Introduction to Neural Networks

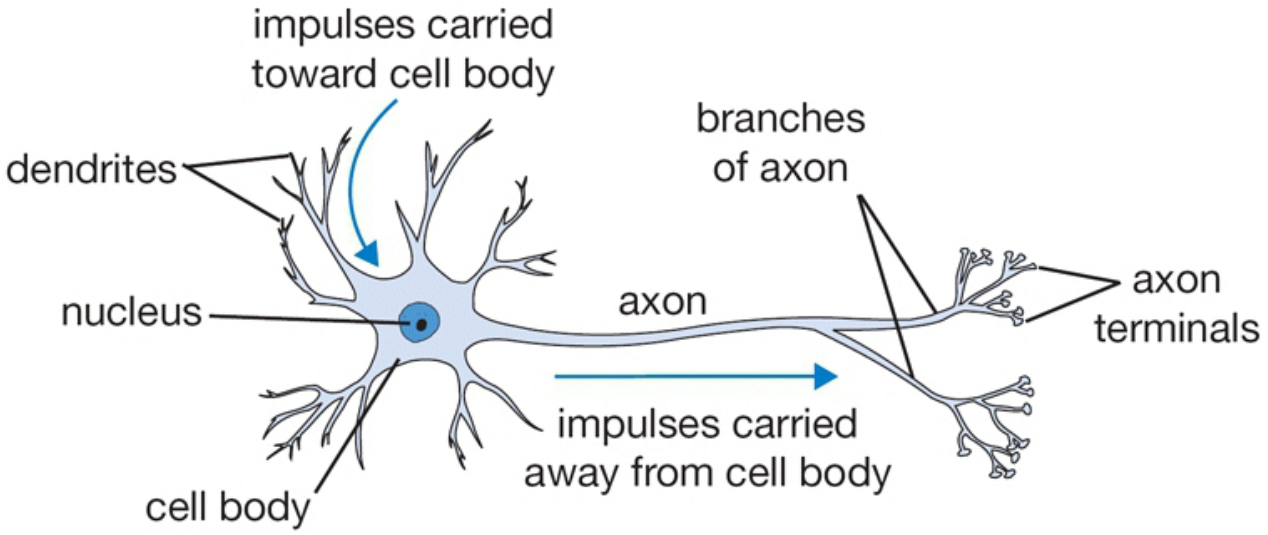
SAPIENZA
UNIVERSITÀ DI ROMA

Roberto Capobianco

Dipartimento di Ingegneria Informatica, Automatica e Gestionale

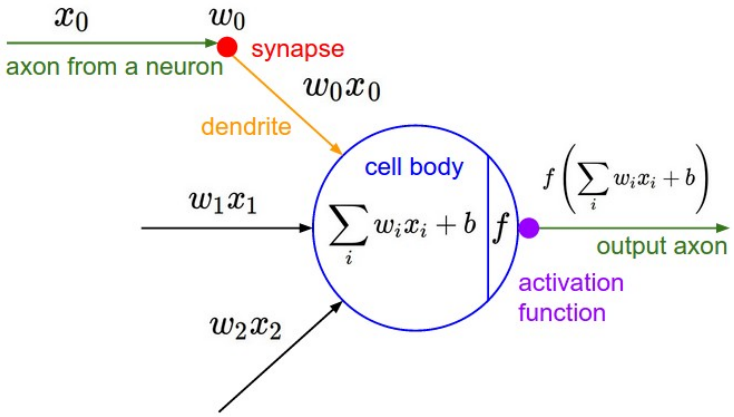
Neural Networks: Motivation

- Initial goal: model biological neural systems
  - basic computational unit: neuron
  - ~86 billion neurons in the human nervous system
  - connected with ~10^{14k}-10^{15} synapses
  - signals on axons interact multiplicatively with dendrites of other neurons based on some synaptic strength
  -
-



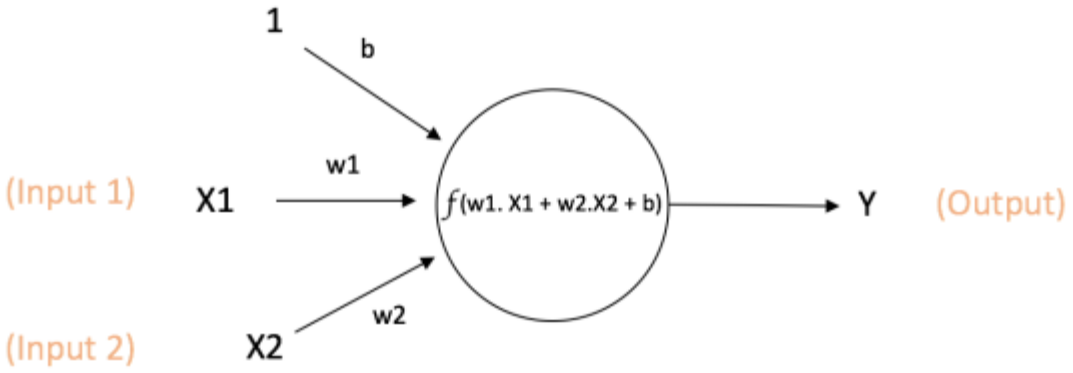
Neural Networks: Implementation

- Diverged from biological model
  - engineered to achieve good results in ML tasks (different from real neurons!)
  - idea: synaptic strengths can be learned
  - model: dendrites carry signals that get summed in the cell body; if sum is above some threshold neuron fires
  - neurons fire with a frequency that depends on the activation function
  -
-



Artificial Neuron

- takes numerical **inputs** (x)
- has a **weight** associated to each input (w)
- has a **bias** in the form of an additional input 1 with weight b
- applies an activation function (f) to the weighted sum of inputs



Output of neuron = Y= $f(w1.\,X1 + w2.X2 + b)$

Activation Function

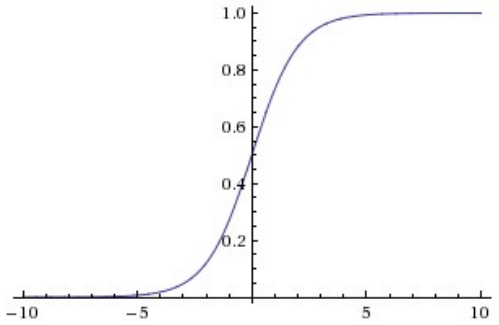
- typically non-linear, aims at introducing non-linearity in the output of a neuron
- takes numbers as input
- performs a fixed mathematical operation on it

e.g., **softmax** function: takes a vector of arbitrary real-valued scores (in z) and squashes it to a vector of values between zero and one that sum to one.

Types of Activation Function

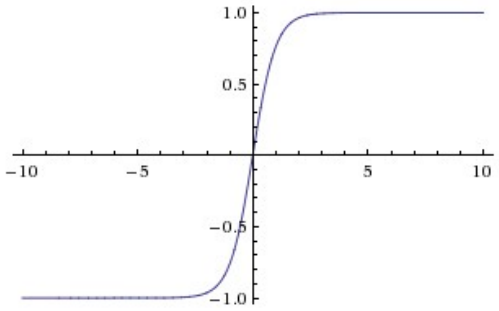
- sigmoid (bad!): takes a real-valued input and squashes it to range between 0 and 1

  $\sigma(x) = 1 / (1 + \exp(-x))$



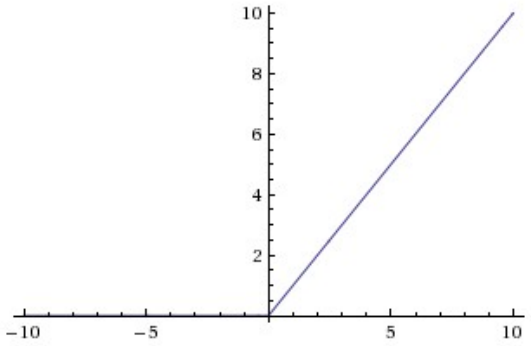
-
- tanh: takes a real-valued input and squashes it to the range [-1, 1]

  $\tanh(x) = 2\sigma(2x) - 1$

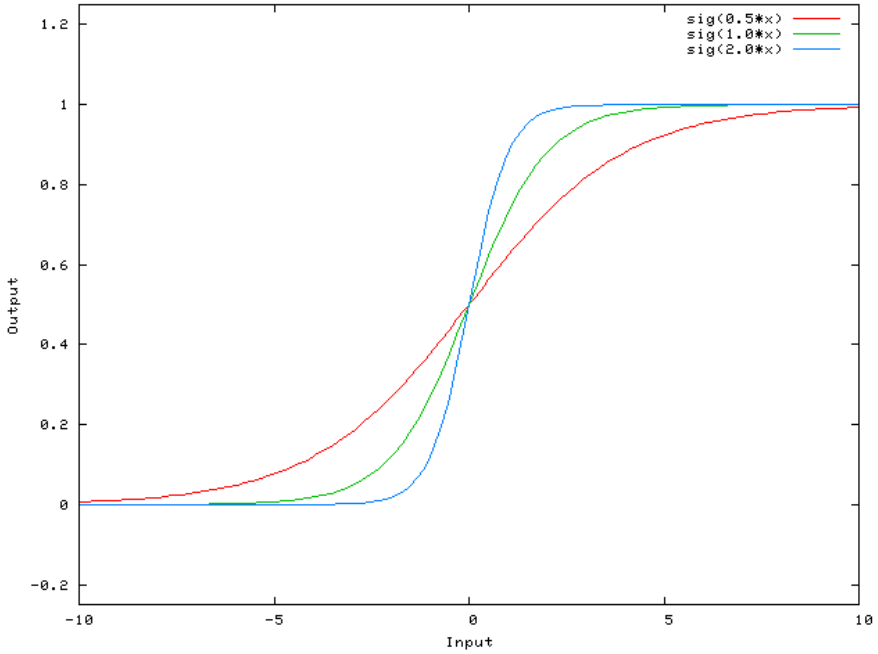


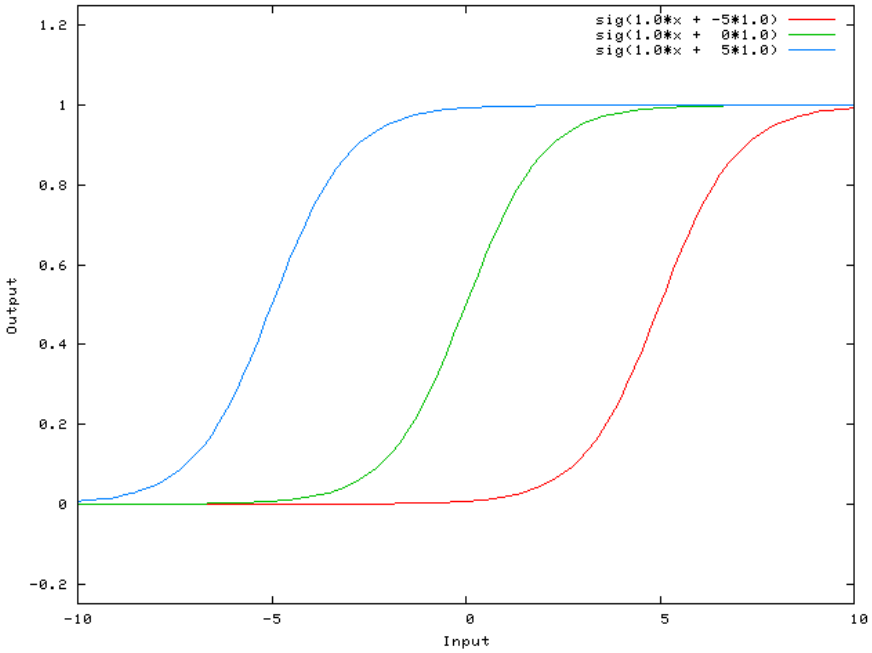
- ReLU: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

  $f(x) = \max(0, x)$



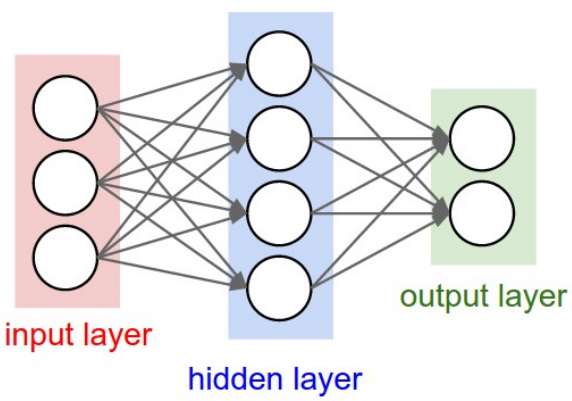
Role of Bias

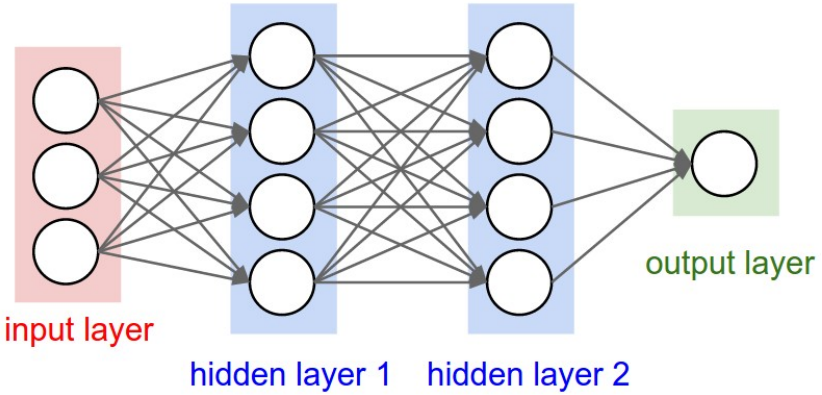

sig(0.5*x)
sig(1.0*x)
sig(2.0*x)
Output
Input
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-5
0
5
10


sig(1.0*x + -5*1.0)
sig(1.0*x + 0*1.0)
sig(1.0*x + 5*1.0)
Output
Input
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-10
-5
0
5
10

Neural Network Architecture

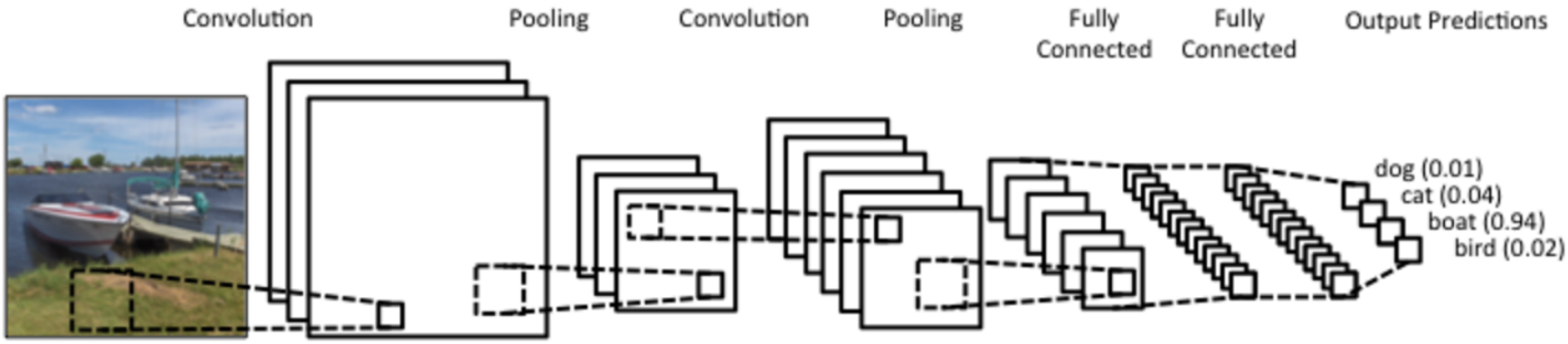
- regular neural networks are neurons connected in an acyclic graph
- 1 or more layers
- typically fully connected layers (no connection inside the same layer)
- output layer typically without activation function
- naming convention: input layer is not counted
  - single-layer networks directly map input to output





Alternative Types of Neural Networks

- **Convolutional Neural Networks**
- unit connectivity pattern inspired by the organization of the visual cortex
- units respond to stimuli in a restricted region of space known as the receptive field
- receptive fields partially overlap, over-covering the entire visual field



Alternative Types of Neural Networks

- **Recurrent Neural Networks**
- connections between unit form a directed cycle
- propagate data forward AND backward
- use memory to process sequence of inputs
- useful for tasks like speech recognition and sequence processing

Training Neural Networks

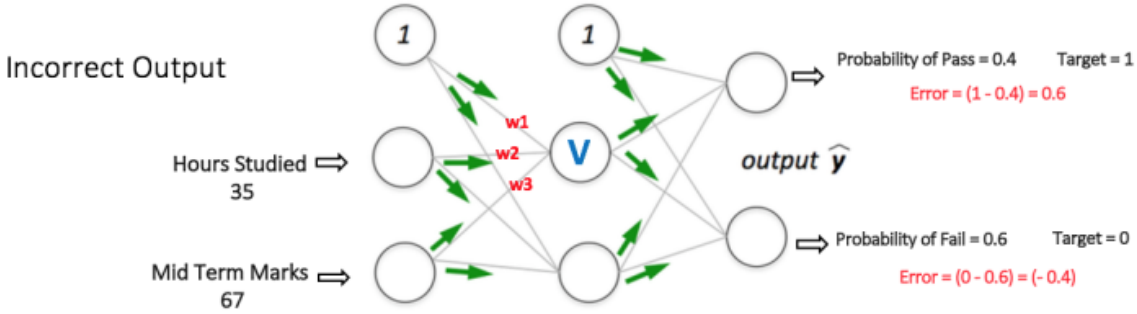
- **Backpropagation**

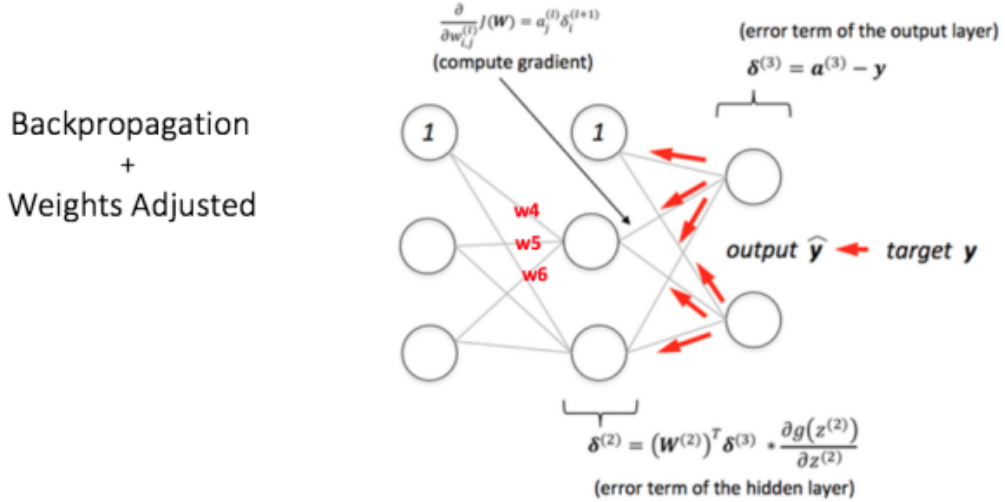
1) Initialize all edge weights randomly
2) For every input, until error < threshold:
   a) provide it to the neural network and observe output
   b) compare output with desired output / label
   c) propagate the error back to the previous layer

Additional resources:

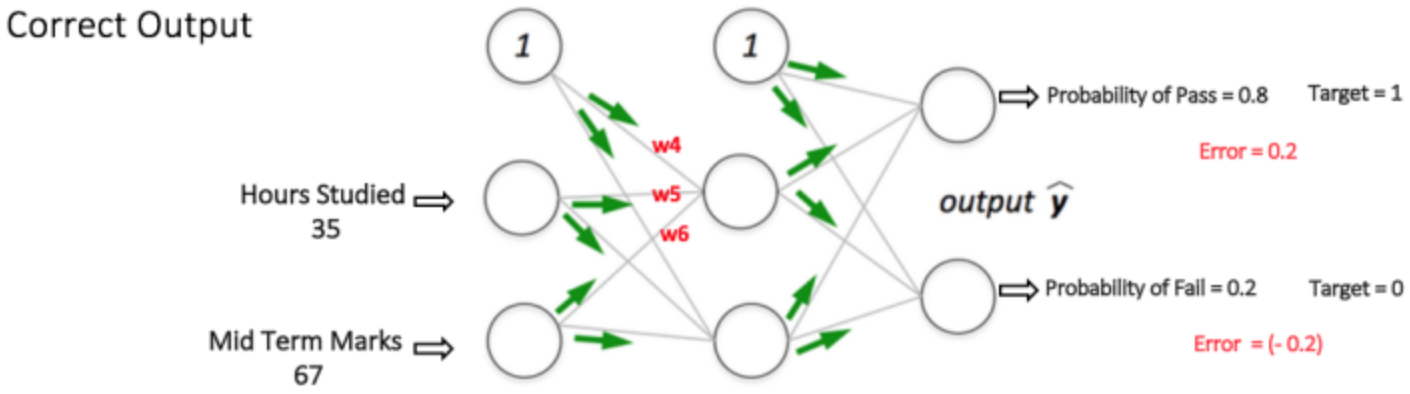
https://github.com/rasbt/python-machine-learning-book/blob/master/faq/visual-backpropagation.md
https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/
3D visualization of neural networks: http://scs.ryerson.ca/~aharley/vis/fc/

Backpropagation

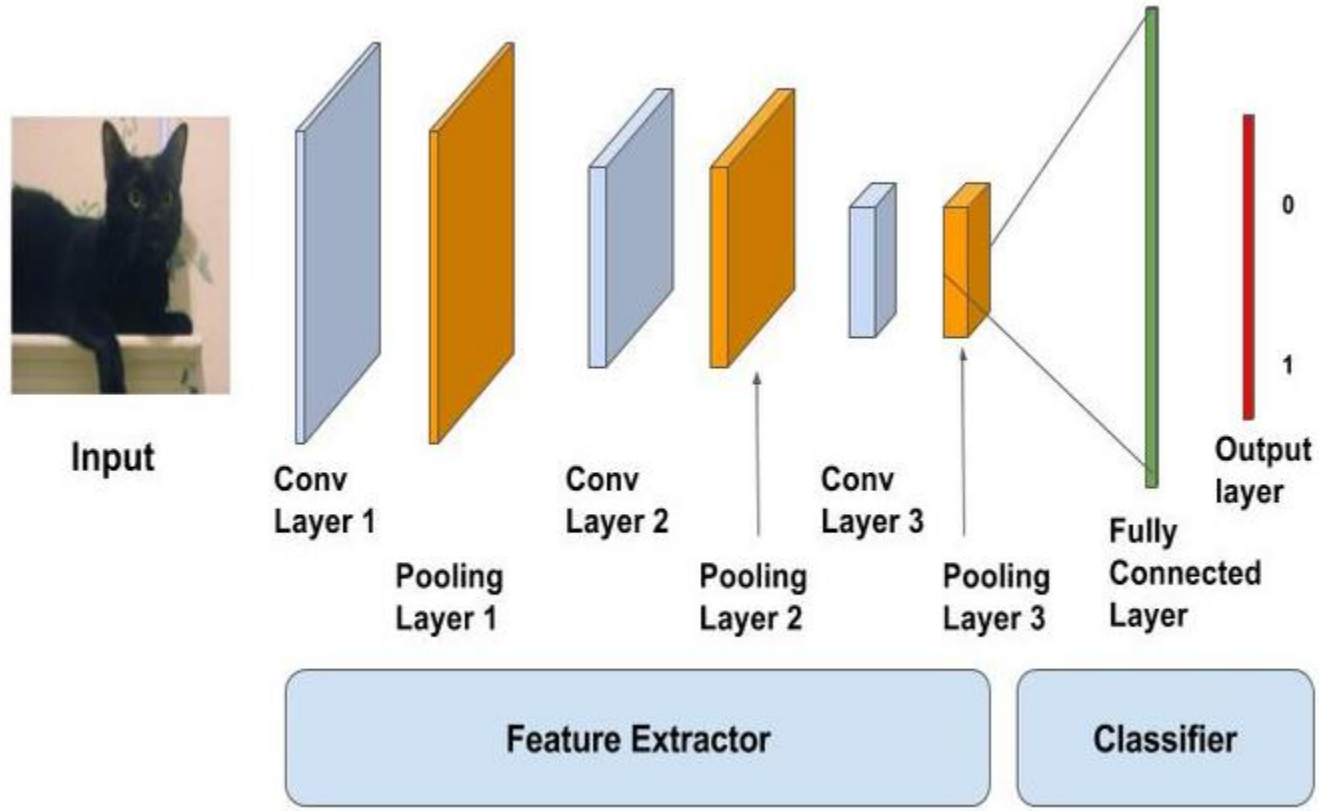




Backpropagation

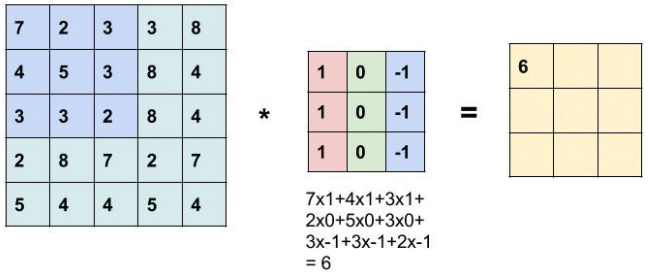


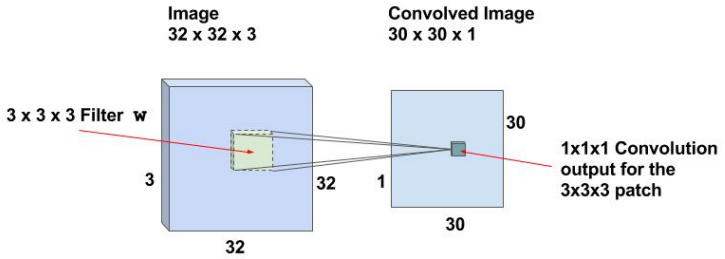
Convolutional Neural Network



Convolutional Layer

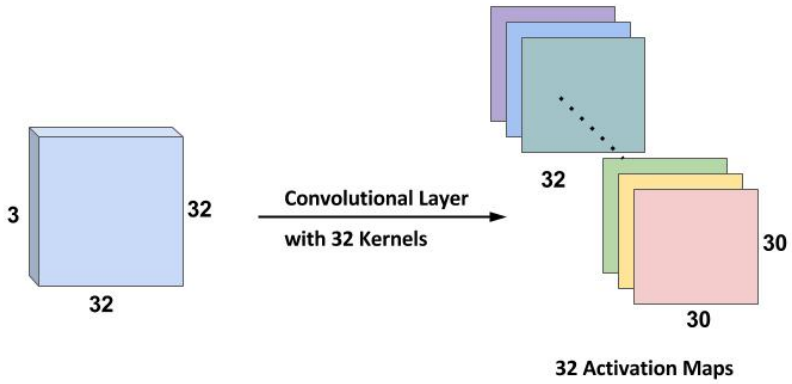
- ‘eyes’ of the neural network
- neurons look for specific features
- to calculate convolution at a particular location (x, y):
  - extract a chunk from the image centered at location (x,y)
  - multiply the values in this chunk element-by-element with the convolution filter (sized as the chunk)
  - add them all to obtain a single output
- weights and biases of neurons corresponding to one filter are shared
-





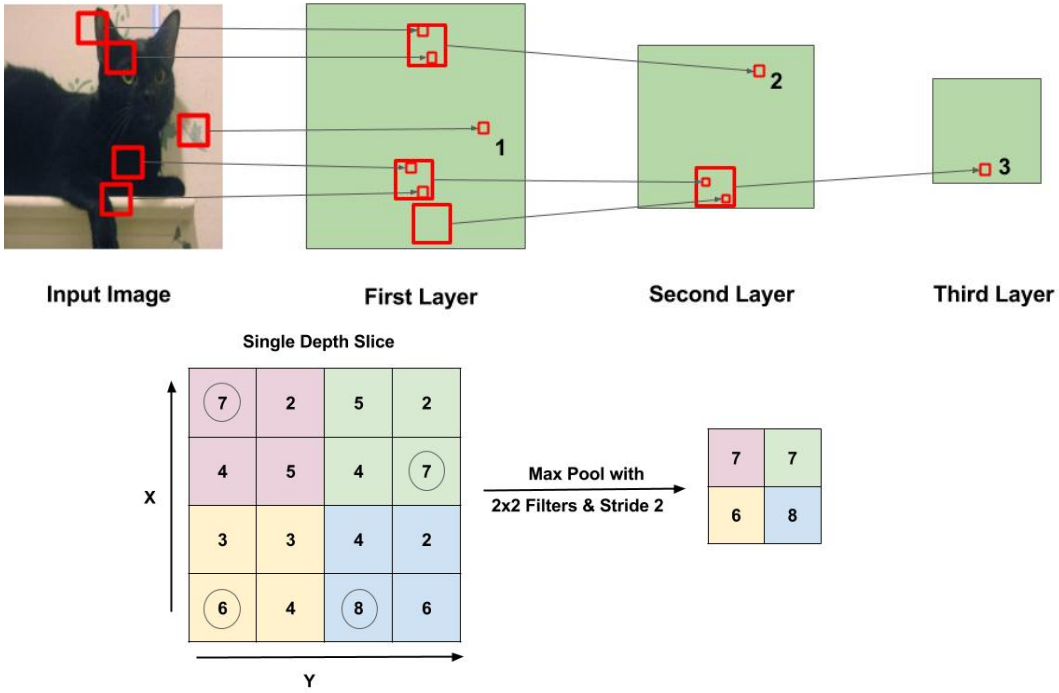
Stride, Multiple Filters and Padding

- stride: how many pixels we use to slide the filter
- after each convolution the output reduces in size (filter 3x3x3 → activation map 30x30x32)
- pad zeros to the boundary of the input layer such that the output is the same size as input layer

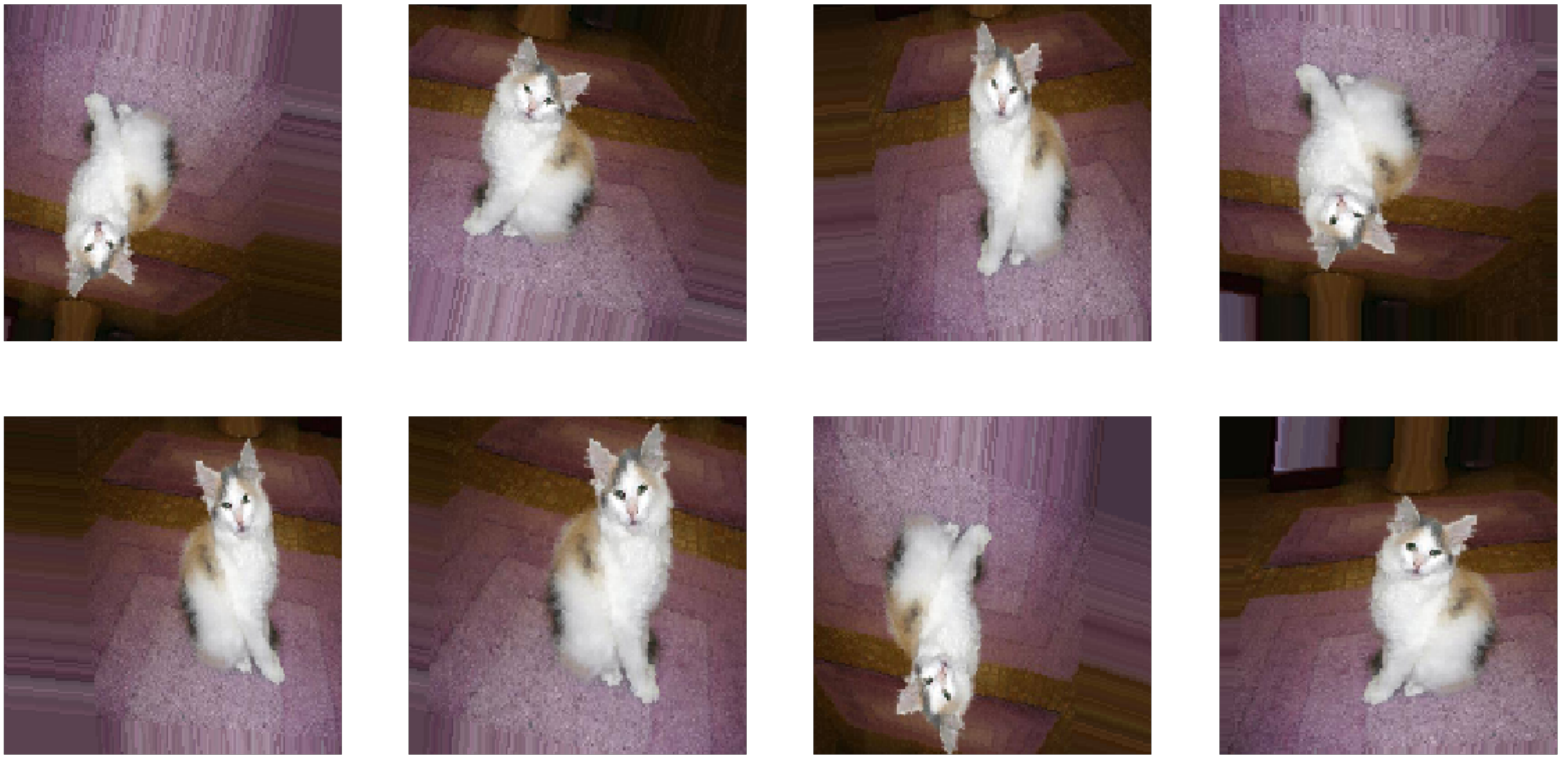


Hierarchical Features & Max Pooling

- pooling reduces spatial size and number of parameters
- avoids overfitting
- commonly: max pooling



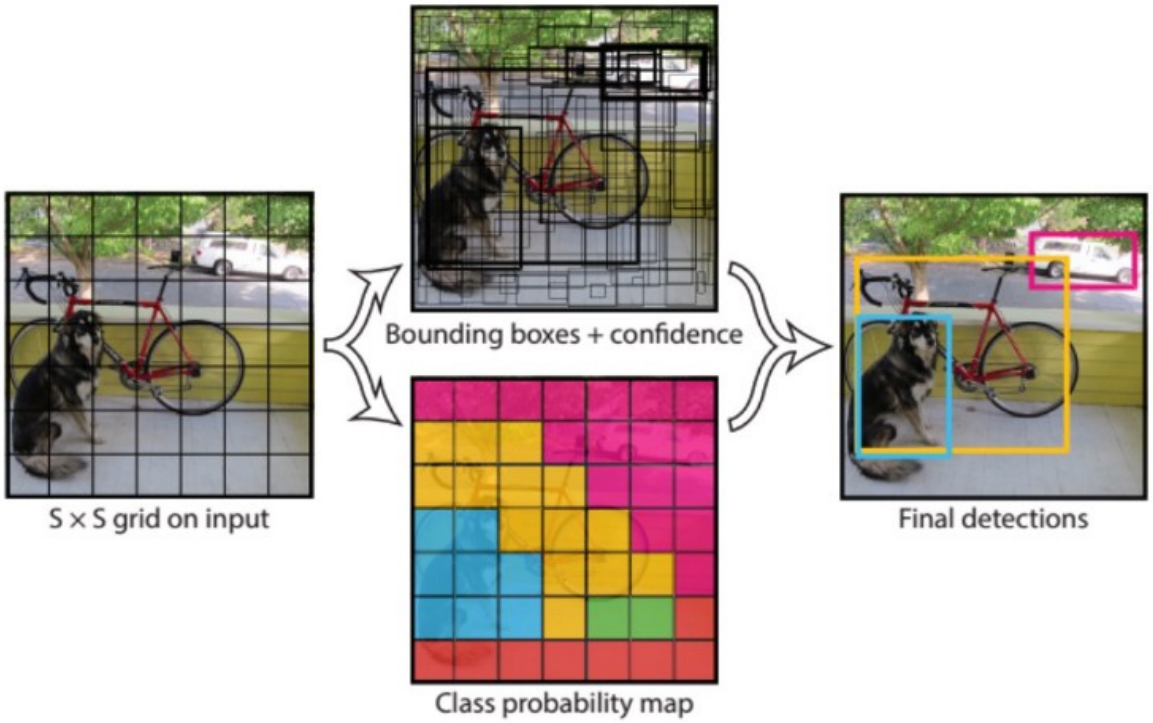
Data Augmentation



YOLO

- YOLO: You Only Look Once
- state-of-the-art real-time object detection system
- https://pjreddie.com/darknet/yolo/
-

YOLO: Idea


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

YOLO: Architecture & Limitations

- YOLO: You Only Look Once
- state-of-the-art real-time object detection system
- https://pjreddie.com/darknet/yolo/
-

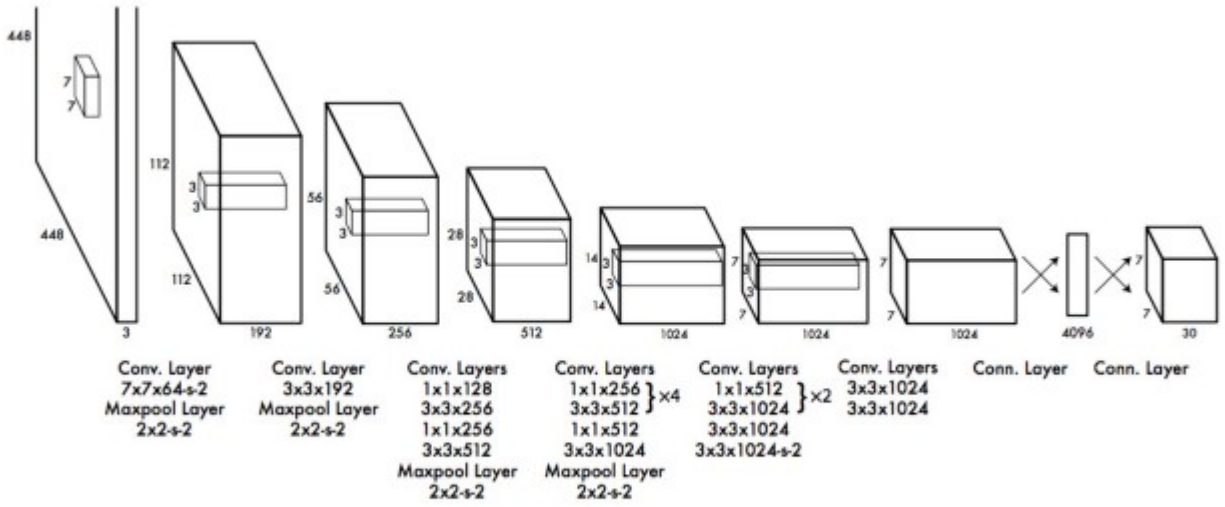


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Homework

Download and install MxNet or TensorFlow (choose if you want to use CPU or also GPU)

https://mxnet.incubator.apache.org/install/index.html
https://www.tensorflow.org/install/install_linux

Follow the MNIST tutorial:
MxNet
https://mxnet.incubator.apache.org/tutorials/python/mnist.html
TensorFlow
https://www.tensorflow.org/versions/r1.1/get_started/mnist/beginners
https://www.tensorflow.org/versions/r1.2/get_started/mnist/pros

Change parameters, layers, activation function, etc. of your network and compare results.

Write a short summary / report to discuss at correction time.