CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

CONVOLUTIONAL NEURAL NETWORKS

Motivation – Image Data

- So far, the structure of our neural network treats all inputs interchangeably.
- No relationships between the individual inputs
- Just an ordered set of variables

- We want to incorporate domain knowledge into the architecture of a Neural Network.

Motivation

- Image data has important structures, such as;

- "Topology" of pixels
- Translation invariance
- Issues of lighting and contrast
- Knowledge of human visual system
- Nearby pixels tend to have similar values
- Edges and shapes
- Scale Invariance – objects may appear at different sizes in the image.

Motivation – Image Data

- Fully connected would require a vast number of parameters
- MNIST images are small (32 x 32 pixels) and in grayscale
- Color images are more typically at least (200 x 200) pixels x 3 color channels (RGB) = 120,000 values.
- A single fully connected layer would require $(200x200x3)^2$ = 14,400,000,000 weights!
- Variance (in terms of bias-variance) would be too high
- So we introduce “bias” by structuring the network to look for certain kinds of patterns

Motivation

- Features need to be “built up”
- Edges -> shapes -> relations between shapes
- Textures

- Cat = two eyes in certain relation to one another + cat fur texture.
- Eyes = dark circle (pupil) inside another circle.
- Circle = particular combination of edge detectors.
- Fur = edges in certain pattern.

Kernels

- A *kernel* is a grid of weights "overlaid" on image, centered on one pixel
- Each weight multiplied with pixel underneath it
- Output over the centered pixel is $\sum_{p=1}^{P} W_p \cdot pixel_p$
- Used for traditional image processing techniques:
  - Blur
  - Sharpen
  - Edge detection
  - Emboss

Kernel: 3x3 Example

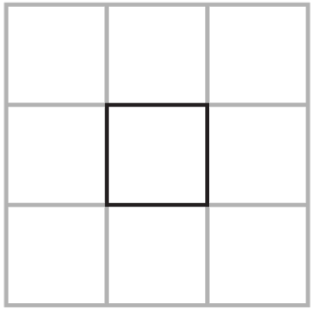
Input

3	2	1
1	2	3
1	1	1

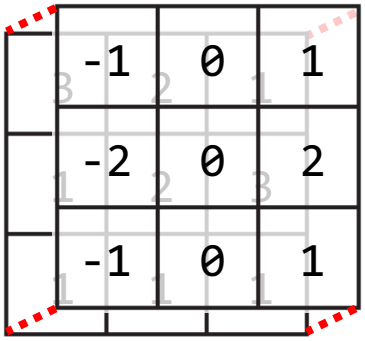
Kernel

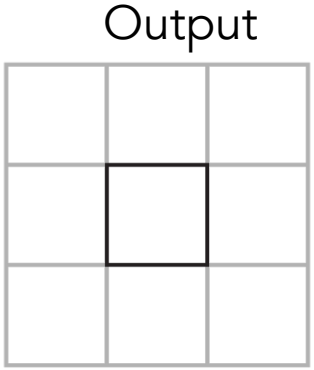
-1	0	1
-2	0	2
-1	0	1

Output



Kernel: 3x3 Example





Kernel: 3x3 Example

Input

3	2	1
1	2	3
1	1	1

Kernel

-1	0	1
-2	0	2
-1	0	1

Output

	2	

$$= (3 \cdot -1) + (2 \cdot 0) + (1 \cdot 1)$$
$$+(1 \cdot -2) + (2 \cdot 0) + (3 \cdot 2)$$
$$+(1 \cdot -1) + (1 \cdot 0) + (1 \cdot 1)$$

$$= -3 + 1 - 2 + 6 - 1 + 1 = 2$$

Kernel: Example

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur

0	-1	0
-1	5	-1
0	-1	0

Kernel to make image sharper

-1	-1	-1
-1	9	-1
-1	-1	-1

Intensified sharper image

Gaussian Blur

Sharpened image

Kernels as Feature Detectors

Can think of kernels as a "local feature detectors"

Vertical Line Detector

-1	1	-1
-1	1	-1
-1	1	-1

Horizontal Line Detector

-1	-1	-1
1	1	1
-1	-1	-1

Corner Detector

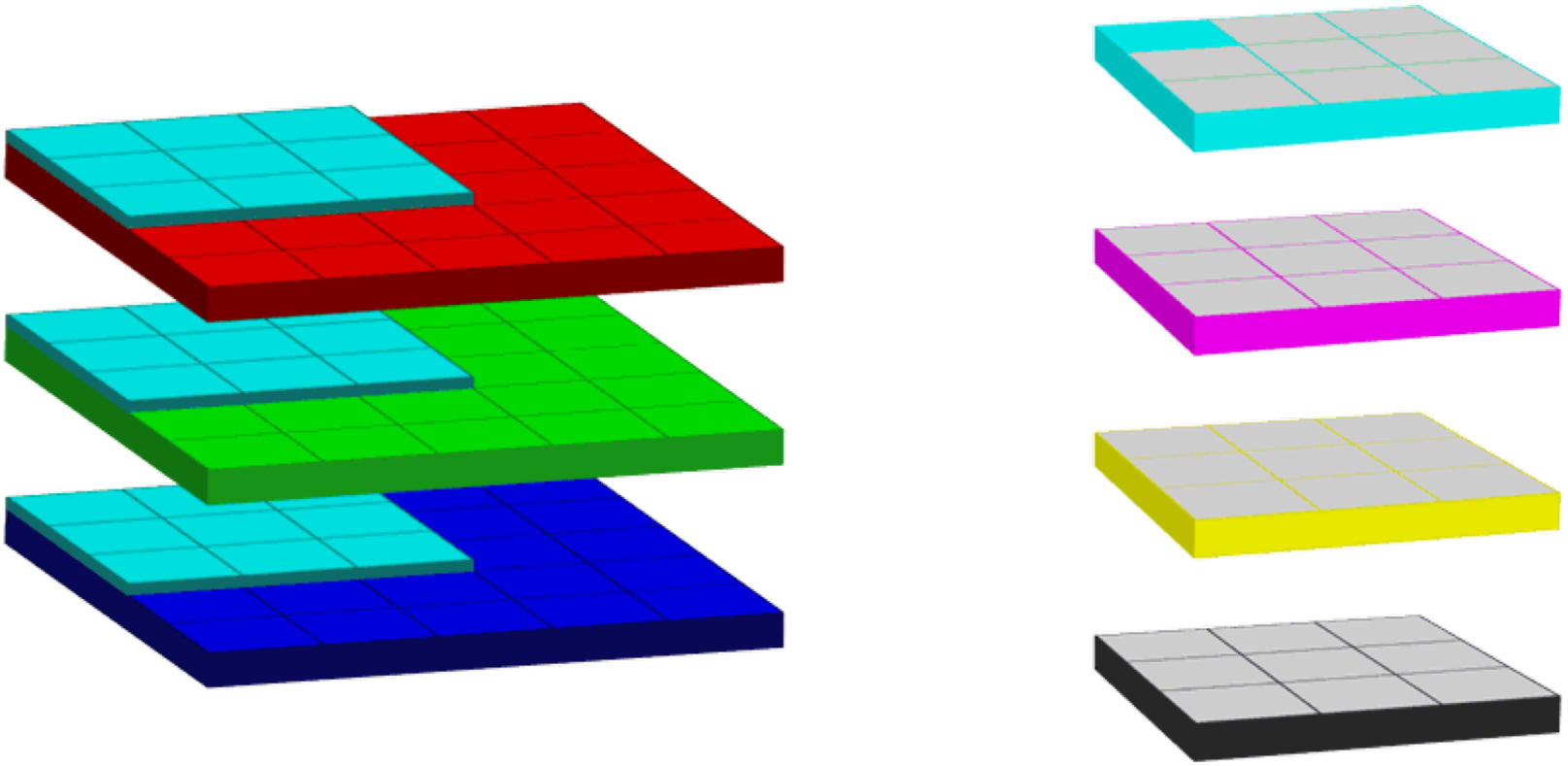
-1	-1	-1
-1	1	1
-1	1	1

Convolutional Neural Nets

Primary Ideas behind Convolutional Neural Networks:

- Let the Neural Network learn which kernels are most useful
- Use same set of kernels across entire image (translation invariance)
- Reduces number of parameters and “variance” (from bias-variance point of view)

Convolutions

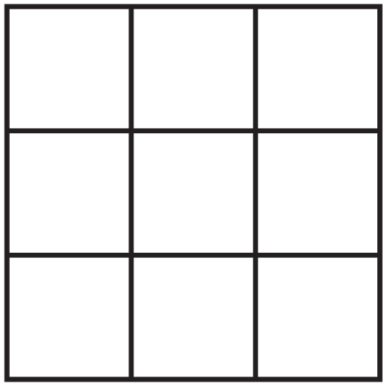


Convolution Settings – Grid Size

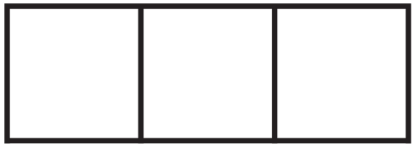
Grid Size (Height and Width):

- The number of pixels a kernel "sees" at once
- Typically use odd numbers so that there is a "center" pixel
- Kernel does not need to be square

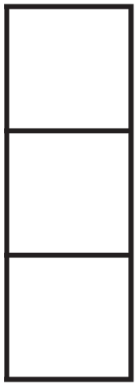
Height: 3, Width: 3



Height: 1, Width: 3



Height: 3, Width: 1

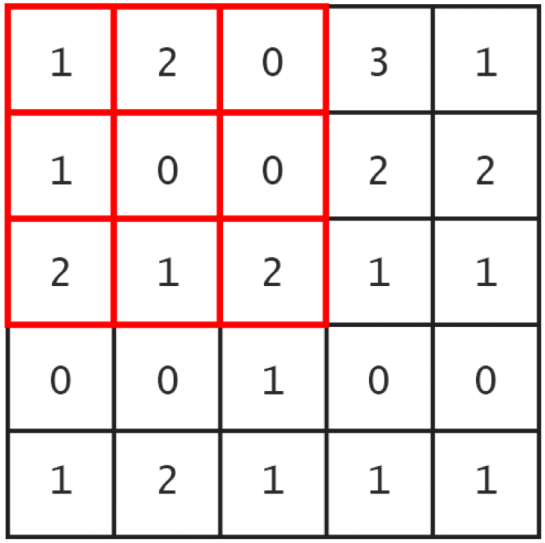


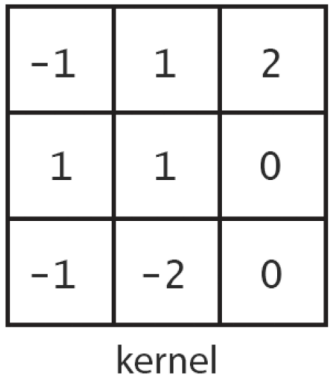
Convolution Settings - Padding

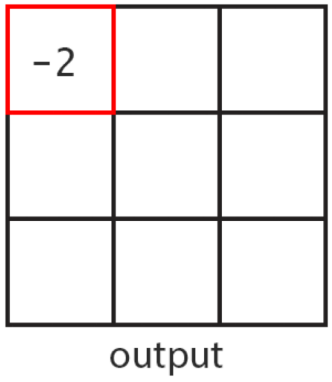
Padding

- Using Kernels directly, there will be an “edge effect”
- Pixels near the edge will not be used as “center pixels” since there are not enough surrounding pixels
- Padding adds extra pixels around the frame
- So every pixel of the original image will be a center pixel as the kernel moves across the image
- Added pixels are typically of value zero (zero-padding)

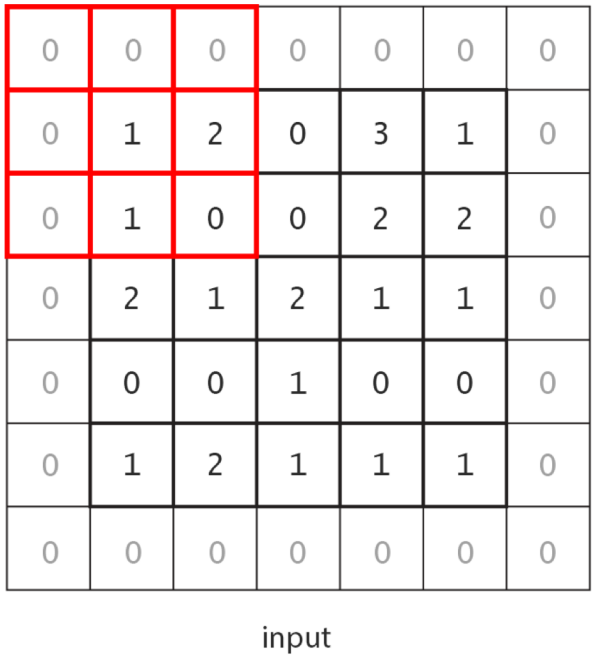
Without Padding



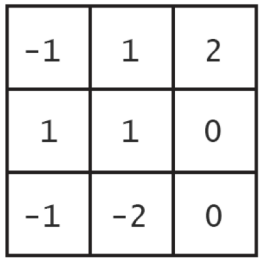




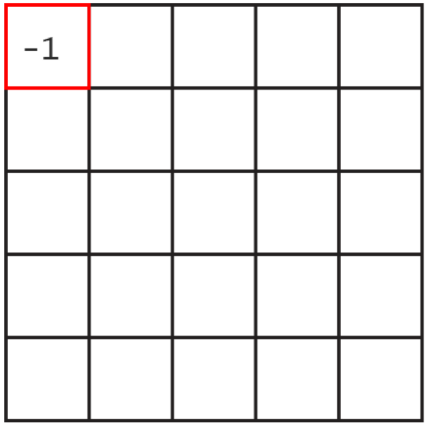
With Padding



input



kernel



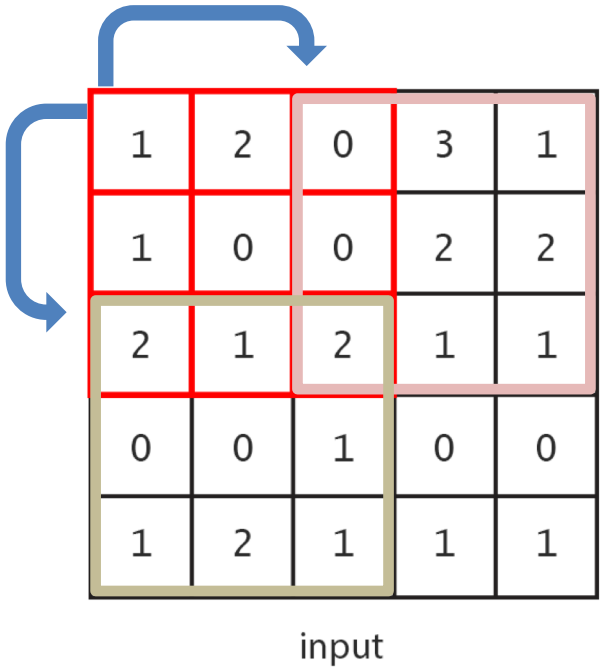
output

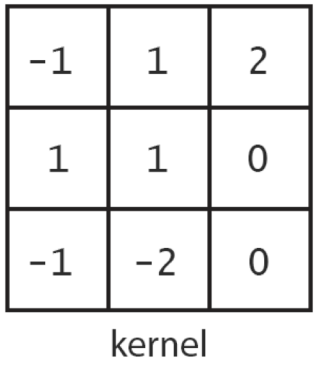
Convolution Settings

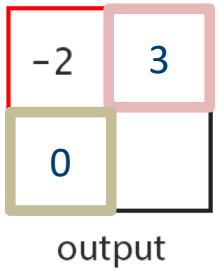
Stride

- The "step size" as the kernel moves across the image
- Can be different for vertical and horizontal steps (but usually is the same value)
- When stride is greater than 1, it scales down the output dimension

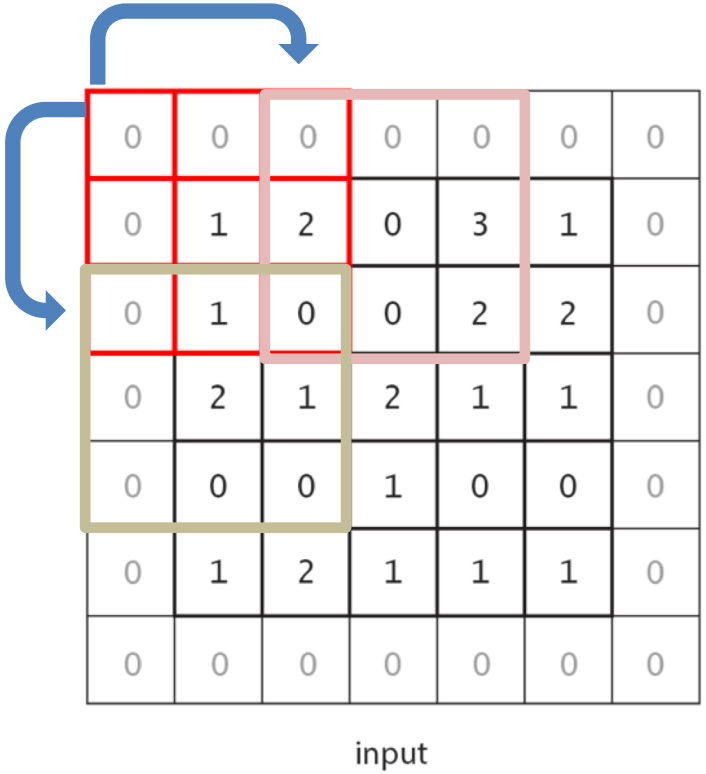
Stride 2 Example – No Padding

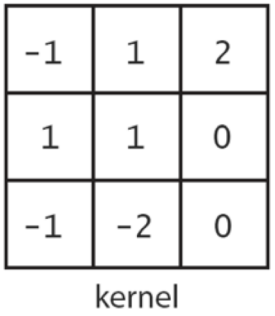


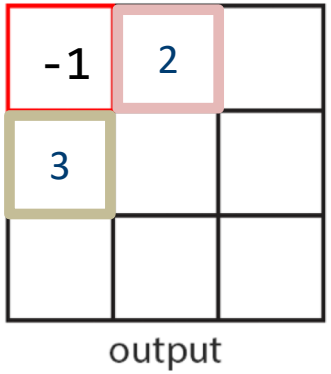




Stride 2 Example – With Padding







Convolutional Settings - Depth

- In images, we often have multiple numbers associated with each pixel location.
- These numbers are referred to as “channels”
  - RGB image – 3 channels
  - CMYK – 4 channels
- The number of channels is referred to as the “depth”
- So the kernel itself will have a “depth” the same size as the number of input channels
- Example: a 5x5 kernel on an RGB image
  - There will be 5x5x3 = 75 weights

Convolutional Settings - Depth

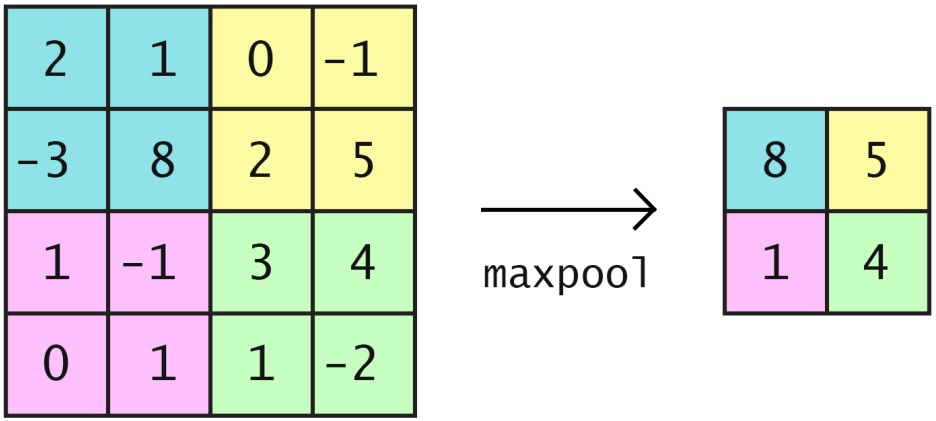
- The output from the layer will also have a depth
- The networks typically train many different kernels
- Each kernel outputs a single number at each pixel location
- So if there are 10 kernels in a layer, the output of that layer will have depth 10.

Pooling

- Idea: Reduce the image size by mapping a patch of pixels to a single value.
- Shrinks the dimensions of the image.
- Does not have parameters, though there are different types of pooling operations.

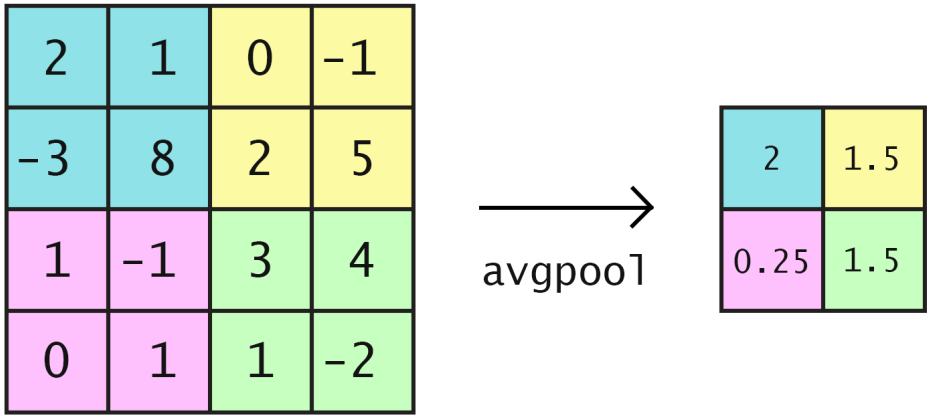
Pooling: Max-pool

- For each distinct patch, represent it by the maximum
- 2x2 maxpool shown below

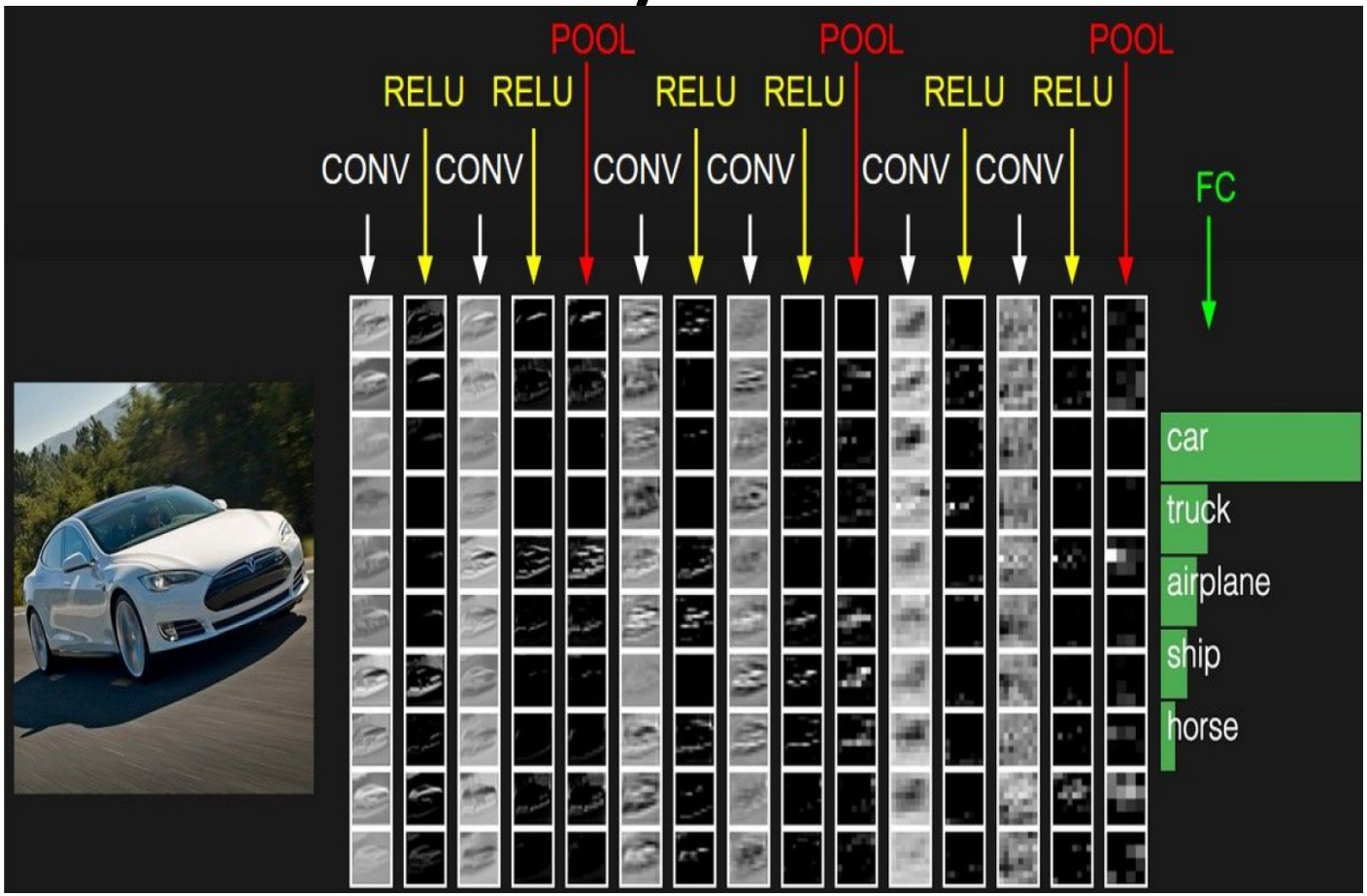


Pooling: Average-pool

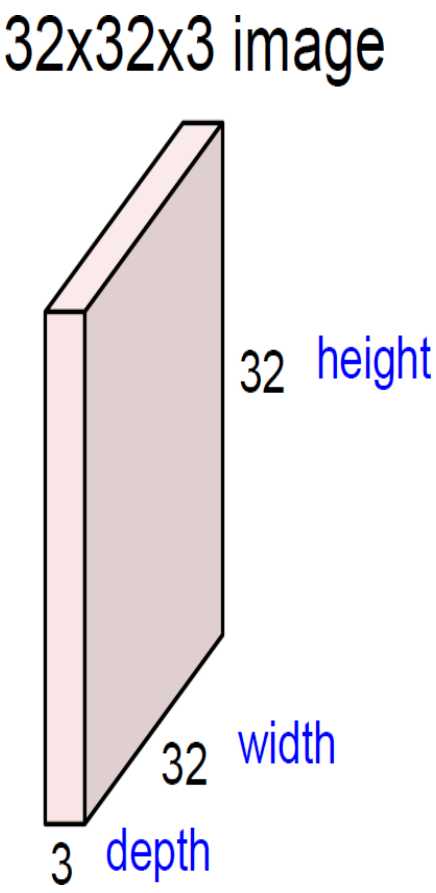
- For each distinct patch, represent it by the average
- 2x2 avgpool shown below.



ConvNet: CONV, RELU, POOL and FC Layers

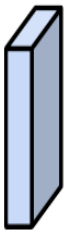


Convolution Layer



Filters always extend the full depth of the input volume

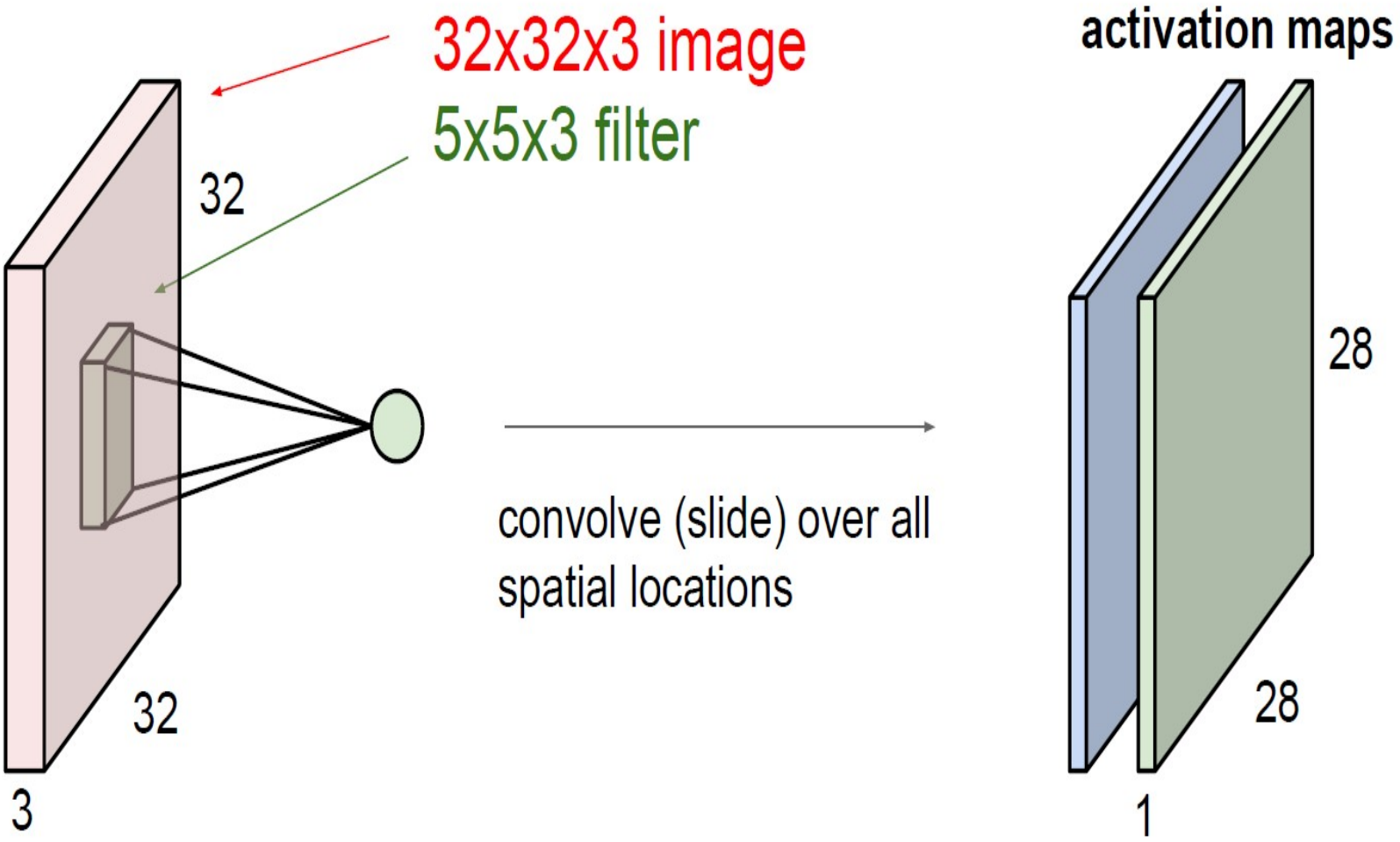
5x5x3 filter



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

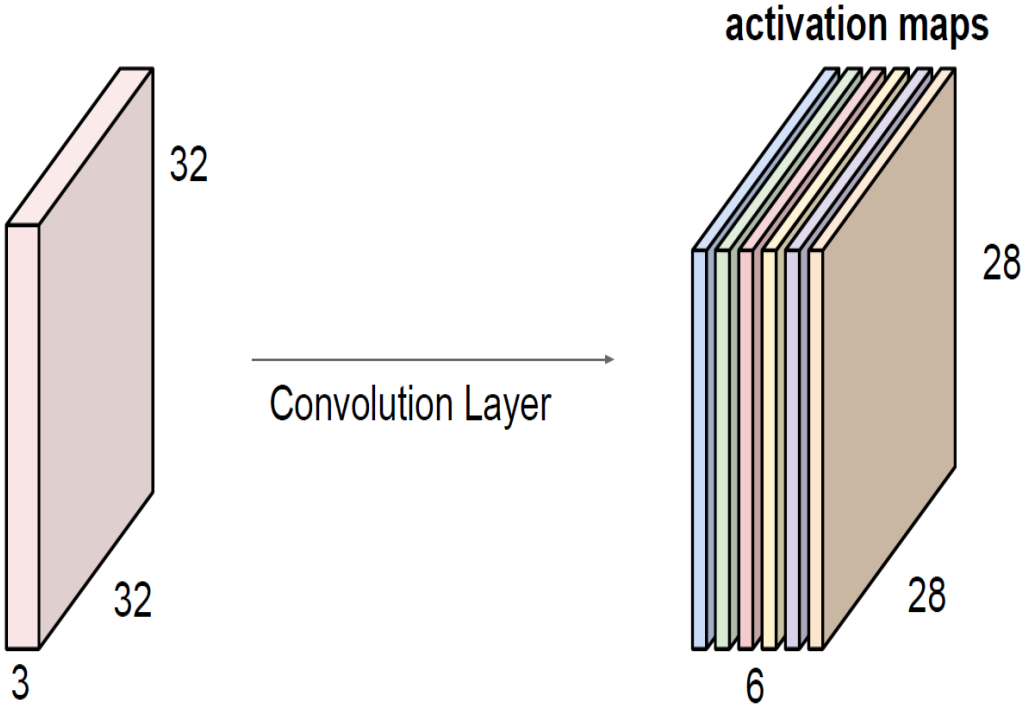
Convolution Layer

consider a second, green filter



Convolution Layer

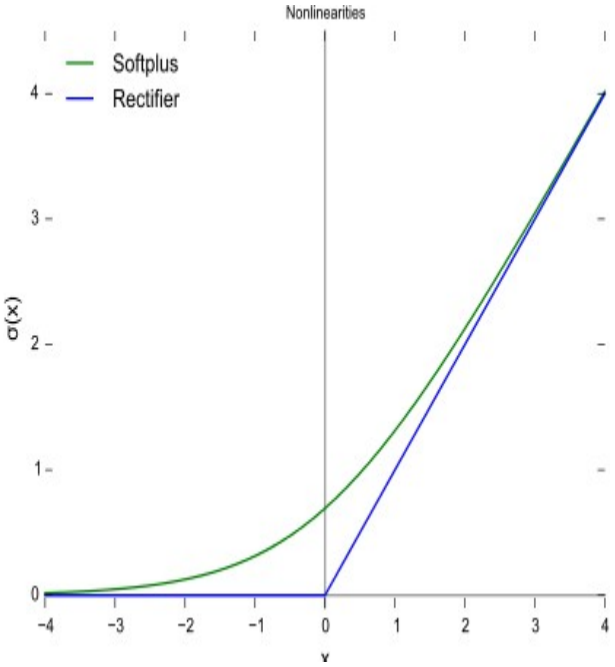
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

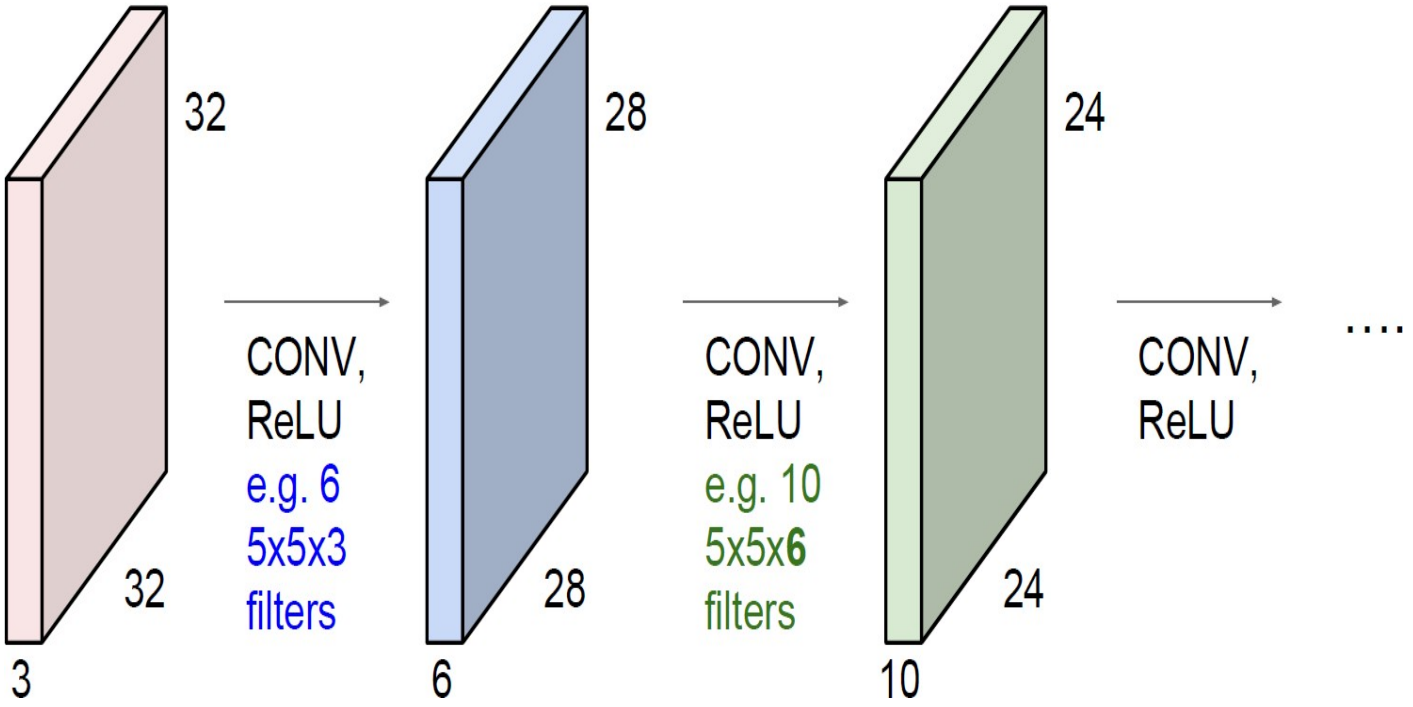
ReLU (Rectified Linear Units) Layer

- This is a layer of neurons that applies the activation function f(x)=max(0,x).
- It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.
- Other functions are also used to increase nonlinearity, for example the hyperbolic tangent f(x)=tanh(x), and the sigmoid function.
- This is also known as a ramp function.

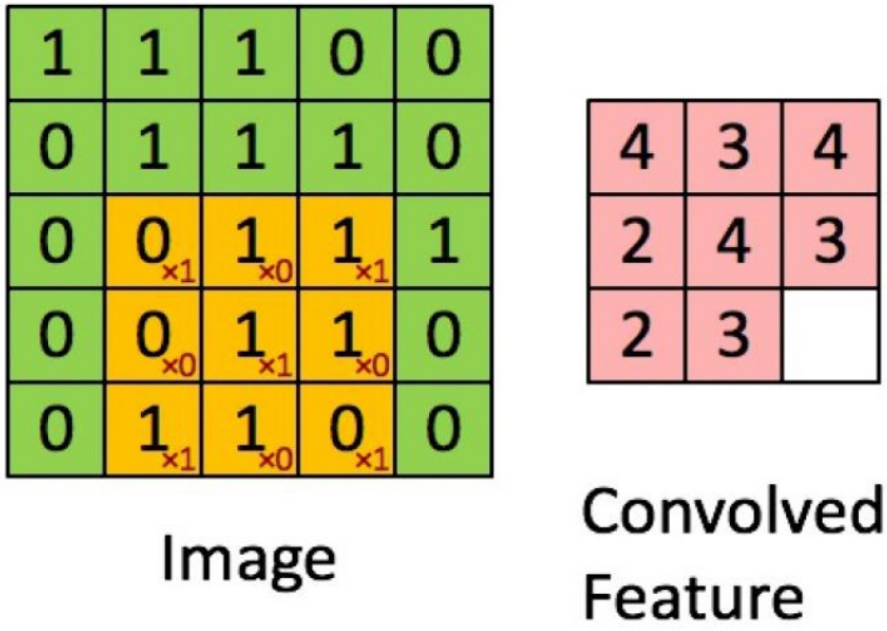


A Basic ConvNet

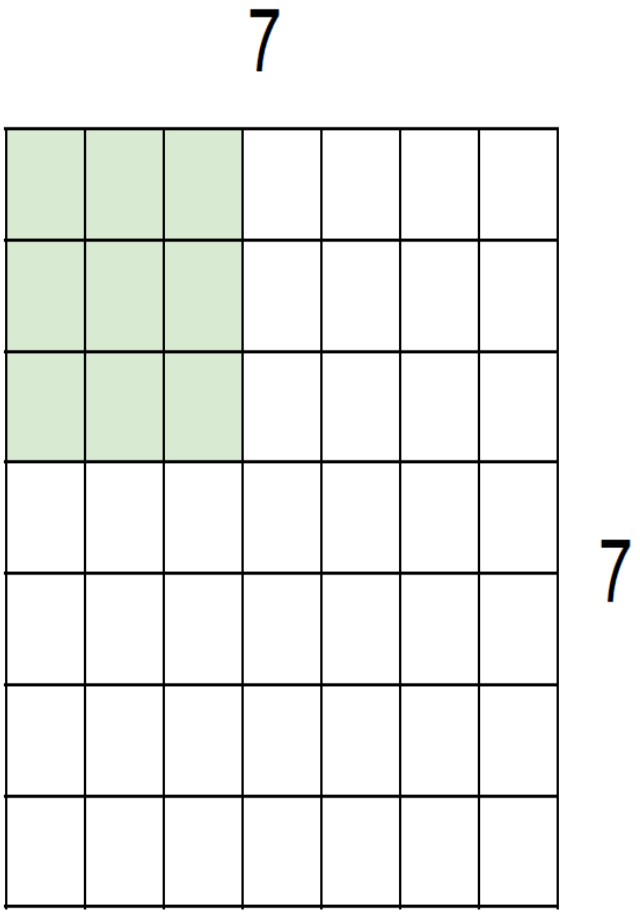
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



What is convolution of an image with a filter



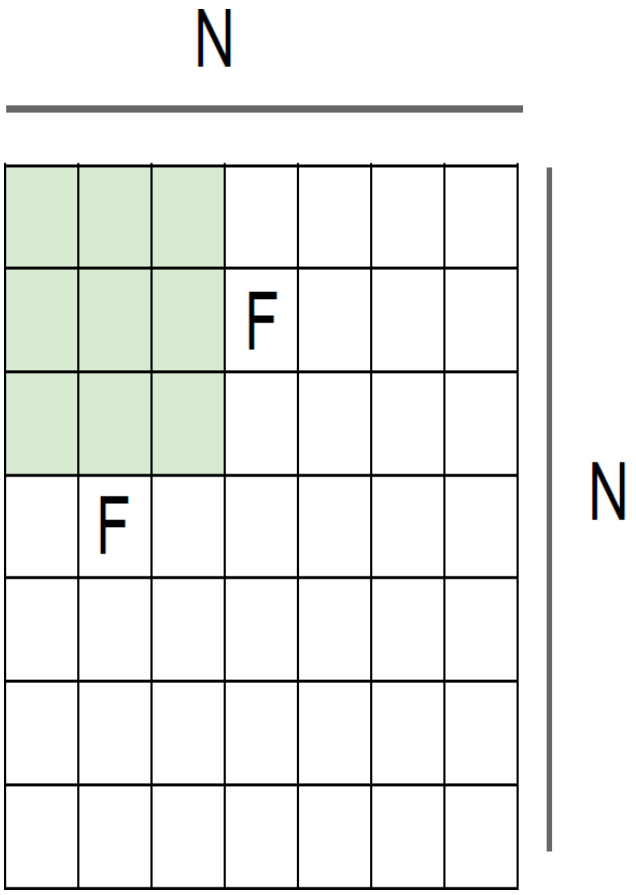
Details about the convolutionlayer



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on 7x7 input with stride 3.

Details about the convolutionlayer

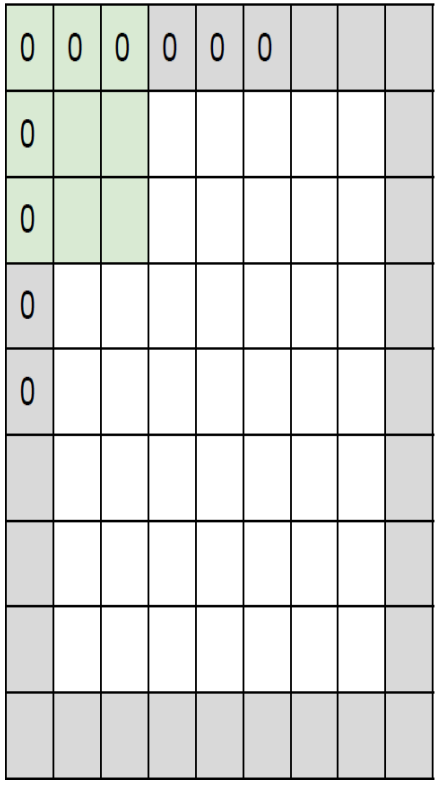


Output size:
(N - F) / stride + 1

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

Details about the convolutionlayer

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!
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

Convolution layer examples

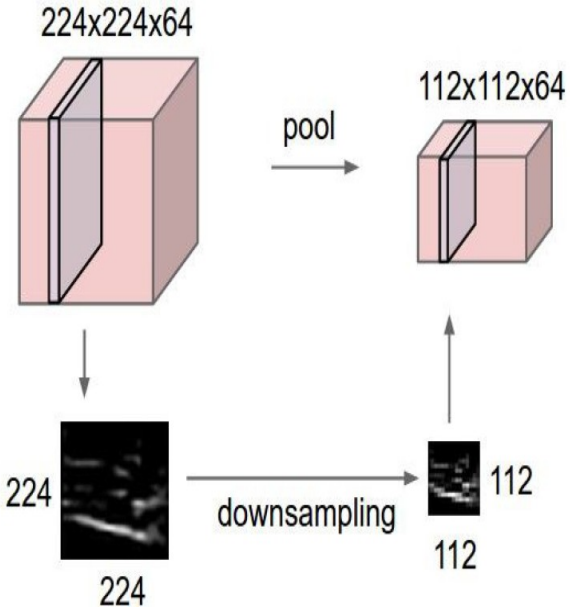
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

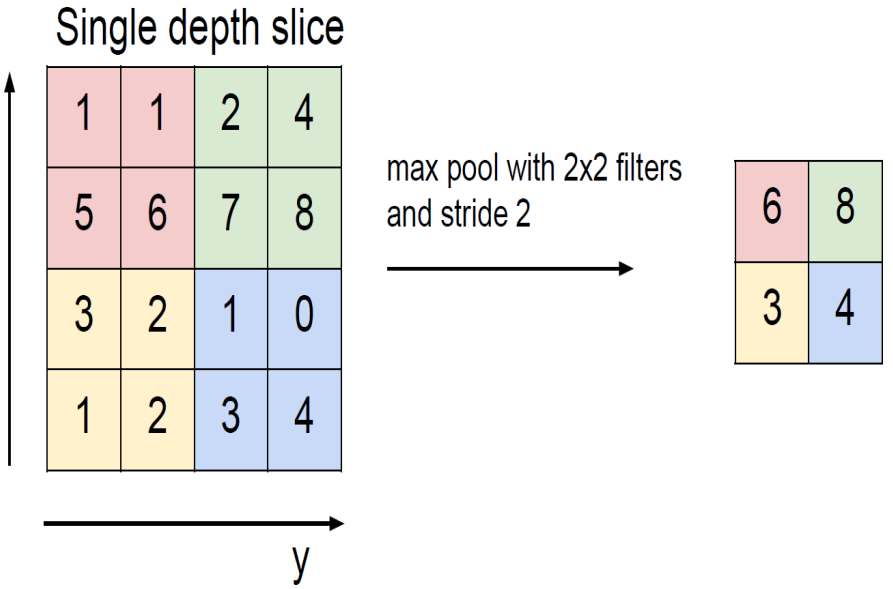
Output volume size: ?

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

Pooling Layer

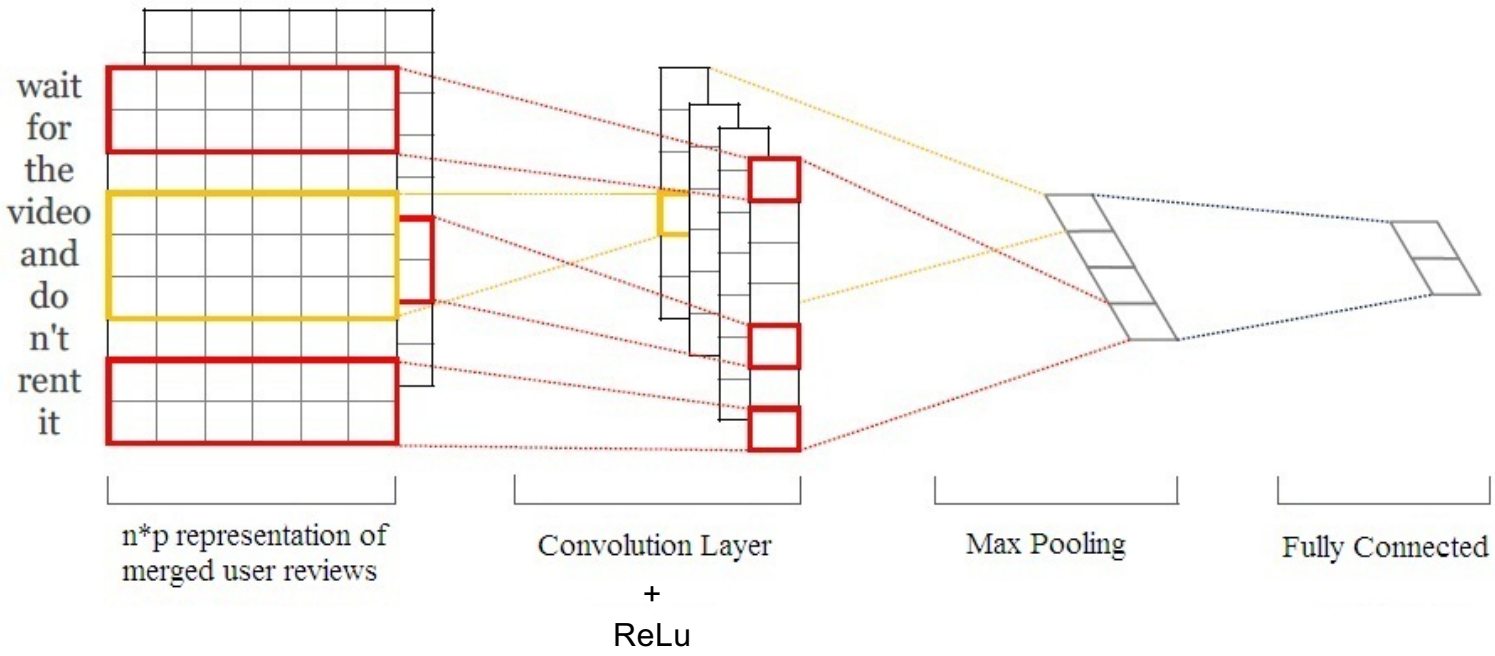
makes the representations smaller and more manageable
operates over each activation map independently:





- Invariance to image transformation and increases compactness to representation.
- Pooling types: Max, Average, L2 etc.

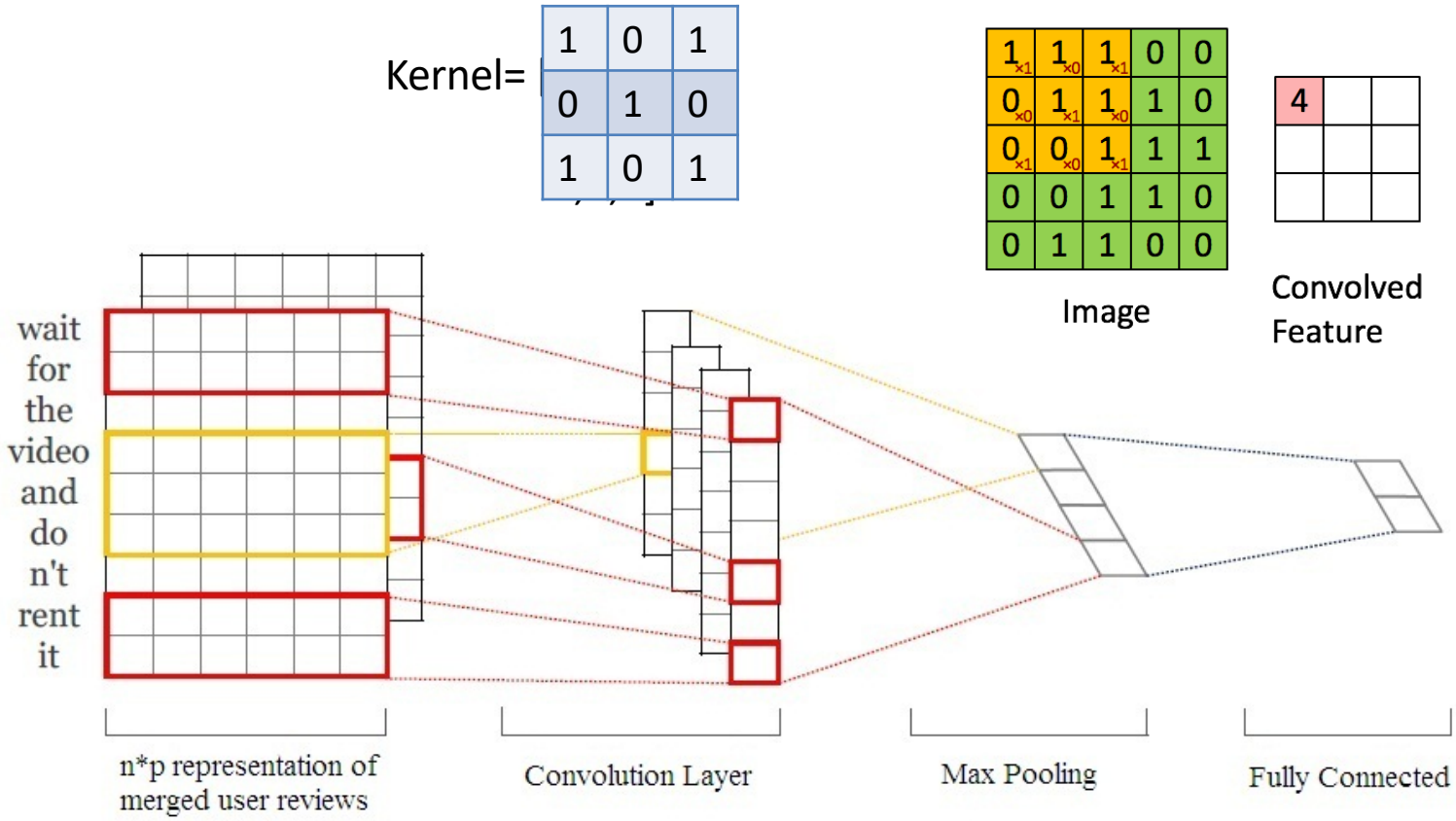
Convolutional Neural Networks



$$z_j = f(V_{1:n}^u * K_j + b_j)$$

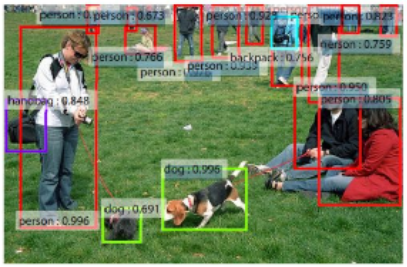
Where ReLu is used as f.

Convolutional Neural Networks



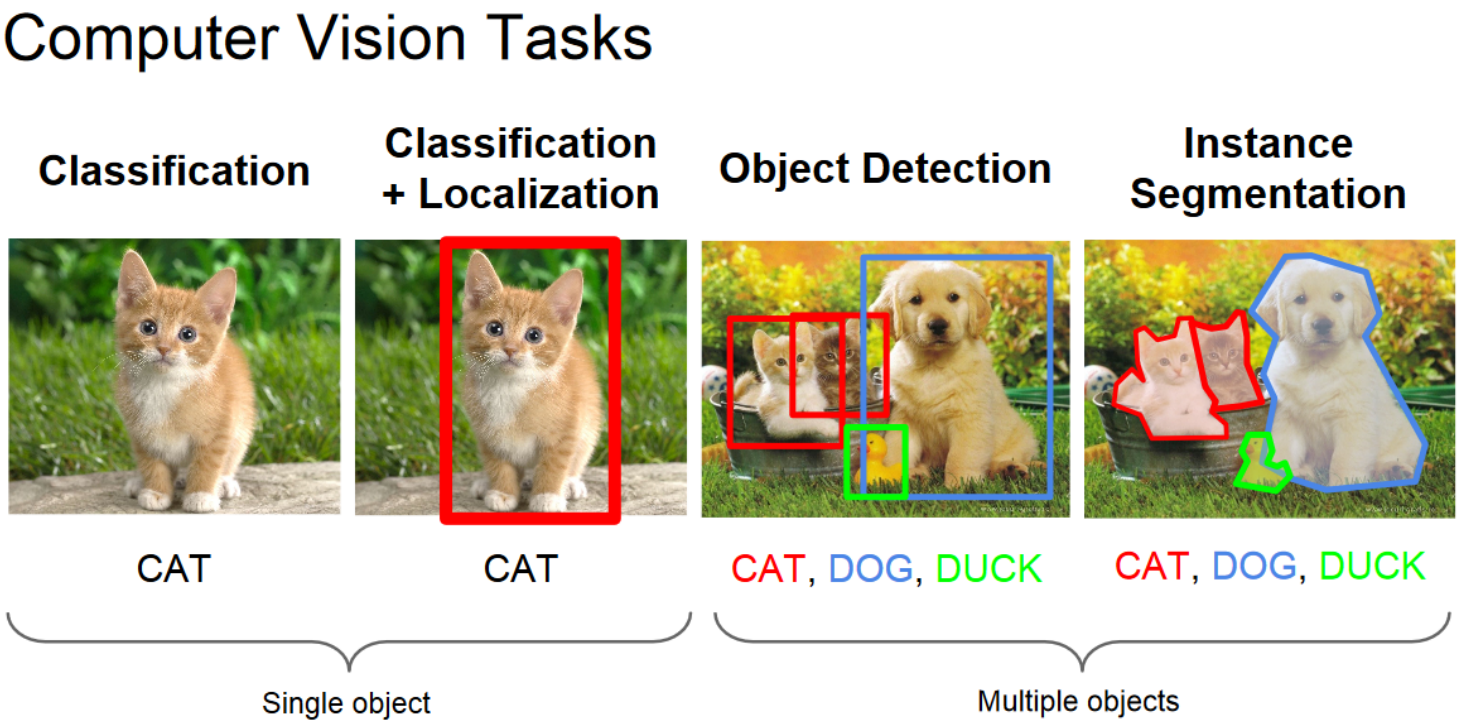
Applications

Localization and Detection



Results from Faster R-CNN, Ren et al 2015

Applications



Is deep learning all about CNNs?

- Consider a language modelling task
- Given a vocabulary, the task is to predict the next word in a sentence
- Sequence information of words are important
- Typically in cases where sequential data is involved, **recurrent neural networks (RNNs)** are widely used

References

- CS231n: Convolutional Neural Networks for Visual Recognition. Andrej Karpathy http://cs231n.github.io/convolutional-networks/
- **https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5**
- **CSE 446 - Machine Learning - Spring 2015,** University of Washington. Pedro Domingos. https://courses.cs.washington.edu/courses/cse446/15sp/

- Thanks