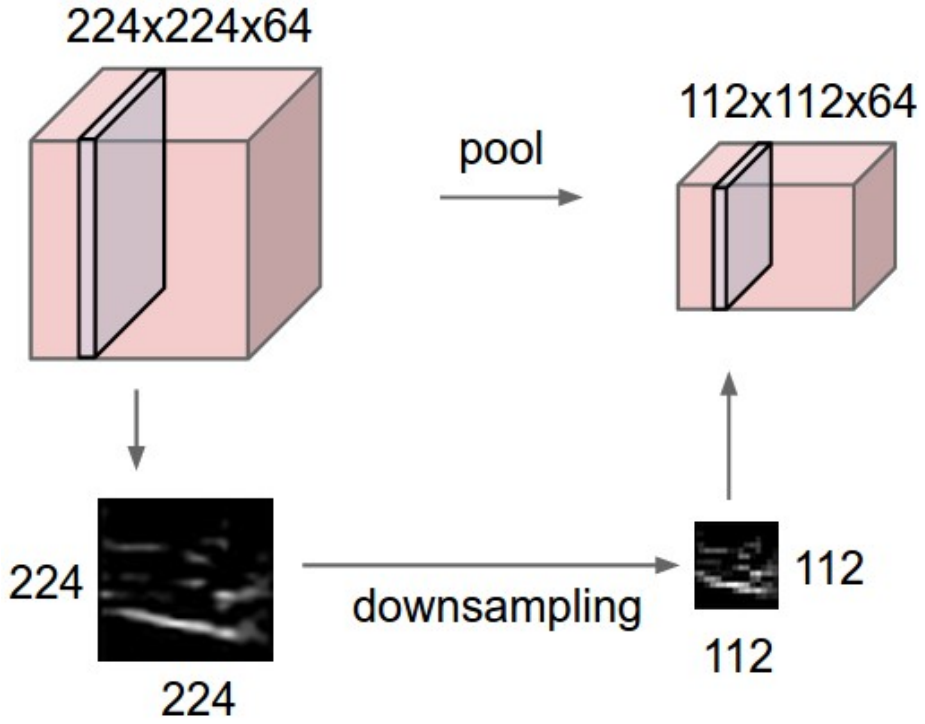


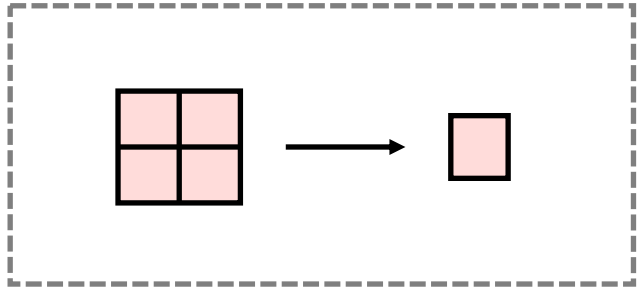
ReLU max(0,x)



Pooling Layer

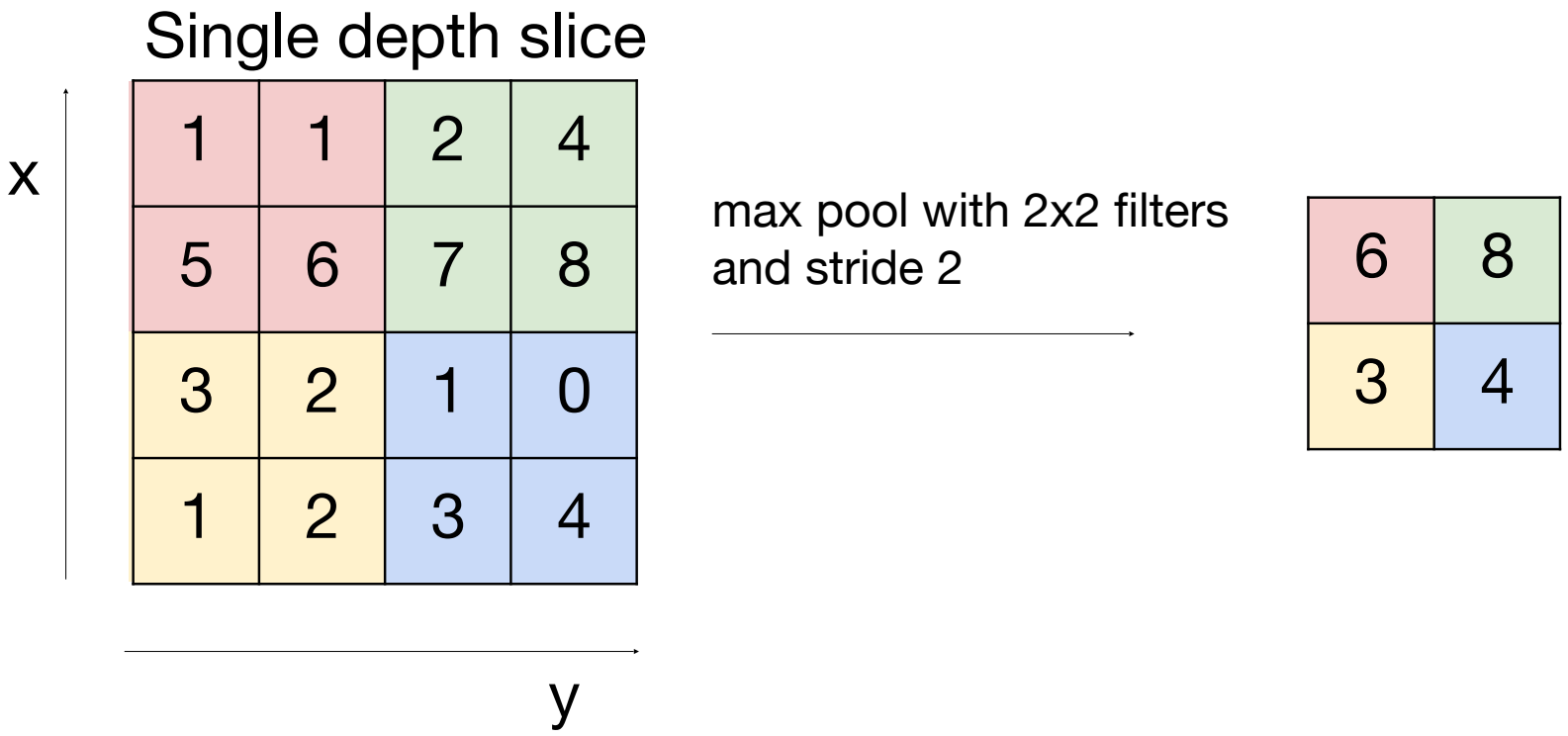
- Makes the representations smaller and more manageable
- operates over each activation map independently





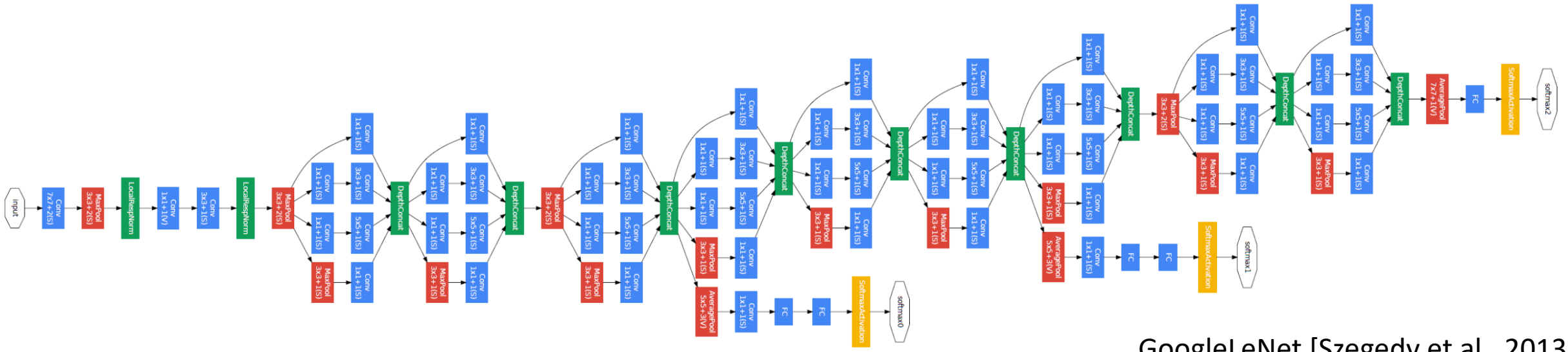
Pooling

Max Pooling

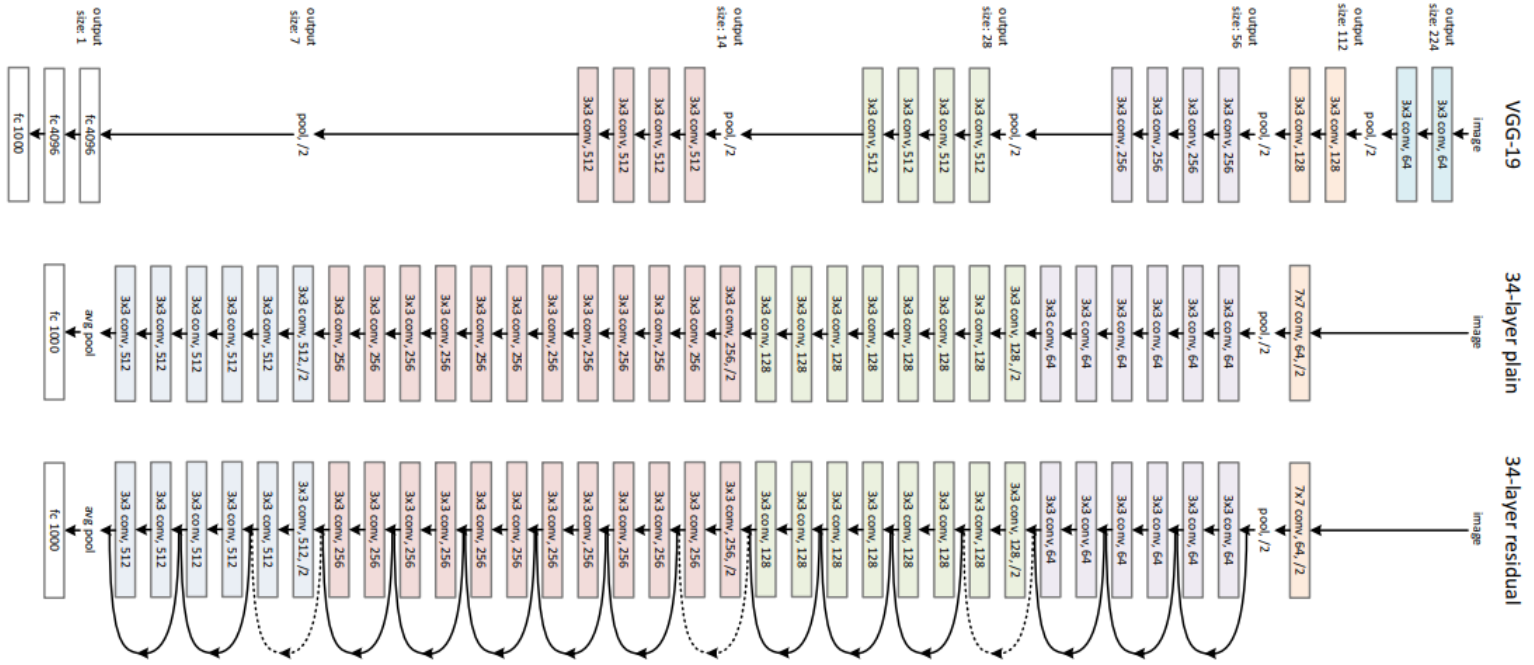


Max Pooling,
Average Pooling,
Sum Pooling
… …

Examples of Deep Convolutional Neural Networks

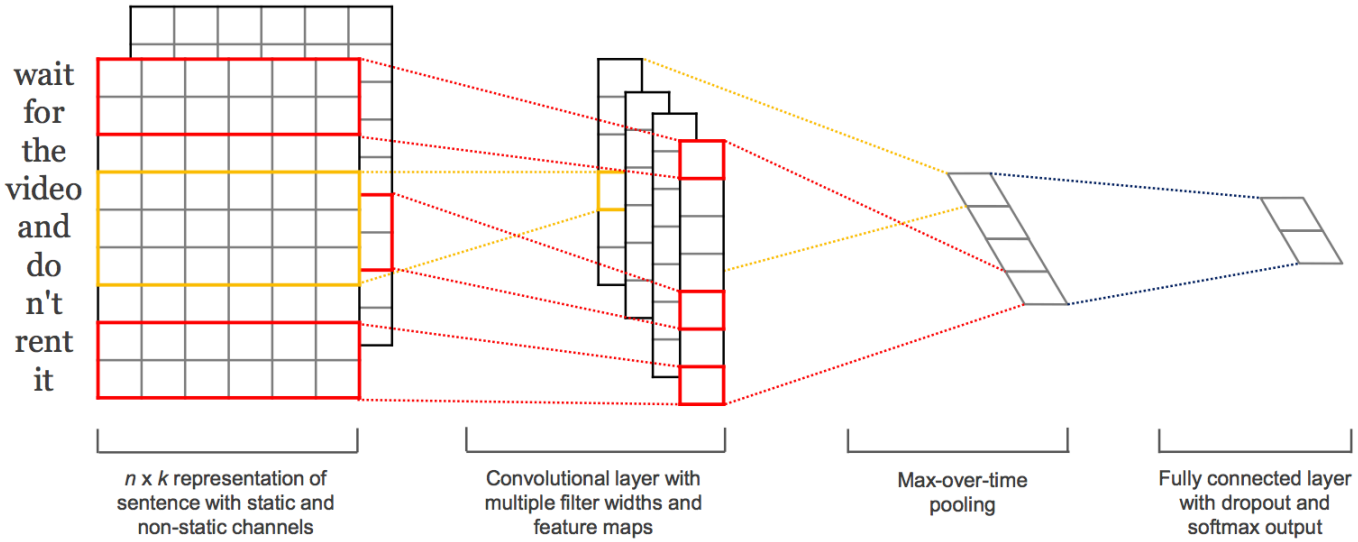


GoogleLeNet [Szegedy et al., 2013]

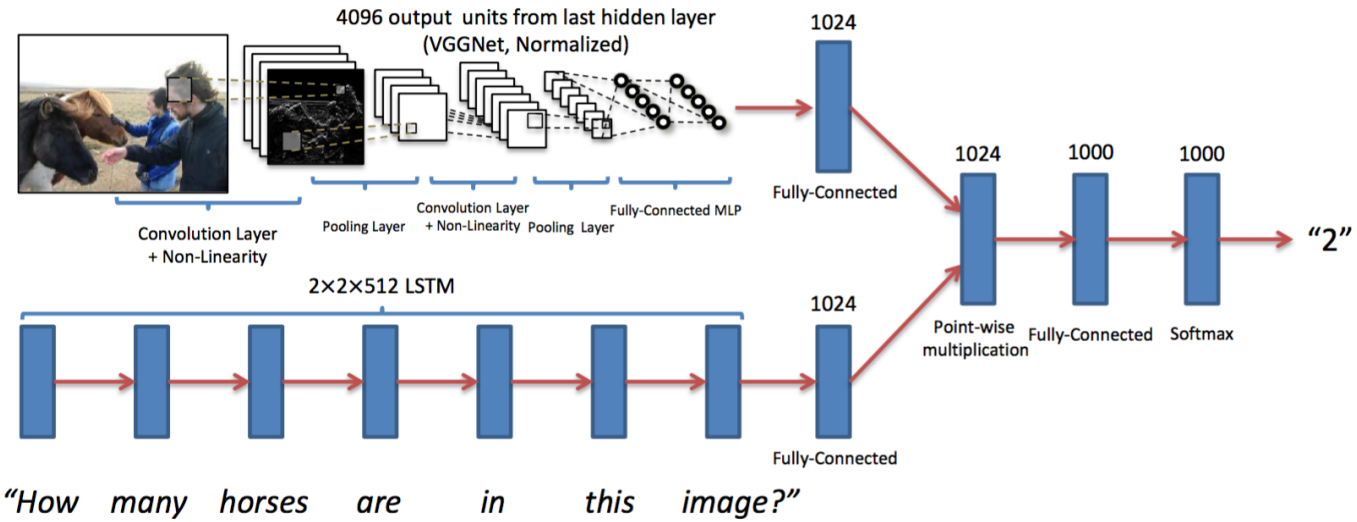


ResNet
[He et al.,, 2015]

Application of Convolutional Neural Networks



Text Classification



(Visual) Question Answering