DS 4400

Machine Learning and Data Mining I
Spring 2022

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

April 13 2022

Announcements

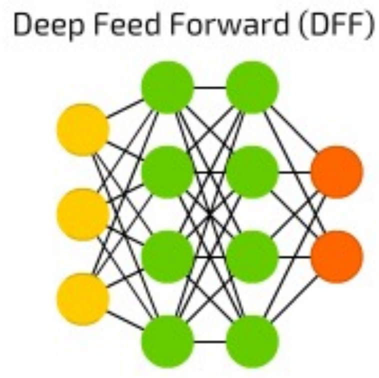
- Project Milestone
  - Due today, April 13
- Final Project
  - Project video recording (5 minute presentation) due on May 2
  - Project report due on May 2 (6-8 pages)
- Final exam: April 20

Outline

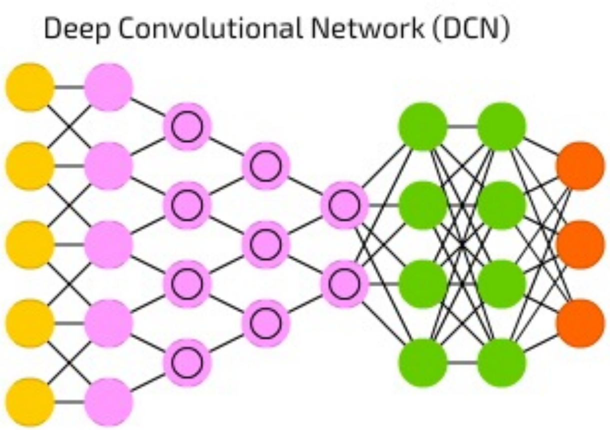
- Convolutional neural networks
  - Convolution layer
  - Max pooling
  - Well-known convolutional networks architectures
- Regularization methods for neural networks
  - Weight decay
  - Dropout
- Transfer learning
- Final exam review

Neural Network Architectures

Feed-Forward Networks



Convolutional Networks



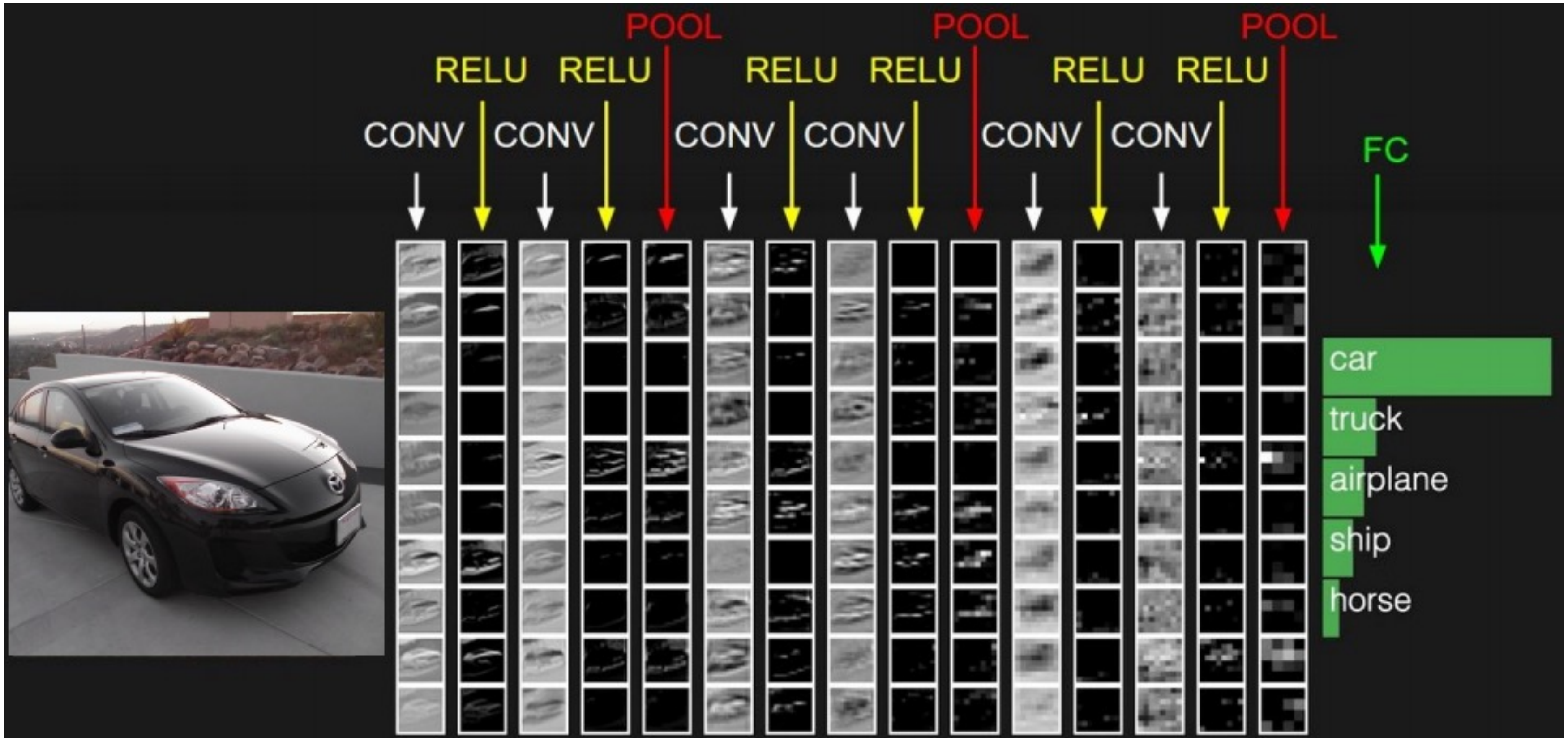
Review FFNN

- Feed-Forward Neural Networks are common neural networks architectures
  - Fully connected networks are called Multi-Layer Perceptron (MLP)
  - Usually use 1 or 2 hidden layers
- Input, output, and hidden layers
  - Linear matrix operations followed by non-linear activations at every layer
- Activations:
  - ReLU, tanh for hidden layers
  - Sigmoid (binary classification) and softmax (for multi-class classification) at last layer
- Forward propagation: process of evaluating input through the network

Convolutional Nets

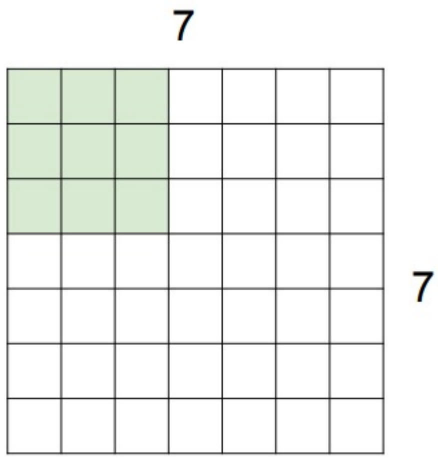
- Neurons are connected from layer to the next
  - Invented by [LeCun 89]
- Applicable to data with natural grid topology
  - Time series
  - Images
- Use convolutions on at least one layer
  - Convolution is a linear operation that uses local information
  - Also use pooling operation
  - Used for dimensionality reduction and learning hierarchical feature representations

Convolutional Nets

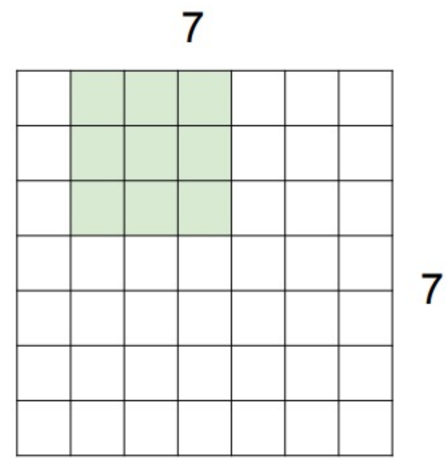


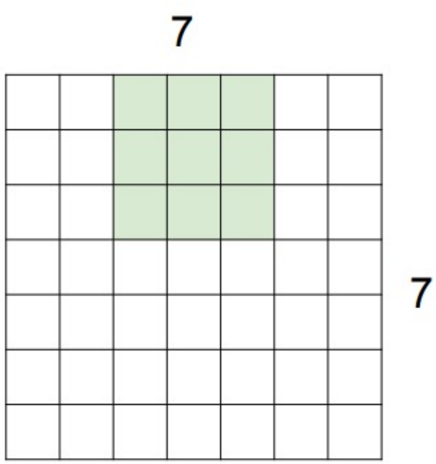
Convolutions

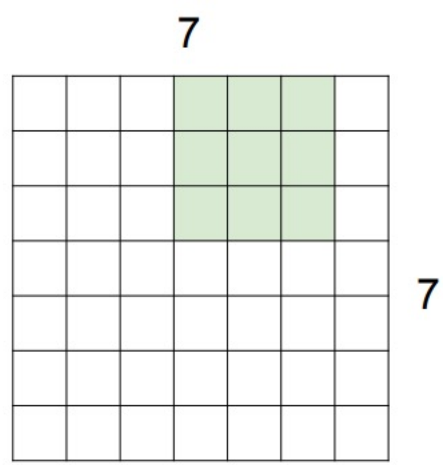
A closer look at spatial dimensions:

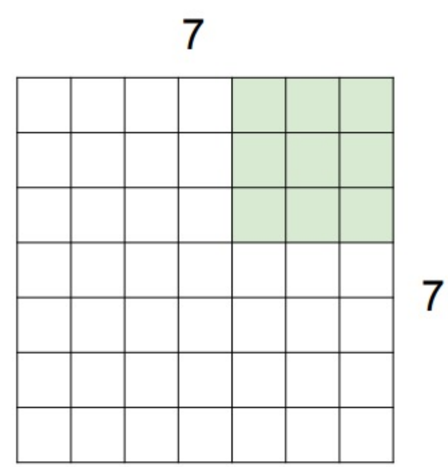


7x7 input (spatially)
assume 3x3 filter

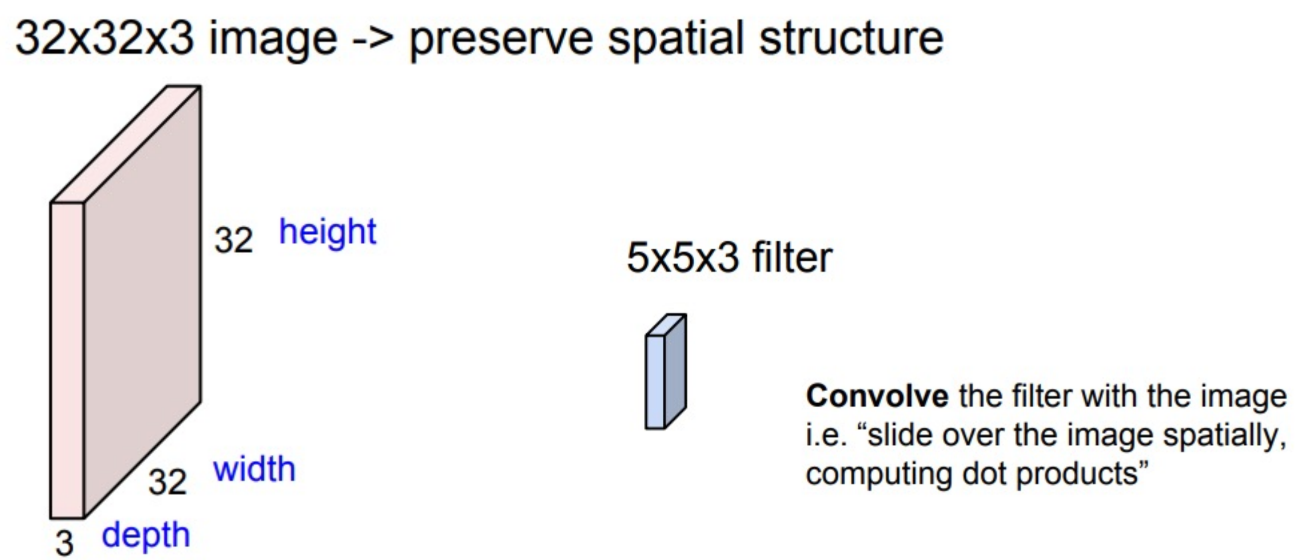








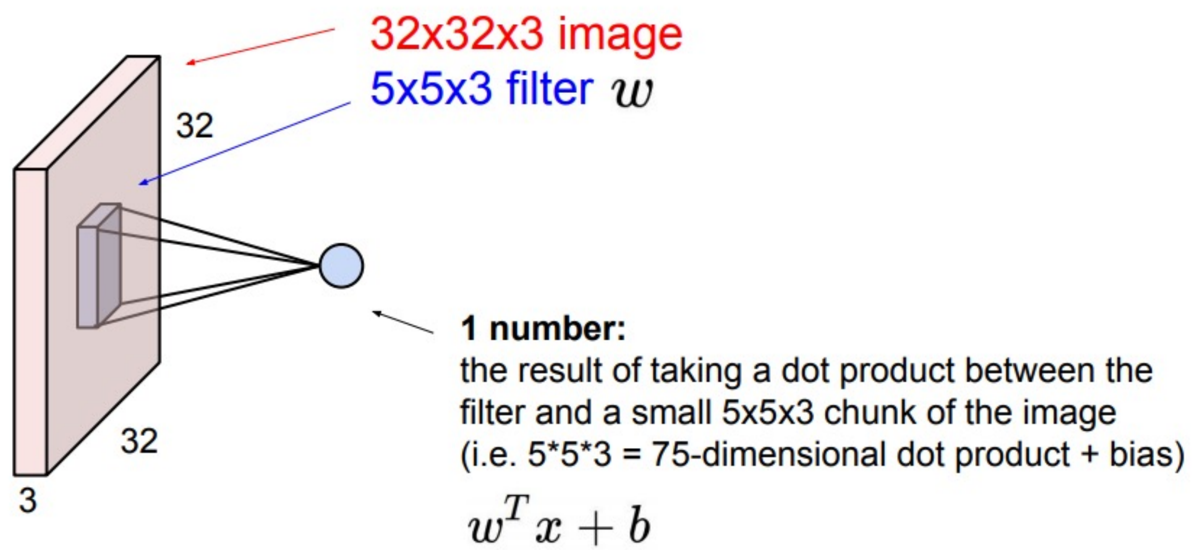
Convolution Layer

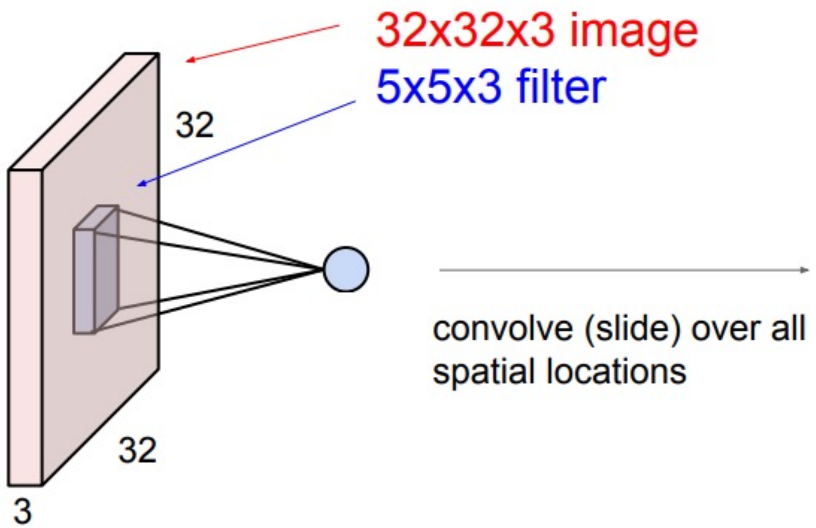


- Depth of filter always depth of input
- Computation is based only on local information

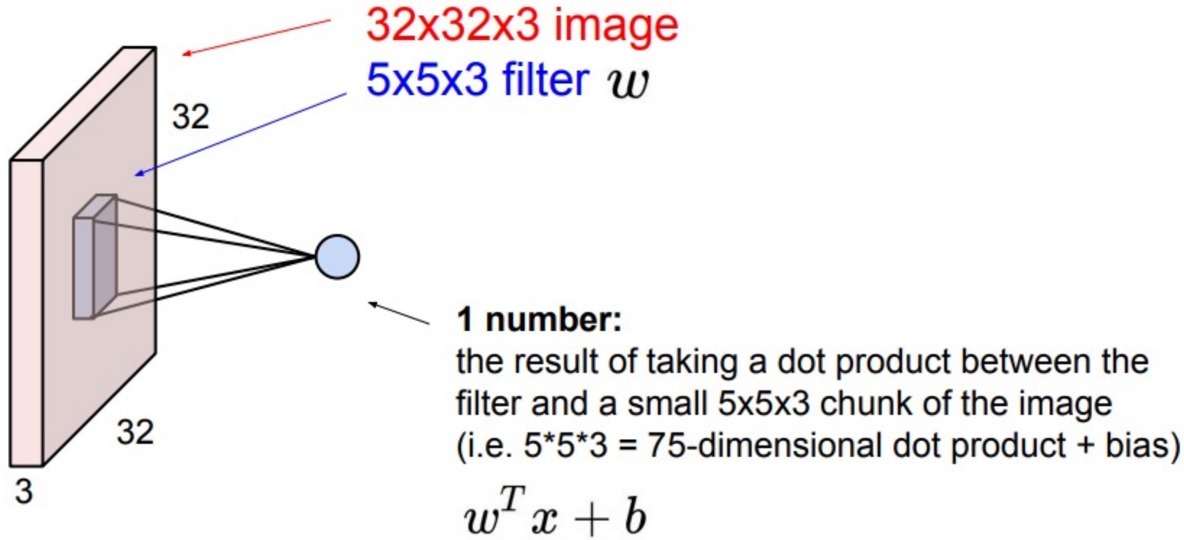
Convolution Operation

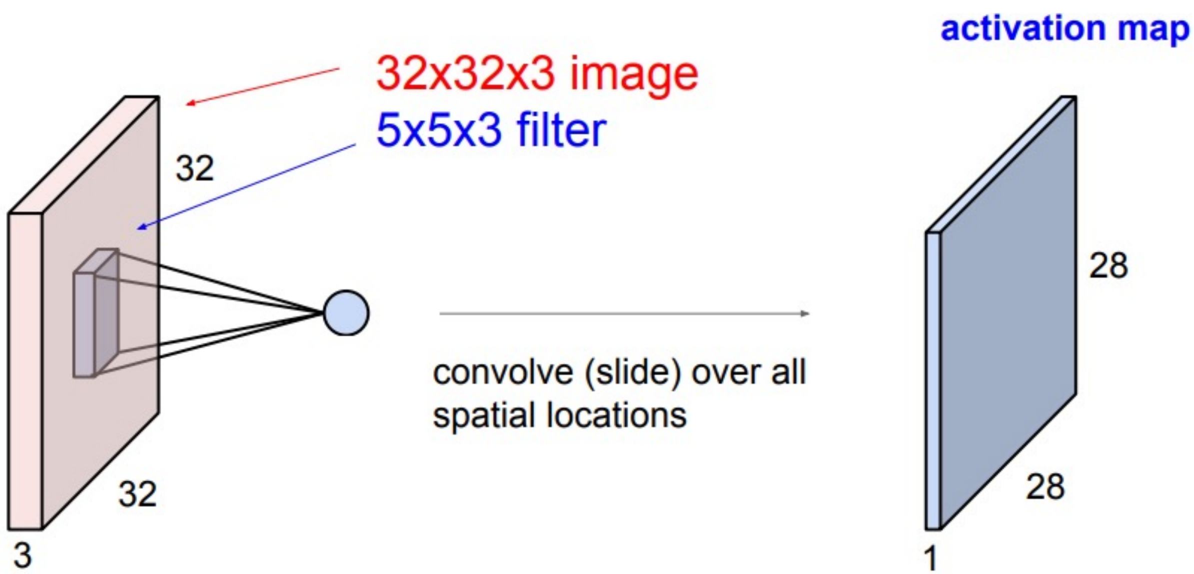
Convolution Layer



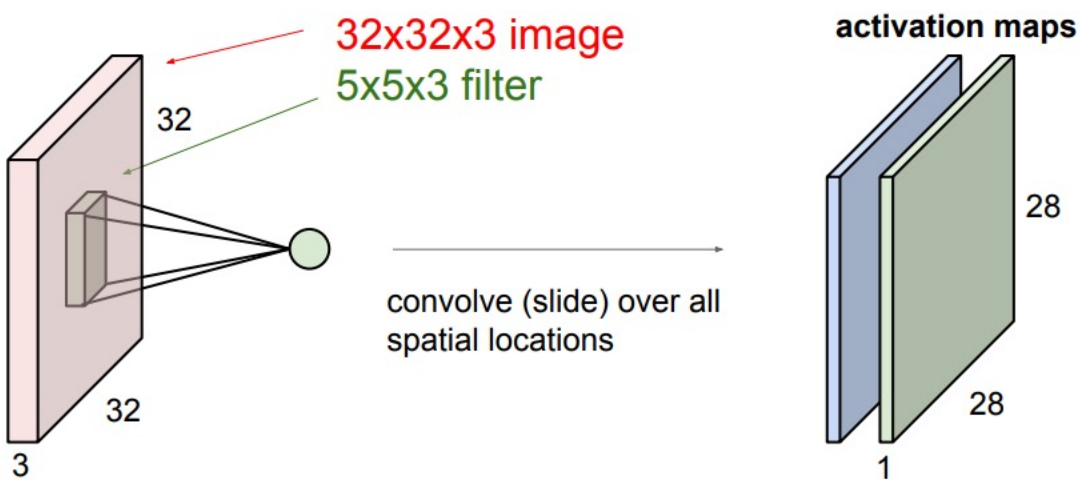


Convolution Layer

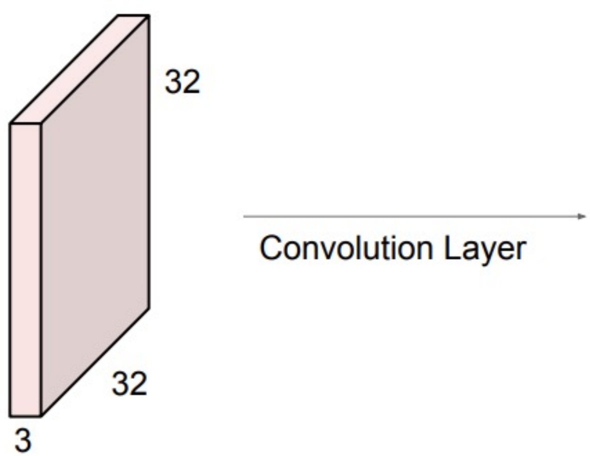




Convolution Layer

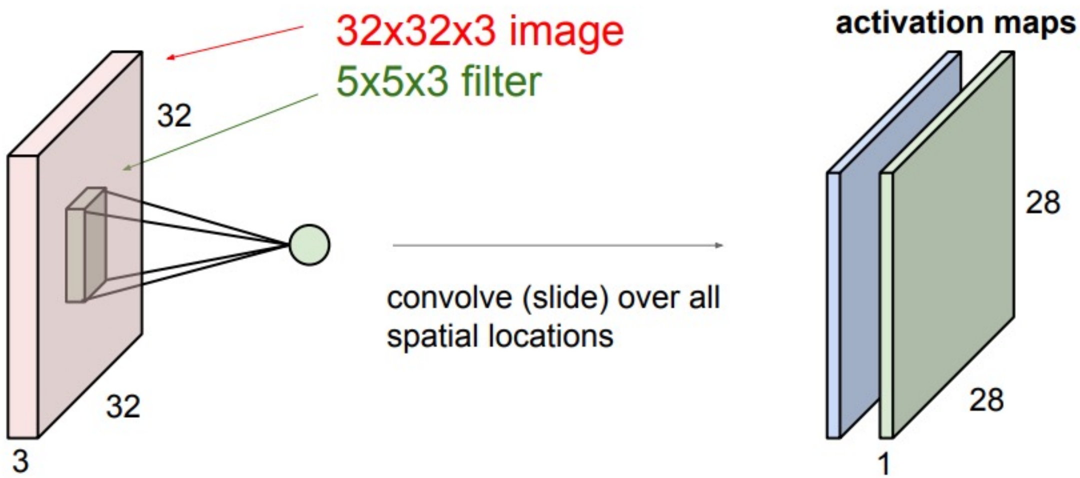


Second, green filter

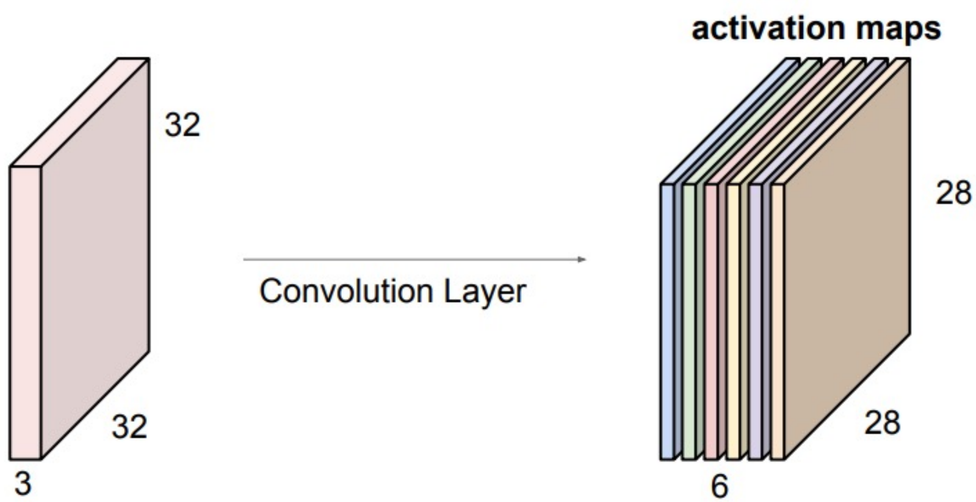


6 filters

Convolution Layer



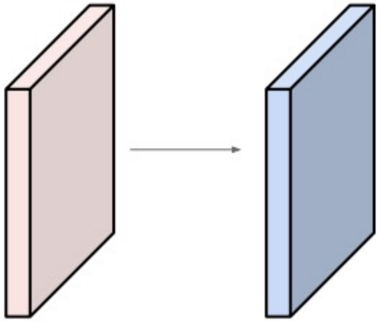
Second, green filter



6 filters

Examples

Examples time:



Input volume: **32x32x3**

10 5x5x3 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)

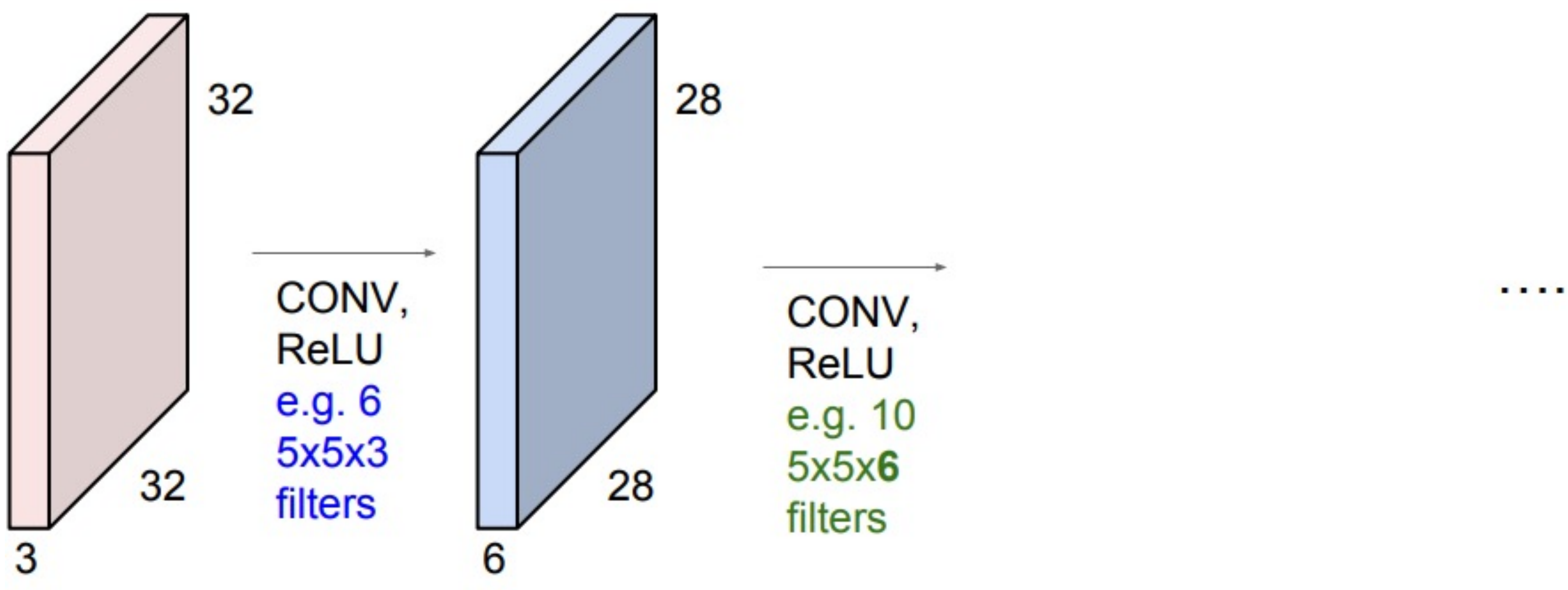
=> 76*10 = **760**

Convolutional Nets

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

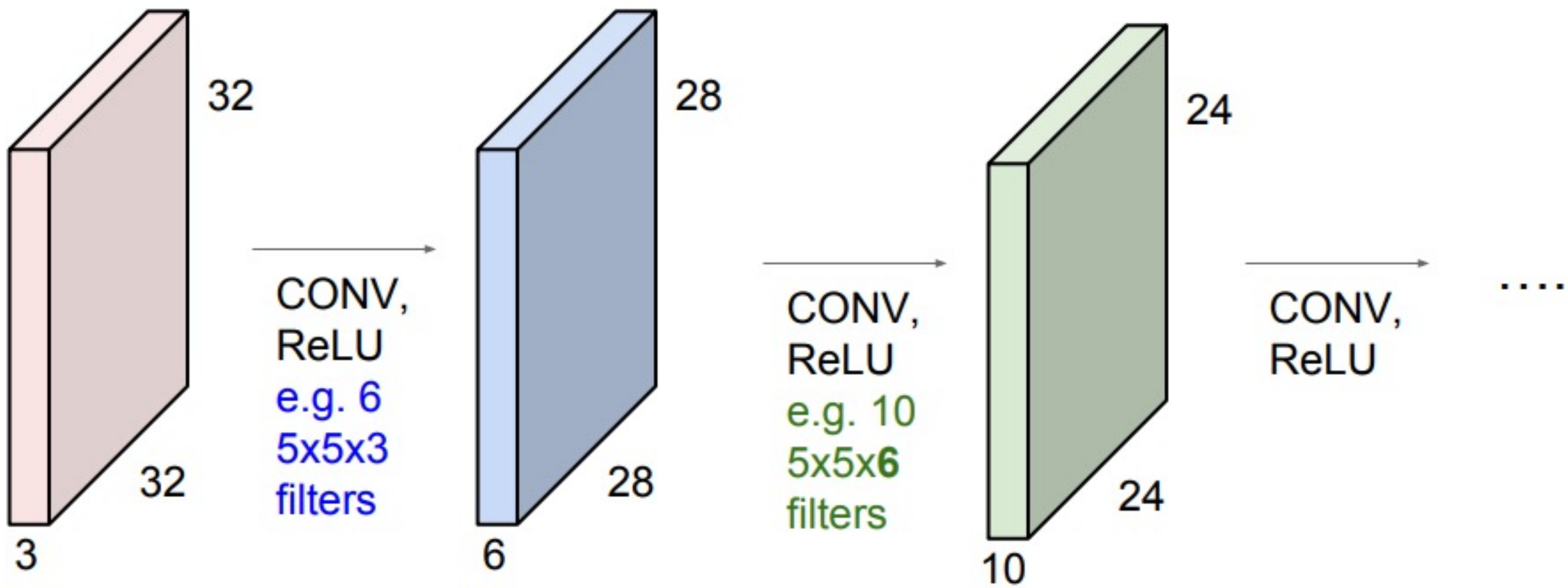
Convolutional Nets

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



Convolutional Nets

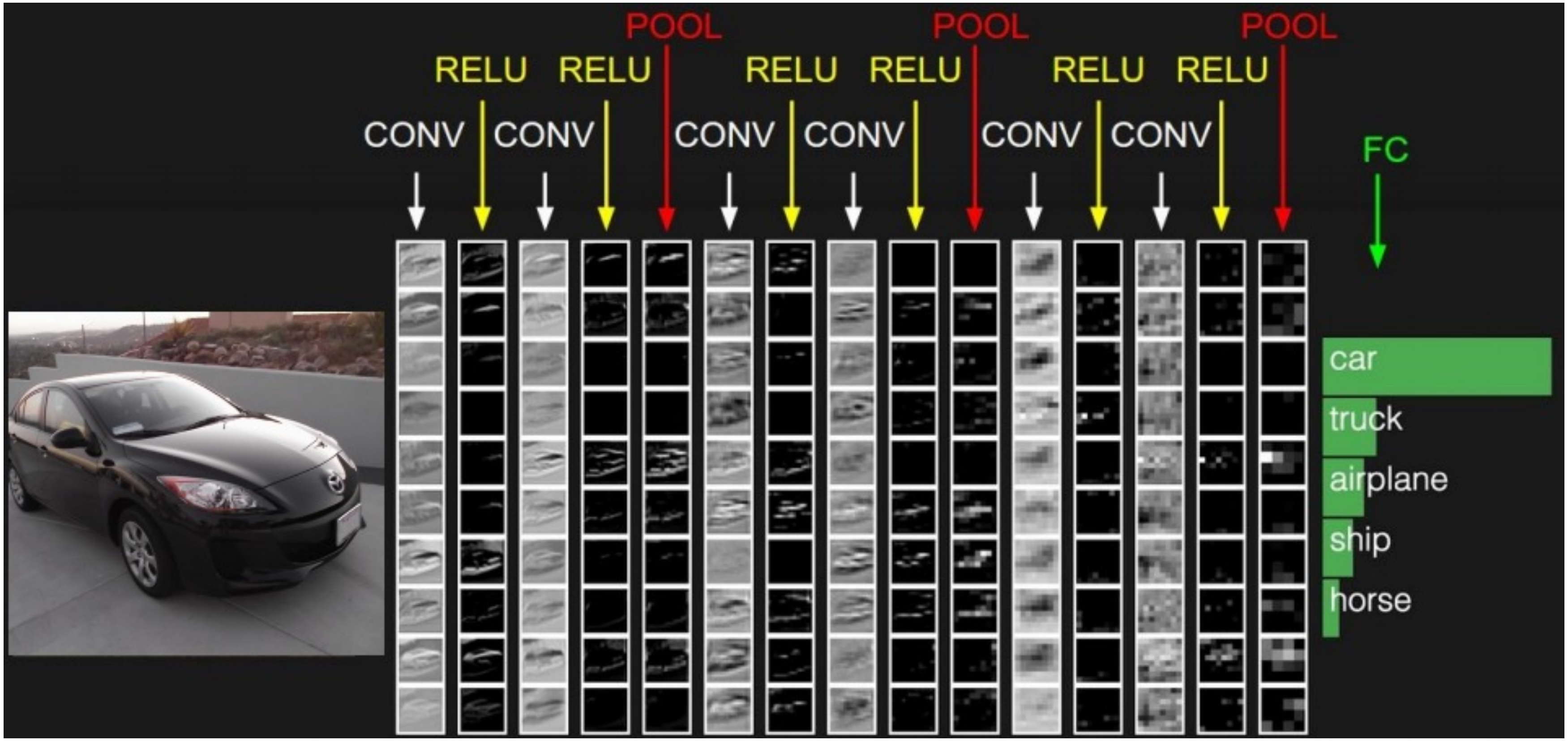
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



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

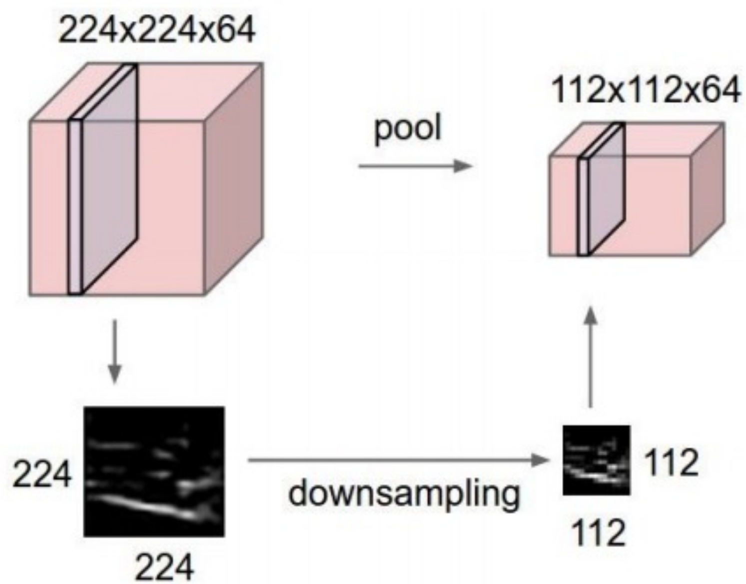
Convolutional Nets



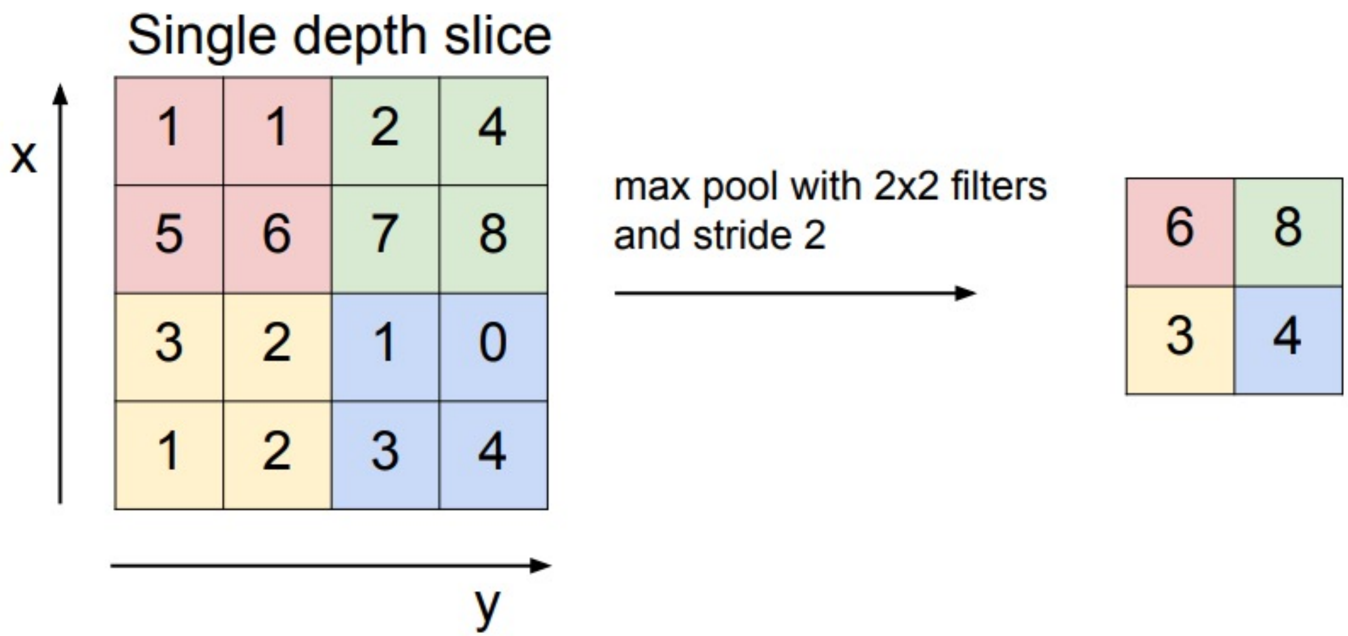
Pooling layer

Pooling layer

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



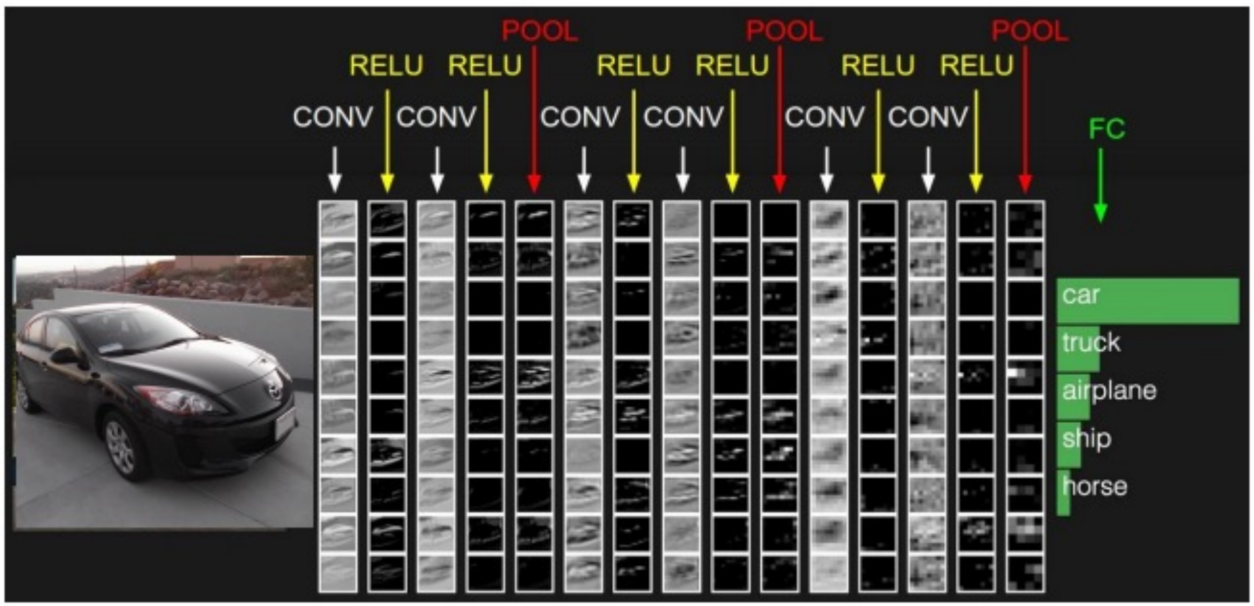
Max Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Convolutional Nets

Fully Connected Layer (FC layer)

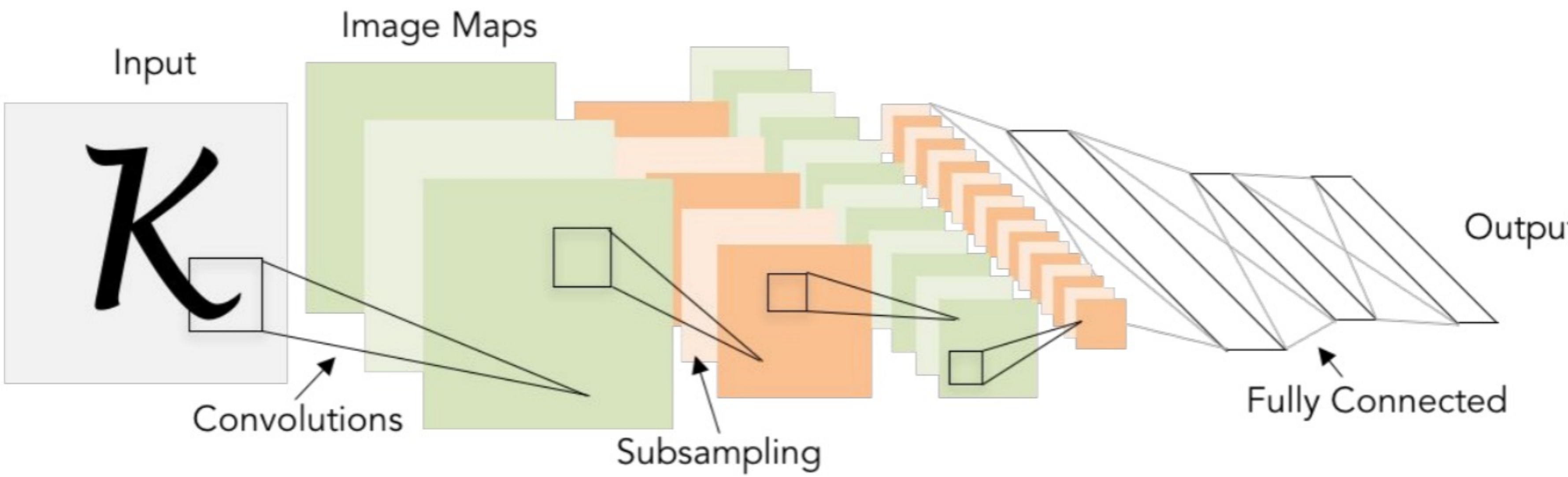
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



- FC layers are usually at the end, after several Convolutions and Pooling layers

LeNet 5

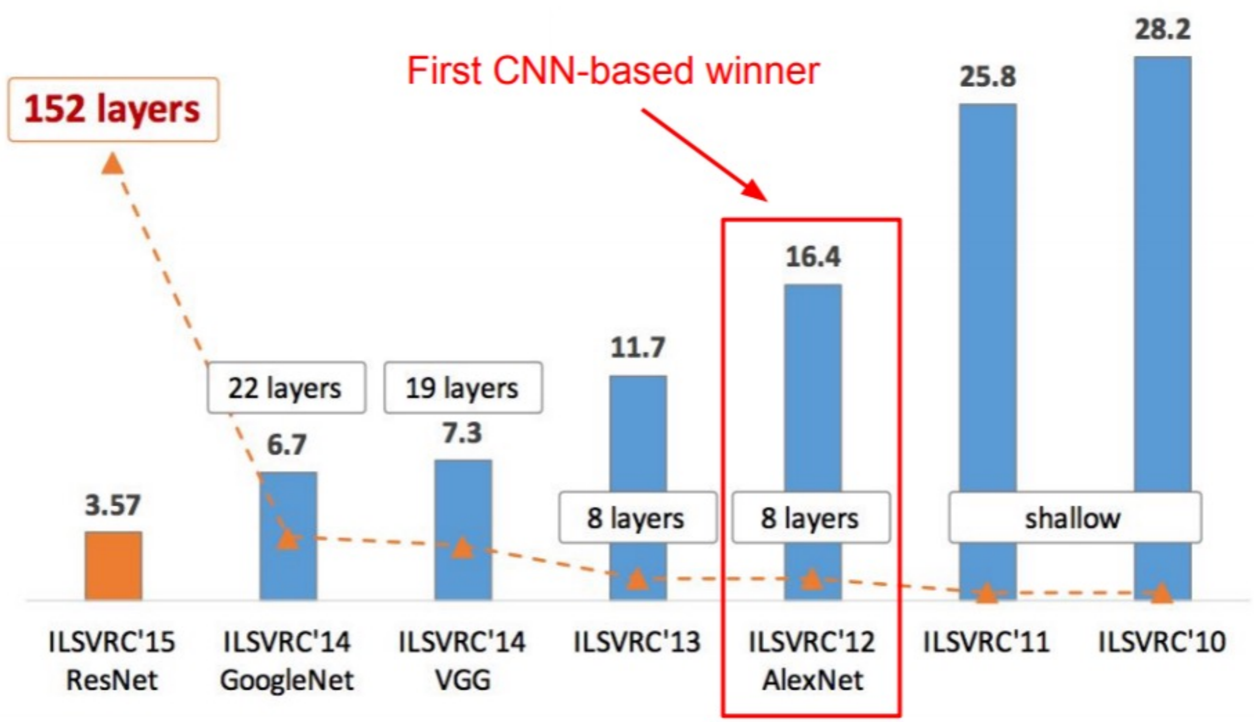
[LeCun et al., 1998]



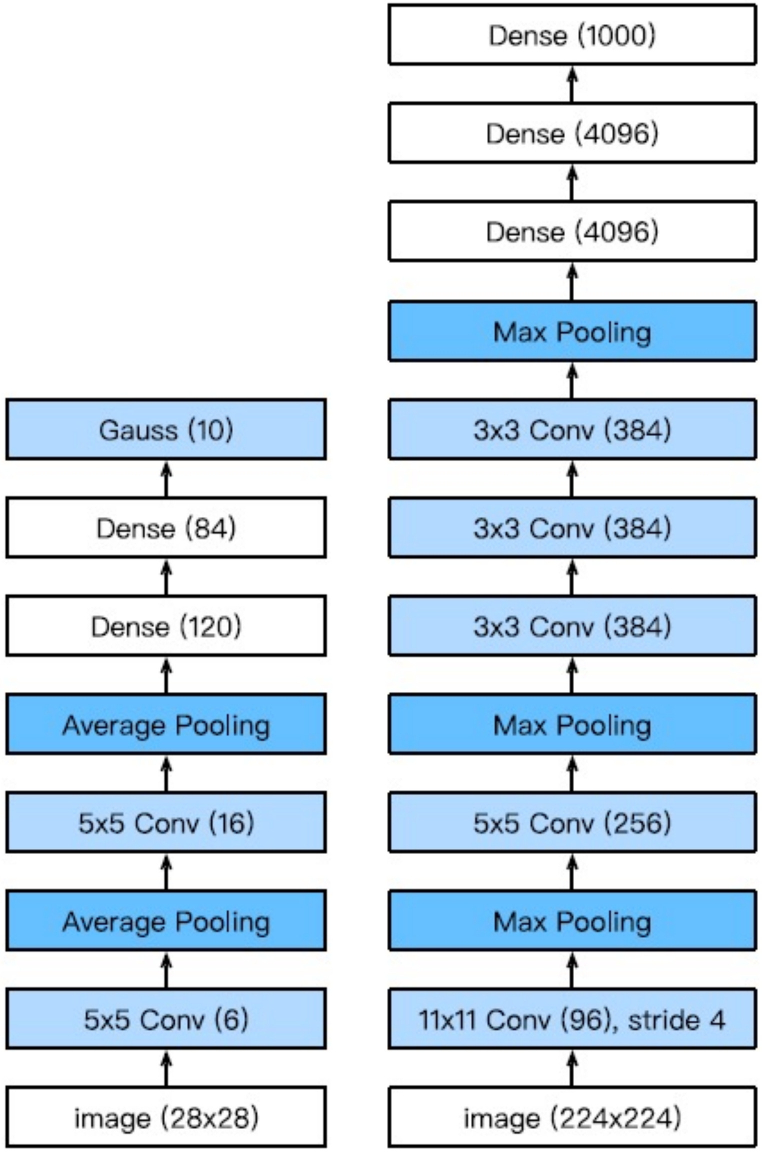
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

History

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



LeNet (left) and AlexNet (right)



Main differences

- Deeper
- Wider layers
- ReLU activation
- More classes in output layer
- Max Pooling instead of Avg Pooling

VGGNet

Case Study: VGGNet

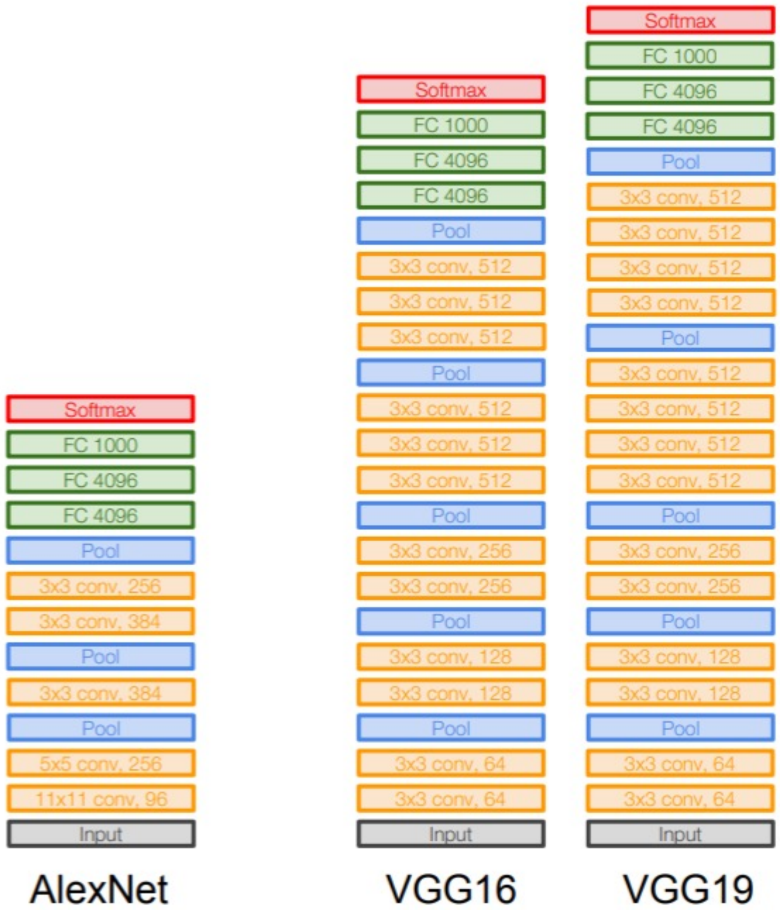
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



138 million parameters

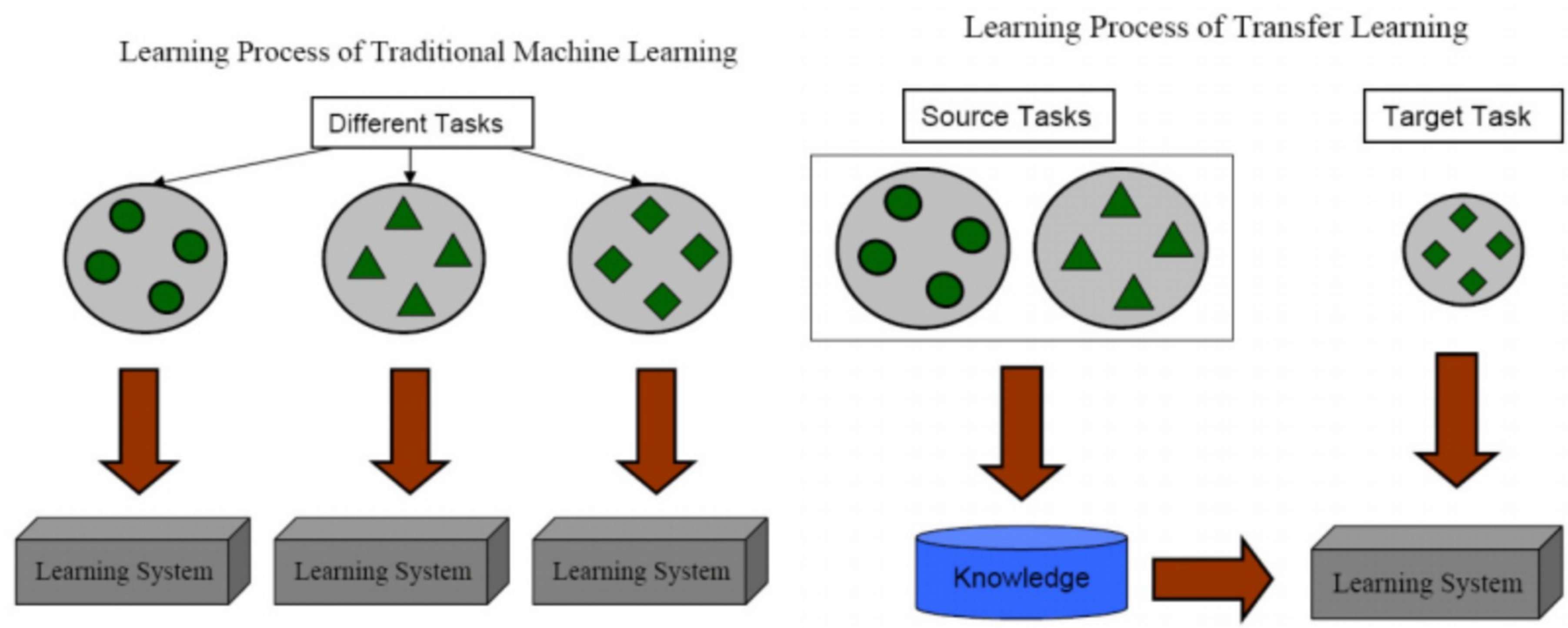
Summary CNNs

- Convolutional Nets have at least one convolution layer and optionally max pooling layers
- Convolutions enable dimensionality reduction, are translation invariant and exploit locality
- Much fewer parameters relative to Feed-Forward Neural Networks
  - Deeper networks with multiple small filters at each layer is a trend
- Fully connected layer at the end (fewer parameters)
- Learn hierarchical feature representations
  - Data with natural grid topology (images, maps)
- Reached human-level performance in ImageNet in 2014

Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Motivation: Reuse representations learned by expensive training procedures that cannot be easily replicated
  - Image classification on ImageNet is very expensive (VGG-16: 138 million, ResNet 50: 23 million parameters)
  - Generative language models very large (BERT: 110 million, GPT-2: 1.5 billion, GPT-3: 175 billion parameters)
- Two major strategies
  - Pretrained Neural Network as fixed feature extractor
  - Fine-tuning the Neural Network

Transfer Learning



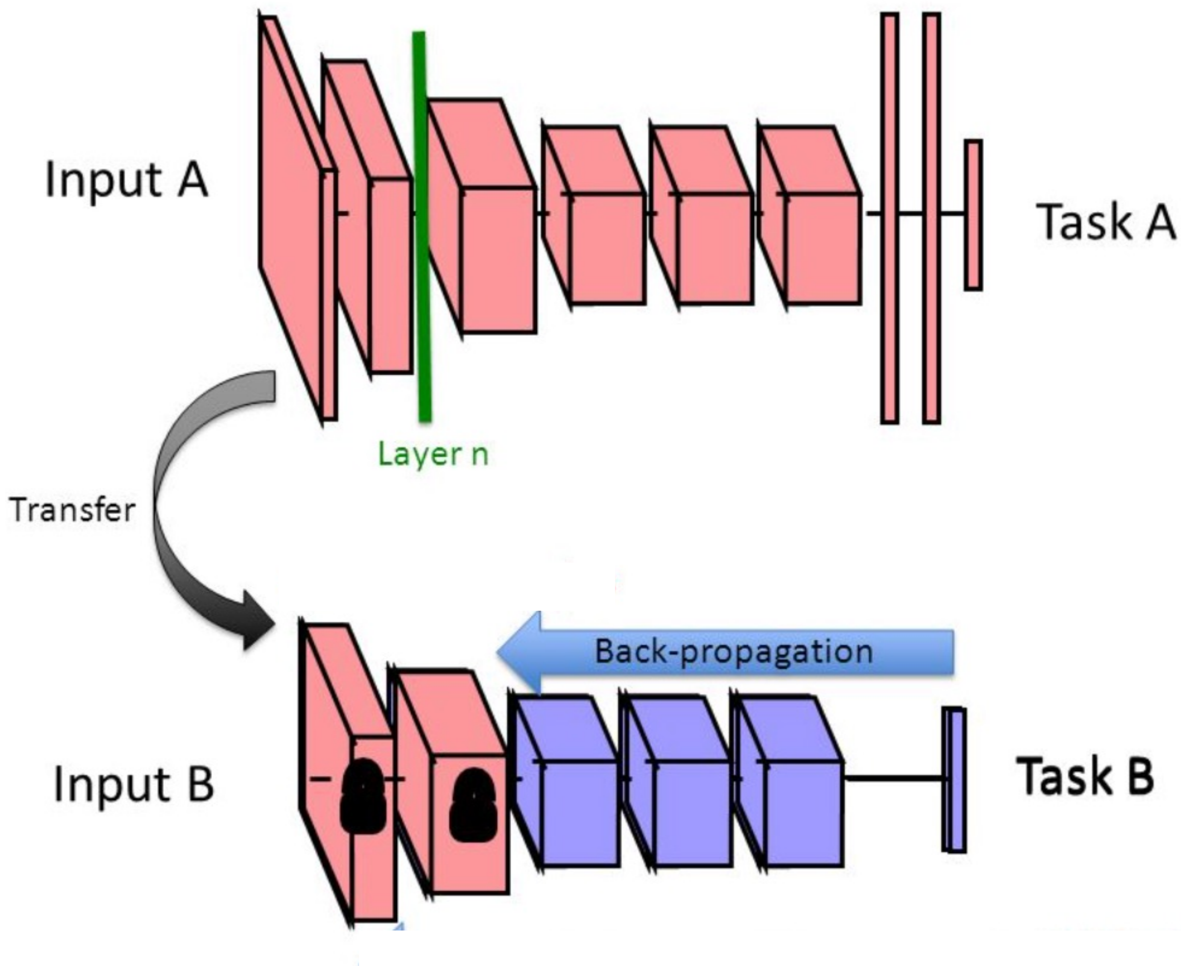
(a) Traditional Machine Learning (b) Transfer Learning

Methods for Transfer Learning

- Use a pre-trained model
  - https://modelzoo.co/

1. Use Convolutional Nets as Feature Extractor
   - Take a ConvNet pretrained on ImageNet
   - Remove the last fully-connected layer
   - Train the last layer on new dataset (usually a linear classifier such as logistic regression or softmax)
2. Fine-tuning
   - Decide to freeze first n layers
   - Train the remaining layers and stop backpropagation at layer n
   - Use a smaller learning rate
   - In the limit fine-tuning can be applied to all layers

Transfer Learning in NN: Freeze Layers



How to do Transfer Learning

Dataset size	Dataset similarity	Recommendation
Large	Very different	Train model B from scratch, initialize weights from model A
Large	Similar	OK to fine-tune (less likely to overfit)
Small	Very different	Train classifier using the earlier layers (later layers won't help much)
Small	Similar	Don't fine-tune (overfitting). Train a linear classifier

Learning Rates

- Training linear classifier: typical learning rate
- Fine-tuning: use smaller learning rate to avoid distorting the existing weights

Transfer Learning Applications

- Image classification (most common): learn new image classes
- Text sentiment classification
- Text translation to new languages
- Speaker adaptation in speech recognition
- Question answering

Final 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, ensembles, neural networks)
- Learn to apply ML algorithms to real datasets
  - Using existing packages in 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

- Bias-variance tradeoff
- Metrics
- Evaluation
- Cross-validation
- Regularization
- Gradient Descent

Linear Regression

Linear algebra

Probability and statistics

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
  - Ensembles randomize the training data in each model (bootstrap samples)
  - Neural networks use dropout or weight decay

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
  – Once you have a model know how to evaluate a point and generate a prediction
    - Example: predict output by Naïve Bayes, decision tree, or neural network

LDA Training and Testing

Given training data $(x_i, y_i), i = 1, \dots, n, y_i \in \{1, \dots, K\}$

1. Estimate sample mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k / n.$$

Given testing point x, predict k that maximizes:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Naïve Bayes Classifier

TRAIN

- For each class label k
  1. Estimate prior $\pi_k = P[Y = k]$ from the data
  2. For each value v of attribute X_j
     - Estimate $\mathrm{P}\left[X_j = v \middle| Y = k\right]$

TEST on INPUT $x = (x_1, \ldots, x_d)$

- For every k, compute the probabilities
  - $p_k = P[Y = k] \prod_{j=1}^{d} P\left[X_j = x_j \middle| Y = k\right]$
- Classify x to the class k that maximizes p_k

Learning Decision Trees

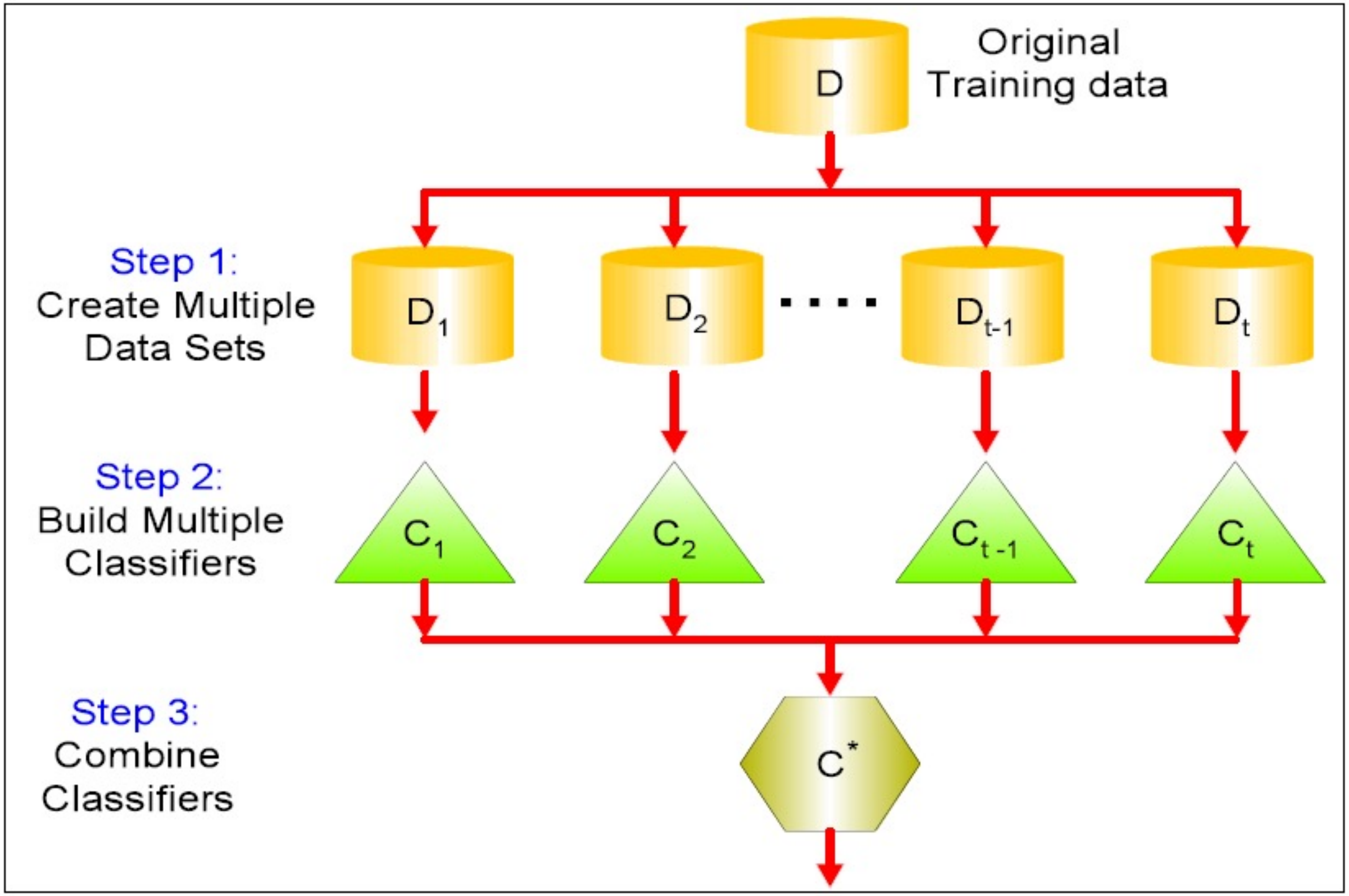
- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

- Recurse

Information Gain reduces uncertainty on Y
Can use Gini index

Bagging



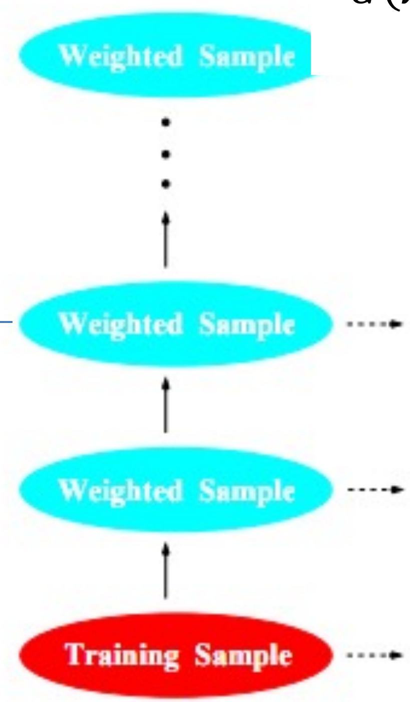
Majority Votes

Boosting

$$G(x) = sign(\sum_{i=1}^{T} \beta_t h_t(x))$$

Better classifiers will get higher weights

- Mis-classified examples get higher weights
- Correct examples get lower weights



$h_3(x)$

$h_2(x)$

$h_1(x)$

Uniform weights

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

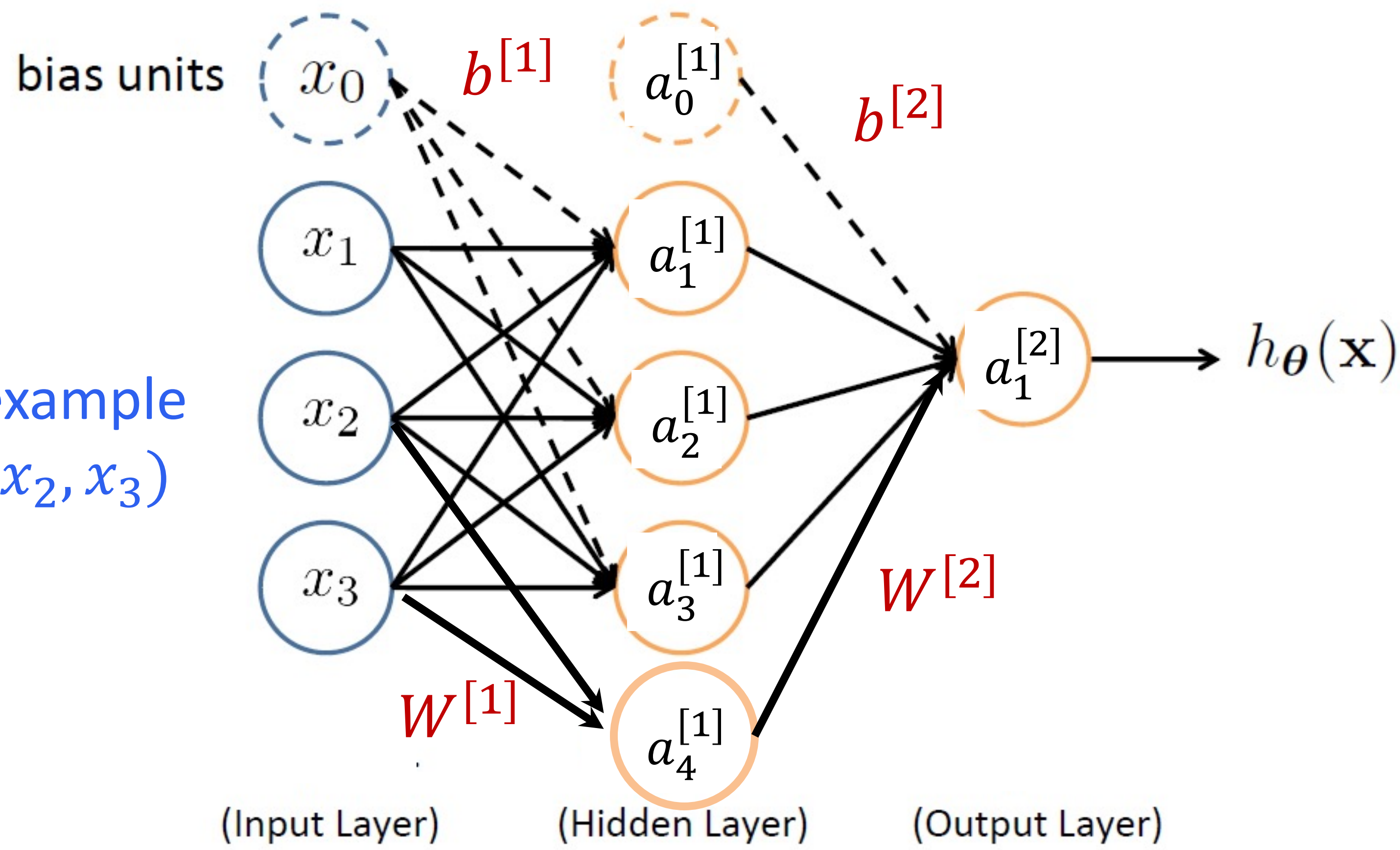
Online Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
    Receive training example (x_i, y_i)
    If $y_i\theta^T x_i \leq 0$ // prediction is incorrect
        $\theta \leftarrow \theta + y_i\, x_i$
Until stopping condition

Online learning – the learning mode where the model update is performed each time a single observation is received

Batch learning – the learning mode where the model update is performed after observing the entire training set

Feed-Forward Neural Network



Training example

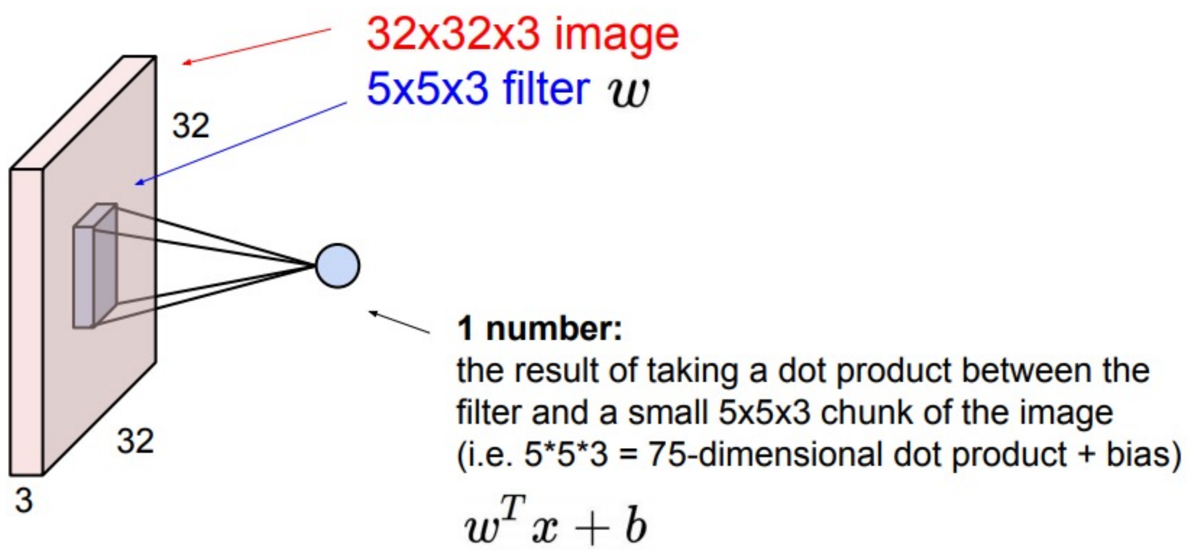
$x = (x_1, x_2, x_3)$

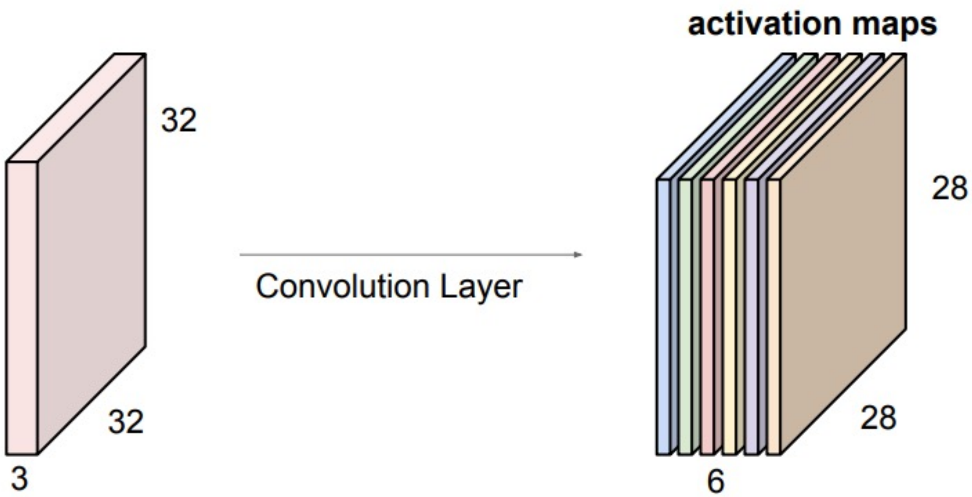
Layer 0 Layer 1 Layer 2

No cycles

$\theta = (b^{[1]}, W^{[1]}, b^{[2]}, W^{[2]})$

Convolution Layer





When to use each model

- Assumptions:
  - 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
  - Reduce variance of single models

Comparing classifiers

Algorithm	Interpretable	Model size	Predictive accuracy	Training time	Testing time
Logistic regression	High	Small	Lower	Low	Low
kNN	Medium	Large	Lower	No training	High
LDA	Medium	Small	Lower	Low	Low
Decision trees	High	Medium	Lower	Medium	Low
Ensembles	Low	Large	High	High	High
Naïve Bayes	Medium	Small	Lower	Medium	Low
Neural Networks	Low	Large	High	High	Low