UCF CENTER FOR RESEARCH IN COMPUTER VISION

CAP5415
Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Administrative

- PA-1 deadline approaching soon
- PA-2 will be released soon
  - There will be strictly no deadline extension
  - Start early
  - If you are visiting office hours right before the deadline, nothing much we can do to help you, of course we will try our best.
- Tutorial for PA-2
  - TODO
- GPU resources
- Course project
- Mid-term: Oct 22, during lecture time via Zoom and in-person

Questions?

Training Neural Networks

Lecture 7

Agenda

- Basics – recap
- Optimization
- Backpropagation
- Practical aspects
- CNN variants

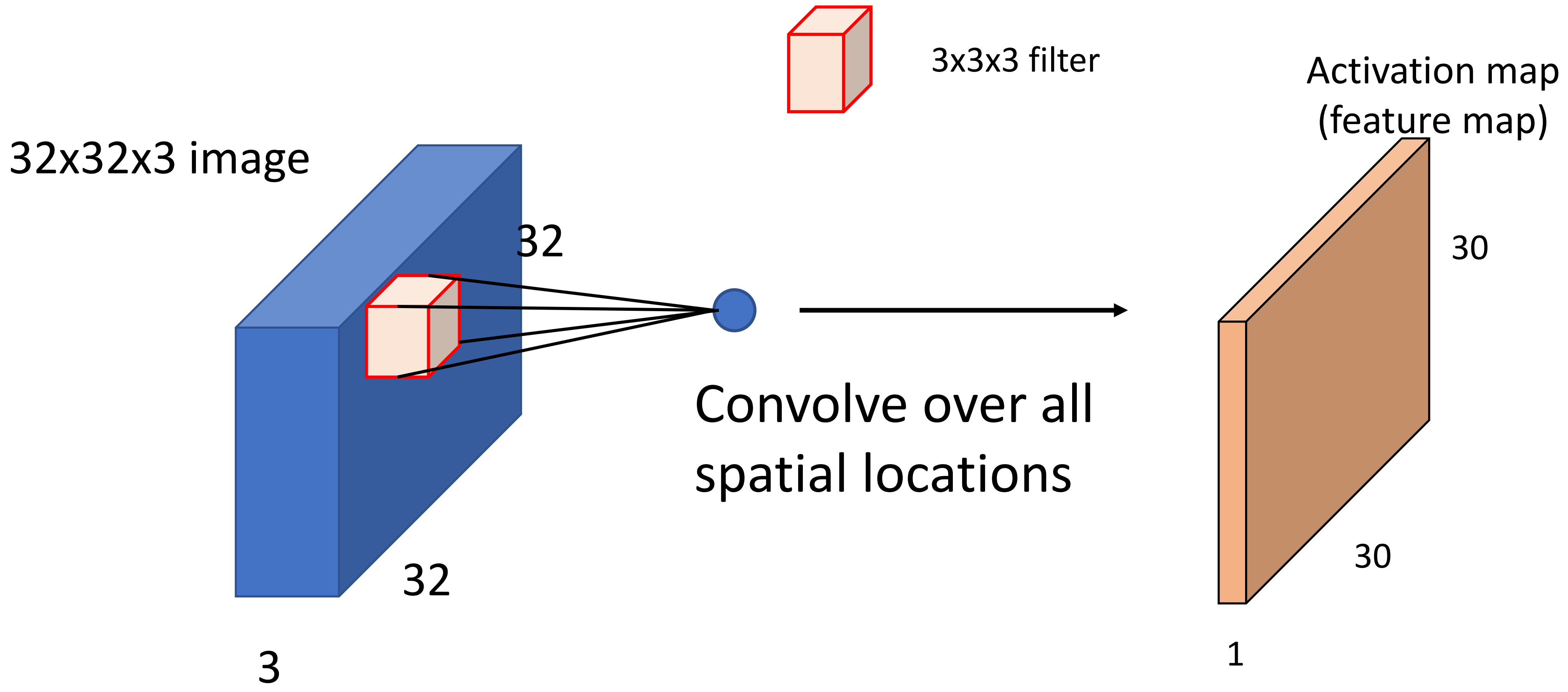
Training Neural Networks

Lecture 7

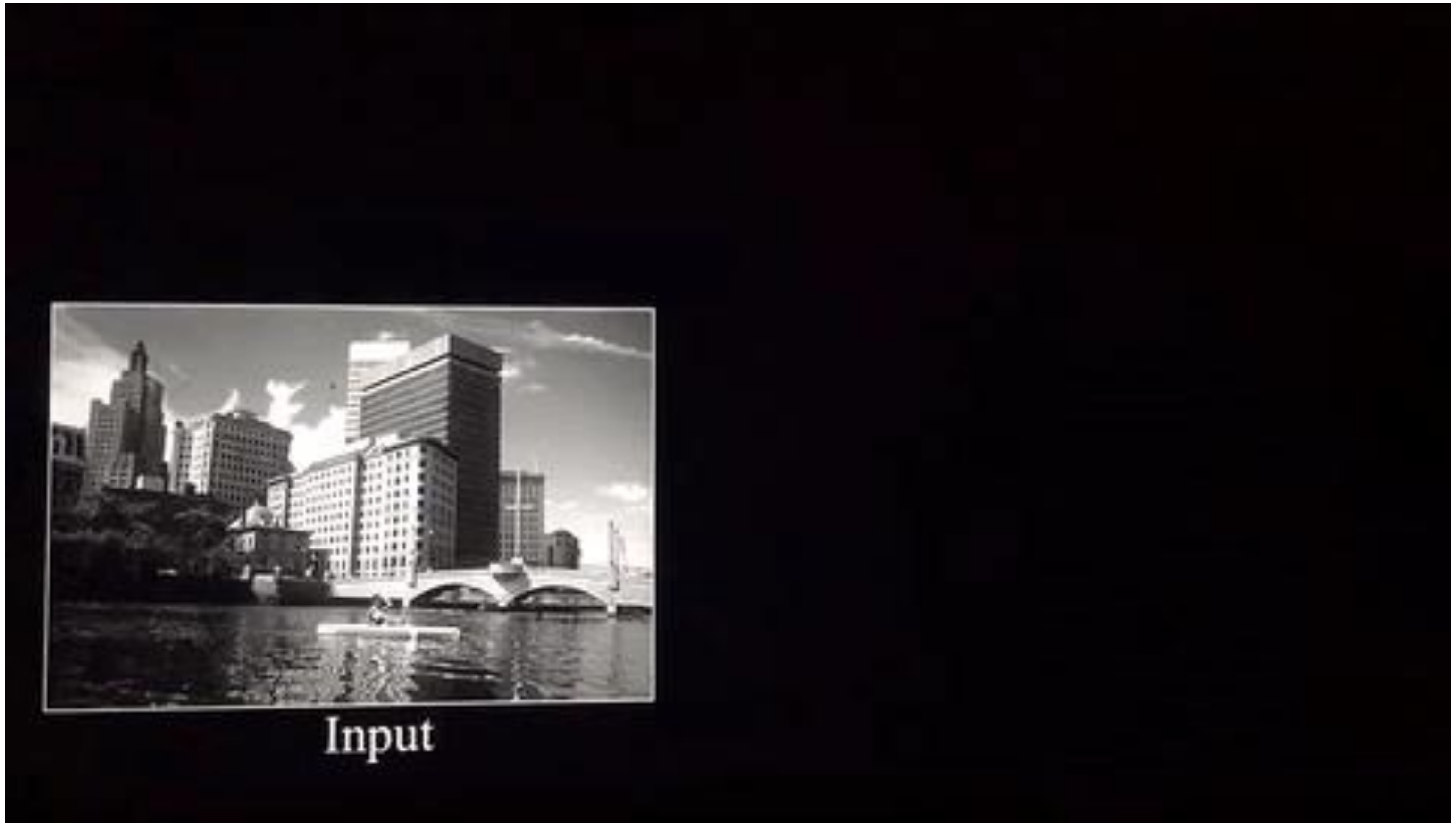
Basics

UCF CENTER FOR RESEARCH IN COMPUTER VISION

Network Parameters - recap

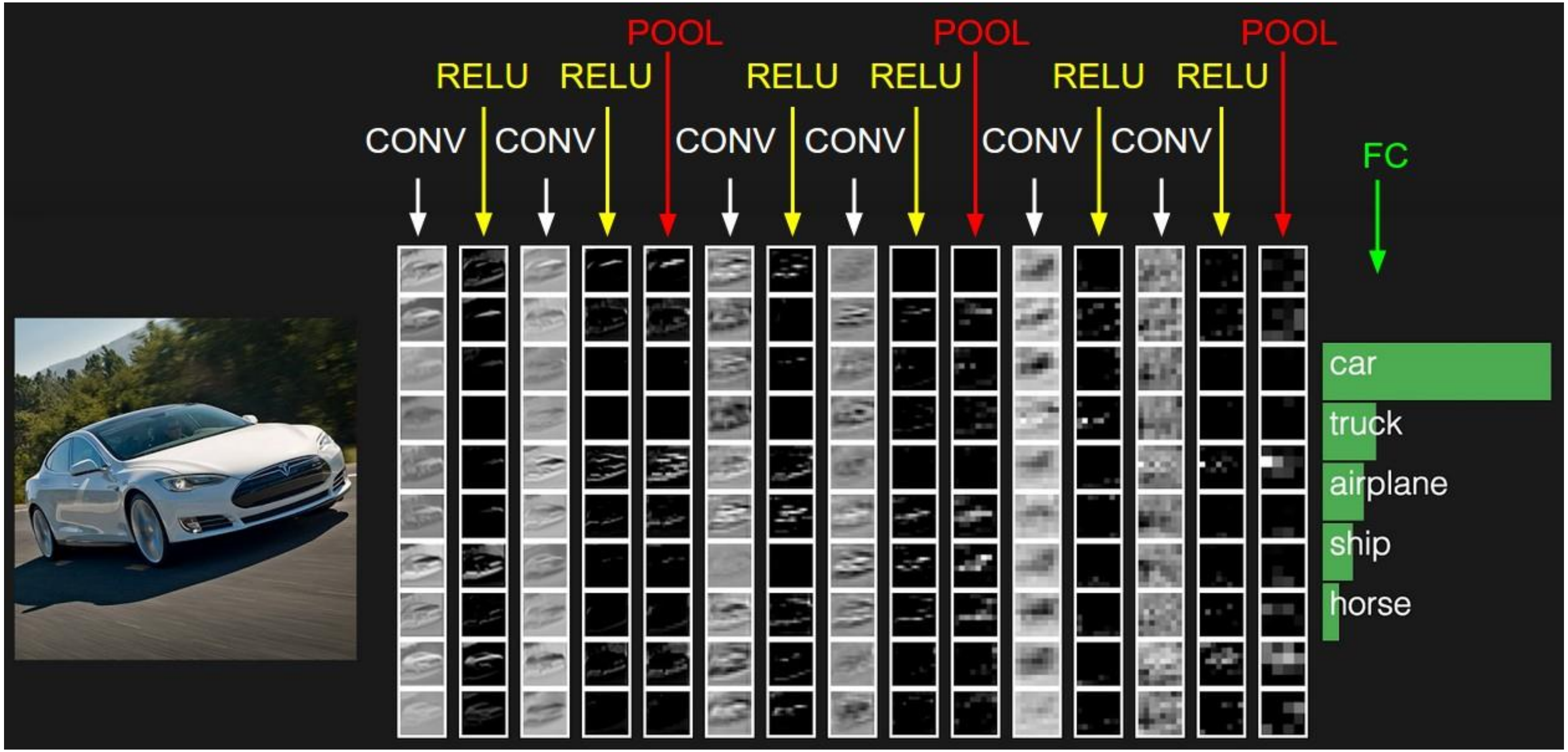


Convolution - Intuition



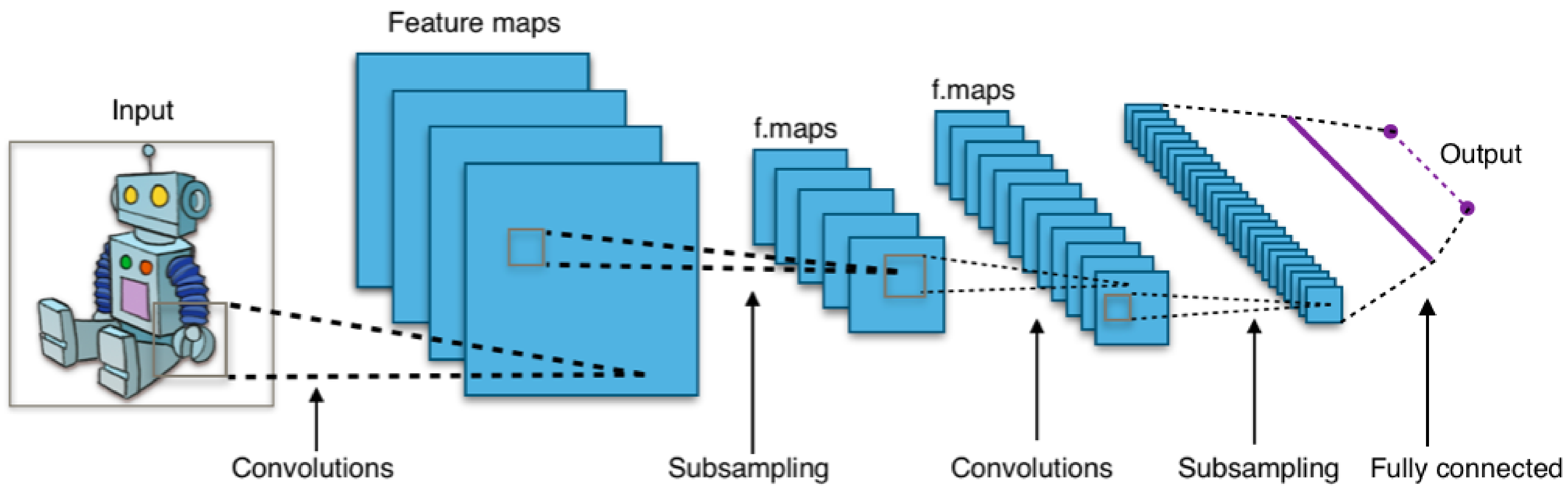
Source : https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Visualizing CNN

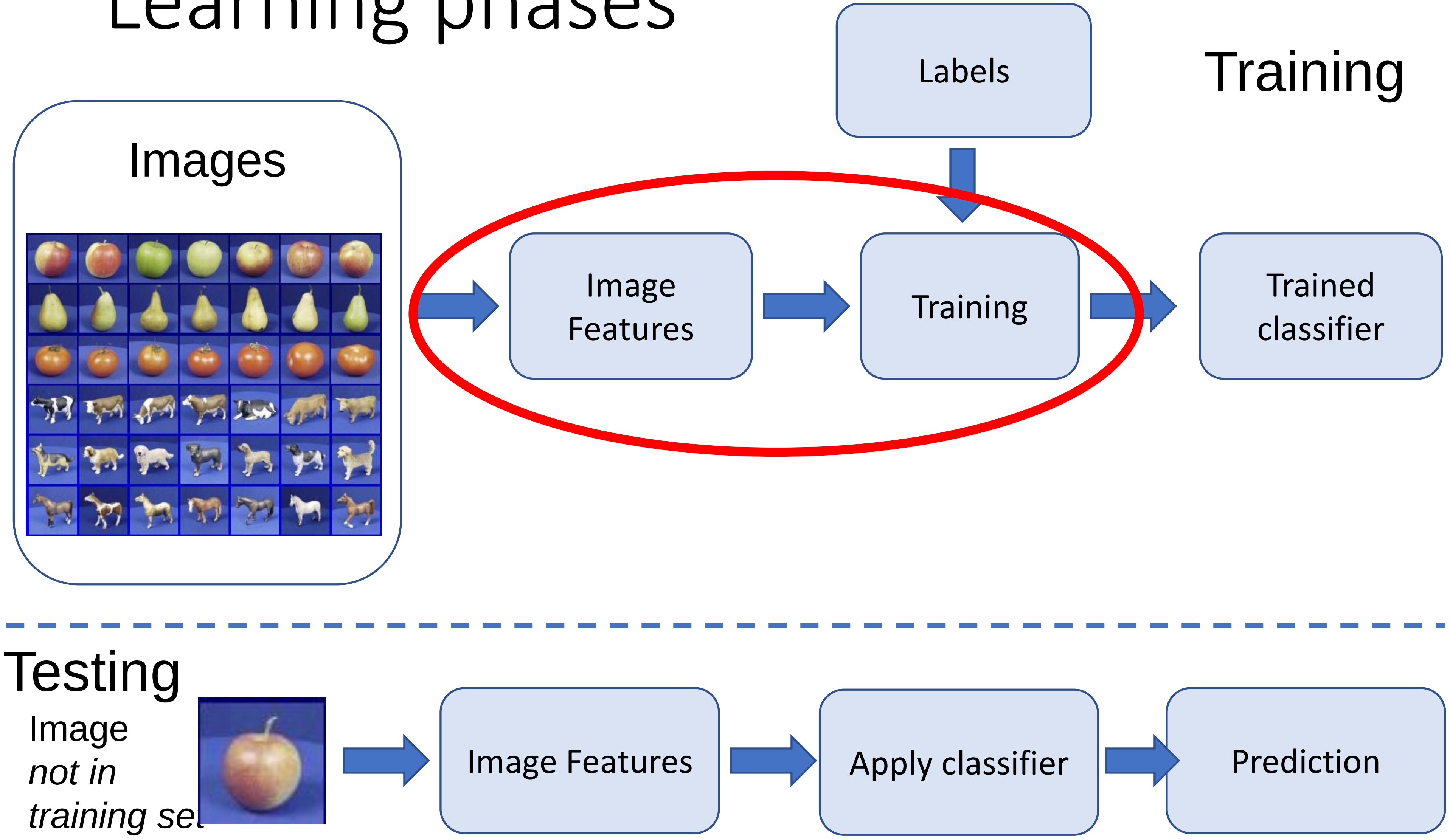


Source : http://cs231n.github.io

General CNN architecture - recap



Learning phases



Slide credit: D. Hoiem and L. Lazebnik

Questions?

Training Neural Networks

Lecture 7

Optimization

Loss Function

- Way to define how good the network is performing
  - In terms of prediction
- Network training (Optimization)
  - Find the best network parameters to minimize the loss

input

Ground truth

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), yi)$$

Network parameters

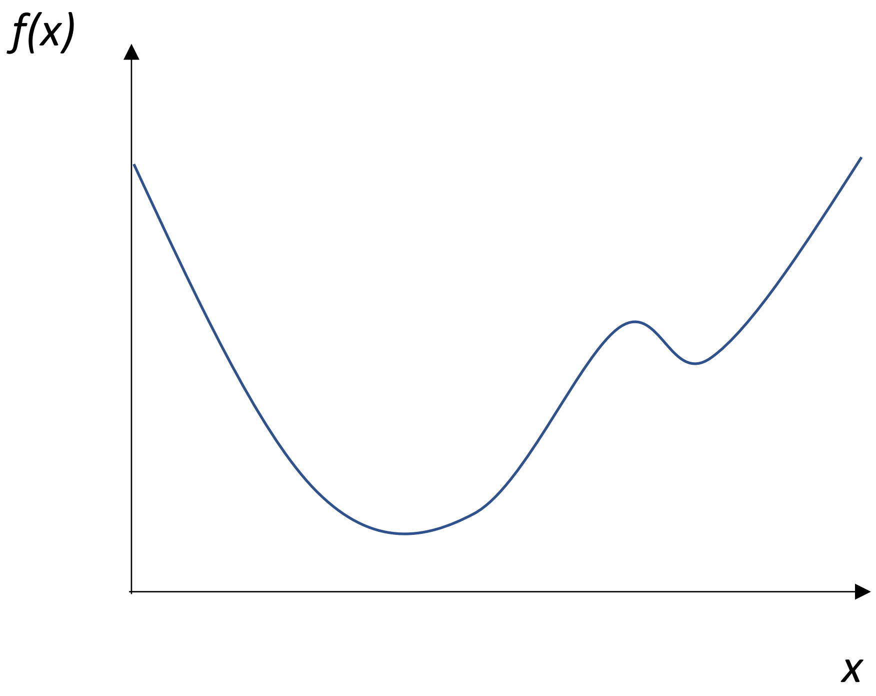
Loss function

network

Network Training – Minimize Cost

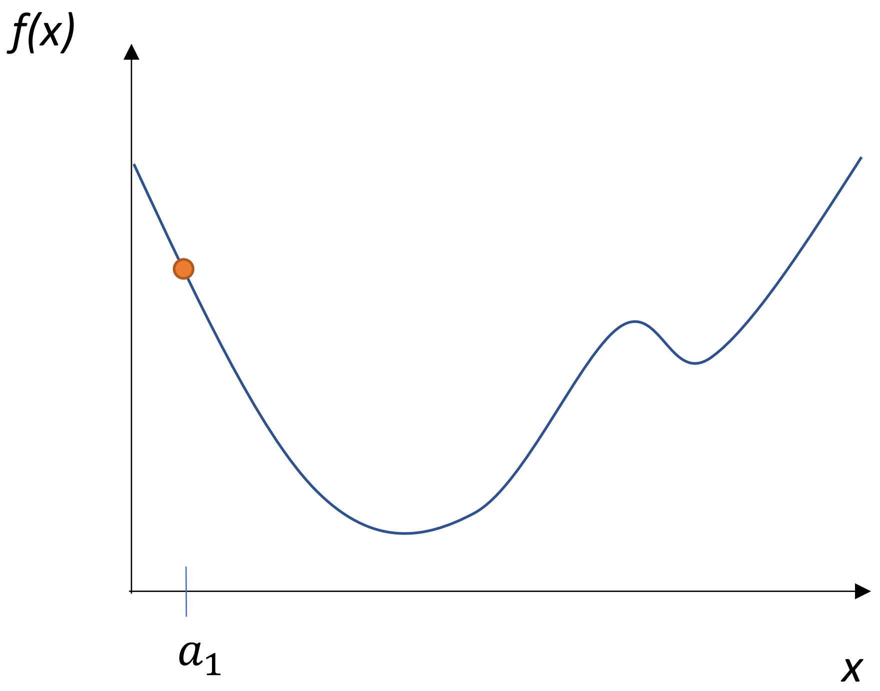
- Gradient Descent
  - Way to minimize a cost function

Gradient descent



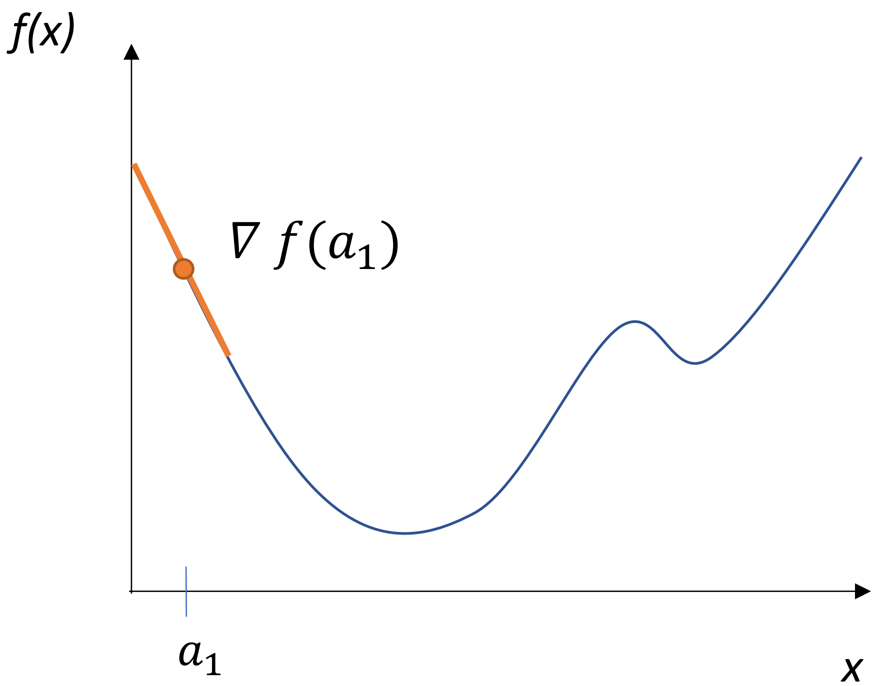
General approach

Pick random starting point.



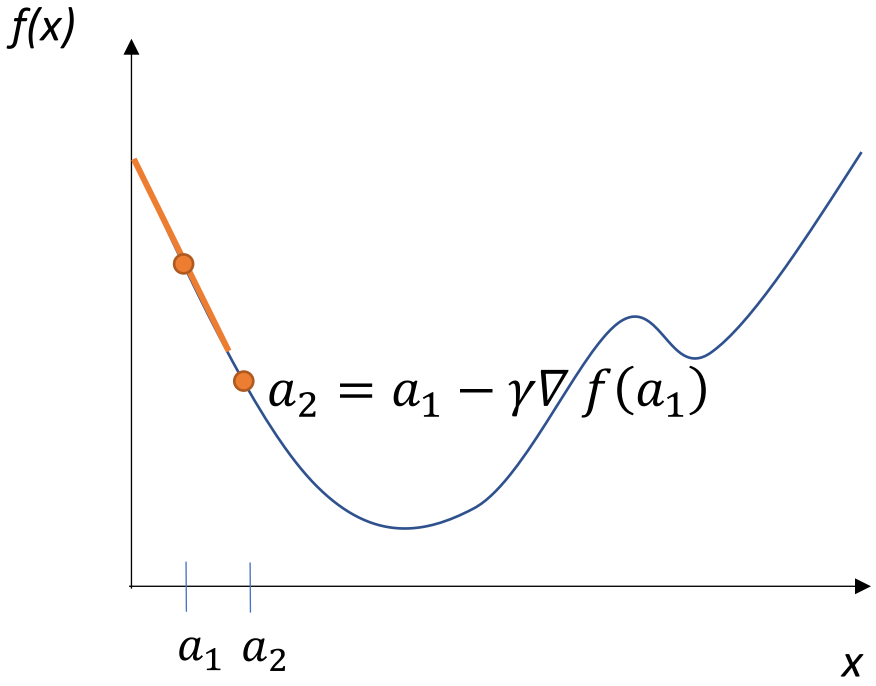
General approach

Compute gradient at point (analytically or by finite differences)



General approach

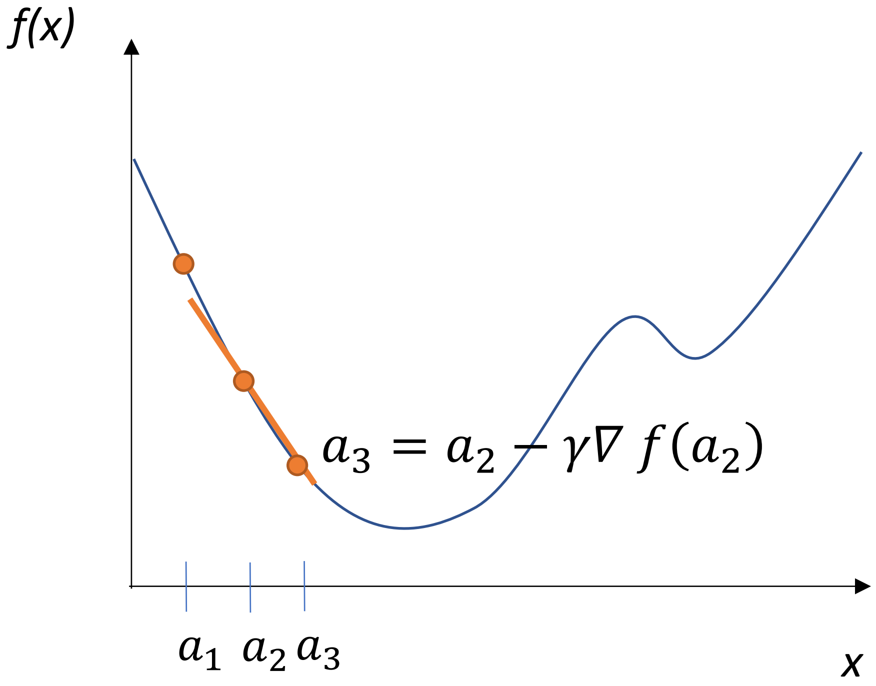
Move along parameter space in direction of negative gradient



γ = amount to move
= *learning rate*

General approach

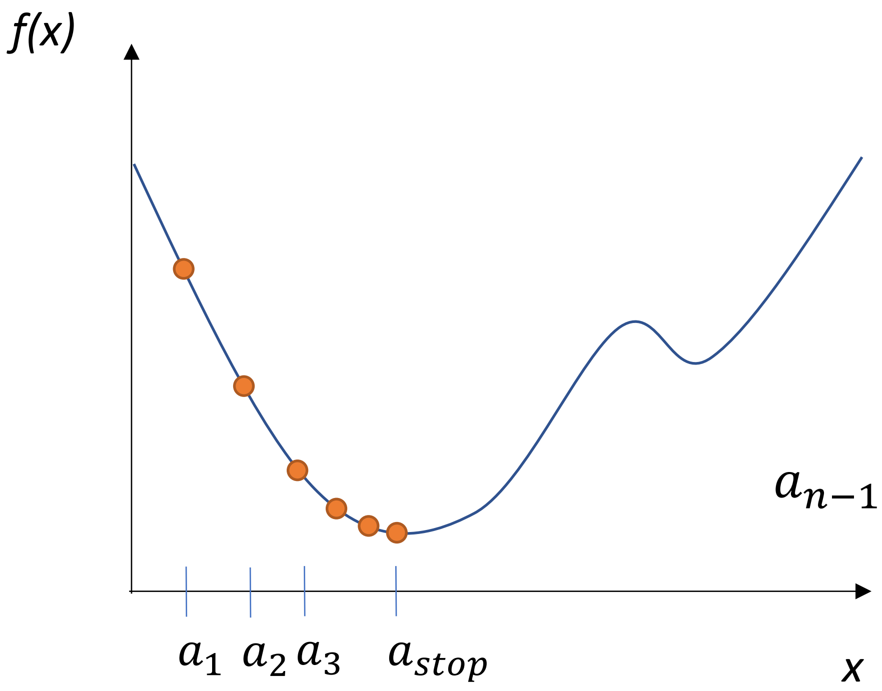
Move along parameter space in direction of negative gradient.



γ = amount to move
= *learning rate*

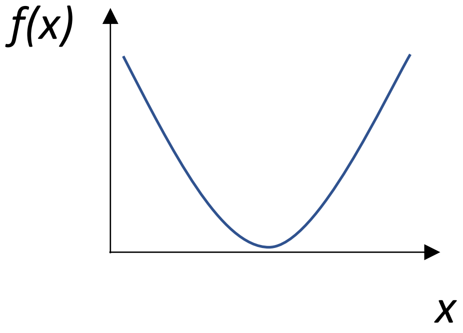
General approach

Stop when we don't move any more.



$$a_{stop}:$$
$$a_{n-1} - \gamma \nabla f(a_{n-1}) = 0$$

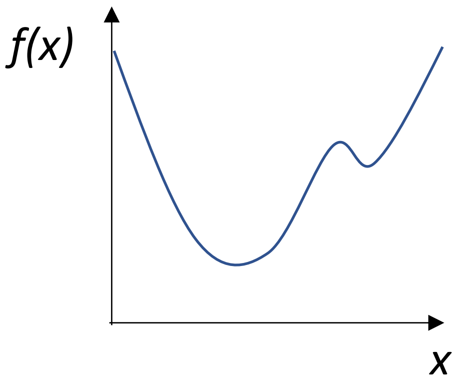
Gradient descent



Optimizer for functions.

Guaranteed to find optimum for convex functions.

- Non-convex = find *local* optimum.
- Most vision problems aren’t convex.



Works for multi-variate functions.

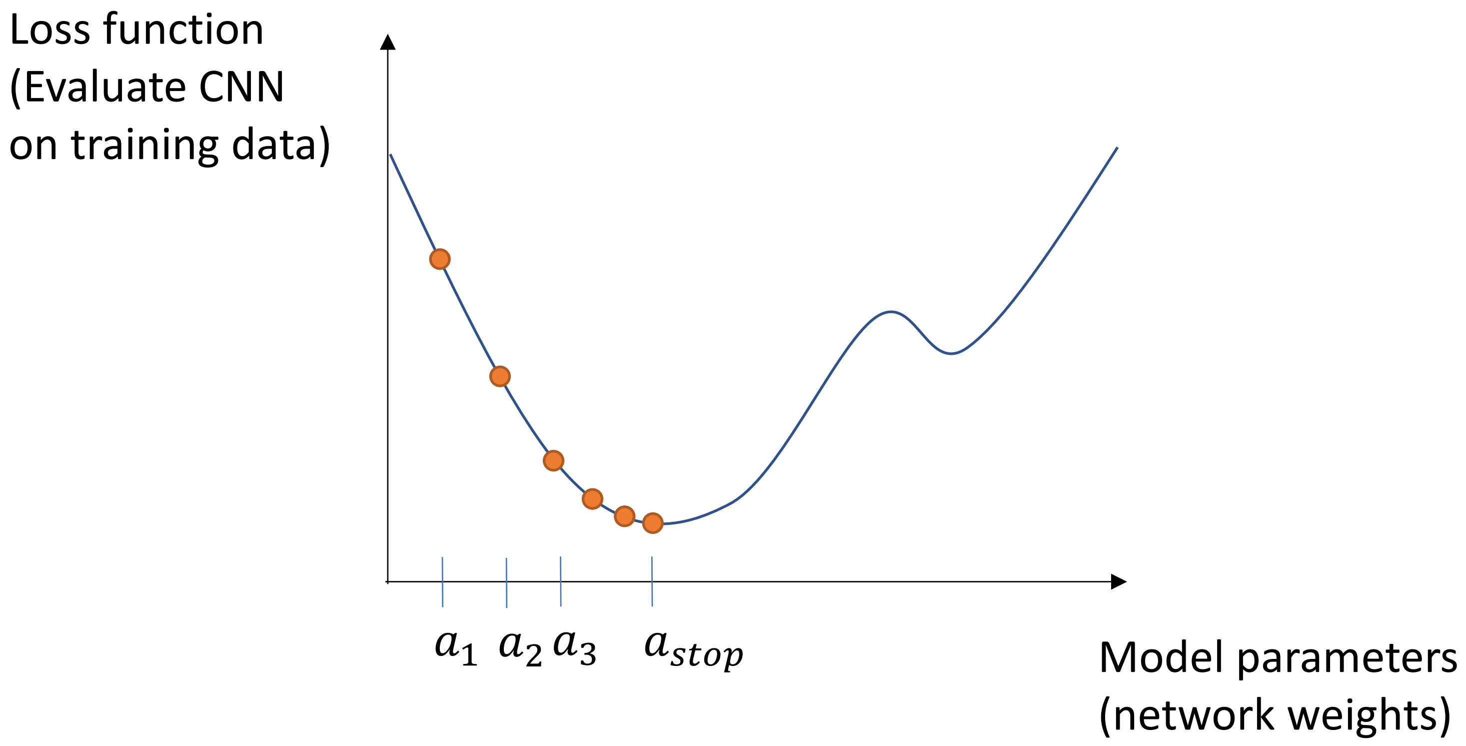
- Need to compute matrix of *partial derivatives (“Jacobian”)*

Train CNN with Gradient Descent

- x^i, y^i = n training examples
- $f(\boldsymbol{x})$ = feed forward network
- $L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ = some *loss function*

Loss function measures how 'good' our network is at classifying the training examples wrt. the parameters of the model (the perceptron weights).

Train CNN with Gradient Descent



Loss Functions

- Cross entropy

Ground-truth

Predicted value

$$-\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1 - y_i)log(1 - \hat{y}_i))$$

- Mean squared error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

CAP5415
Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

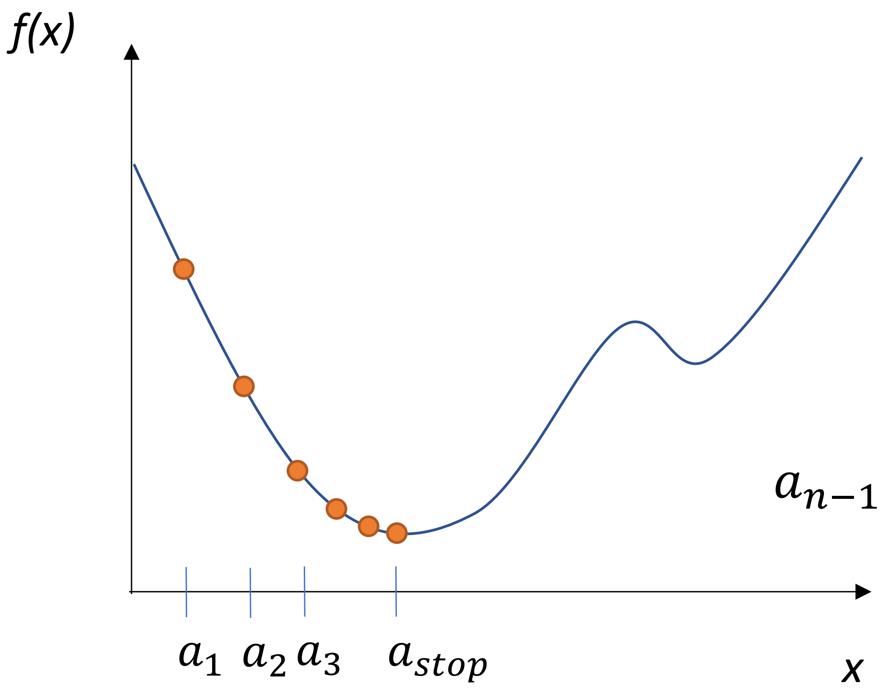
Training Neural Networks

Lecture 7

Backpropagation

General approach - recap

Stop when we don't move any more.



a_{stop}:

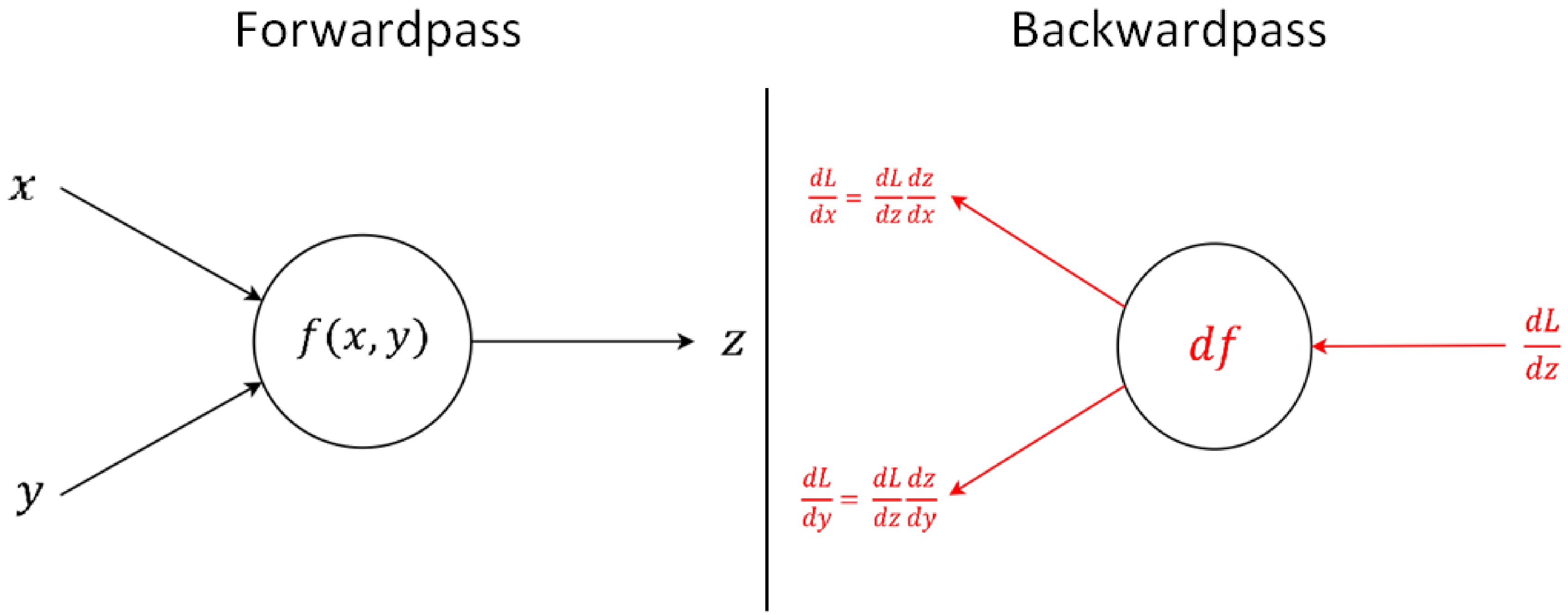
$$a_{n-1} - \gamma \nabla f(a_{n-1}) = 0$$

Differentiability

- Loss function
- Activation function
- Convolution
- Pooling
- …

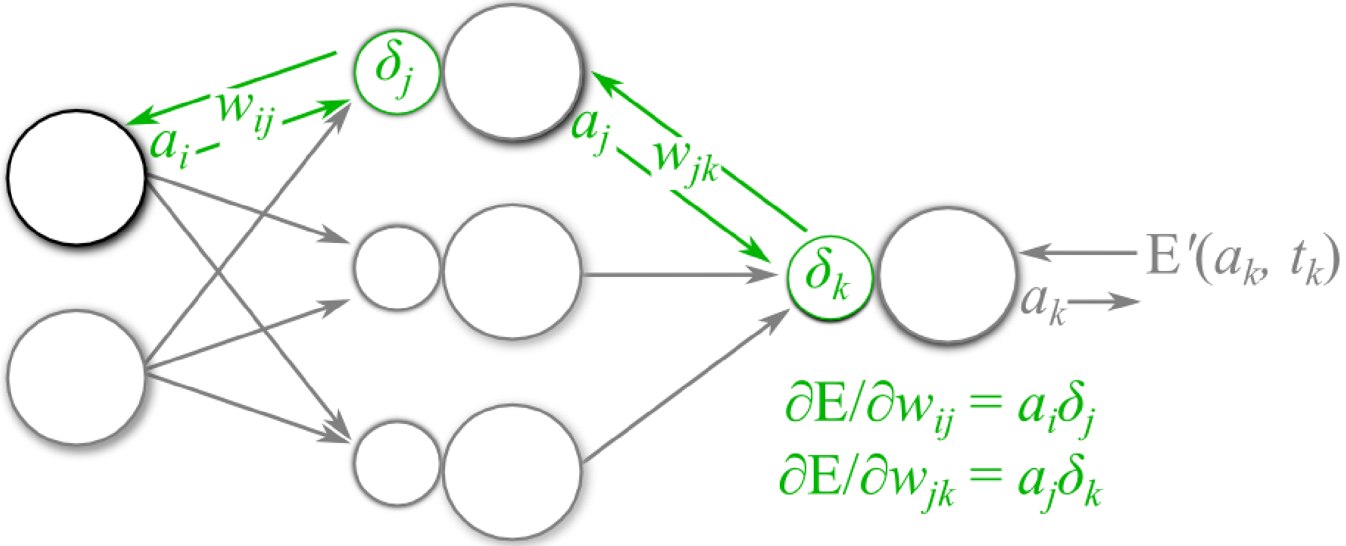
Backpropagation – Chain Rule

- Chain rule $$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$



Backpropagation – Chain Rule

- Add gate:
    - gradient distributor
- Max gate:
    - gradient router
- Mul gate:
    - gradient switcher



Stochastic Gradient Descent

- Dataset can be too large
  - Can not apply gradient descent wrt. all data points.
- Randomly sample a data point
  - Perform gradient descent per sample and iterate.
  - Picking a subset of points: “*mini-batch*”

“Batchsize“

Randomly initialize starting W and pick learning rate γ

While not at minimum:

- Shuffle training set
- For each data point *i=1…n (maybe as mini-batch)*
  - *Gradient descent*

“Epoch“

Questions?

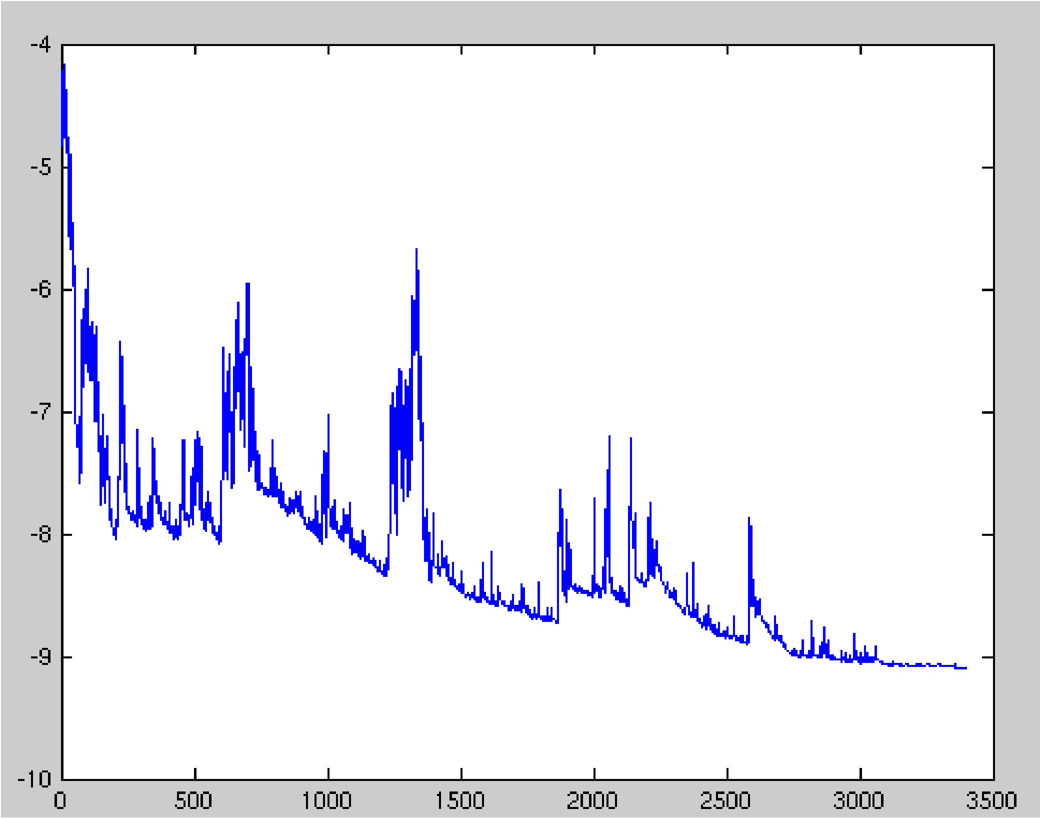
Training Neural Networks

Lecture 7

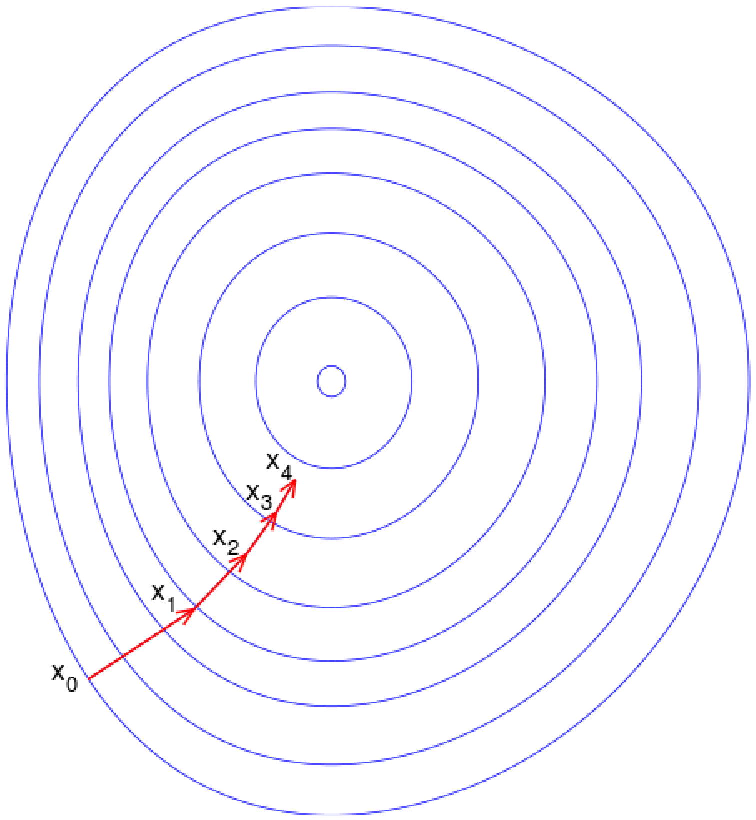
Practical aspects

Stochastic Gradient Descent

- Loss will not always decrease (locally)
  - As training data point is random.
- Still converges over time.

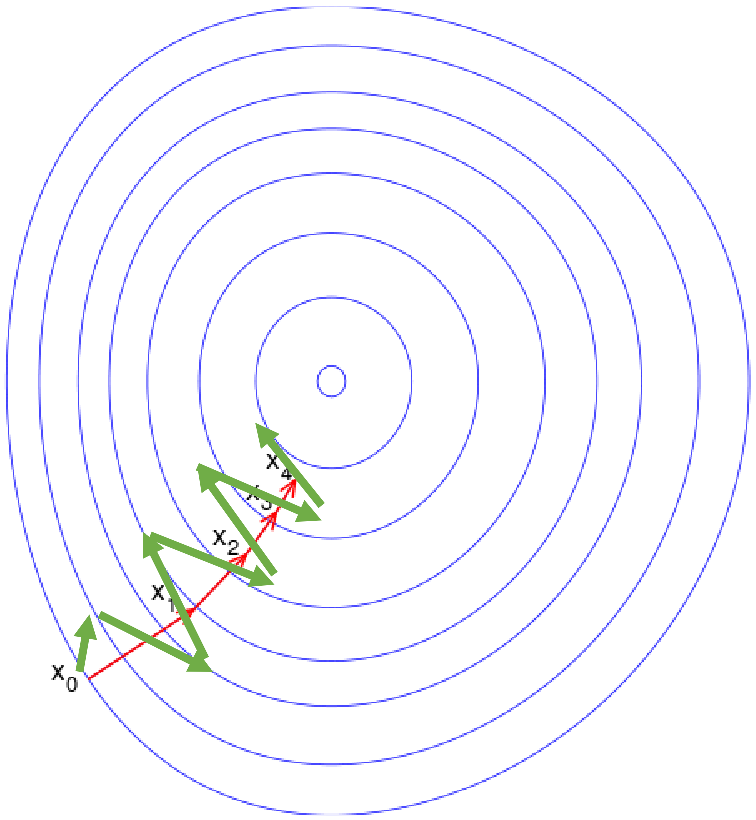


Gradient descent oscillations



Gradient descent oscillations

Slow to converge to the (local) optimum



Momentum

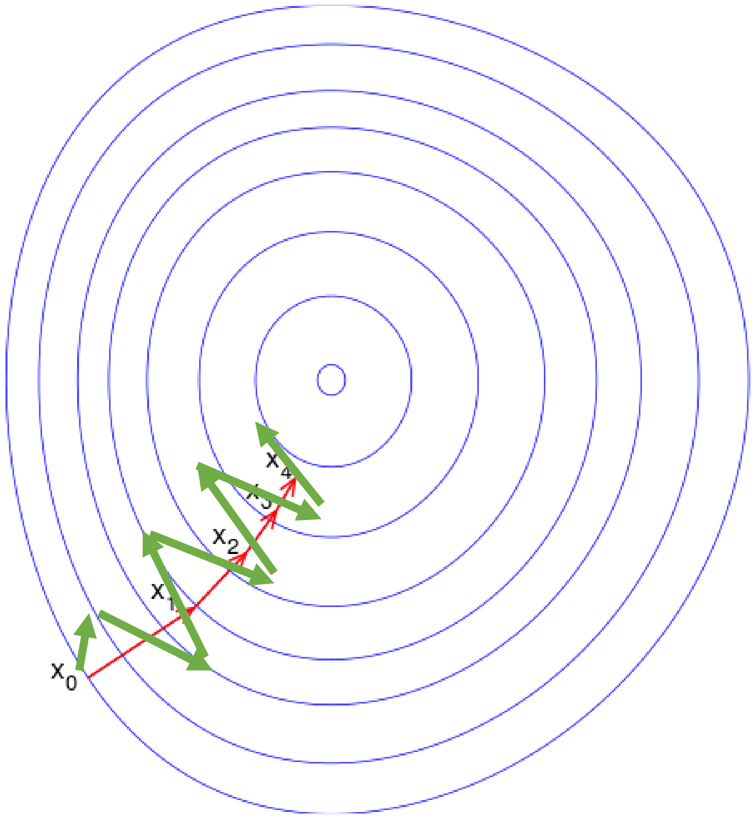
- Adjust the gradient by a weighted sum of the previous amount plus the current amount.

- Without momentum: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \frac{\partial L}{\partial \boldsymbol{\theta}}$

- With momentum (new α parameter):

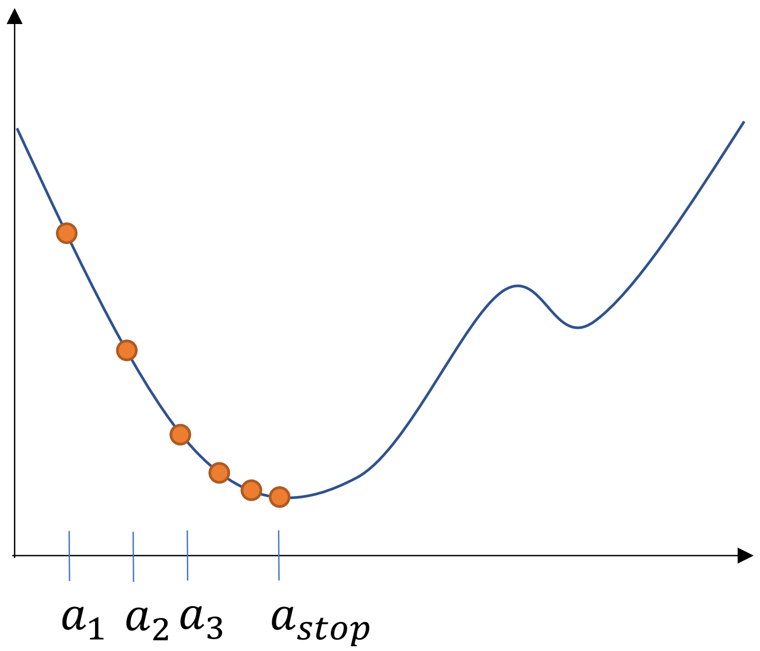
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \left(\alpha \left[\frac{\partial L}{\partial \boldsymbol{\theta}} \right]_{t-1} + \left[\frac{\partial L}{\partial \boldsymbol{\theta}} \right]_t \right)$$

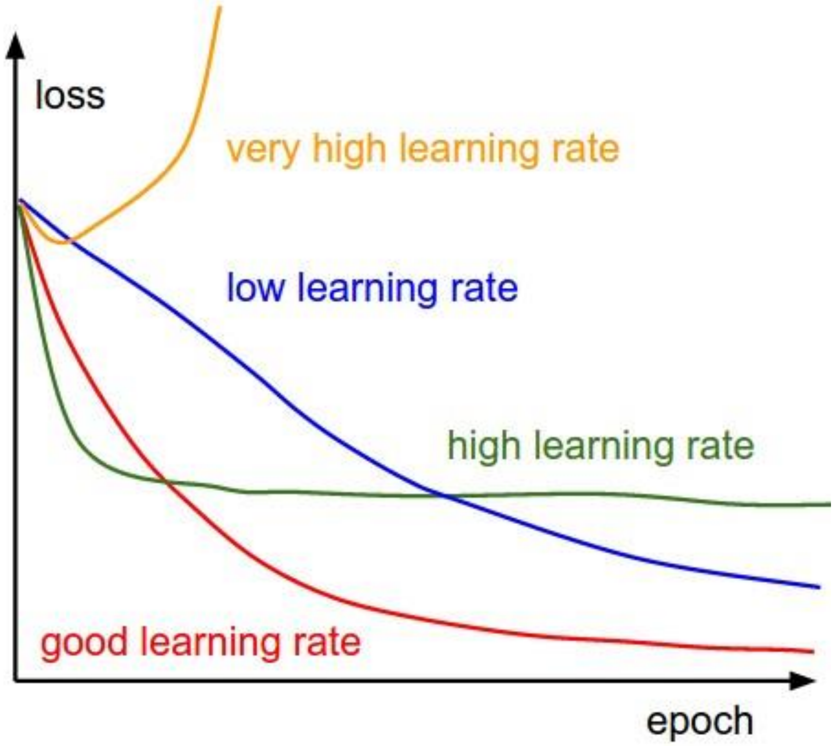
Lowering the learning rate



-Takes longer to get to the optimum

Learning rate



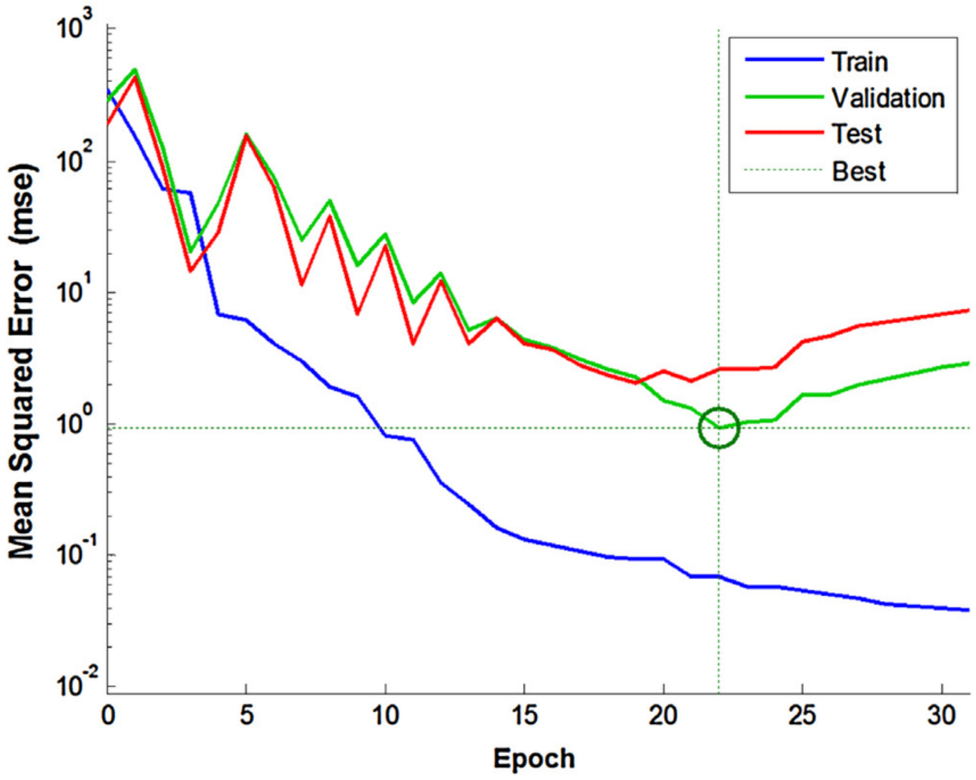


Problem of fitting

- Too many parameters = overfitting
- Not enough parameters = underfitting

- More data = less chance to overfit

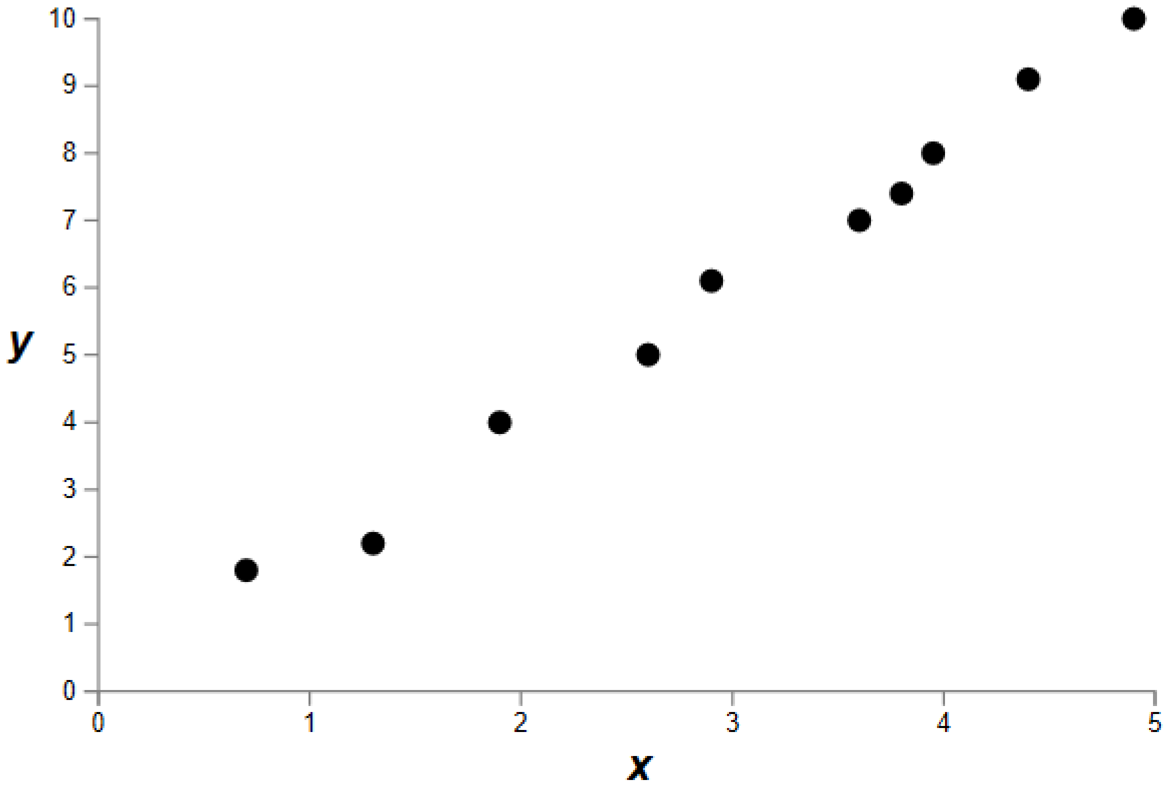
- How do we know what is required?



Regularization

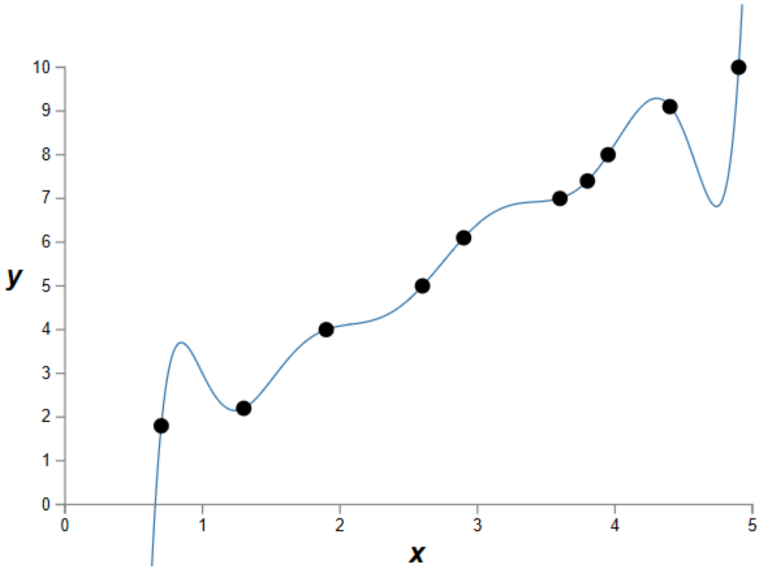
- Attempt to guide solution to *not overfit*
- But still give freedom with many parameters

Data fitting problem

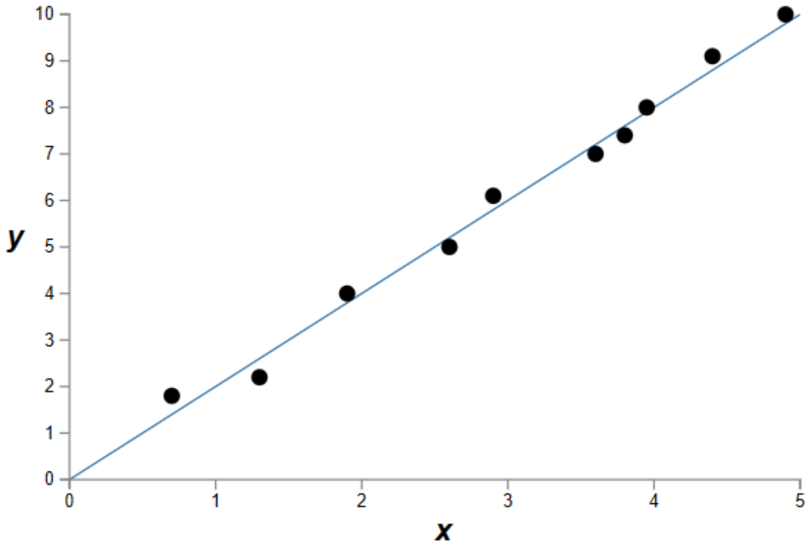


[Nielson]

Which is better?



9th order polynomial

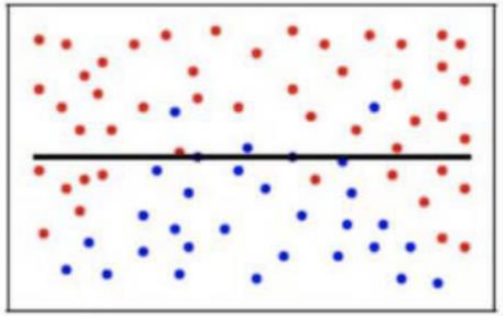


1st order polynomial

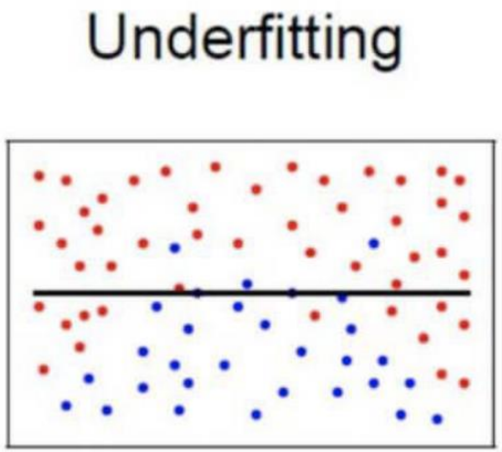
[Nielson]

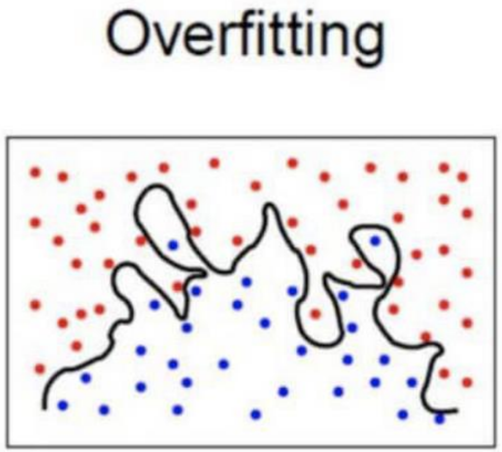
Data fitting problem

Underfitting

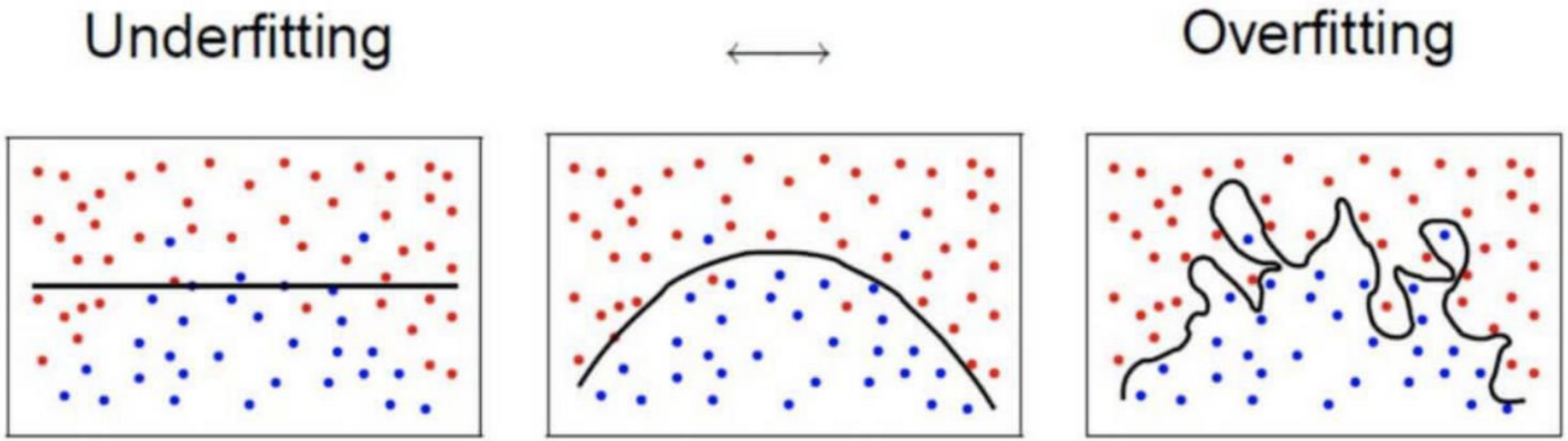


Data fitting problem





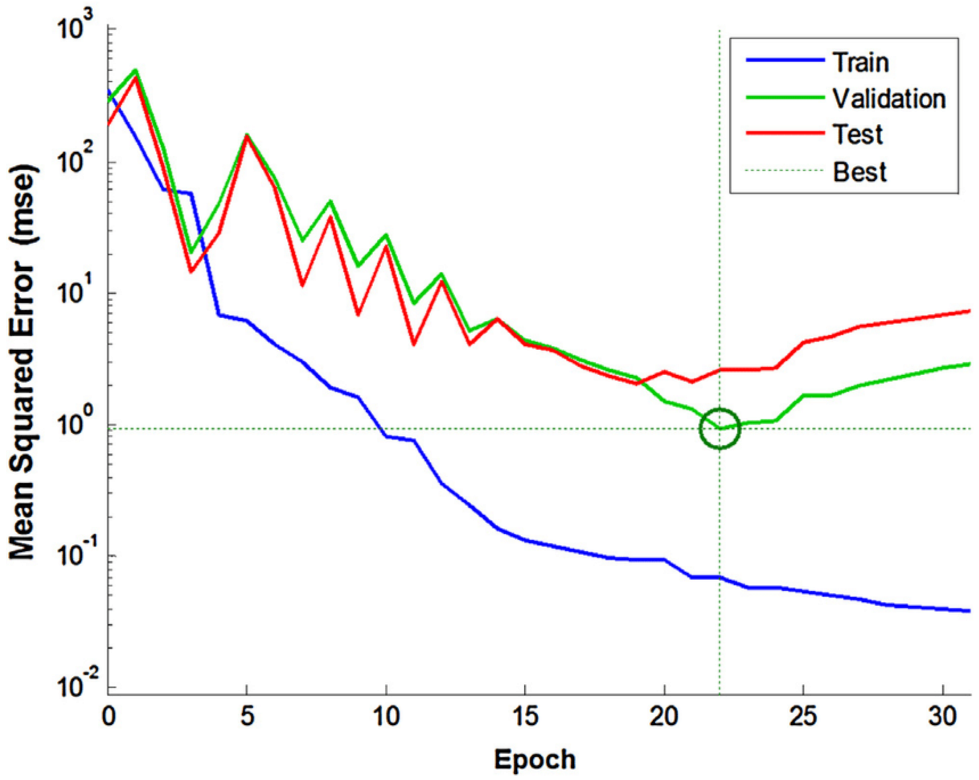
Data fitting problem



- Early stopping
- Regularization
- Dropout
- …

UCF Center for Research in Computer Vision

Early stopping



Regularization

- Attempt to guide solution to *not overfit*
- But still give freedom with many parameters

- Idea:
  Penalize the use of parameters to prefer small weights.

Regularization

- Add a cost to having high weights

Loss, Weight decay, Regularization

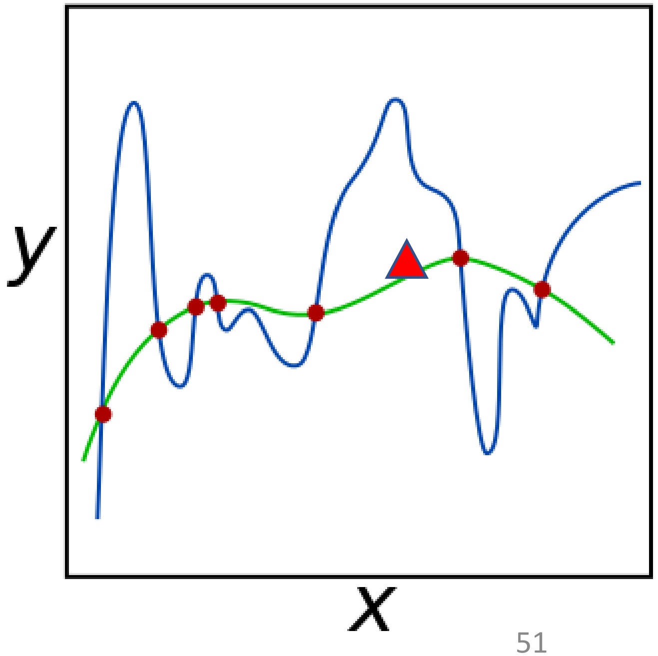
$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), yi) + \lambda R(W)$$

- In common use,

  L1 Norm – $R(W) = \sum_i \sum_j |Wij|$

  L2 Norm – $R(W) = \sum_i \sum_j W^2{}_{ij}$

  Elastic net – $R(W) = \sum_i \sum_j \beta W^2{}_{ij} + |Wij|$

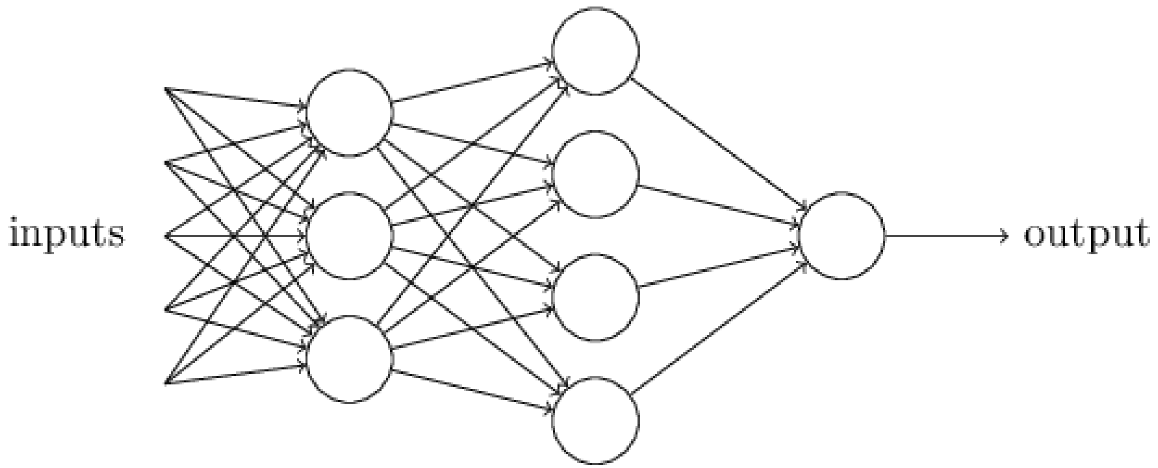


Regularization: Ensemble

Our networks typically start with random weights.

Every time we train = slightly different outcome.

- Why random weights?
- If weights are all equal, response across filters will be equivalent.
  - Network doesn't train.



$$w \cdot x \equiv \sum_j w_j x_j,$$

[Nielson]

Regularization: Ensemble

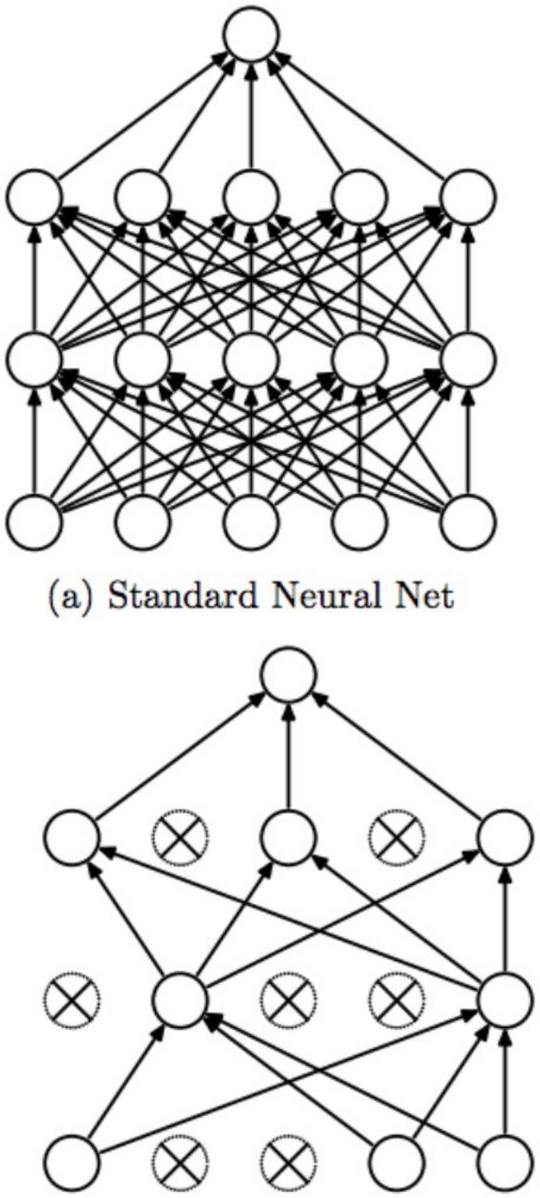
Our networks typically start with random weights.

Every time we train = slightly different outcome.

- Why not train 5 different networks with random starts and vote on their outcome?
  - Works fine!
  - Helps generalization because error due to overfitting is averaged; reduces variance.

Dropout

- Stochastically switch neurons off
  - Each neuron is set to 0 with probability p
  - Hidden units cannot co-adapt to each other
  - Units are useful independently
- Hyperparameter
  - P is usually set to 0.5



(a) Standard Neural Net

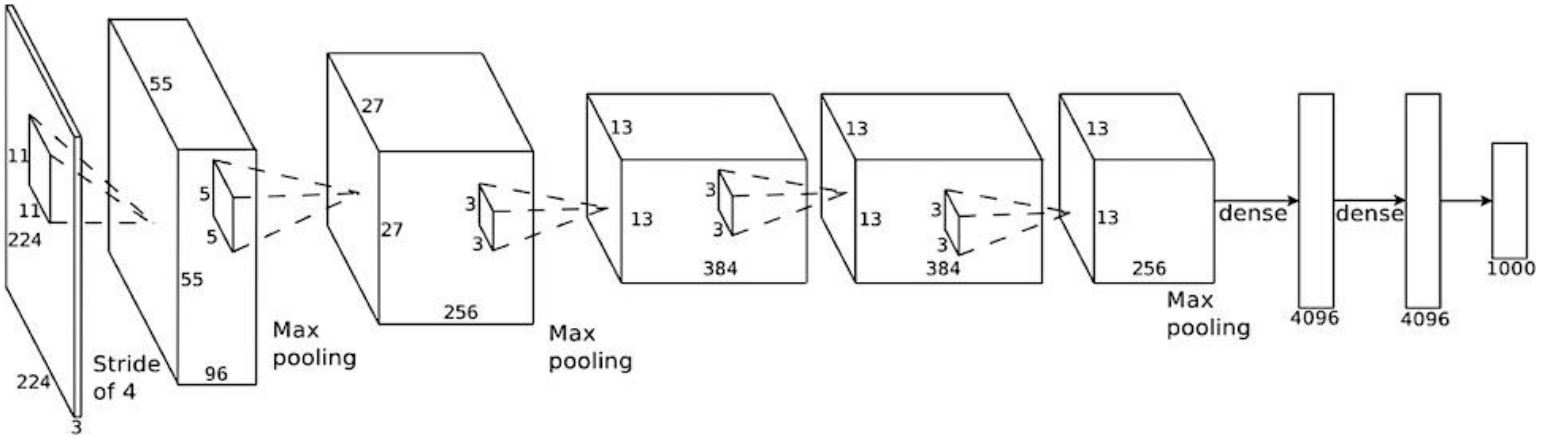
(b) After applying dropout.

Training steps

- Define network
- Loss function
- Initialize network parameters
- Get training data
  - Prepare batches
- Feedforward one batch
  - Compute loss
  - Backpropagate gradients
  - Update network parameters
  - Repeat

AlexNet - Training

- Parameters:
  - First use of ReLU
  - Dropout 0.5
  - Batch size 128
  - Optimizer SGD
  - Momentum 0.9
  - Learning rate 1e-2
  - Decay – lr reduced by 10 manually when val accuracy plateaus
  - L2 weight decay 5e-4



CAP5415
Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

Training Neural Networks

Lecture 7

CNN variants

Residual Networks

- Deep networks performs worse
  - As we add more layers
- Problem
  - Vanishing gradients
- It models
  - H(x) = F(x) + x
- Skip connections
  - Help in backpropagation

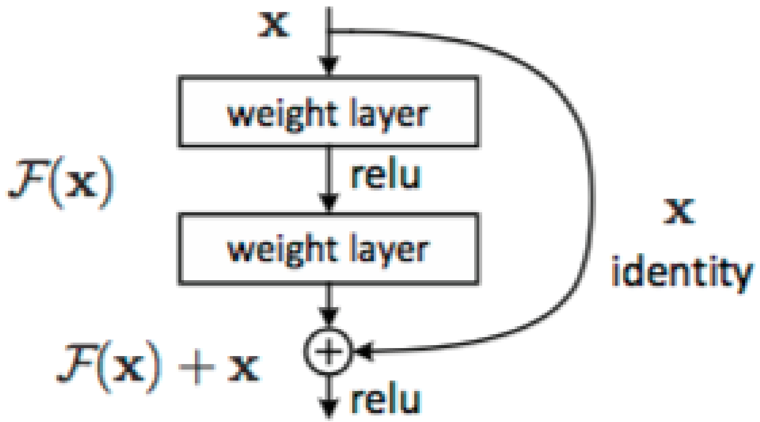
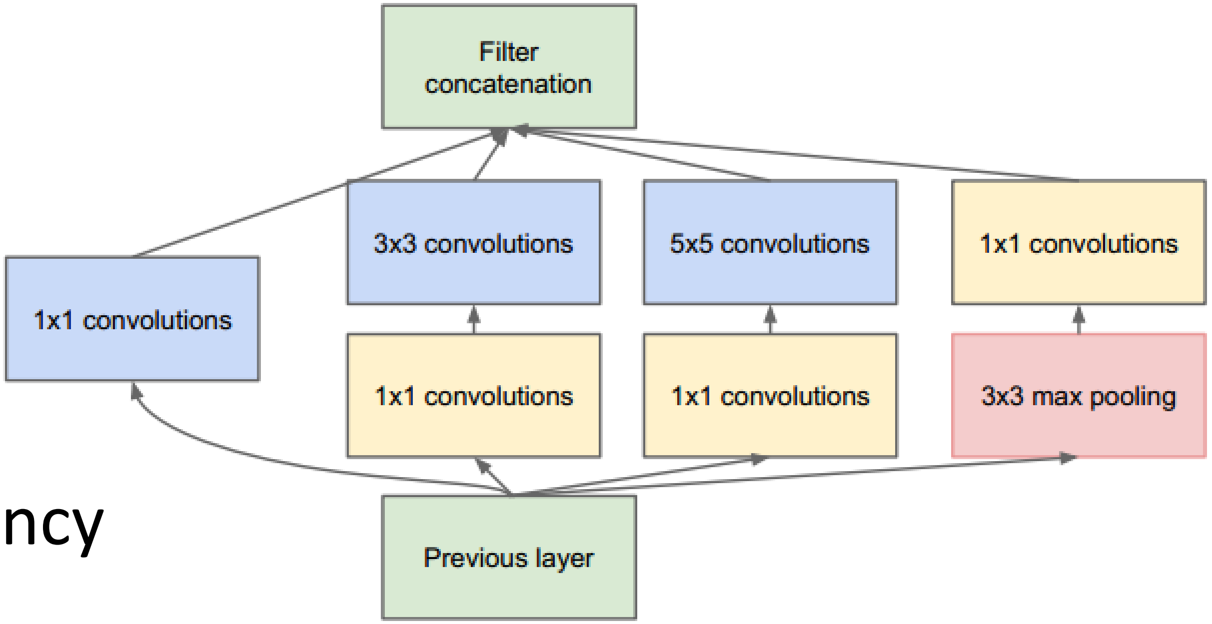


Figure 2. Residual learning: a building block.

He et. al. Deep Residual Learning for Image Recognition, 2015

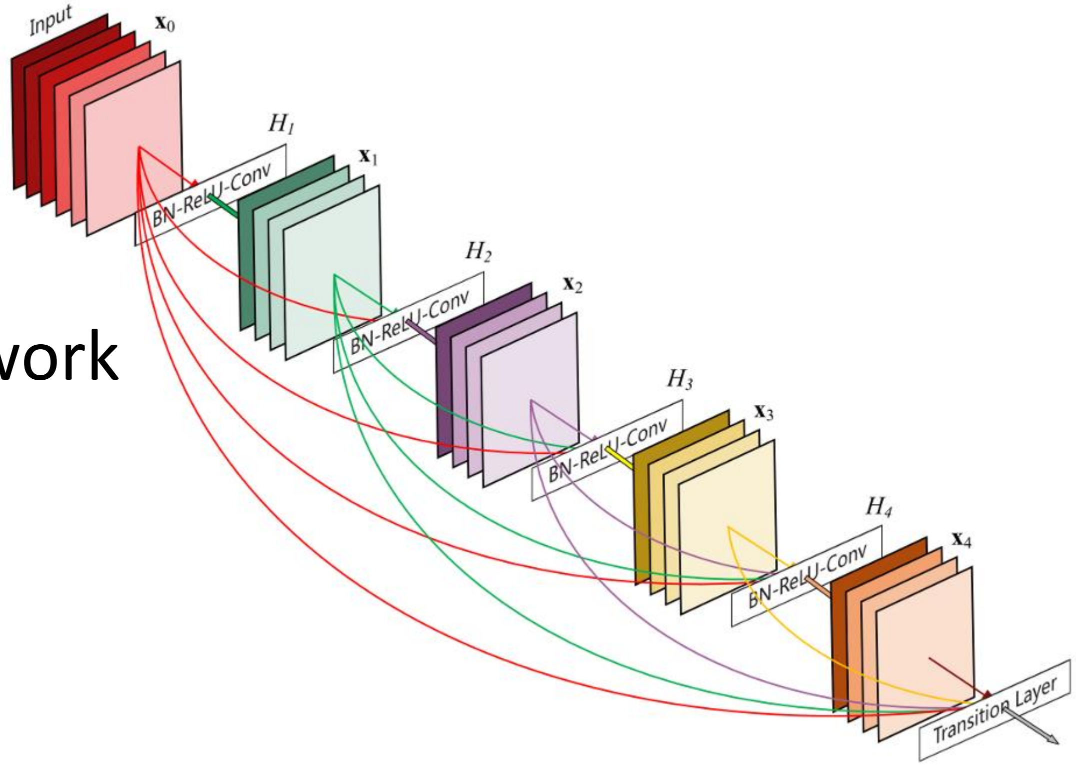
GoogleNet - Inception



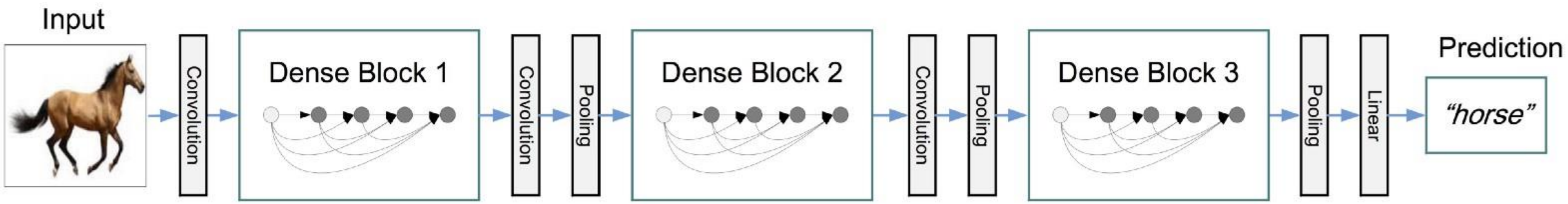
- ResNet is about going deeper
- Inception is about going wider
- Focused on computational efficiency
- The network learns
    - Which features are useful

Going deeper with convolutions, Szegedy et al. (2014)
Inception-v4, Szegedy et al. (2016)

DenseNet



- Densely Connected Convolutional Network
  - Similar to ResNet
  - Concat features instead of summation
  - A layer is passed all previous maps
  - Sequence of dense blocks



Huang, Gao, et al. "Densely connected convolutional networks." (2016).

Questions?

Sources for this lecture include materials from works by Abhijit Mahalanobis, James Tompkin, and Fei Fei Li