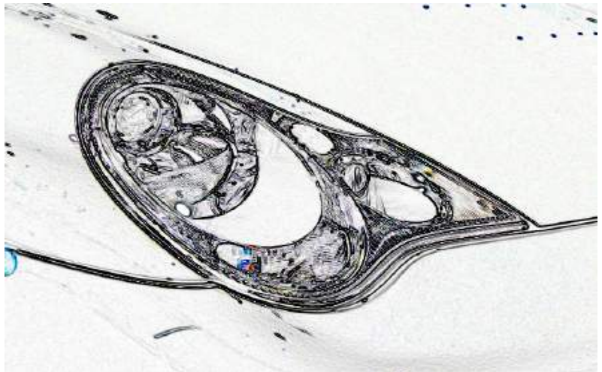
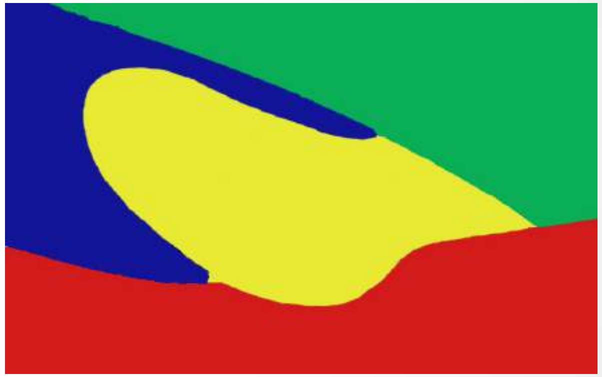
CSE578: Computer Vision

Spring 2019:

Feature Learning with Deep Learning





Avinash Sharma & Anoop M. Namboodiri

Center for Visual Information Technology

IIIT Hyderabad, INDIA

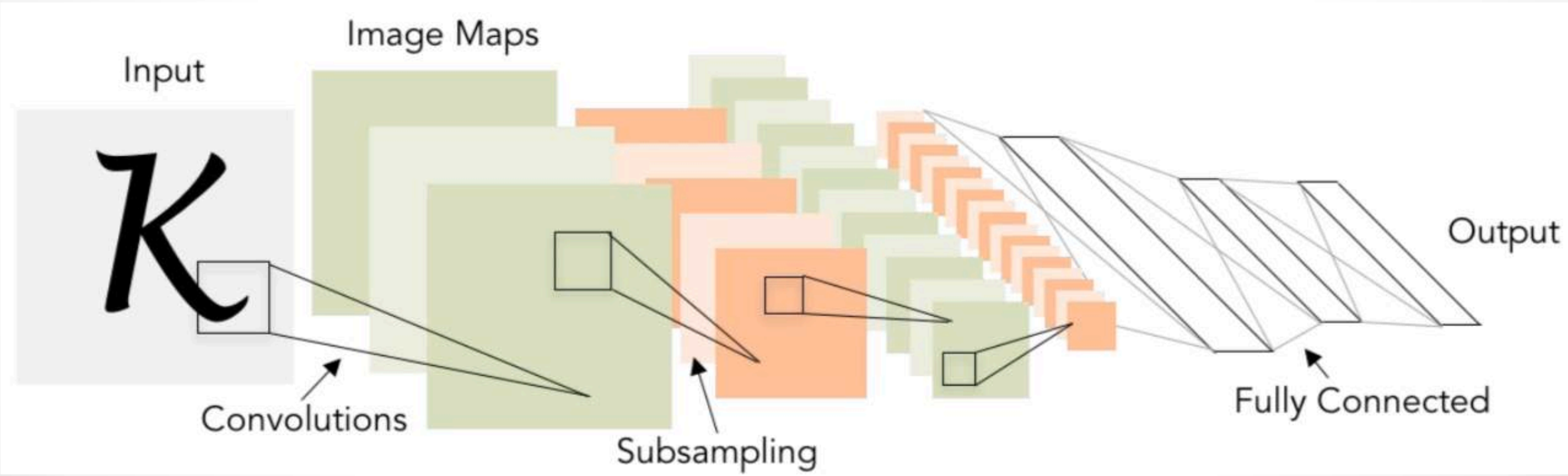
[Content Generously Borrowed from CS231n]

Convolutional Neural Networks

- Course on CNN in Computer Vision at Stanford
  - Fei-Fei Li, Justin Johnson and Serena Yeung
  - 2017 edition on YouTube: https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

Convolutional Neural Networks: History

- LeNet: Digit / Character Recognition, LeCun, Bottou, Bengio, Haffner 1998.



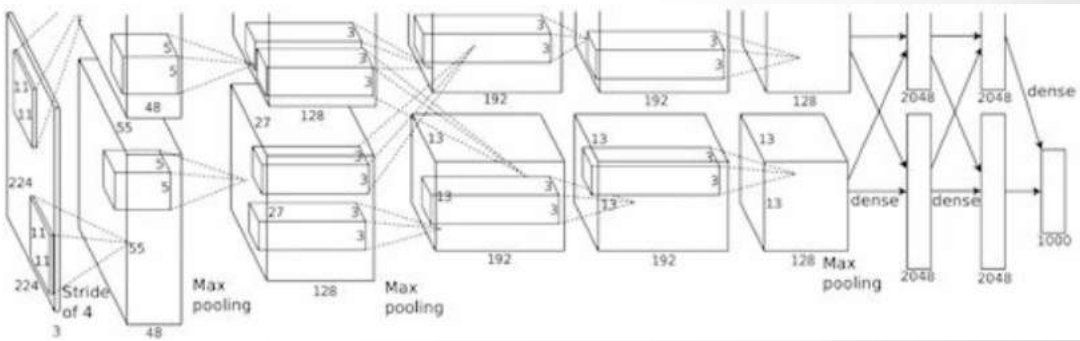
Convolutional Neural Networks: History

- The Mark 1 Perceptron machine (Frank Rosenblatt, ~1957)
- Connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.
- Recognized letters of the alphabet
- Used gradient descent update rule for learning
-

Convolutional Neural Networks: History

Several other efforts

- Adaline/Madaline: Widrow and Hoff, 1960
- Backpropagation: Rumelhart et al. 1986
- RBMs: Pretraining: Hinton and Salakhutdinov 2006
- The watershed moment: “Imagenet Classification with Deep Convolutional Neural Networks”: Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



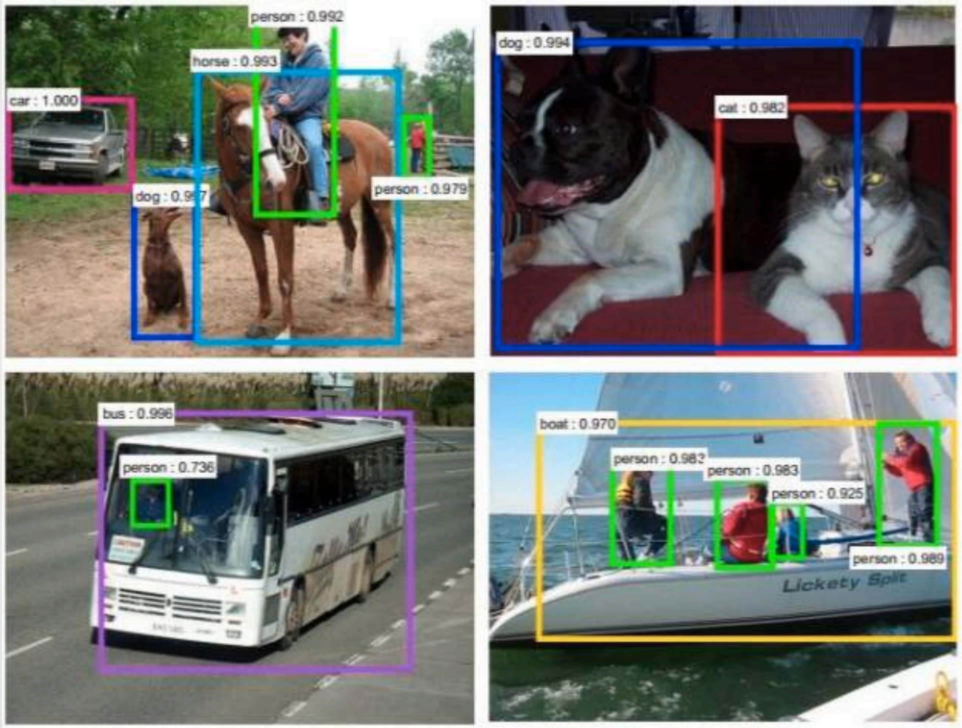
ConvNets tops most vision tasks

Classification

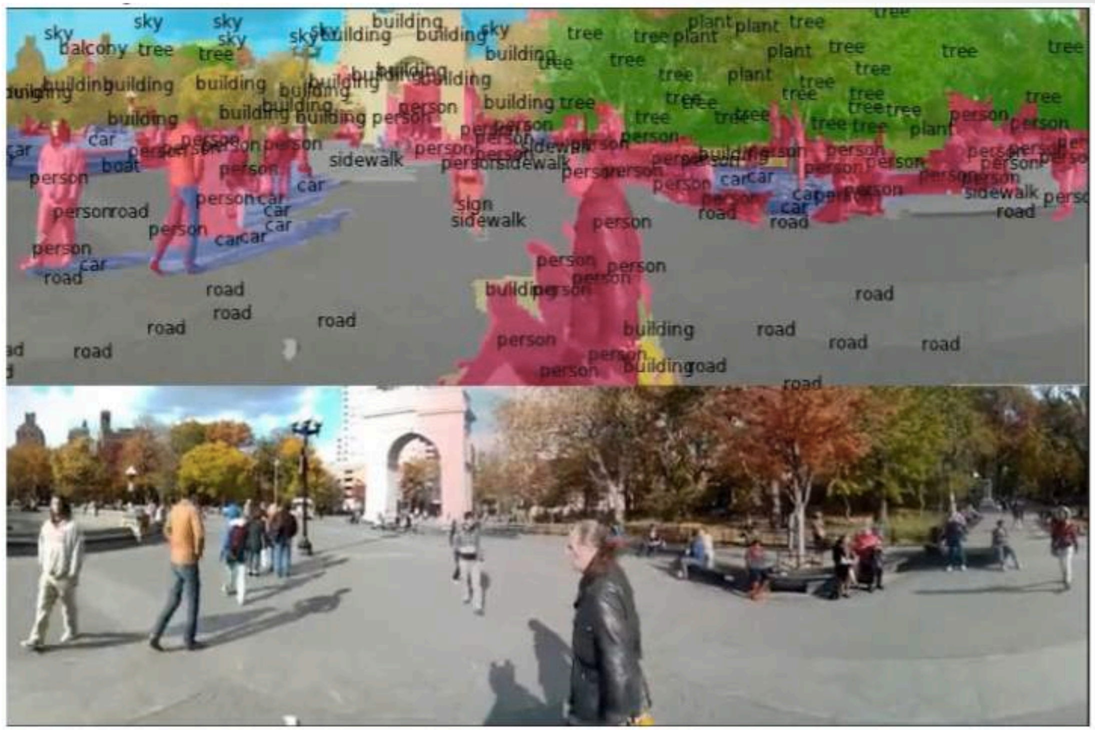
Retrieval

ConvNets tops most vision tasks

Object Detection



Semantic/Instance Segmentation



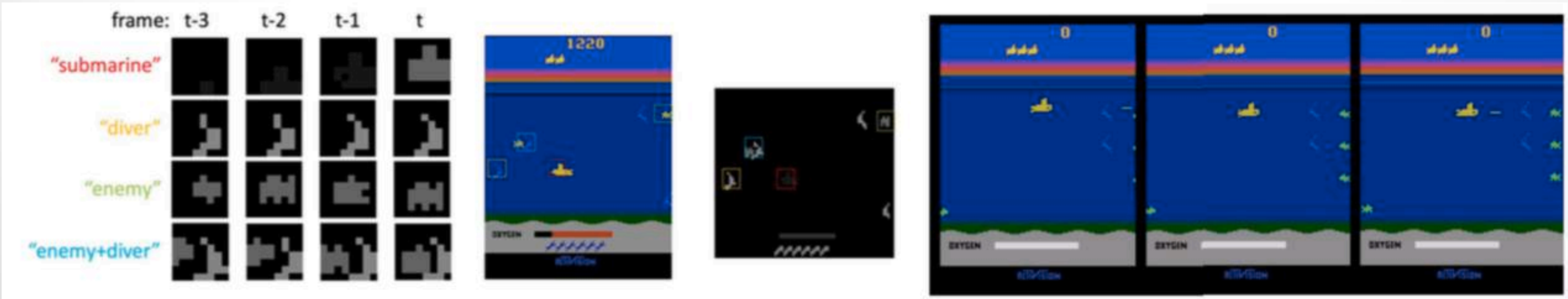
ConvNets tops most vision tasks

Self Driving

Street Sign Recognition

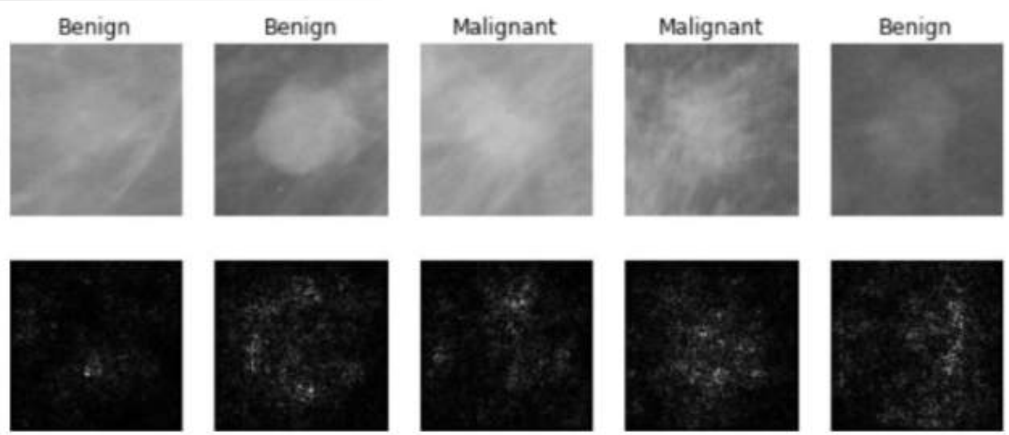
ConvNets tops most vision tasks

- Human Pose Estimation; Video game play



ConvNets tops most vision tasks

- Medical Diagnosis
- Street Detection



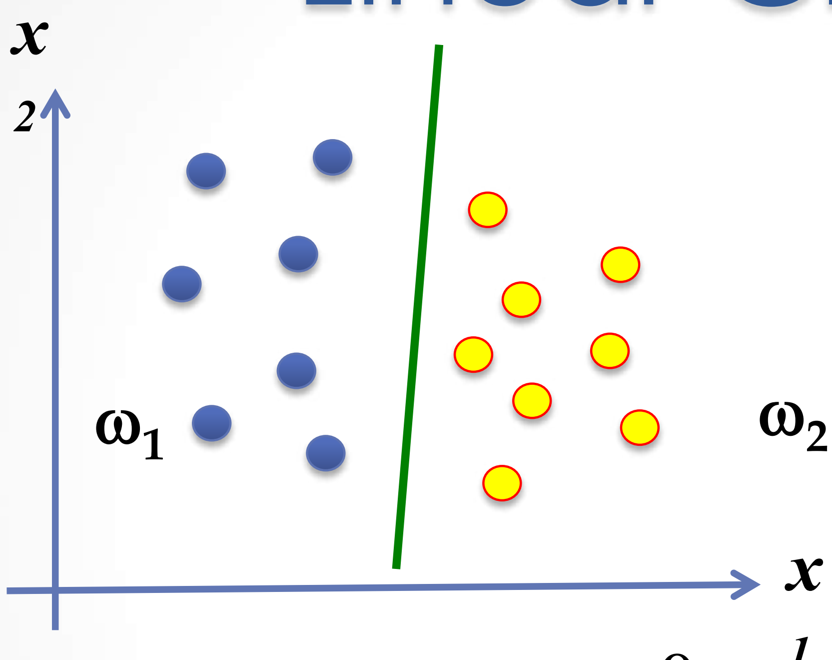
Whale Recognition (Kaggle)

ConvNets tops most vision tasks

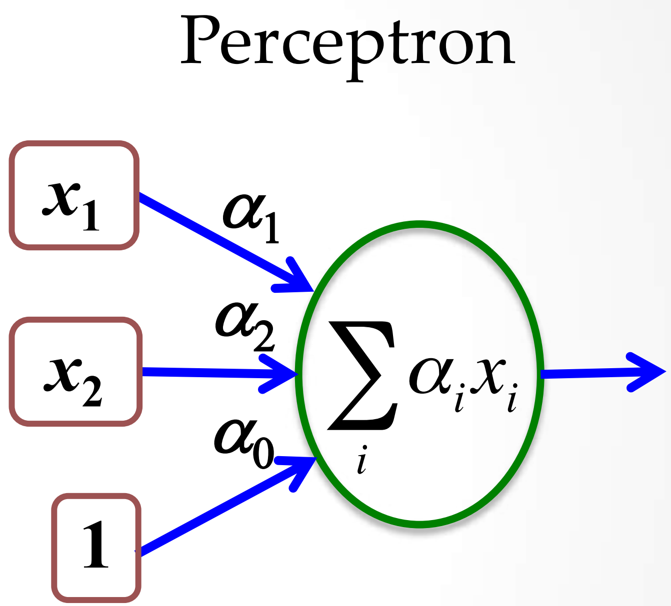
- Person Recognition
- Spoof Detection
- Video Activity Recognition
- Image Captioning
- Image Generation
- Style Transfer
- Image Super-resolution
- Image Coloring
- Lip Reading
- Visual QA
- Video Captioning
- Video Highlight Detection
- Single/few Image 3D Reconstruction
- And many others
  - Just see Kaggle

Deeper into Neural Networks

Linear Classifier

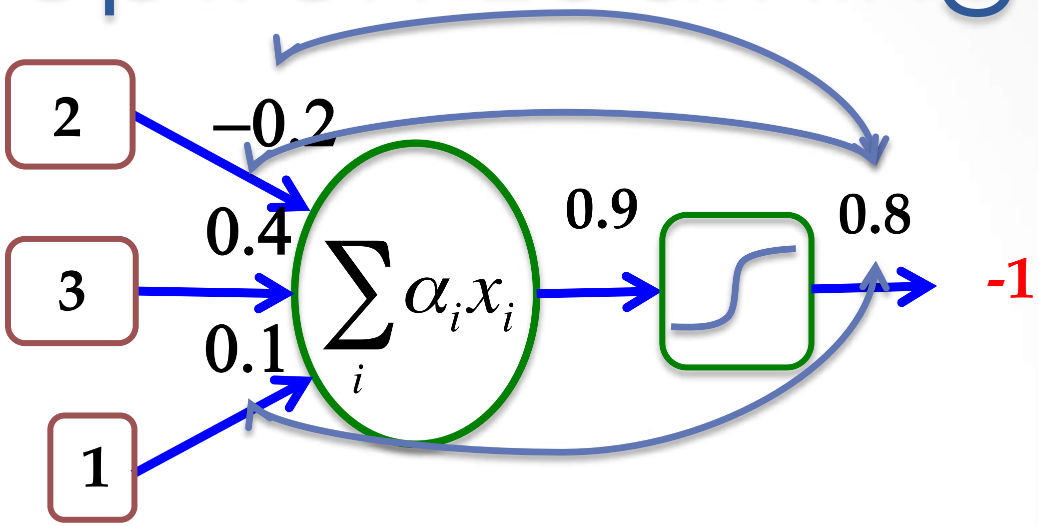


$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_0 = 0$$



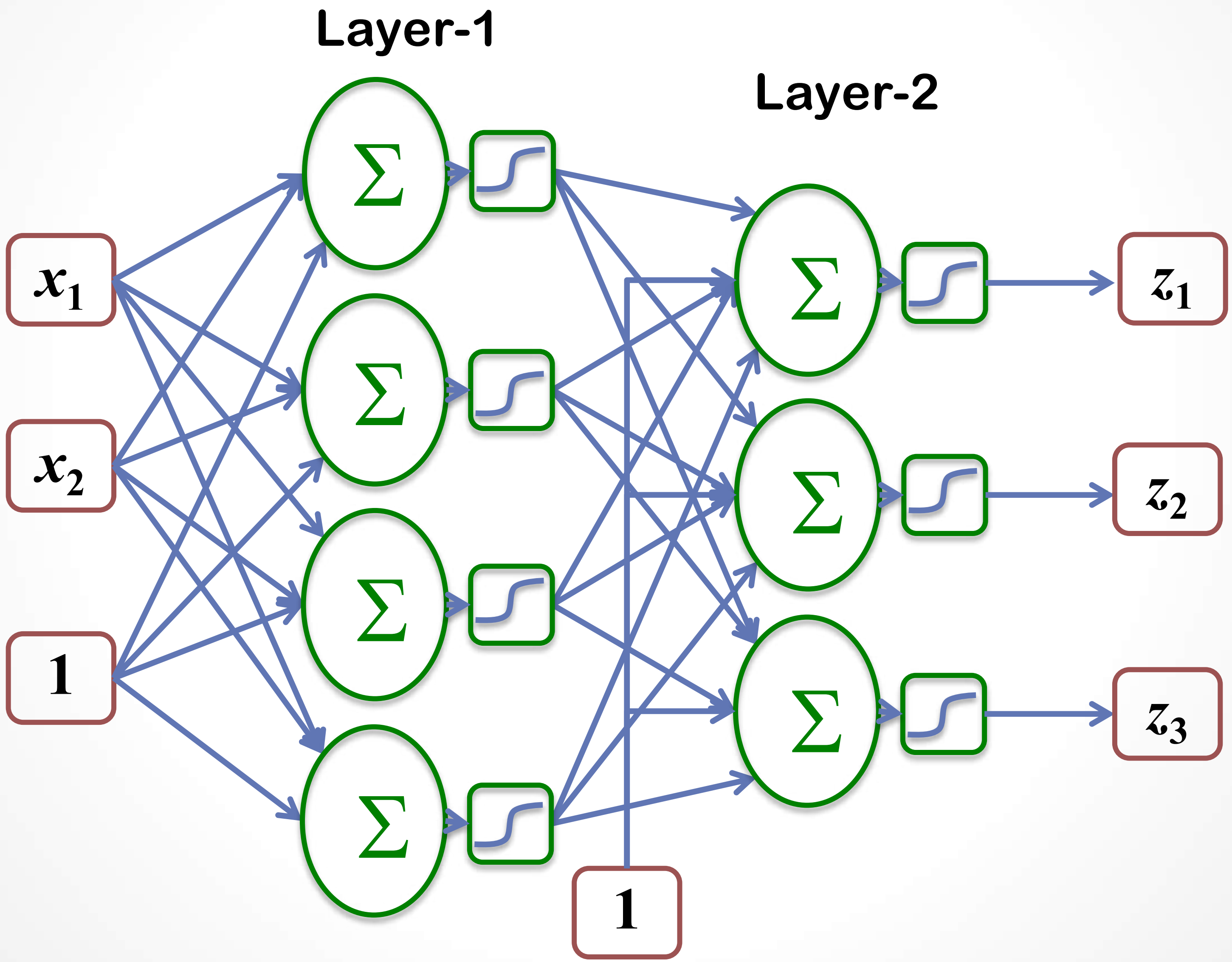
- A linear boundary separates two classes.
- Learning the classifier means learning the weights, α_i.

Perceptron Learning

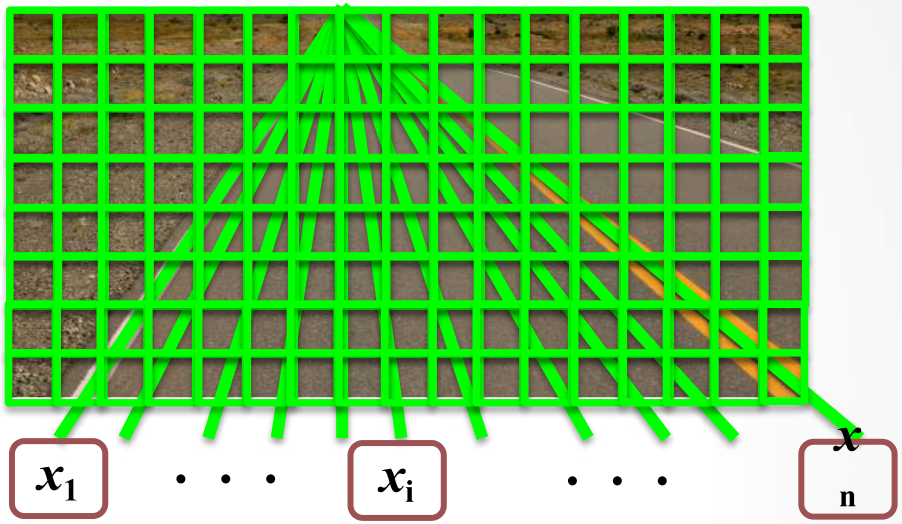


- Randomly Initialize the weights
- For each sample:
  - Feed a sample and find the output (forward pass)
  - Find the difference between actual and desired outputs (cost function)
  - Find the effect of each weight on the cost (derivative)
  - Update the weights with a learning rate (GD)

Multi-layer Perceptron



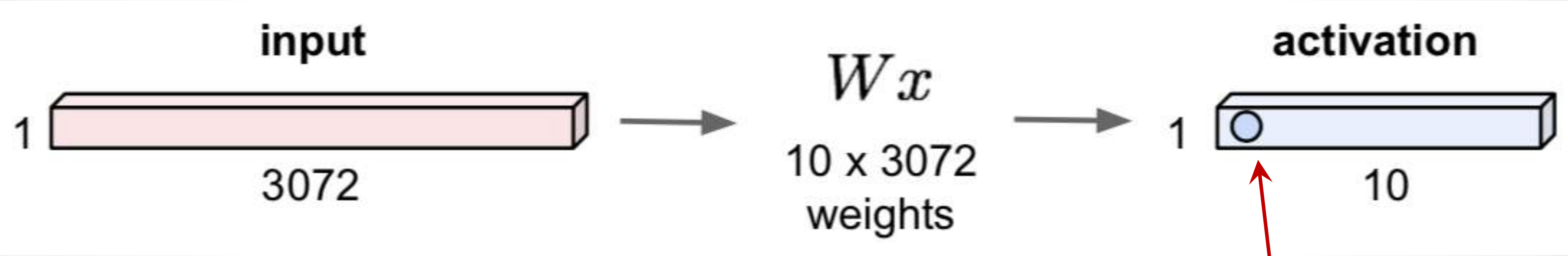
MLP in Computer Vision



- 30x32 “Input Retina”
- 5 hidden units
- 10 output units
    - Sharp Left to Sharp Right

Fully Connected Layer

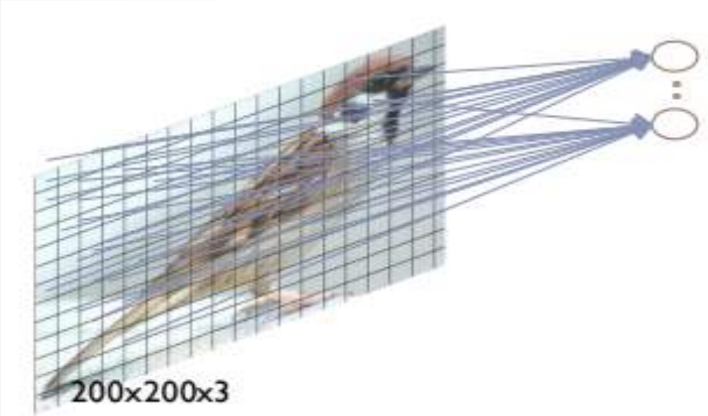
- 32x32x3 image → stretch to 3072 x 1



1 number: The result of taking a dot product of a row of W with the input (a 3072-dim. dot product)

Convolution layer

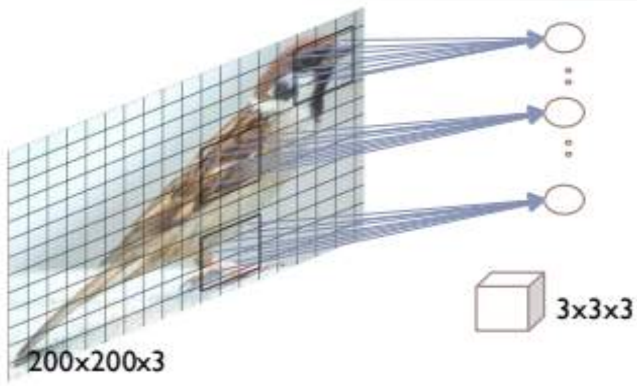
- Fully connected layer



- Image of size 200 X 200 and 3 colours (RGB)
- #Hidden Units: 120,000 (= 200X200X3)
- #Params: 14.4 billion (= 120K X 120K)
- Need huge training data to prevent over-fitting!

- Locally connected layer

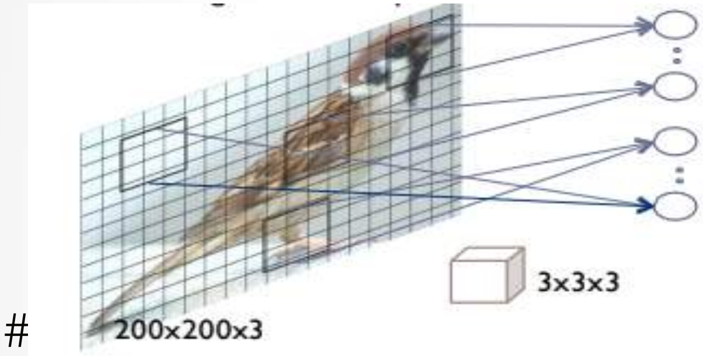
Parameter Calculations



- #
- #Params: 3.2 Million (= 120K X 27)
- Useful when the image is highly registered

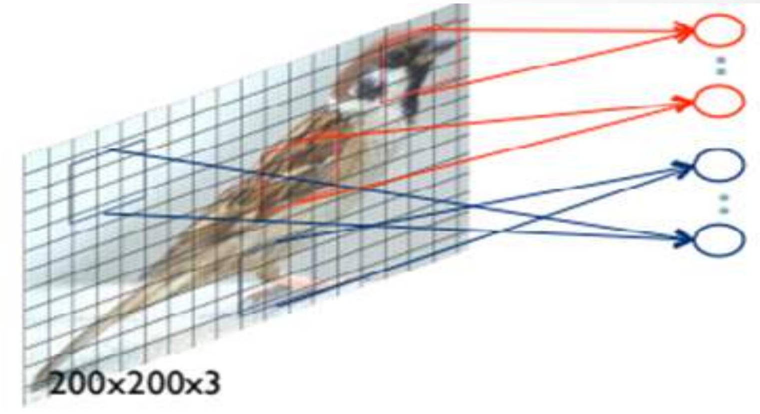
Convolution layer

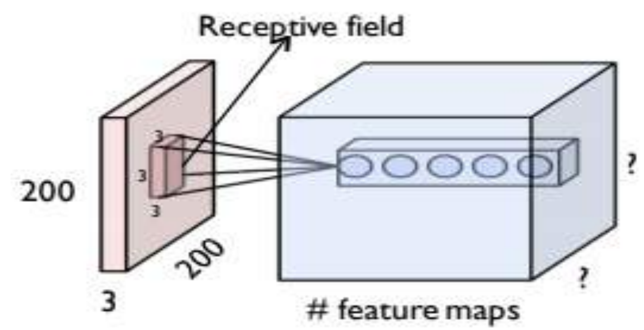
- Convolutional layer with **single** feature map.



- #
- #Params: 27 x #Feature Maps
- **Sharing parameters**
- Exploiting the stationarity property and preserves locality of pixel dependencies

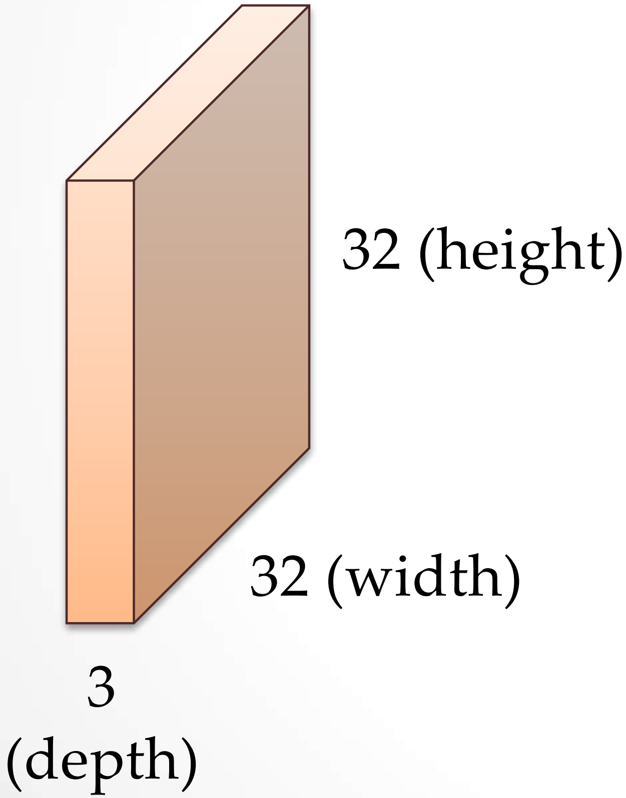
- Convolutional layer with **multiple** feature maps





Convolution Layer

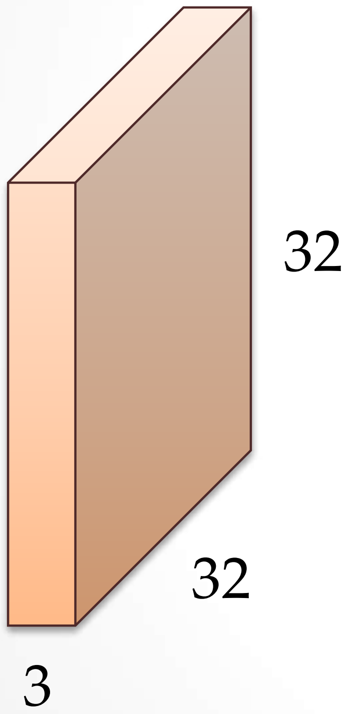
- 32 x 32 x 3 image → Preserve spatial structure



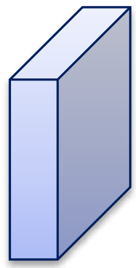
Convolution Layer

Filters always extend the full depth of the input volume

- 32 x 32 x 3 image

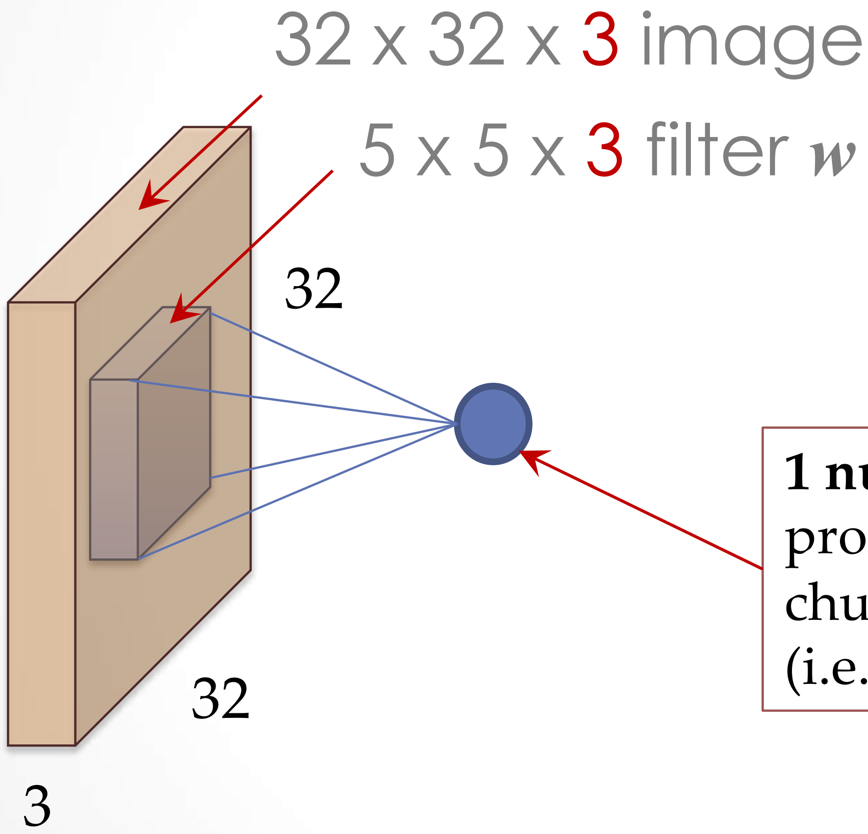


5 x 5 x 3 filter



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

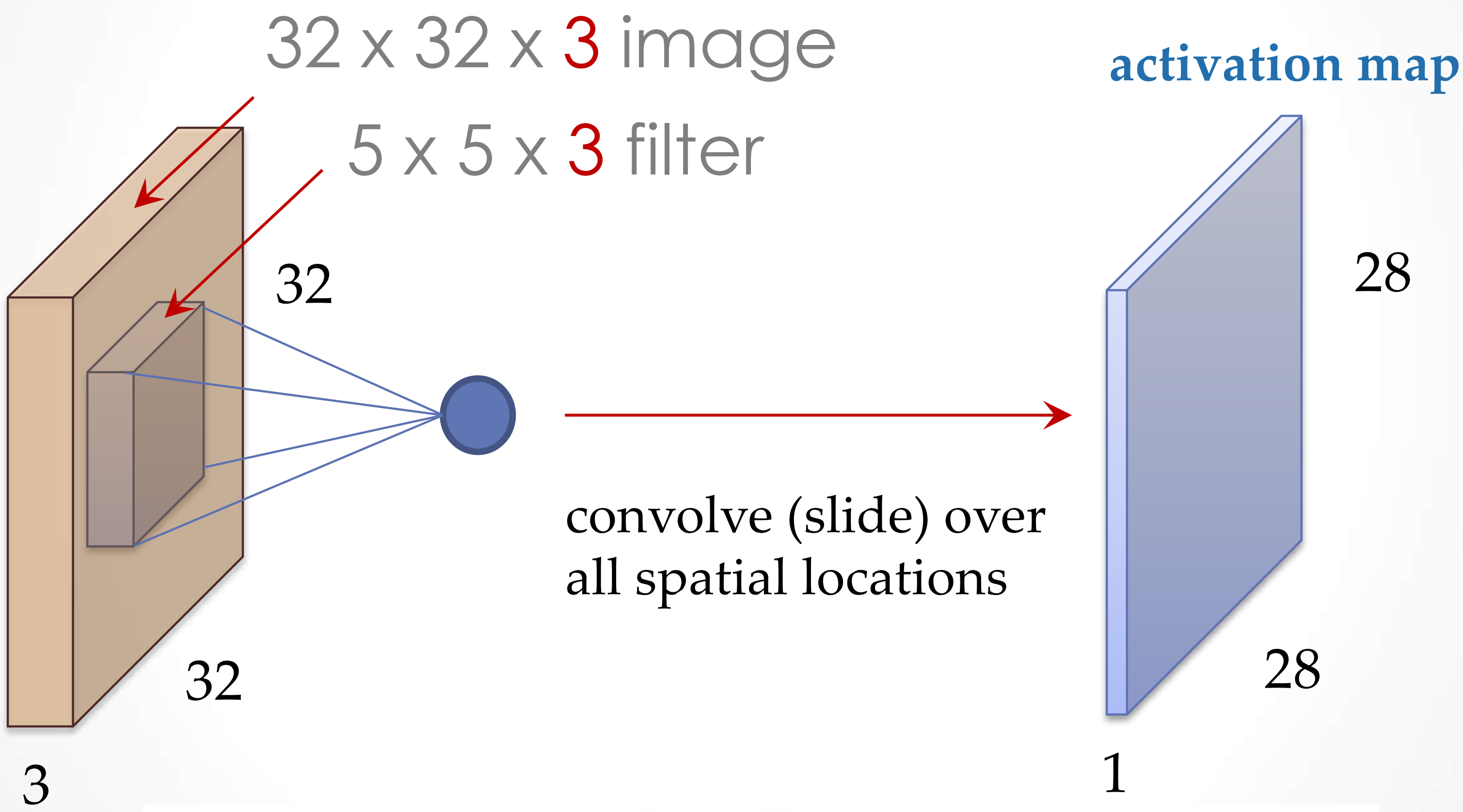
Convolution Layer



1 number: The result of taking a dot product of the filter and a small 5x5x3 chunk of the image
(i.e., 5*5*3 = 75-dim. dot product + bias)

$$\boldsymbol{w}^T\boldsymbol{x} + b$$

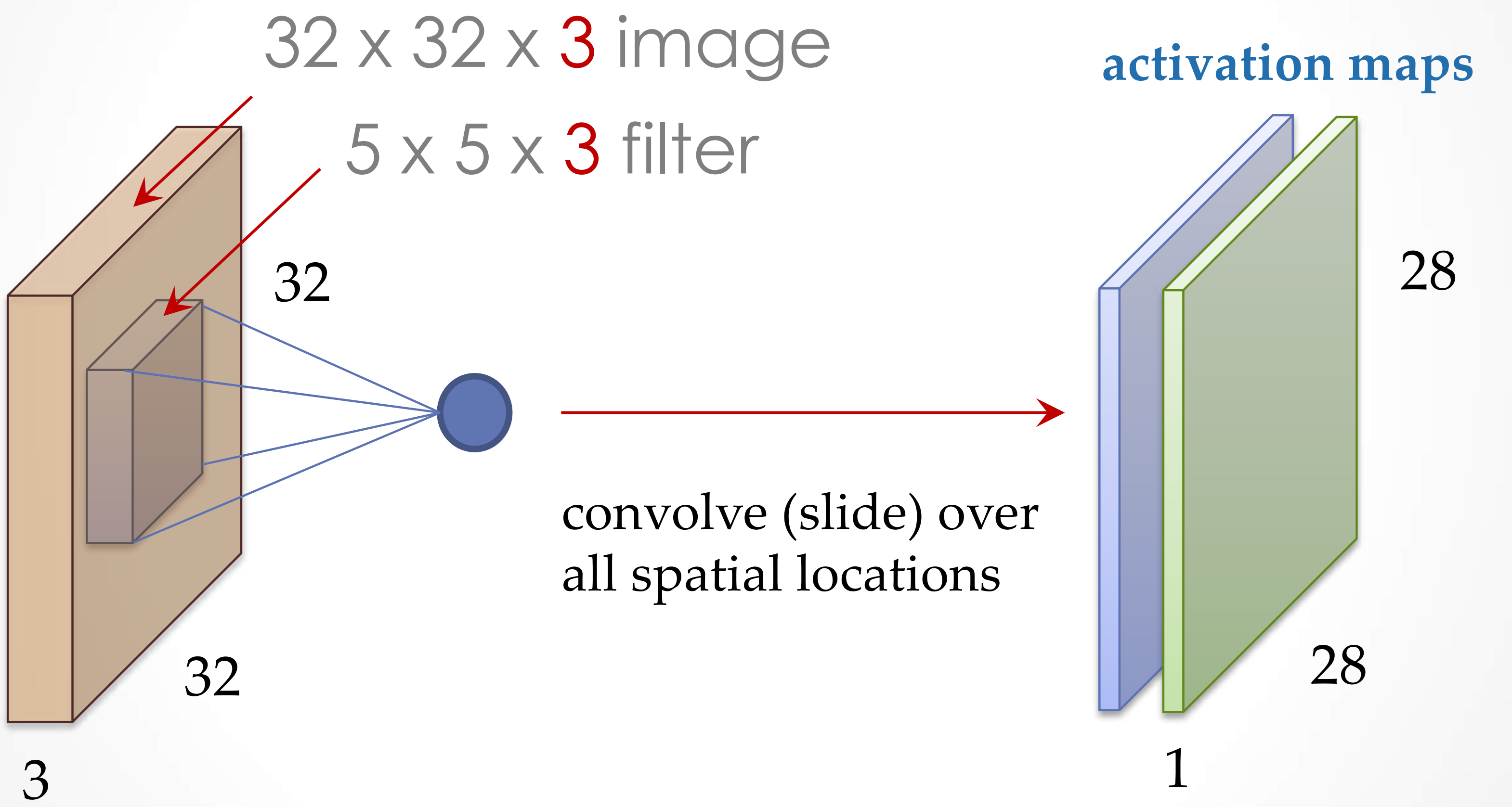
Convolution Layer



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

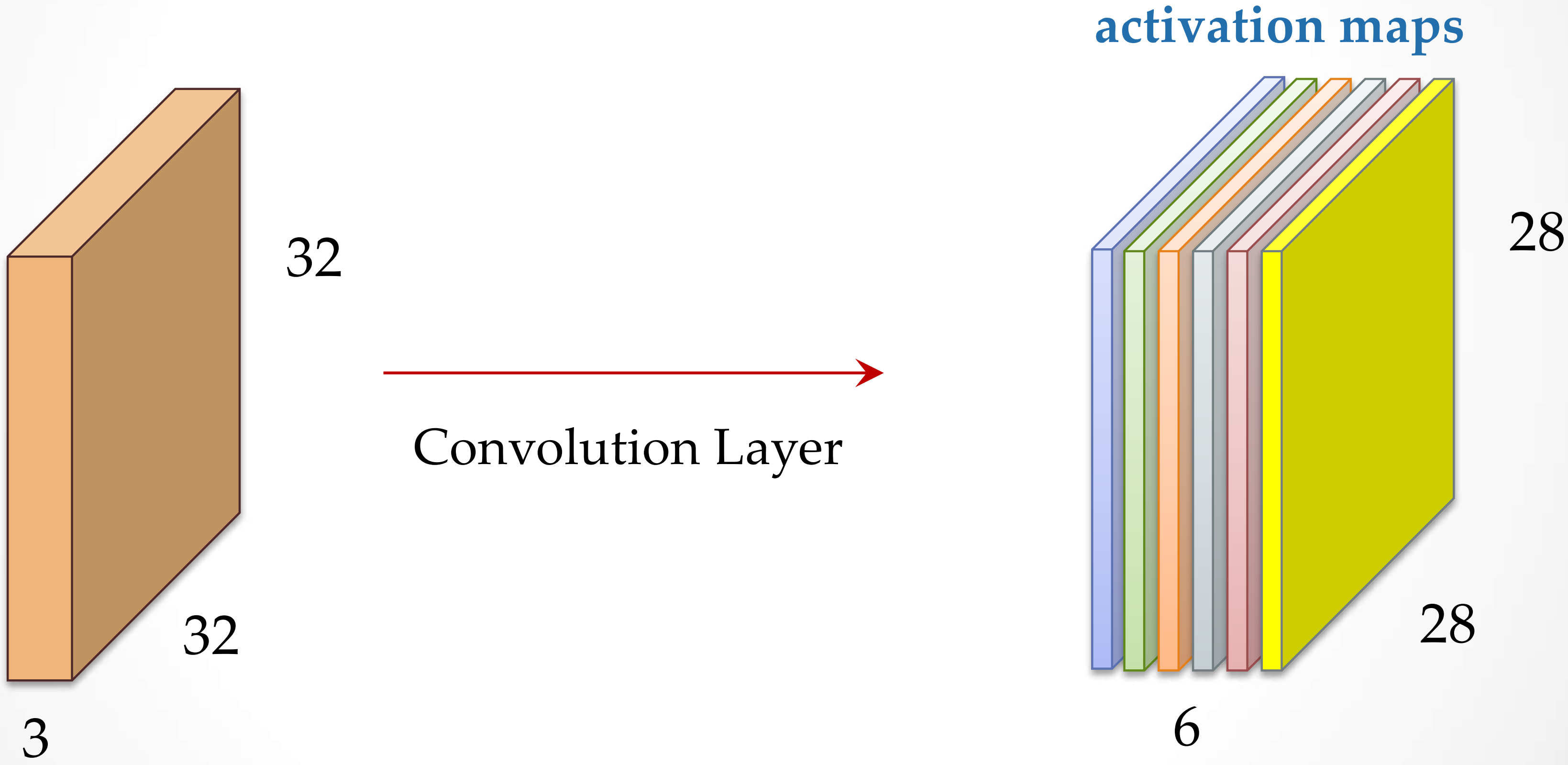
Convolution Layer

Consider a second, green filter



Convolution Layer

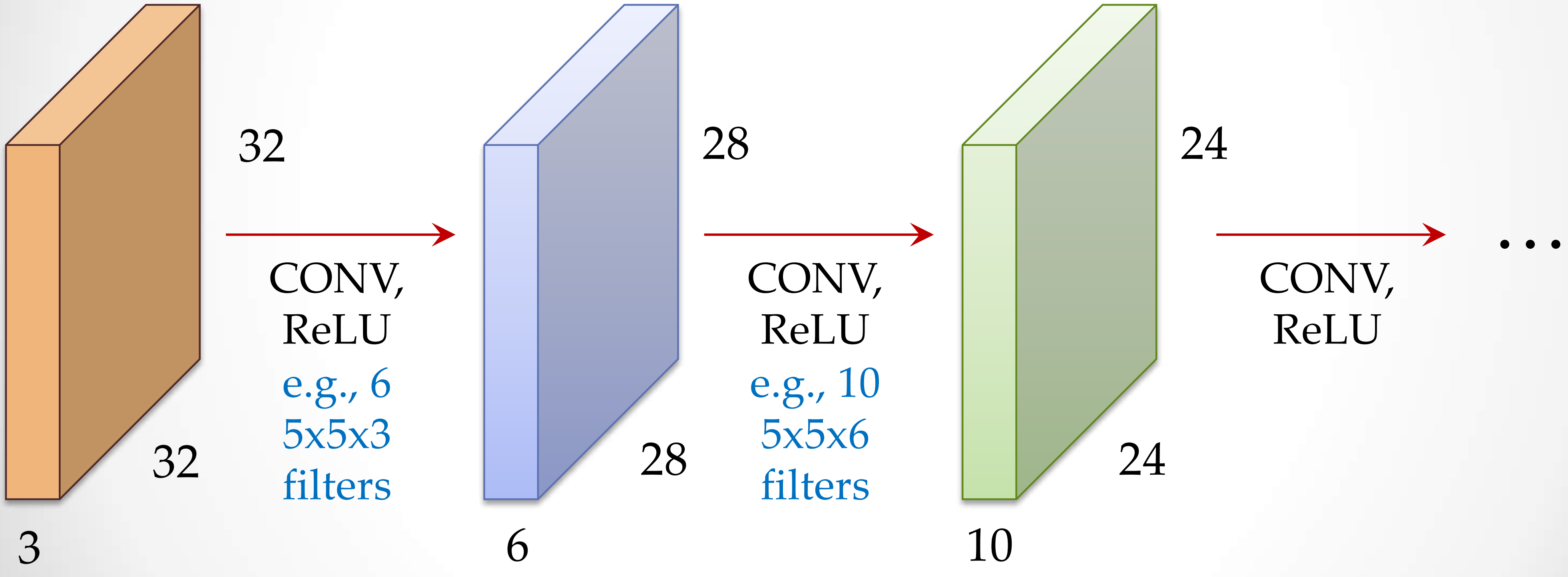
For example, if we had 6 5x5 filters, we will get 6 separate activation maps



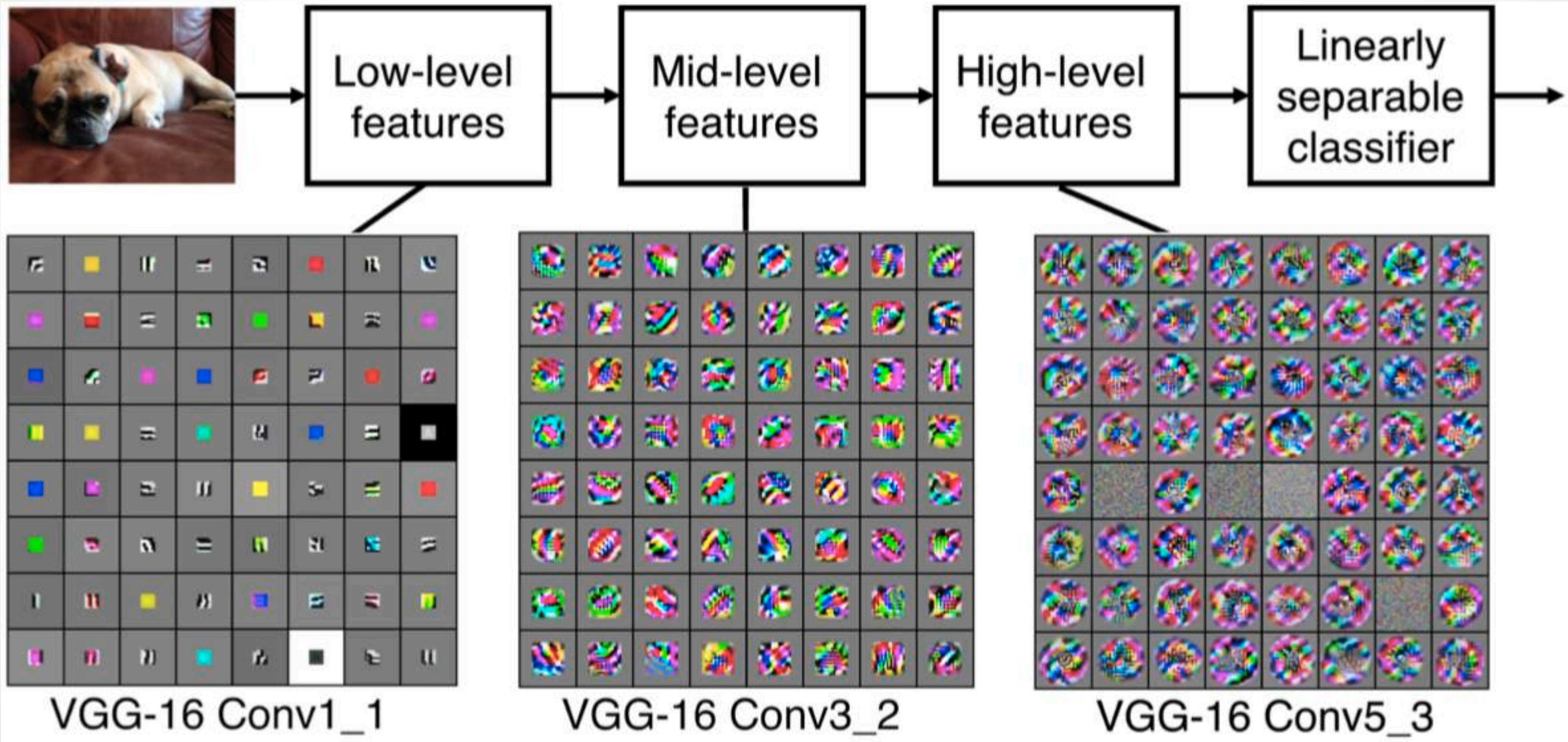
We stack these up to get a “new image” of size 28x28x6 !

Convolutional Neural Net

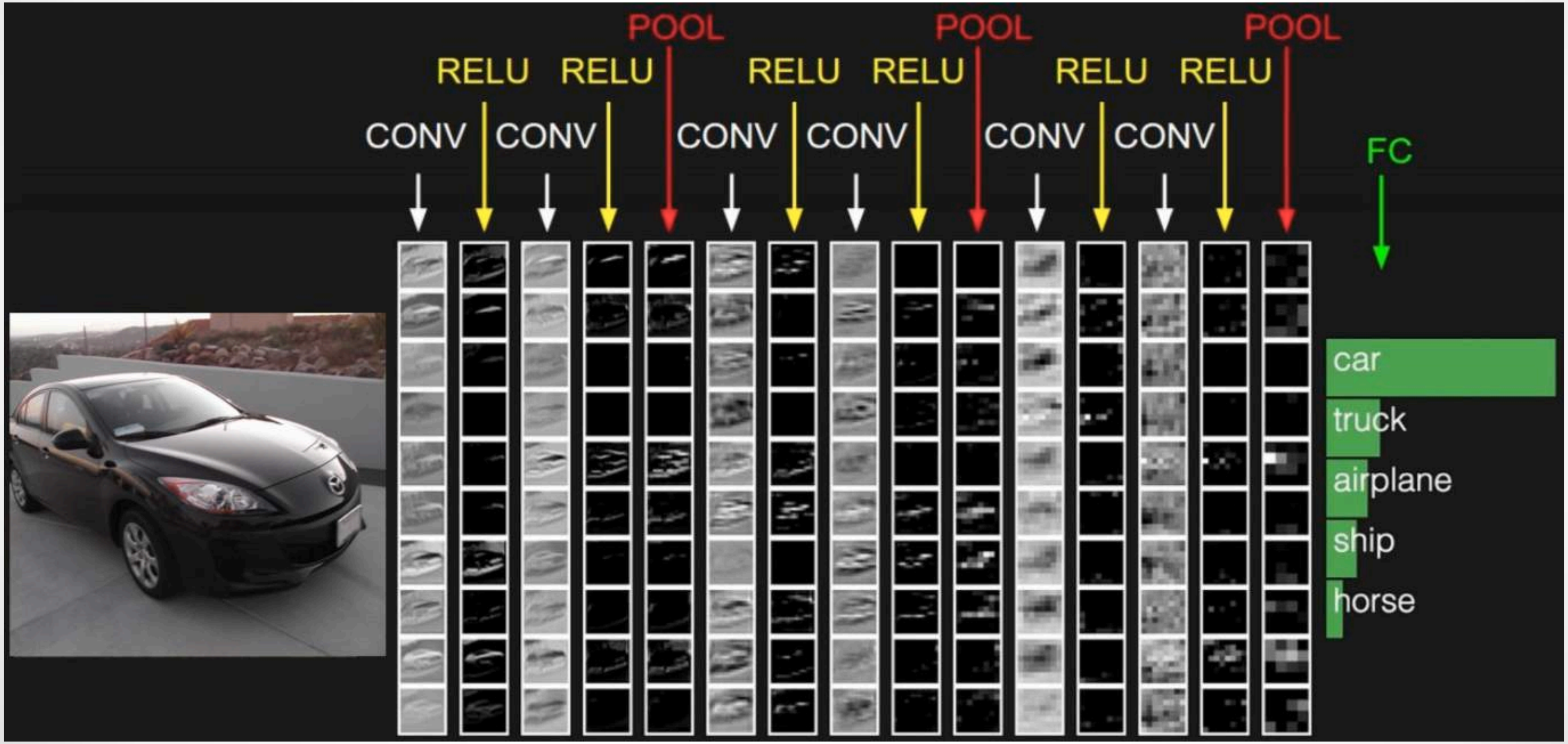
Is a sequence of Conv. Layers, interspersed with activation functions



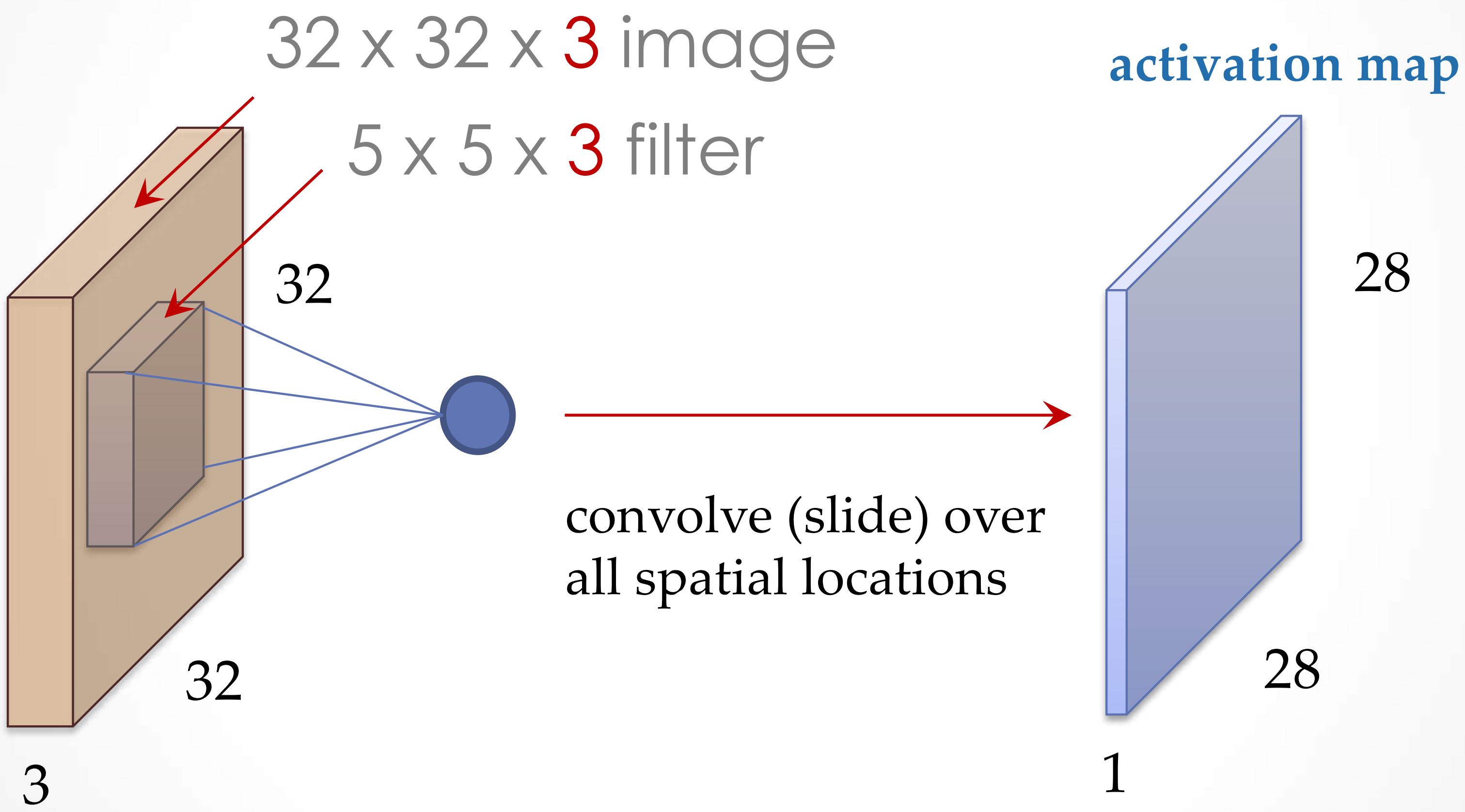
Understanding a Conv. Net.



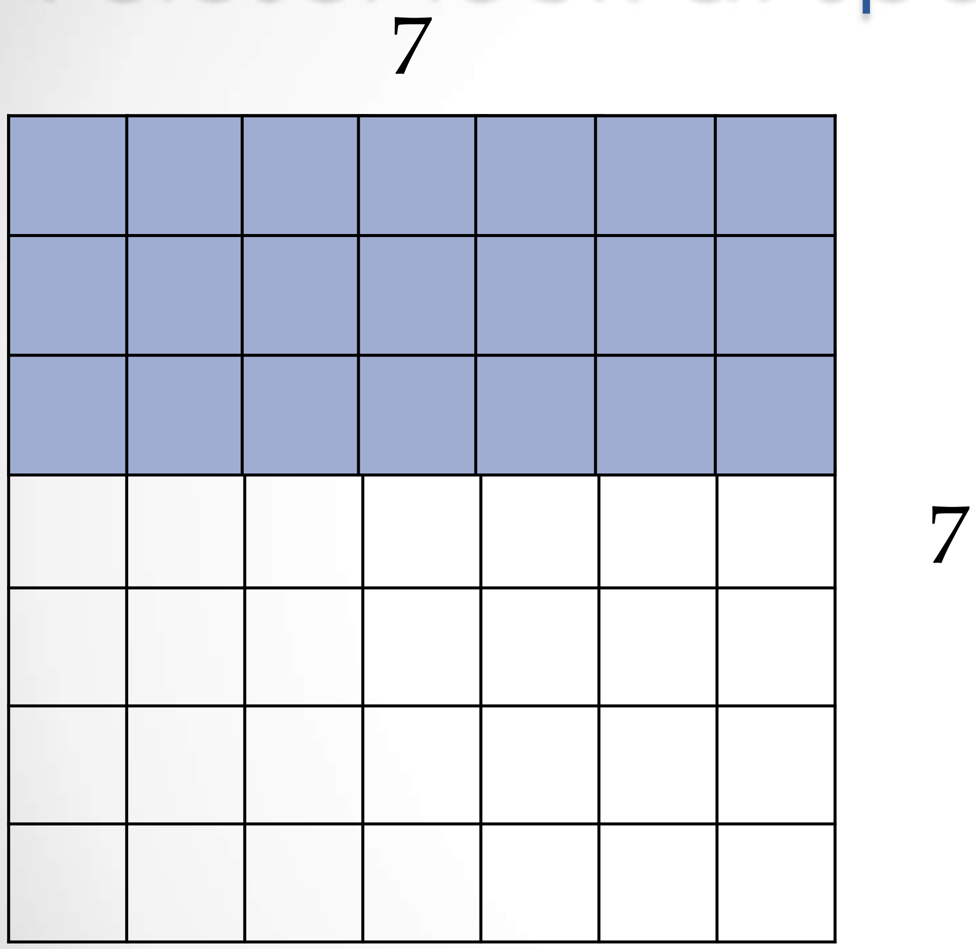
Understanding a Conv. Net.



A closer look at spatial dimensions:



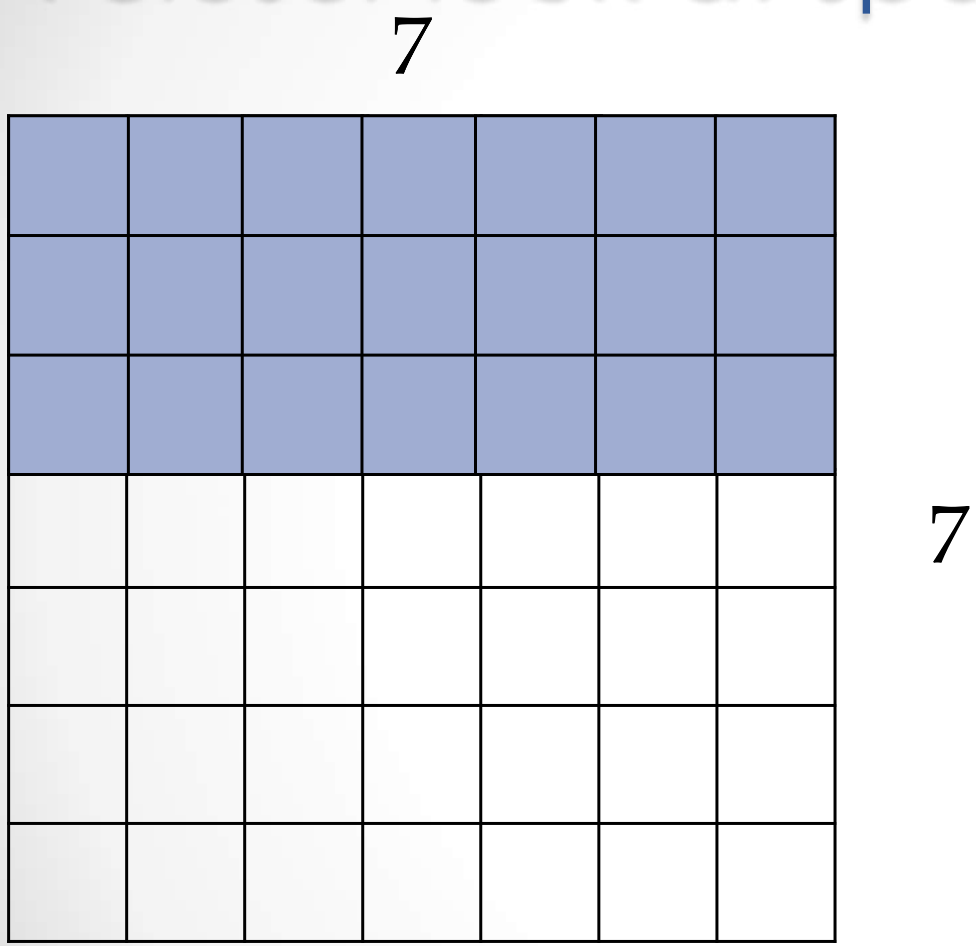
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

→ 5x5 output

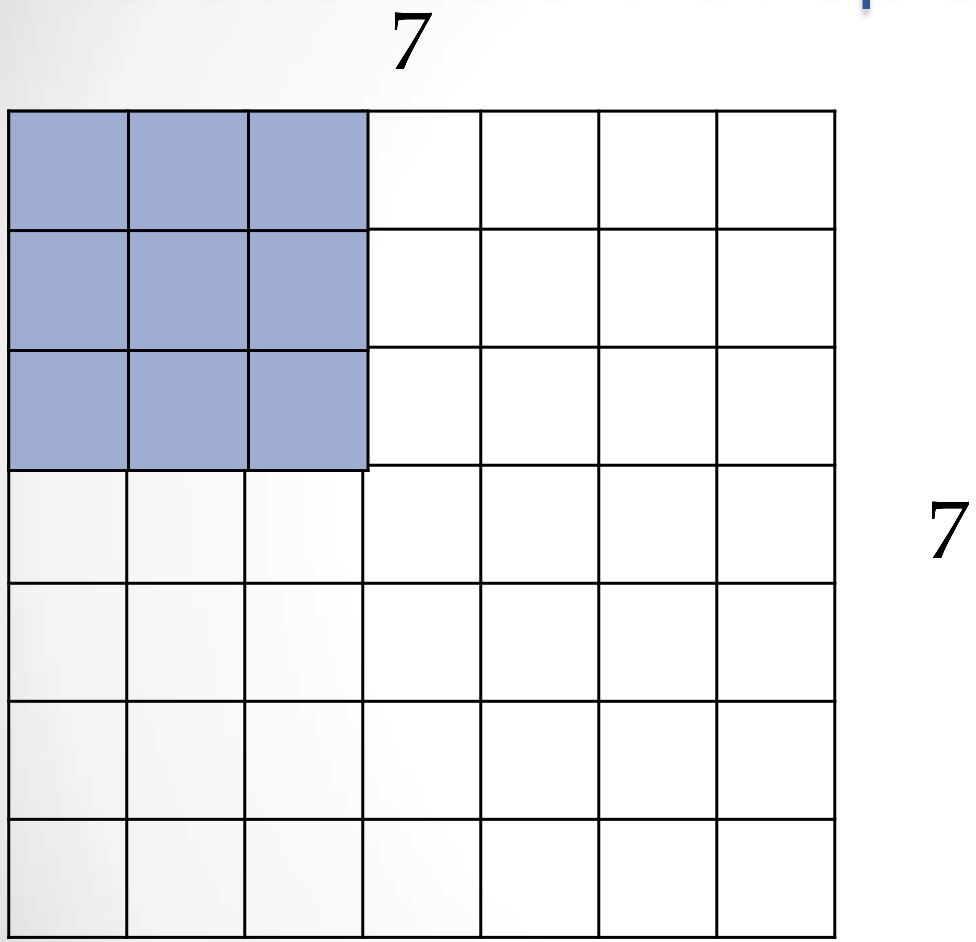
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied with stride 2

→ 3x3 output

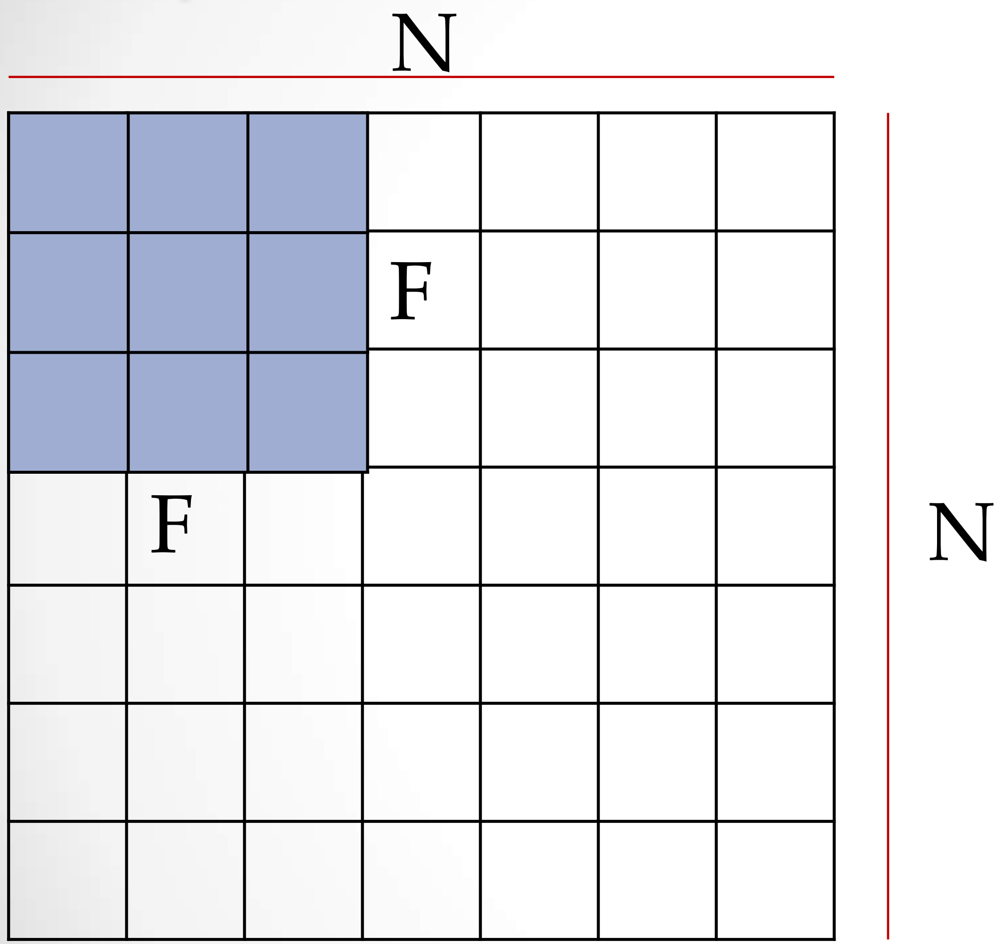
A closer look at spatial dimensions:



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

Doesn't fit !
Cannot apply 3x3 filter on a 7x7 input with stride 3

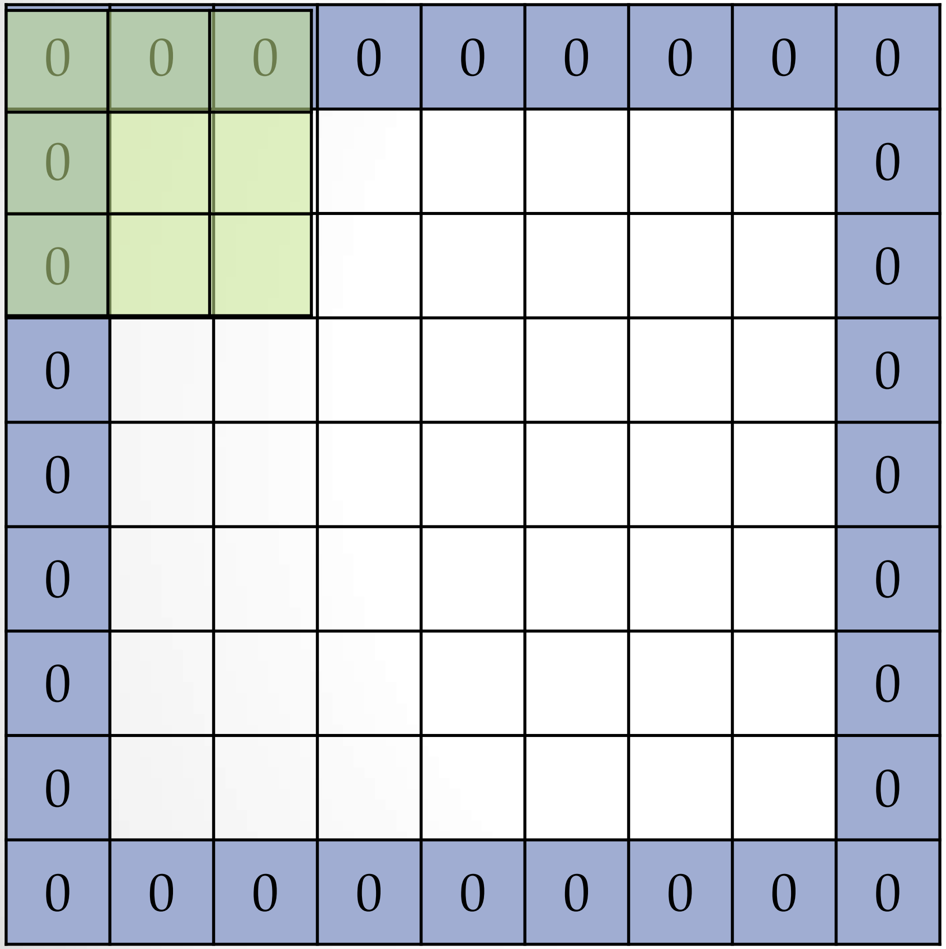
Output Dimensions:



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 !

Common to Zero-pad the border in practice



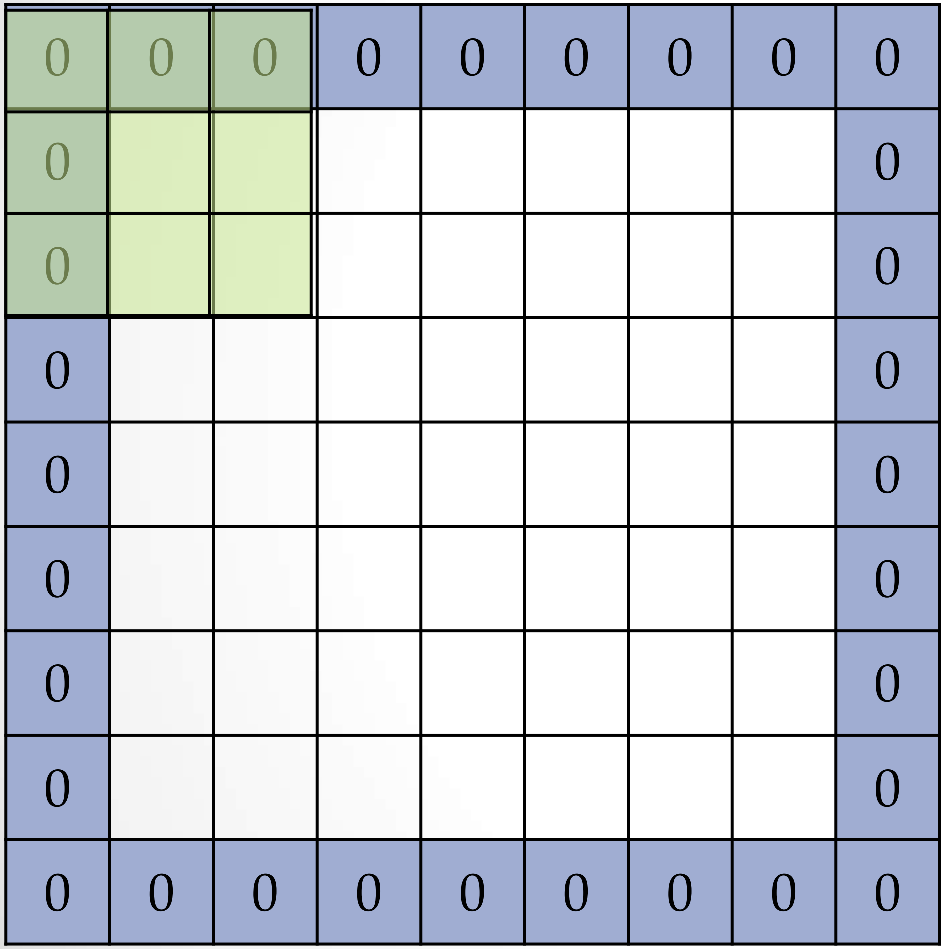
e.g., input 7x7
3x3 filter applied **with stride 1**
pad with 1 pixel border

What is the output size?

Size = 7x7

Note: output Size:
(N-F+2P)/stride + 1

Common to Zero-pad the border in practice



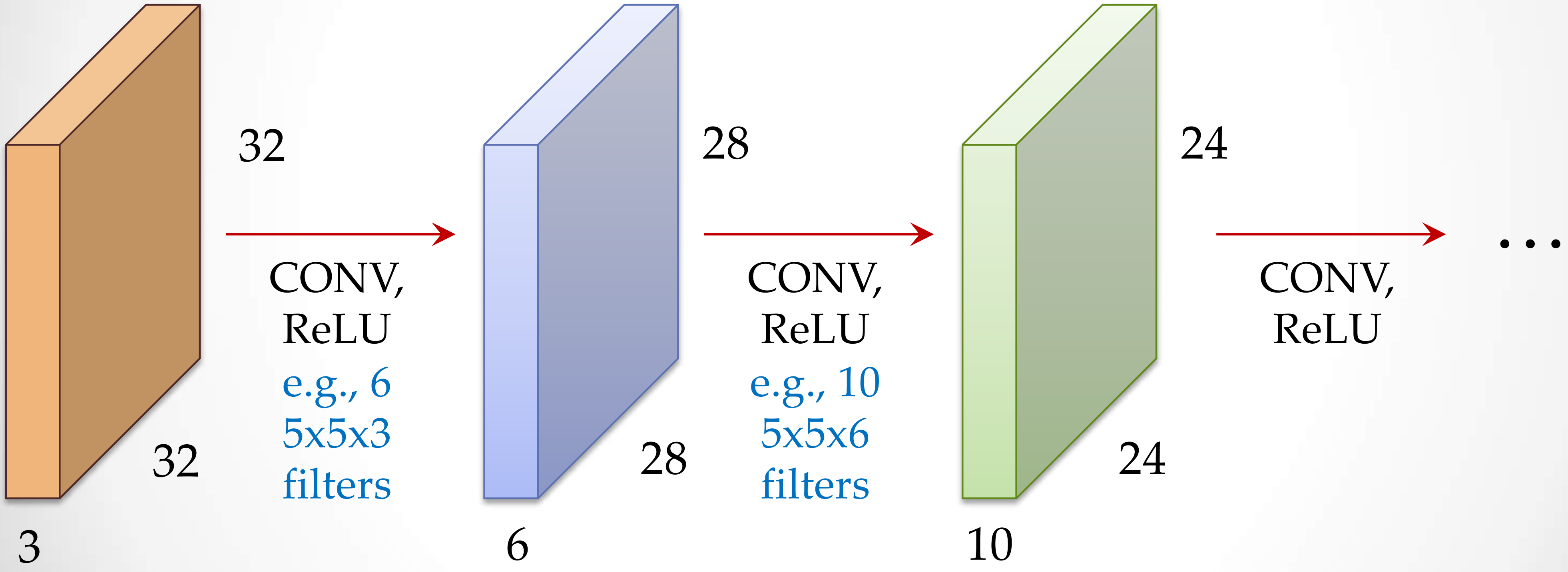
Output Size = 7x7

In general, it is common to have conv layers with **stride 1**, **filter size FxF**, and **zero padding (F-1)/2**, preserving spatial size

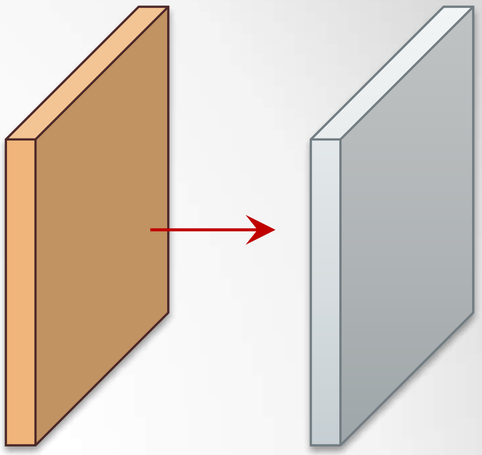
- F=3 → zero pad with 1
- F=5 → zero pad with 2
- F=7 → zero pad with 3

Recollect:

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



Example:

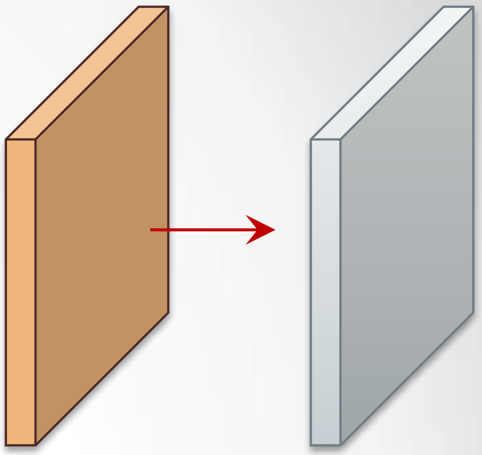


- Input Volume: **32 x 32 x 3**
- 10 **5x5** filters with stride 1, pad 2

- Output volume size?

32 x 32 x 10

Example:



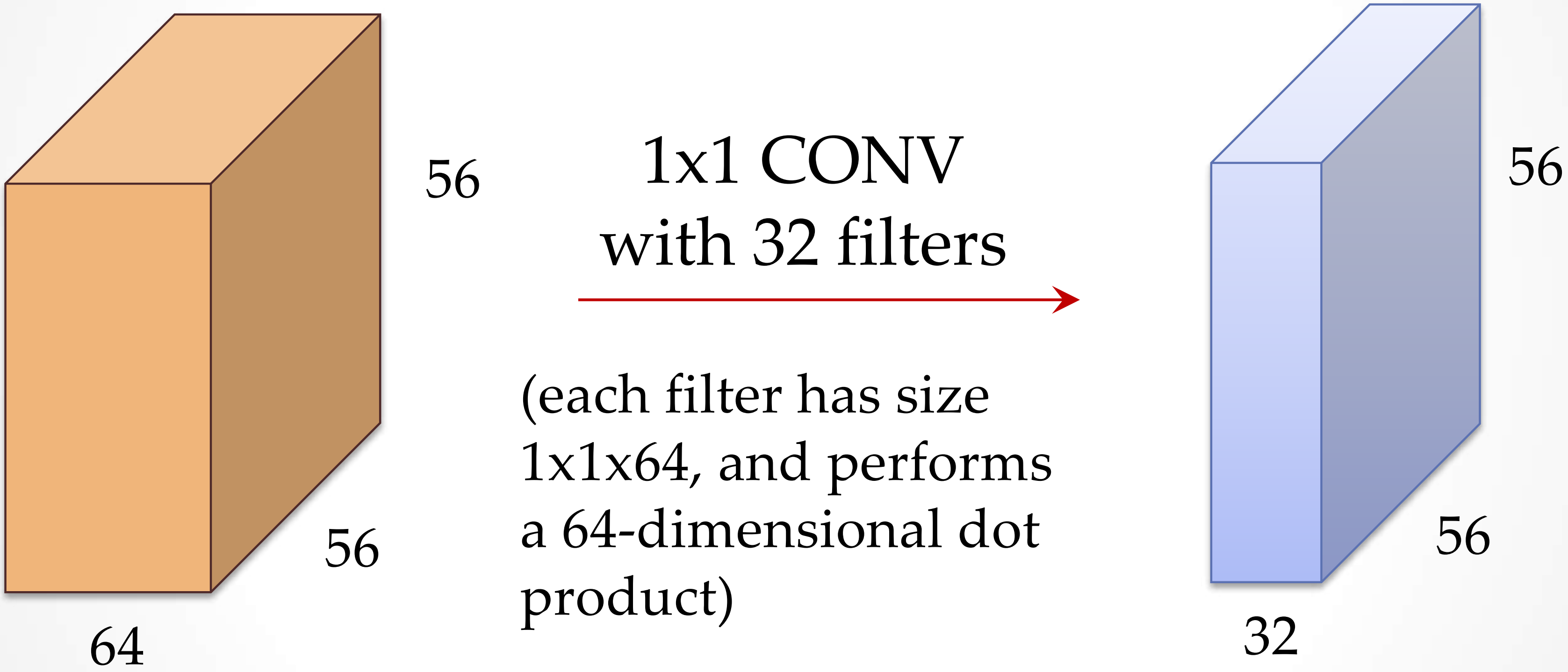
- Input Volume: **32 x 32 x 3**
- 10 **5x5** filters with stride 1, pad 2

- Number of parameters in this layer?

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

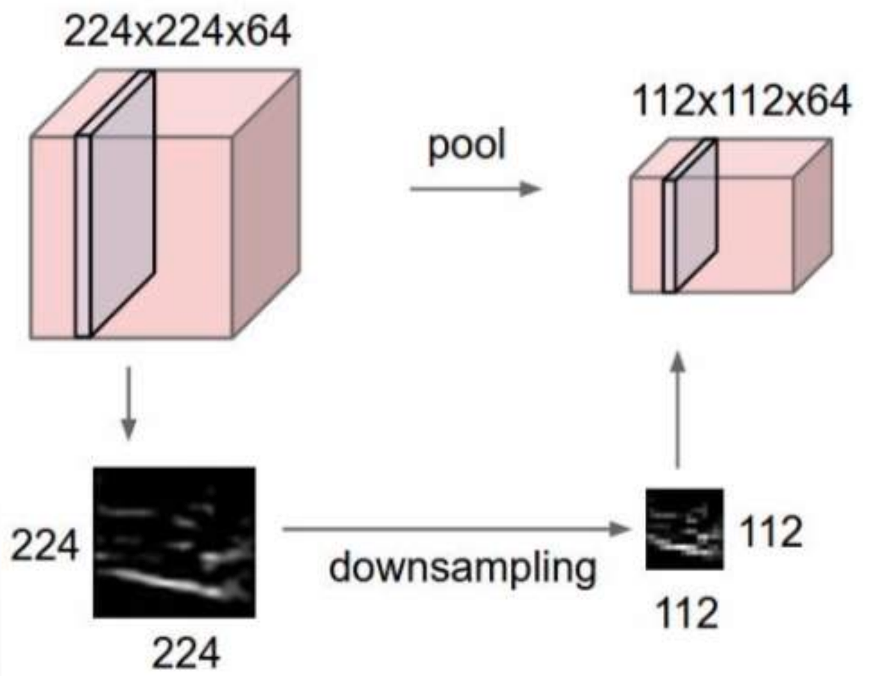
→ **76*10 = 760 parameters in the layer**

Note: 1x1 convolutions are perfectly fine

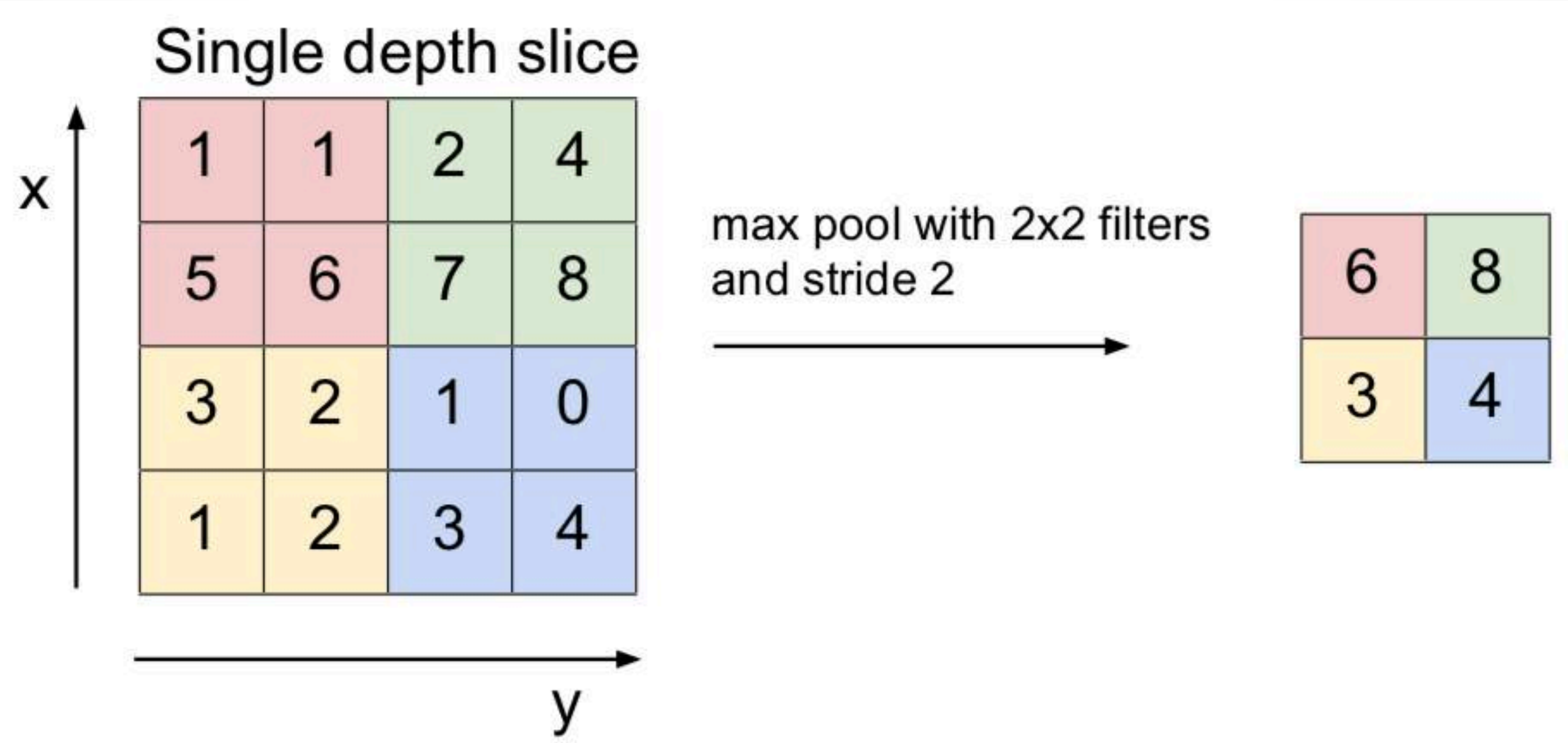


Pooling Layer

- Makes representations smaller and manageable
- Later filters have larger support
- Operates over each activation map independently



Max Pooling (2D)



Summary

- CNNs are a series of CONV, ReLU, Pool, FC layers
- CNNs are computationally efficient and compact
- Parallels to human/animal visual system.
- Learnt features can be used for classification
- Recent Trends:
  - Stick with 3x3 filters, make the network deeper
  - Improve connectivity
  - Several innovations for specific applications