Deep Learning and PyTorch Tutorial

17th Jan 2023
Skanda Vaidyanath
CS 234

Overview

- Motivation
- Neural networks
- Backpropagation
- Training and Optimization
- CNNs
- PyTorch

Motivation

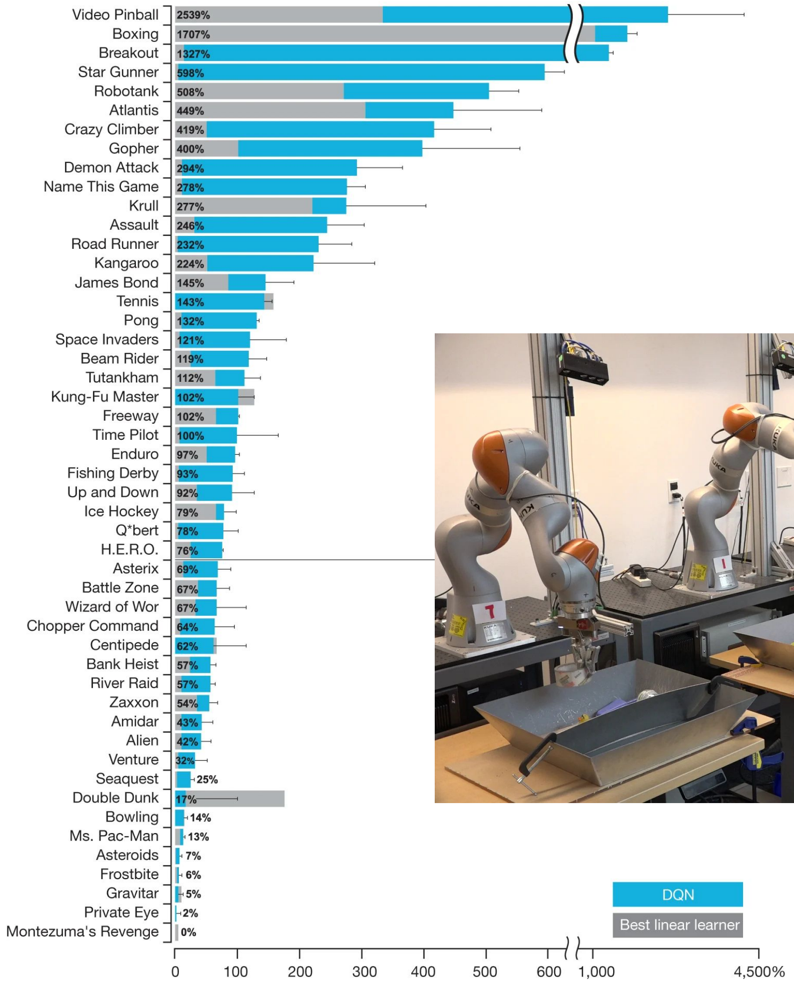
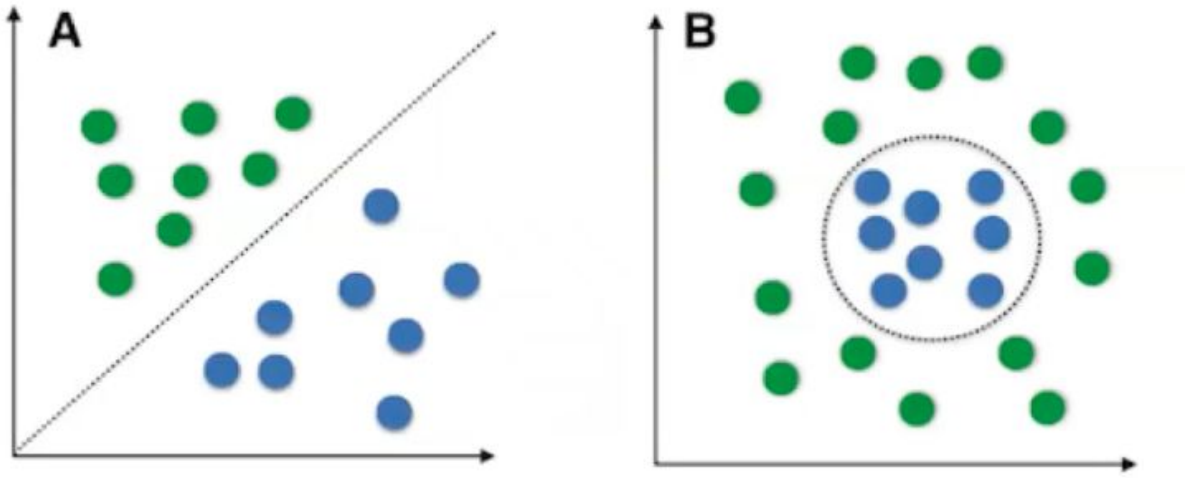
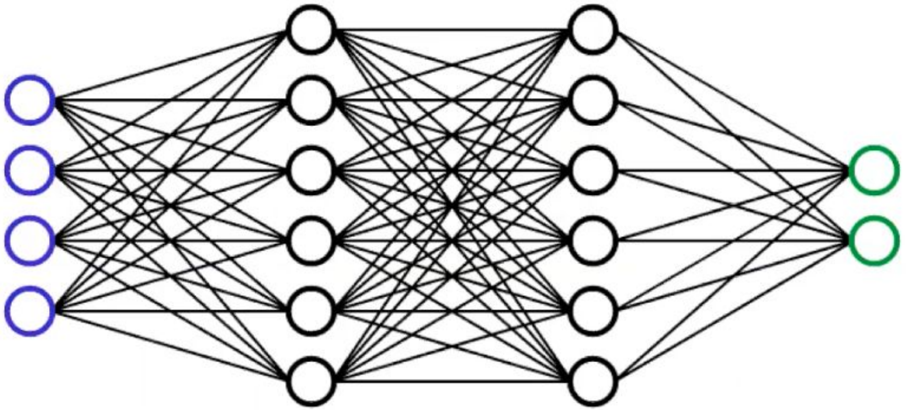


Image source

The need for non-linear models



Neural networks



x $\quad h_1 = \phi(W_1 x + b_1) \quad h_2 = \phi(W_2 h_1 + b_2) \quad y = \phi_{out}(W_3 h_2 + b_3)$

Non-linearities

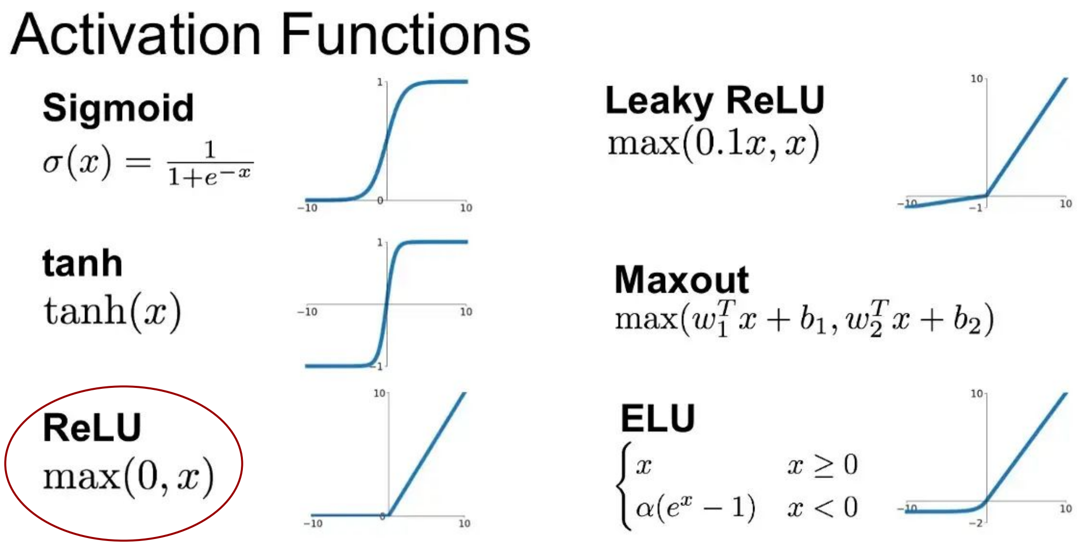


Image credit: Medium article

Computational graphs

Eg: f(x,y,z) = (x+y)z

What is the derivative of f wrt x, y, z?

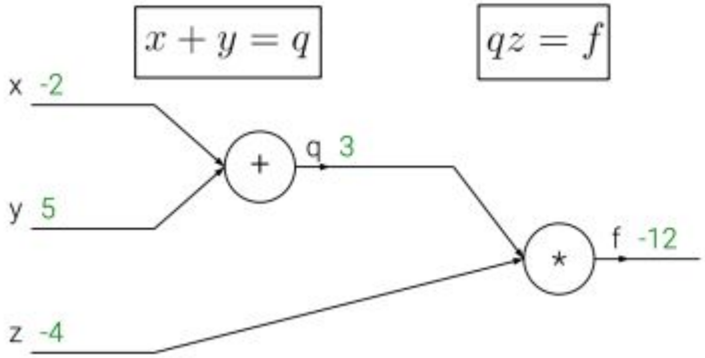
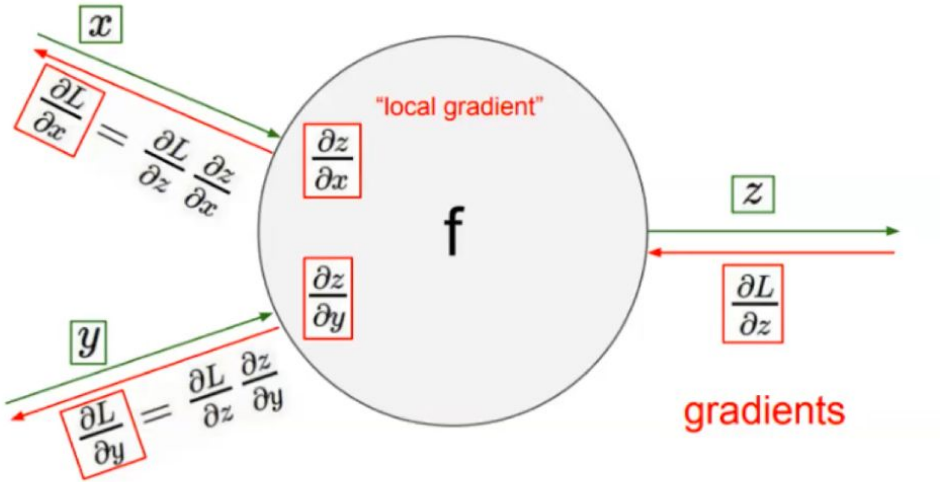


Image credit: [CS231N Slides]

Backpropagation

Using the chain rule to calculate gradients



Training a neural network

Loss function, usually an average over data points:

$$L(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\theta; x, y)$$

Usually MSE loss or cross-entropy loss, etc.

Want to find

$$\theta^\star \in \arg\min_\theta L(\theta)$$

Optimization

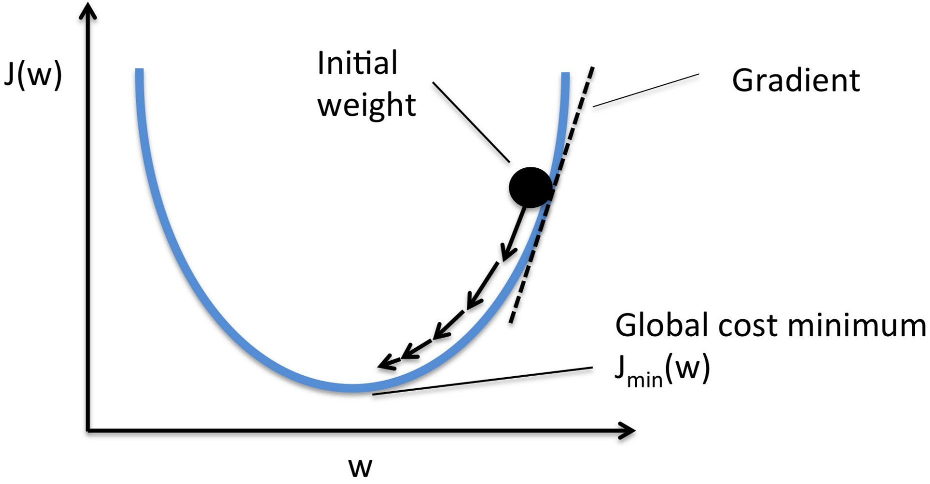


Image credit: [Blog post]

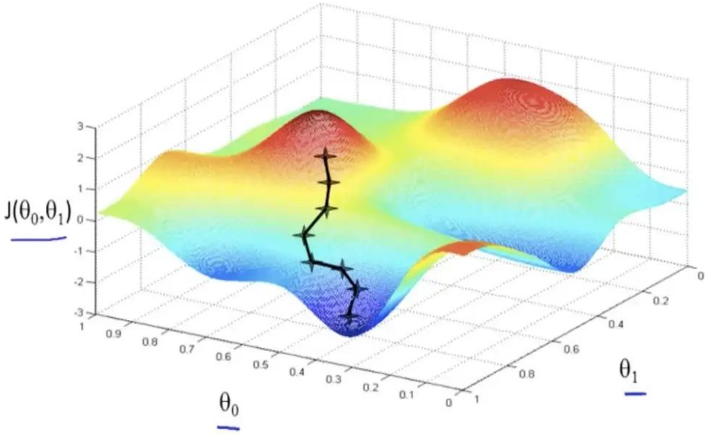


Image credit: [Medium article]

Stochastic Gradient Descent

Often it is infeasible/ inefficient to do full batch gradient descent, so we resort to stochastic gradient descent (SGD)

1. Pick some θ_0, then iterate:
2. $\theta_{k+1} = \theta_k - \alpha_k \nabla L(\theta_k)$

$$\nabla L(\theta) = \frac{1}{|\mathscr{D}|} \sum_{(x,y)\in\mathscr{D}} \nabla_\theta \ell(\theta; x, y) \qquad O(|\mathscr{D}|)$$

SGD: sample mini-batch $B \subseteq \mathscr{D}$, use

$$\hat{\nabla} L(\theta) = \frac{1}{|B|} \sum_{(x,y)\in B} \nabla_\theta \ell(\theta; x, y) \qquad O(|B|)$$

Note $\mathbb{E}[\hat{\nabla} L(\theta)] = \nabla L(\theta)$

Other optimizers

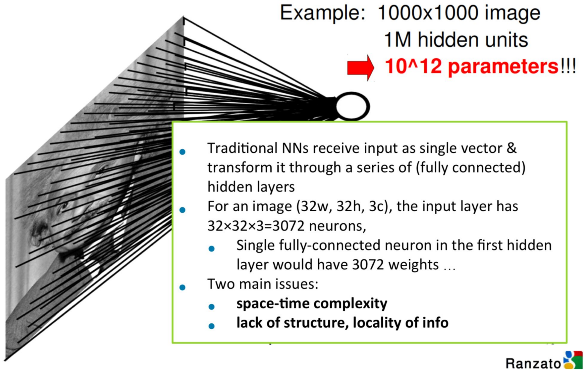
There are several new optimizers that improve on vanilla gradient descent. Some of them are:

- Momentum
- Adagrad
- RMSProp
- Adam

To learn more: https://ruder.io/optimizing-gradient-descent/

Convolutional Neural Networks

- CNNs are extensively used in computer vision
- If we want to go from pixels to decisions, likely useful to leverage insights from visual input
- Why not just use fully connected networks?



Convolutional Neural Networks

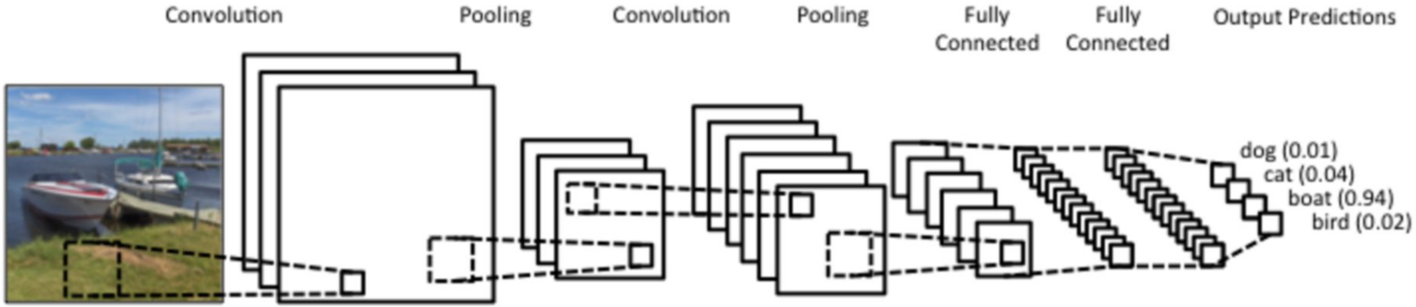
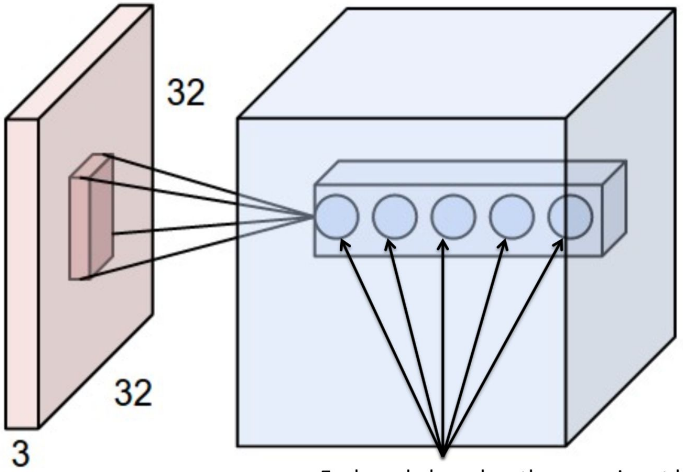


Image: http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-07-at-7.26.20-AM.png

Convolutional Neural Networks

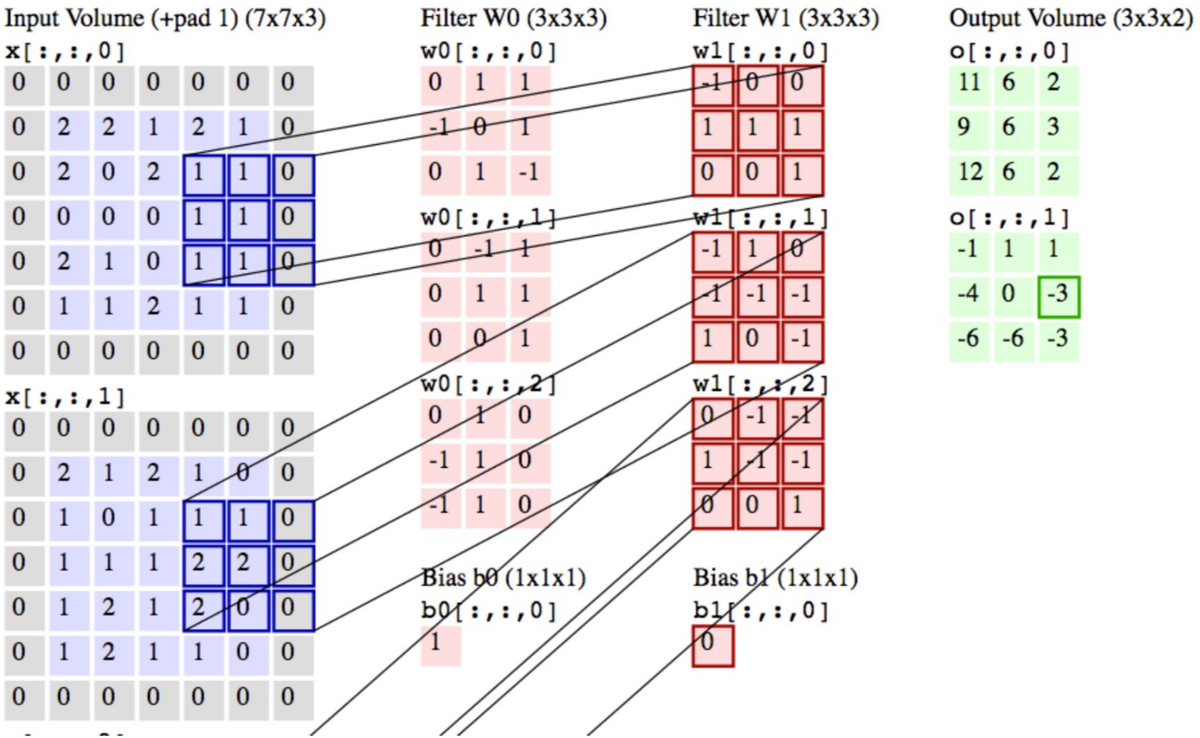
- Image dimensions
- Filters/ kernels
- Stride
- Padding



Each node here has the same input but different weight vectors (e.g. computing different features of same input)

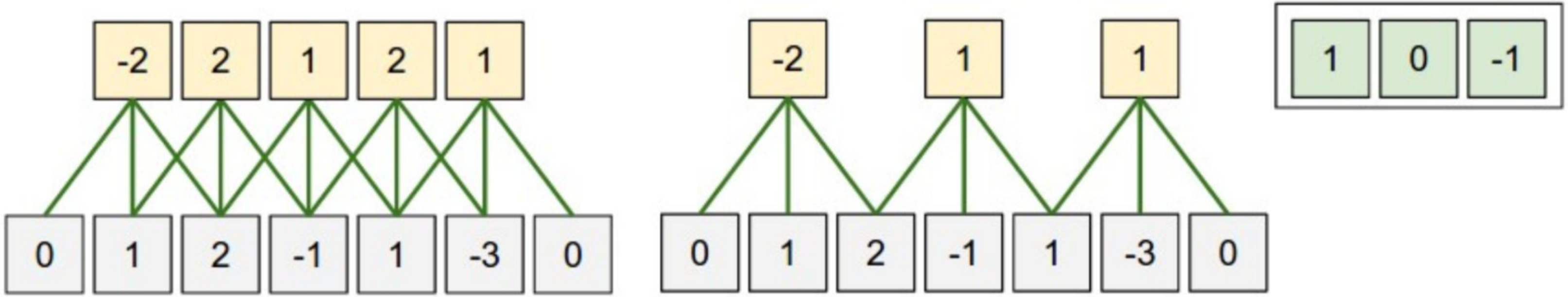
Convolutional Neural Networks

Full demo: https://cs231n.github.io/convolutional-networks/



Convolutional Neural Networks

Simpler example:



PyTorch

PyTorch

- Objective: introduce key PyTorch concepts for CS234
- PyTorch documentation: https://pytorch.org/docs/stable/index.html
- Tutorial: Python notebook will be shared after class