DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

November 12 2020

Exam Review

DS-4400 Course objectives

- Become familiar with machine learning tasks
  - Supervised learning vs unsupervised learning
  - Classification vs Regression
- Study most well-known algorithms and understand their details
  - Regression (linear regression)
  - Classification (Naïve Bayes, decision trees, esnembles, neural networks)
- Learn to apply ML algorithms to real datasets
  - Using existing packages in R and Python
- Learn about security challenges of ML
  - Introduction to adversarial ML

What we covered

Ensembles
- Bagging
- Random forests
- Boosting
- AdaBoost

Deep learning
- Feed-forward Neural Nets
- Architectures
- Forward propagation

Linear classification
- Perceptron
- Logistic regression
- LDA

Non-linear classification
- kNN
- Decision trees
- Naïve Bayes

- Metrics
- Evaluation
- Cross-validation
- Regularization
- Gradient Descent

Linear Regression

Linear algebra

Probability and statistics

ML Models

- Categorization
  - Is it a linear or non-linear?
  - Is it generative or discriminative?
  - Is it an ensemble?
- For each ML model
  - Understand how training is done
  - Take a small example and train a model
    - In class or homework we have done linear regression, Naïve Bayes, decision tree
  - Once you have a model know how to evaluate a point and generate a prediction
    - Example: predict probability by logistic regression model

When to use each model

- Assumptions:
  - LDA assumes Gaussian data distribution
  - Naïve Bayes assumes conditional independence between features given class
- Linear models work well for linearly separable data
- Decision trees work well for categorical data
- Ensembles are powerful models
  - Need a lot of training data available

How to measure performance

- Regression: MSE
- Why we need multiple metrics
  - Accuracy, error
  - Precision, recall
  - Confusion matrix
  - F1 score
  - ROC curves, AUC
- Compute these metrics on small examples

Bias-Variance Tradeoff

- Why learning is hard
- What overfitting means
- How to avoid it
  - Regularization
  - Cross validation to report performance
- How different models improve generalization
  - Decision trees: limit tree depth
  - Linear and logistic regression: Lasso and ridge regularization
  - Ensembles randomize the training data in each model (bootstrap samples)

Type I: Conceptual

- Example 1: Describe difference between classification and regression
- Example 2: What are the two methods to design ensembles and how are they different
- Example 3: Provide advantages and disadvantages, and compare the following:
  - Linear classifiers compared to more complex ones
  - Gradient descent vs closed form solution for linear regression
  - Naïve Bayes versus LDA

Type II: Computational

- Example 1: Given a small dataset, train a particular ML model
  - E.g., linear regression, Naïve Bayes, etc.
  - Evaluate model on some small training and testing data
- Example 2: Given a particular model, describe the training process and count the number of parameters
- Example 3: Compute different metrics: true positives, false positives, precision, recall

Type III: Case Study

- Example: Consider the problem of predicting a patient’s risk to a disease. The features include demographic information (address, zip code), as well as measurements from blood test results in the last 2 years. Assume there is a datasets including patients with and without the disease.

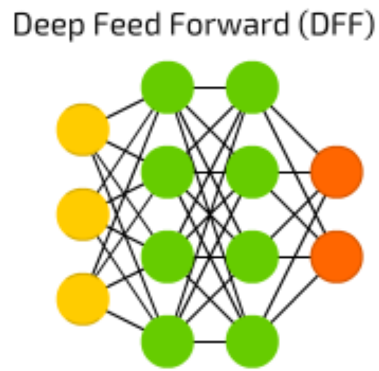
Describe the process to:

1. Represent the features in a format suitable for ML
2. How would you do feature selection
3. Describe what models you would use and why

Neural Network Architectures

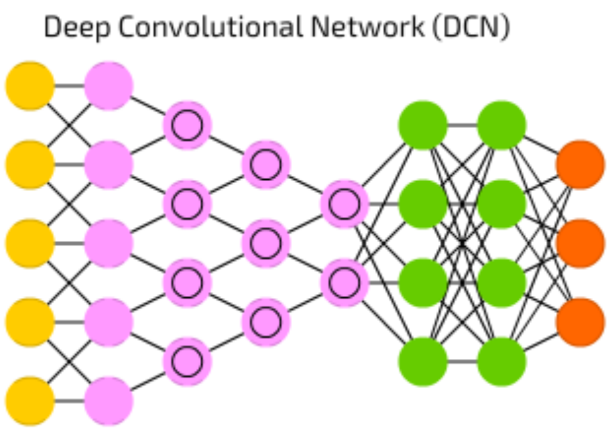
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



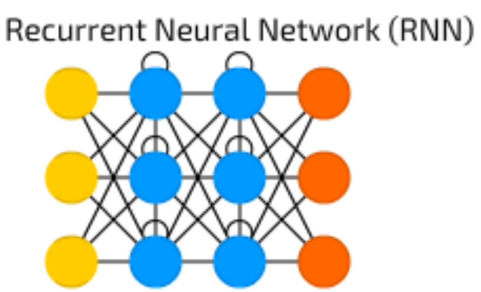
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



Recurrent Networks

- Keep hidden state
- Have cycles in computational graph

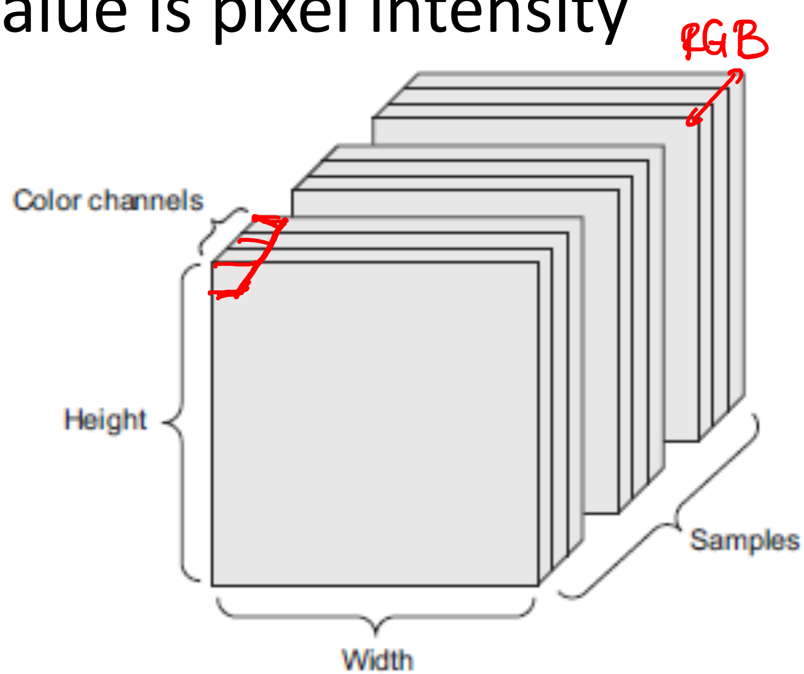


Convolutional Nets

- Particular type of Feed-Forward Neural Nets
  - Invented by [LeCun 89]
- Applicable to data with natural grid topology
  - Time series
  - Images
- Use convolutions on at least one layer
  - Convolution is a linear operation that uses local information
  - Also use pooling operation
  - Used for dimensionality reduction and learning hierarchical feature representations

Image Representation

- Image is 3D “tensor”: height, width, color channel (RGB)
- Black-and-white images are 2D matrices: height, width
  - Each value is pixel intensity



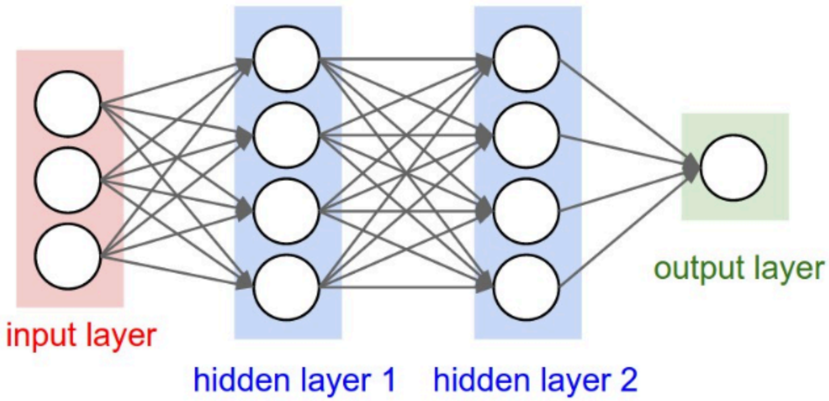
Computer vision principles

- Task: image classification (object identification)
- Translation invariance
  - Classification should work if object appears in different locations in the image => All image regions are treated the same
- Locality
  - Focus on local regions for object detection => computation should be local
- Mathematical operation: Convolution

Convolutional Neural Networks

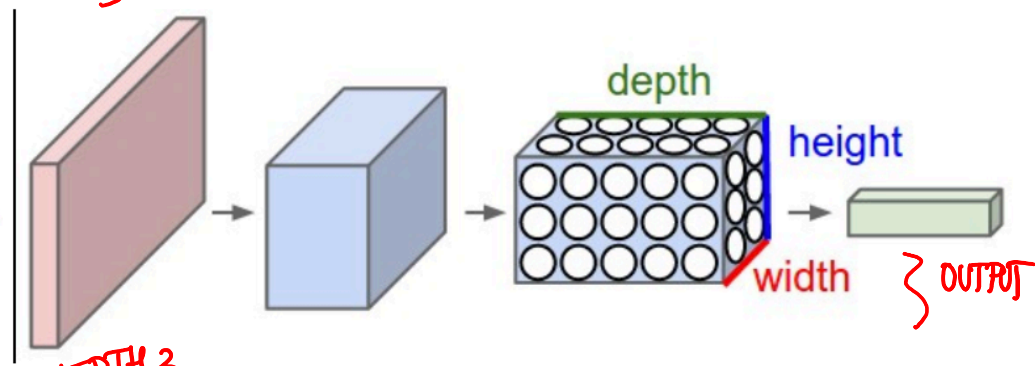
VECTORS

Feed-forward network



Convolutional network

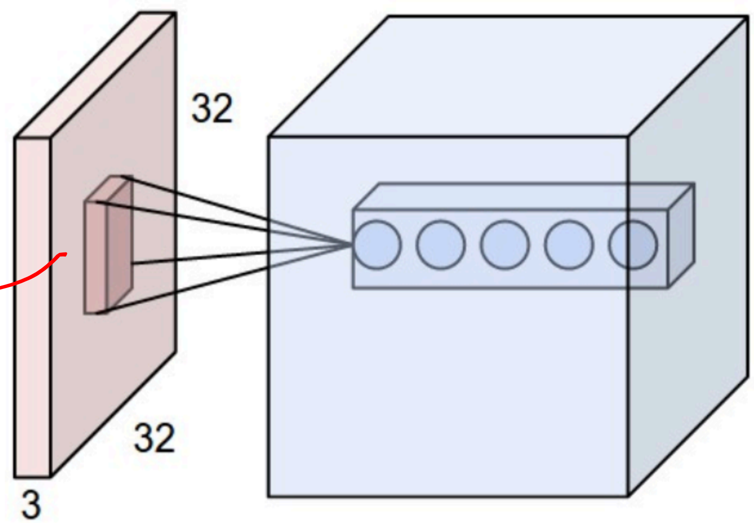
3D



DEPTH 3

OUTPUT

Filter



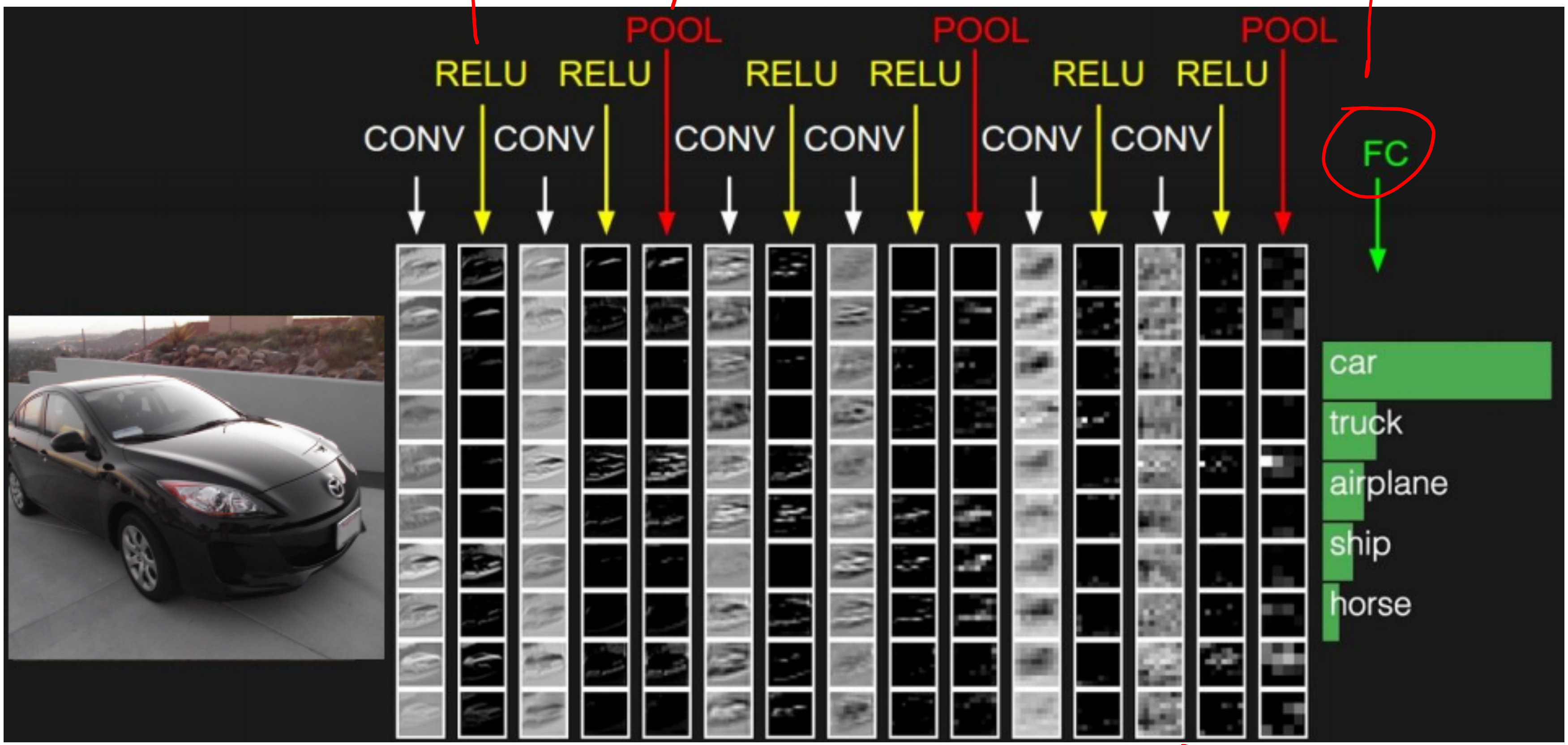
https://www.youtube.com/watch?v=f0t-OCG79-U

Convolutional Nets

DENSE

ACTIVATION

DIM. REDUCTION



INPUT

OUTPUT

TENSORS (3D)

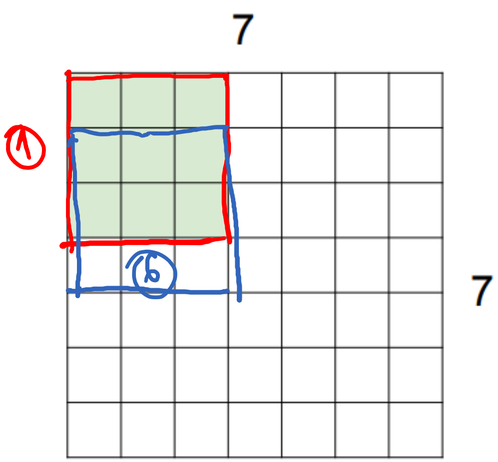
LINEAR

VECTOR

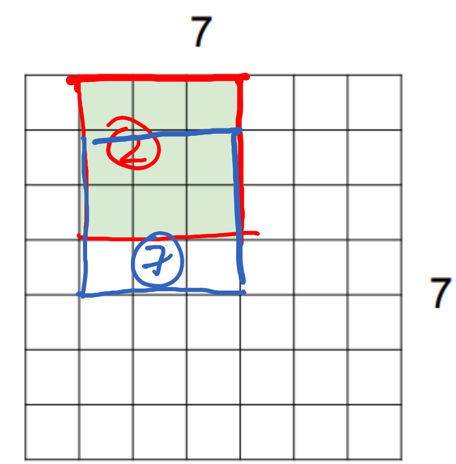
Convolutions

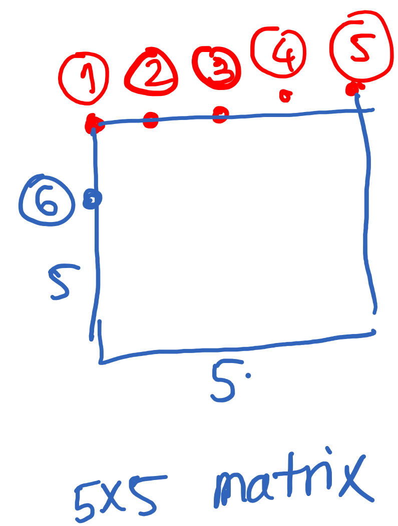
2D

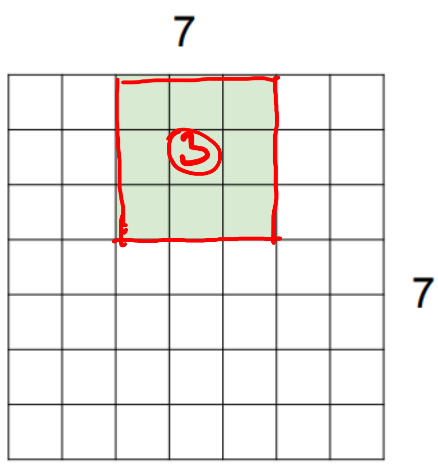
A closer look at spatial dimensions:

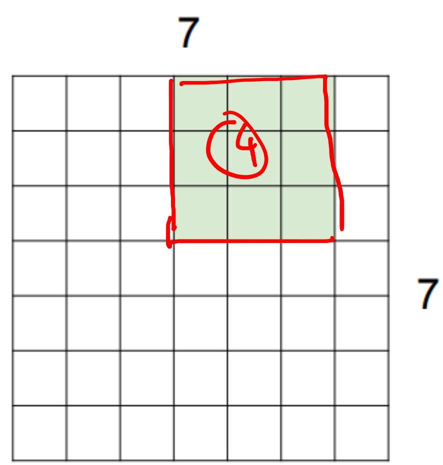


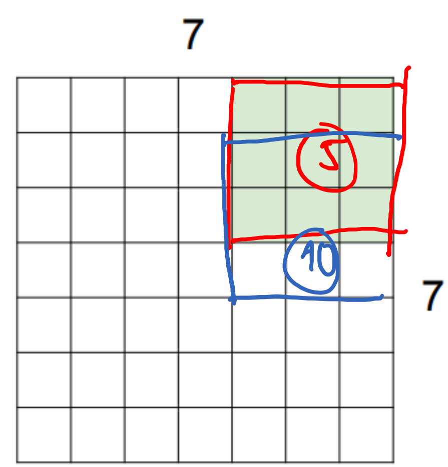
7x7 input (spatially)
assume 3x3 filter











Example

R2

R1

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

2

2

Input Filter Output

REGION 1

0	1	3	4

DOT

FILTER

0	1	2	3

$0 \cdot 0 + 1 \cdot 1 + 3 \cdot 2 + 4 \cdot 3$ + BIAS

REGION 2

1	2	4	5

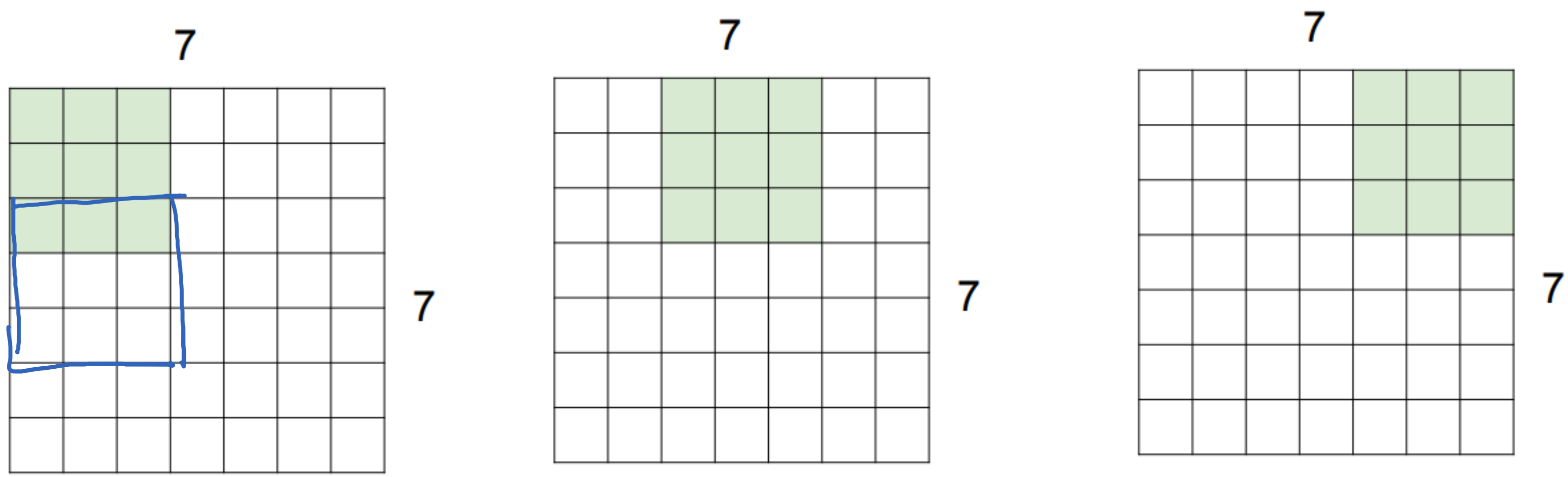
FILTER

0	1	2	3

CONVOLUTION

Convolutions with stride

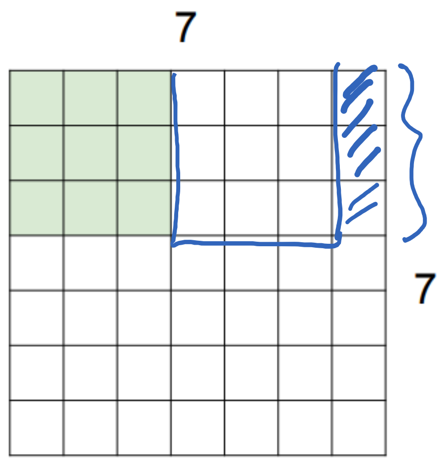
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

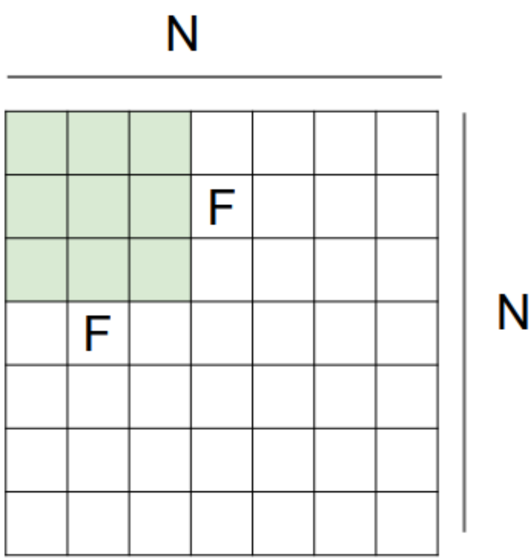


=) 3X3 OUTPUT

Convolutions with stride

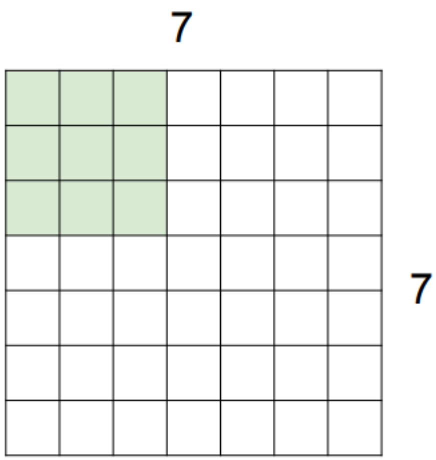
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**



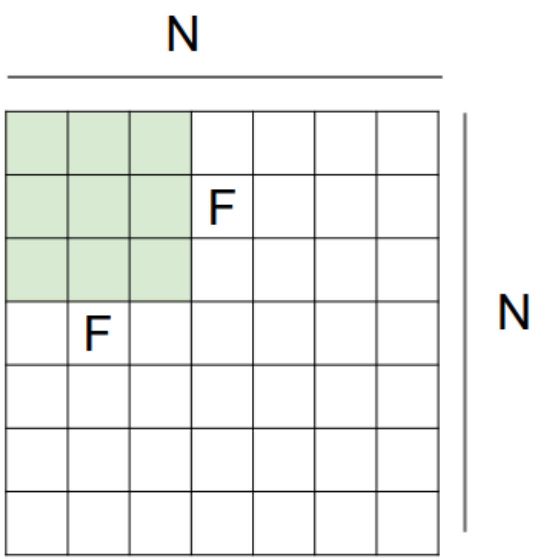


Convolutions with stride

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



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

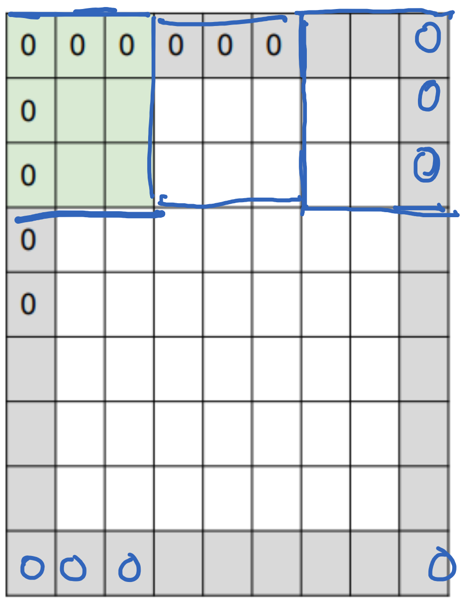


Output size:
(N - F) / stride + 1

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 :\

Padding

In practice: Common to zero pad the border



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

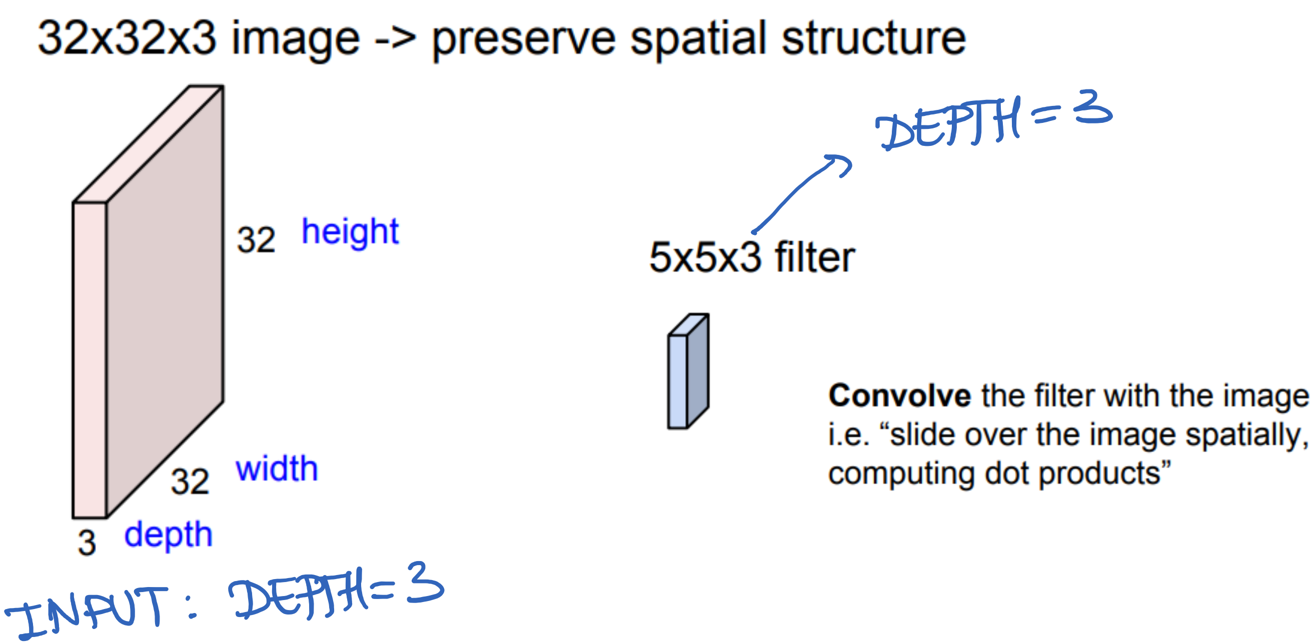
(recall:)
(N - F) / stride + 1

PAD = 1.

OUTPUT: 3X3

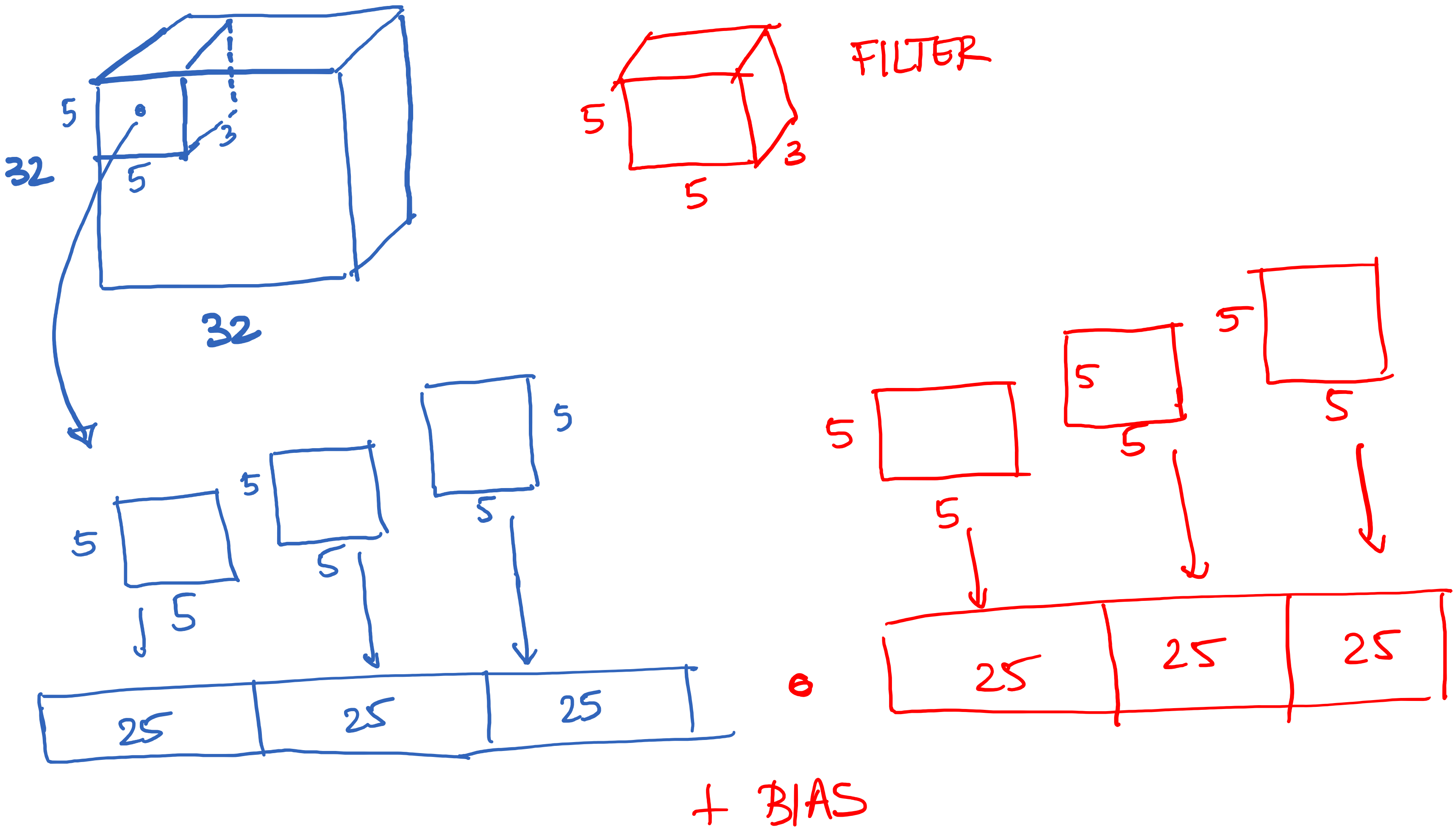
3D

Convolution Layer

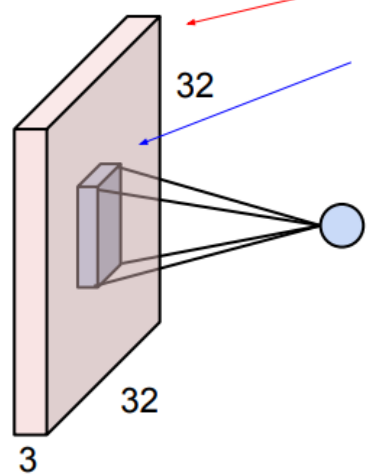


- Depth of filter always depth of input
- Computation is based only on local information

Convolution Operation



Convolution Layer



32x32x3 image
5x5x3 filter w
BIAS b

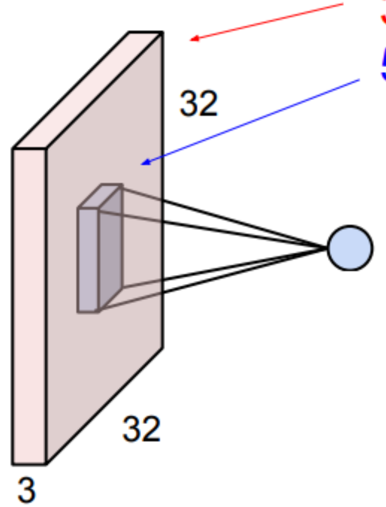
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$ LINEAR

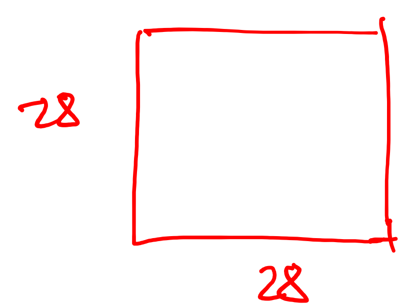
28X28 Output.

32x32x3 image
5x5x3 filter

STRIDE 1

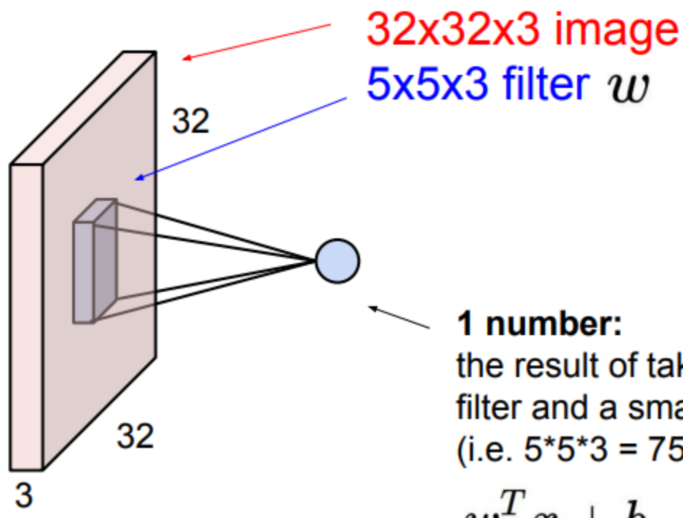


convolve (slide) over all spatial locations



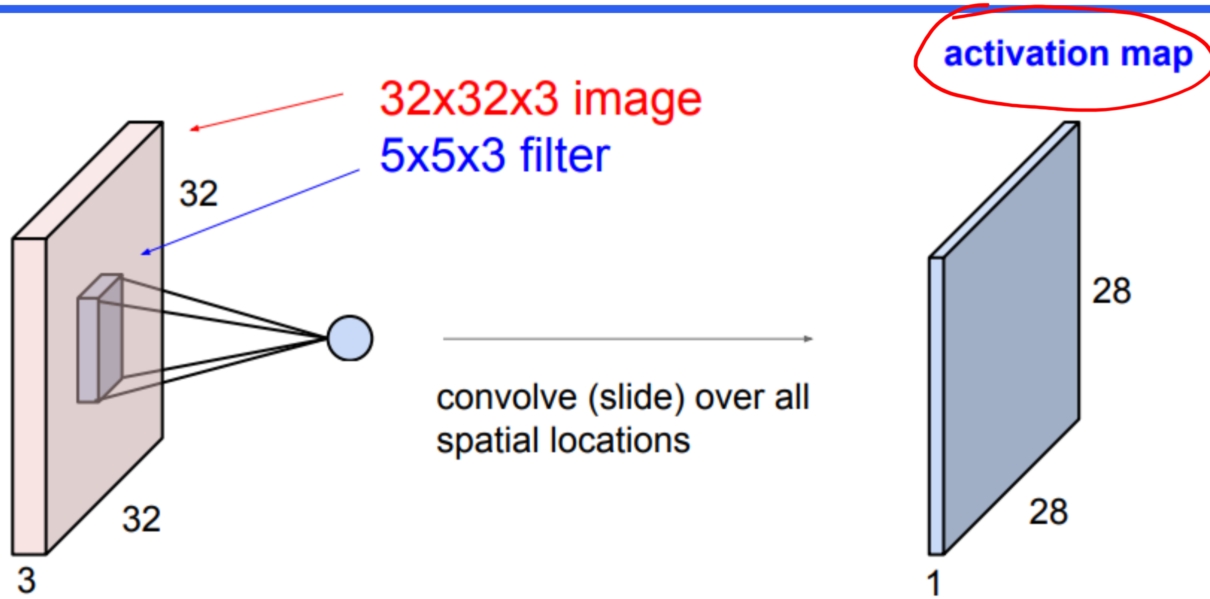
DEPTH = 1

Convolution Layer



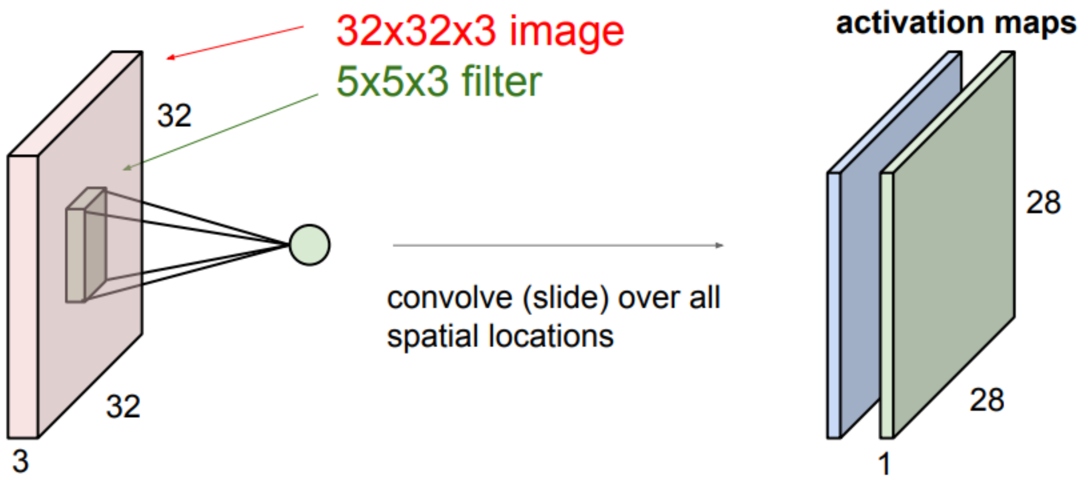
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

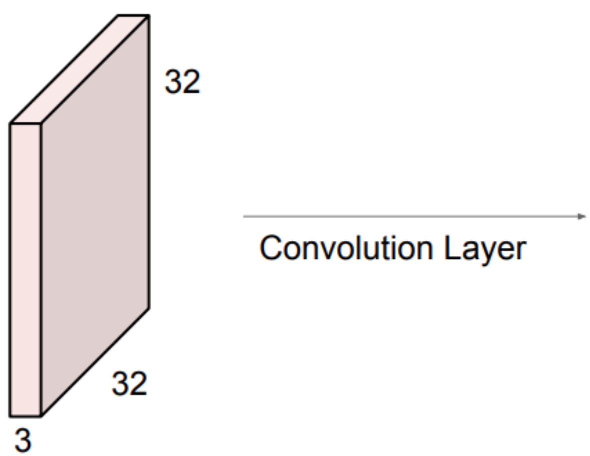


CONV WITH 1 FILTER

Convolution Layer

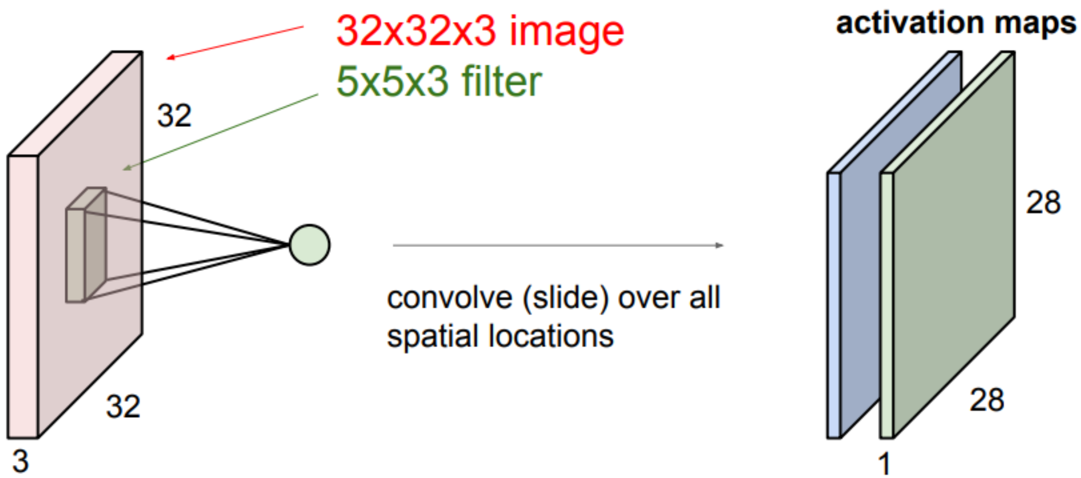


Second, green filter

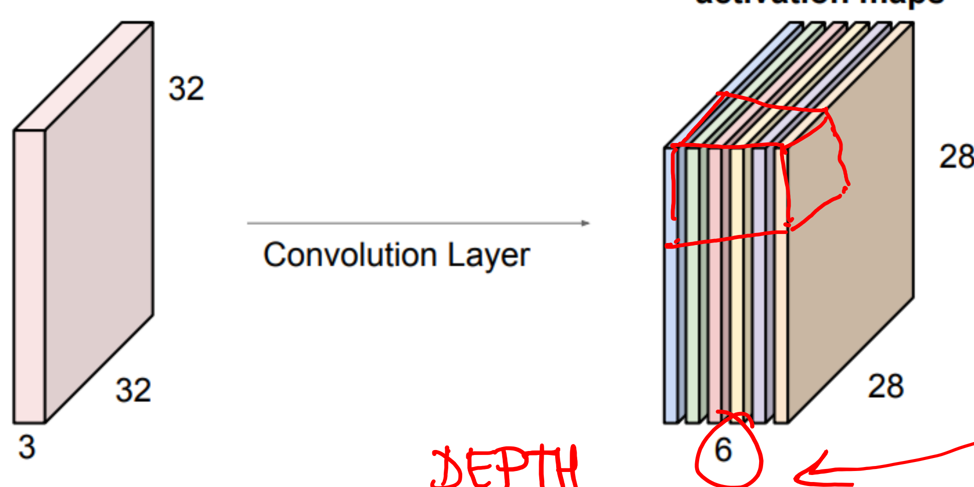


6 filters

Convolution Layer



Second, green filter



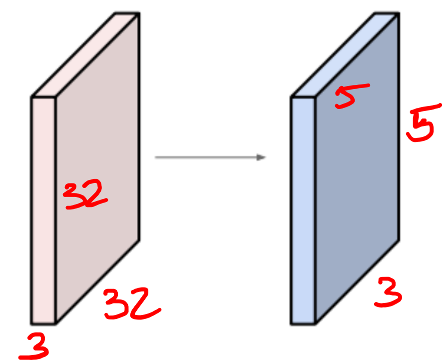
CONV LAYER

FILTER HAS DEPTH 6

6 filters

Examples

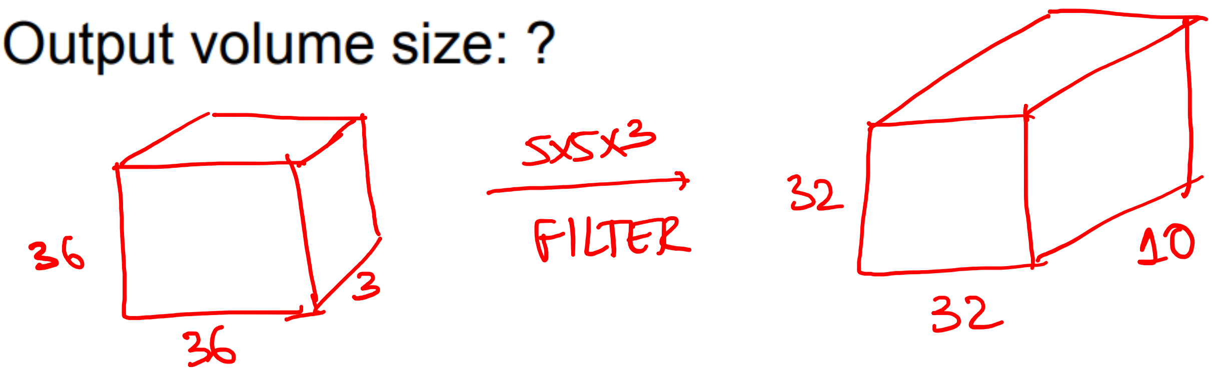
Examples time:



Input volume: **32x32x3**

10 5x5x3 filters with stride 1, pad 2

Output volume size: ?



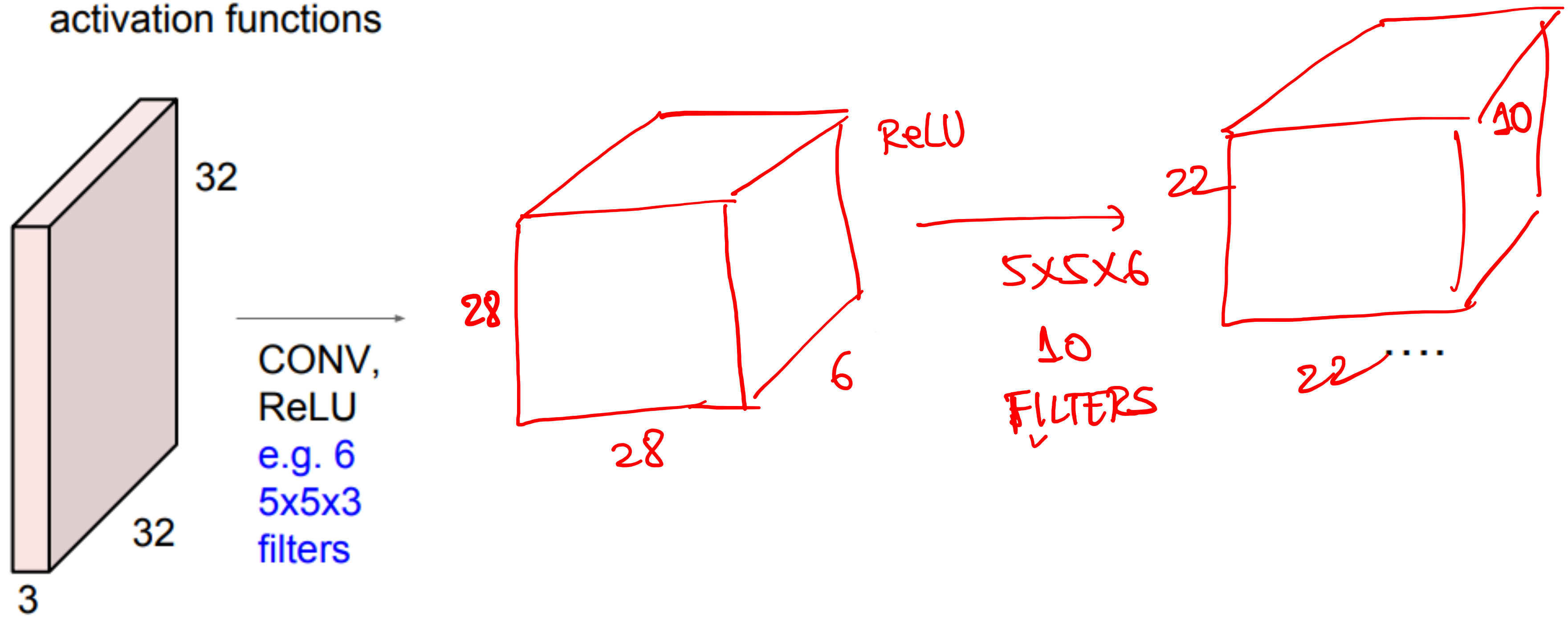
Number of parameters in this layer?

EACH FILTER : 5x5x3 ⤳ 75 + 1 BIAS = 76

10 FILTERS : 760

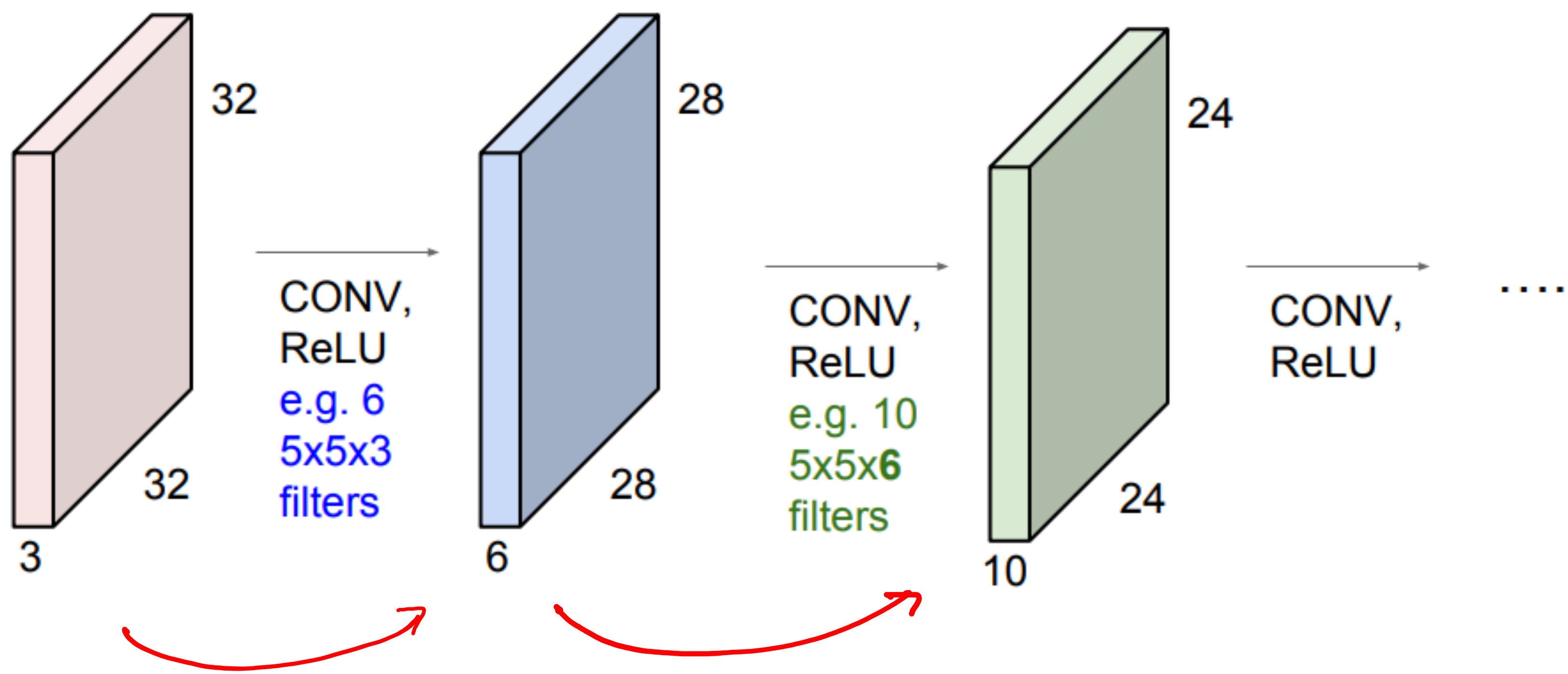
Convolutional Nets

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



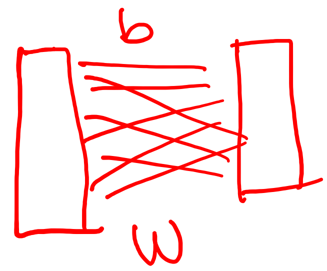
Convolutional Nets

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



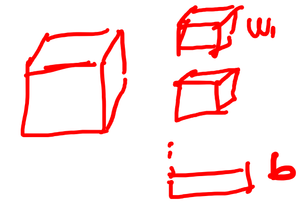
Summary: Convolution Layer

FFNN



TRAIN W,b

CNN



TRAIN FILTER PARAMS AND BIAS

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.

$F \times F \times D_1$

- 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.

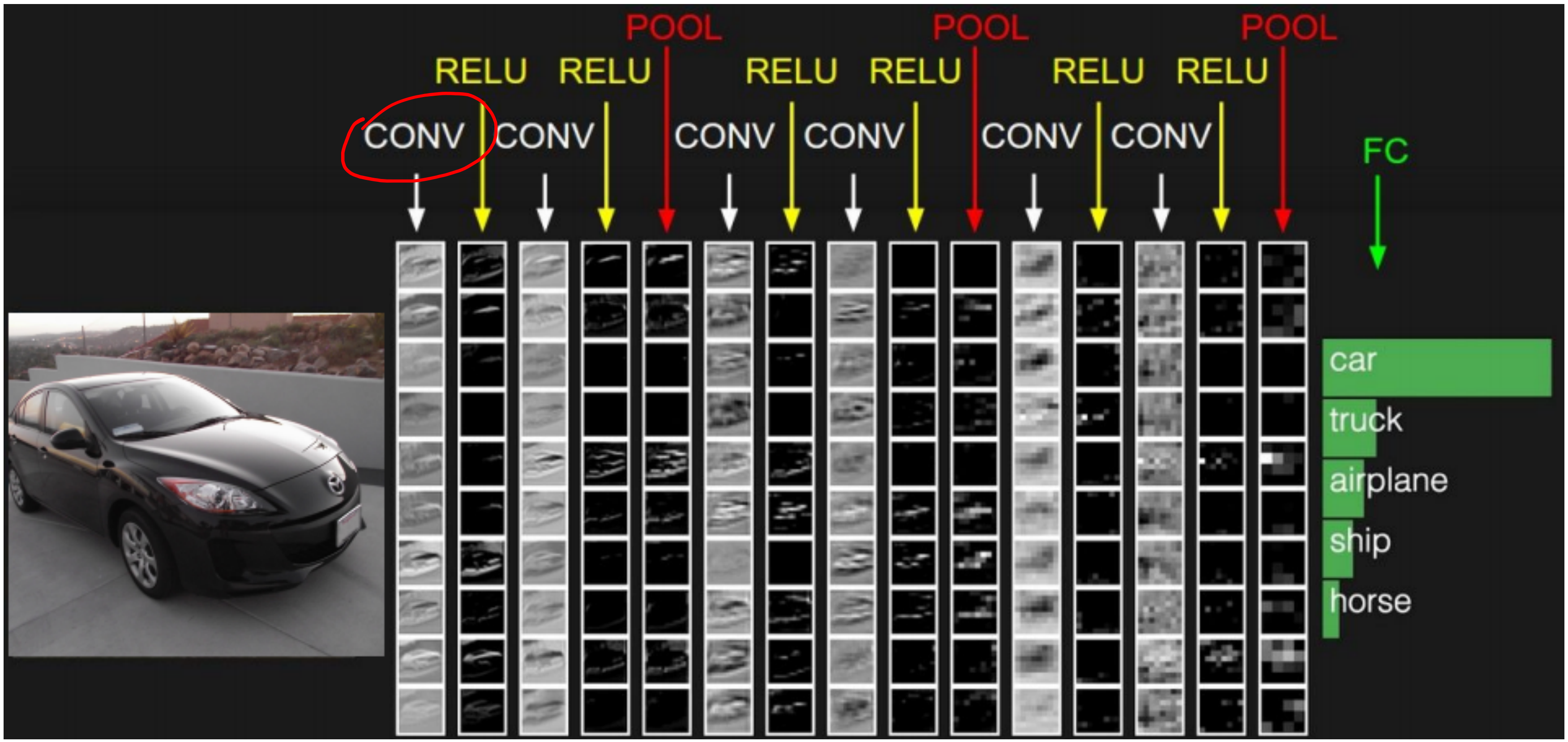
NUMBER OF FILTERS

Convolution layer: Takeaways

- Convolution is a linear operation
  - Reduces parameter space of Feed-Forward Neural Network considerably
  - Capture locality of pixels in images
  - Smaller filters need less parameters
  - Multiple filters in each layer (computation can be done in parallel)
- Convolutions are followed by activation functions
  - Typically ReLU

Convolutional Nets

MAX POOLING



Acknowledgements

- Slides made using resources from:
  - Yann LeCun
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!