Deep Learning Overview

Gabriel Maher
gdmaher@stanford.edu

Institute for Computational and Mathematical Engineering,
Stanford University

ICME

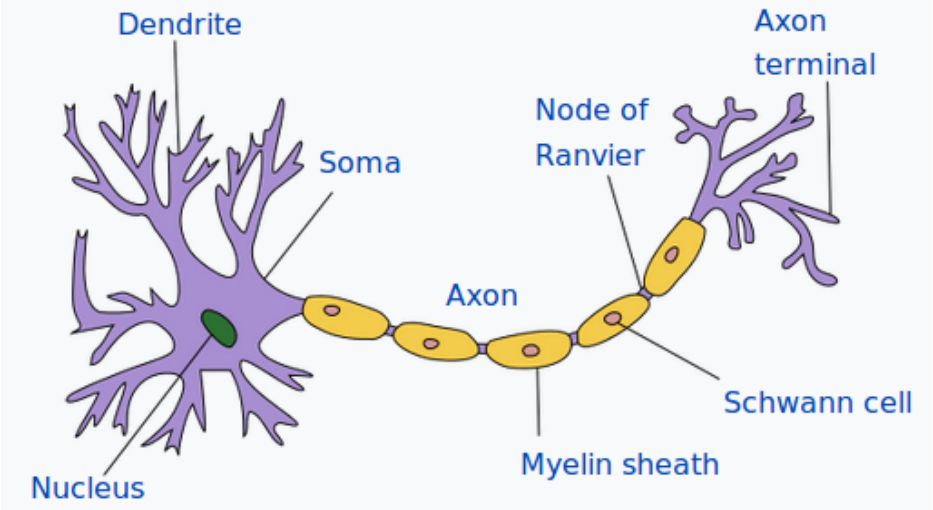
What is Deep Learning?

- Deep Learning means Machine Learning where our model is a **Neural Network**
- Can be both supervised and unsupervised
- Both classification and Regression possible

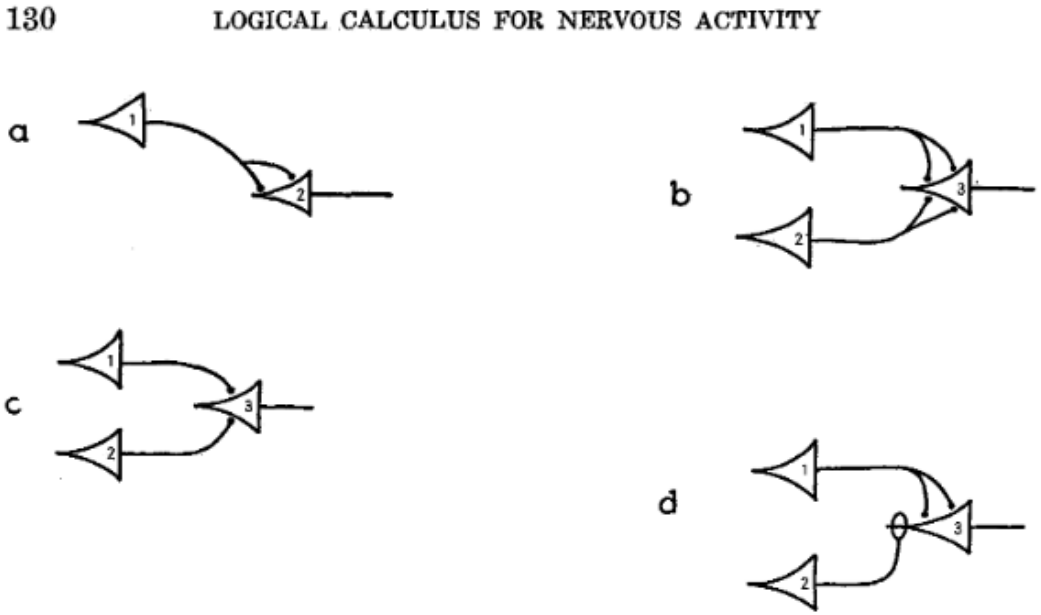
Neural Networks

- Type of Machine Learning model inspired by the brain
- **1943**: McCulloch and Pitts propose modeling the brain as temporal logic circuits
- **1958:** Rosenblatt develops the Perceptron and applies it to classification problems
- **1975:** Werbos derives the backpropagation algorithm, allowing calculation of NN gradients
- However NN development was slow until advances in GPUs and Distributed Computing allowed faster training

Neural Networks



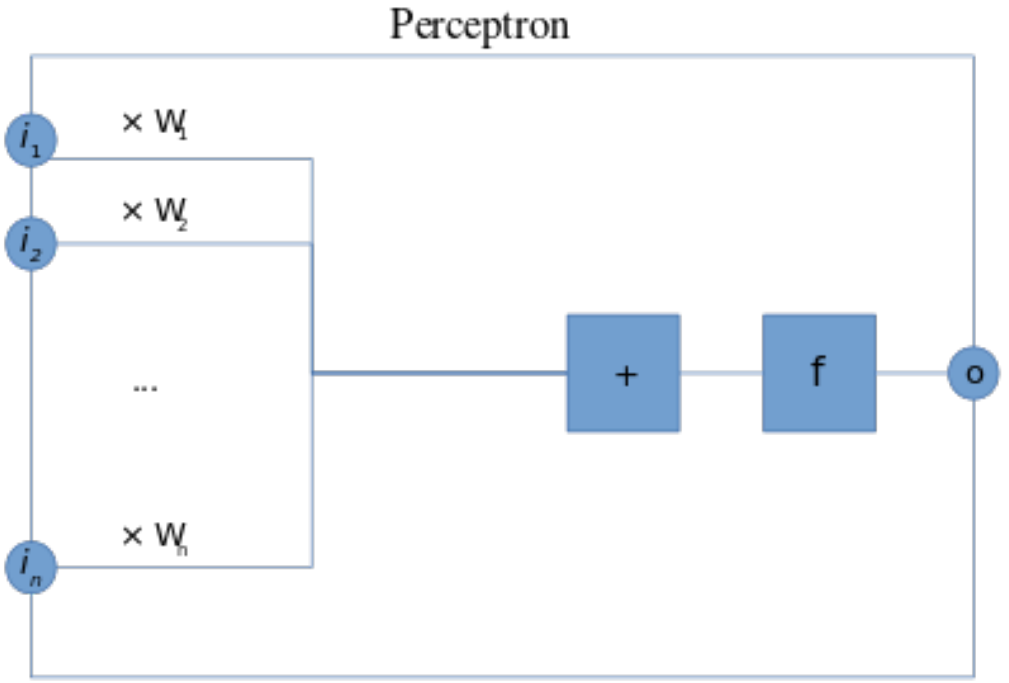
Neuron



McCulloch and Pitts (1943): Models
Of neuronal networks as logic circuits

Perceptron

- Perceptron is a one layer Neural Network
- Performs linear classification



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

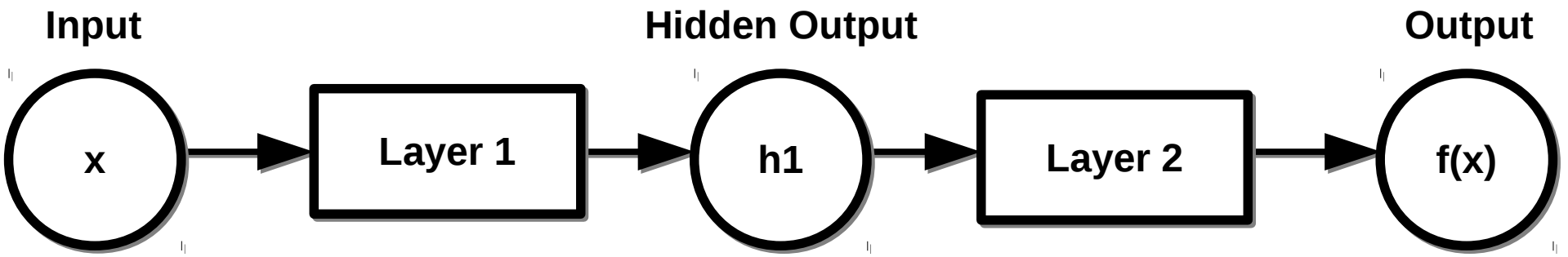
Why the Hype?

- **Universal approximation theorem**
  - Neural Network with a single hidden layer can fit any continuous function, given enough hidden units
  - Output of each layer must pass through a nonlinear function (**Activation**)
  - Does not tell us how many units needed or how to find the right values!
- **Question:** Why haven't neural networks solved all data science problems yet?

Anatomy of Neural Networks

Feed Forward Neural Networks

- Network Composed of multiple layers
- Output of a layer becomes input for the next layer
- Each layer performs an operation on the input
- No cycles or loops
- Still satisfies universal approximation theorem



ICME

Feed Forward Neural Networks

- Many different types of layers, each with different behavior
  - Fully-connected
  - Convolutional
  - Pooling
  - Recurrent
  - Dropout
  - Batch-normalizaton
  - …
- But first lets talk about **Activation Functions**

Activation Functions

- Inspired by Neurons
  - Neurons receive electric signals
  - Stores the signal via chemical reaction
  - When enough signal is stored, releases emitting signal to other neurons
- Main requirement, activation should be nonlinear
- Including activations makes universal approximation possible
- Output layer activation function determines types of output NN can produce

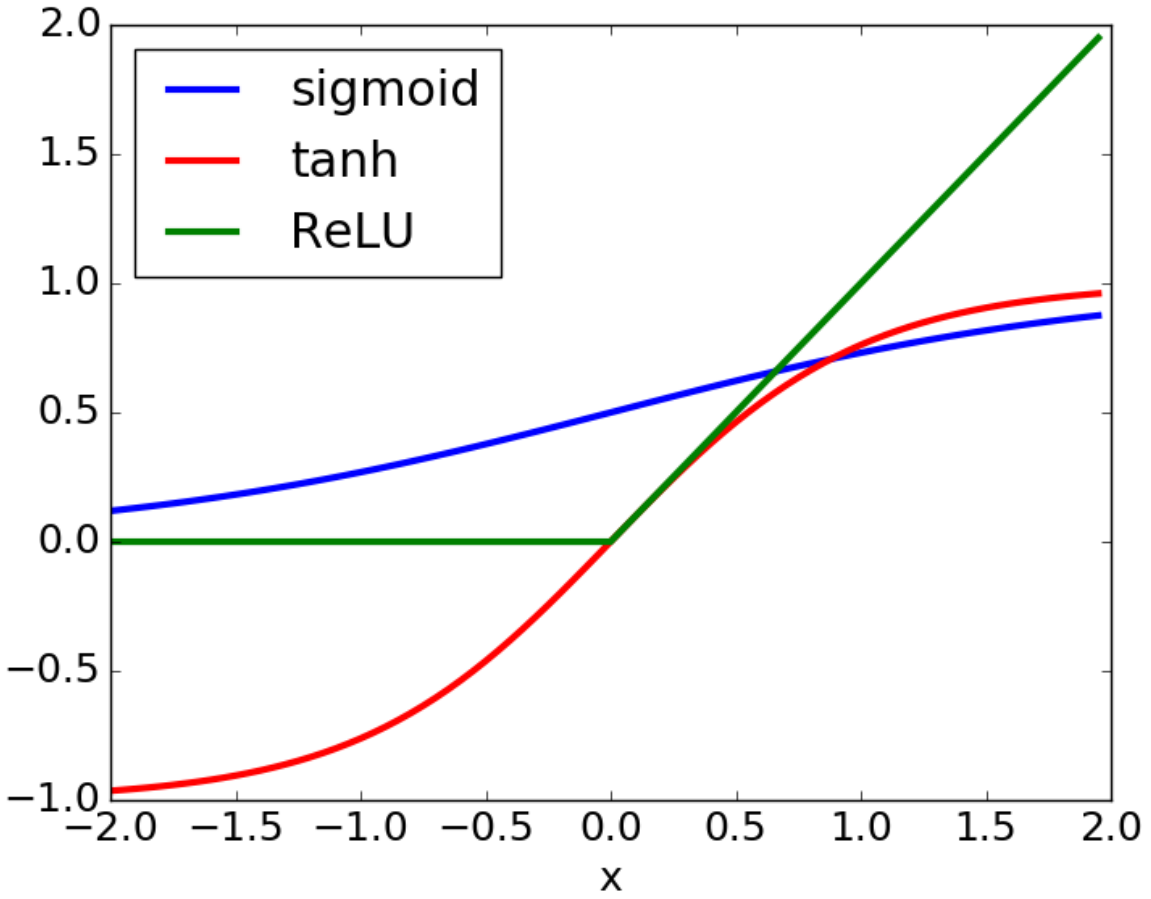
Activation Functions

Most commonly used:

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



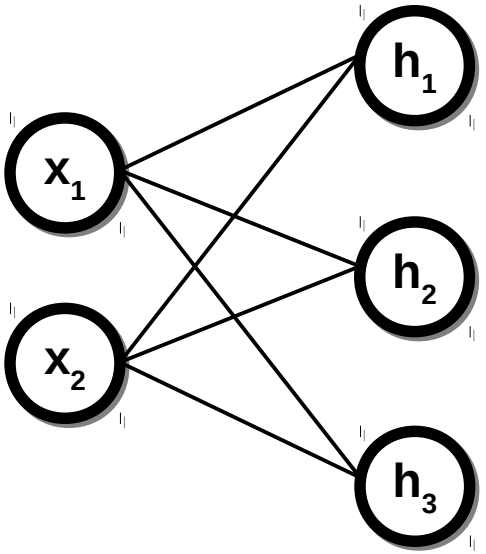
ICME

Neural Network Layers

- Fully-Connected layers
  - Applies a linear transformation to its input

$$h = Wx + b$$

  - ***W*** and ***b*** are the learned parameters
  - x size N, h size M then W has MxN parameters



$$h_1 = W_{11}x_1 + W_{12}x_2$$

$$h_2 = W_{21}x_1 + W_{22}x_2$$

$$h_3 = W_{31}x_1 + W_{32}x_2$$

Neural Network Layers

- Convolutional Layers
  - for processing images, data with spatial structure
  - Convolves input with filter of learnable weights

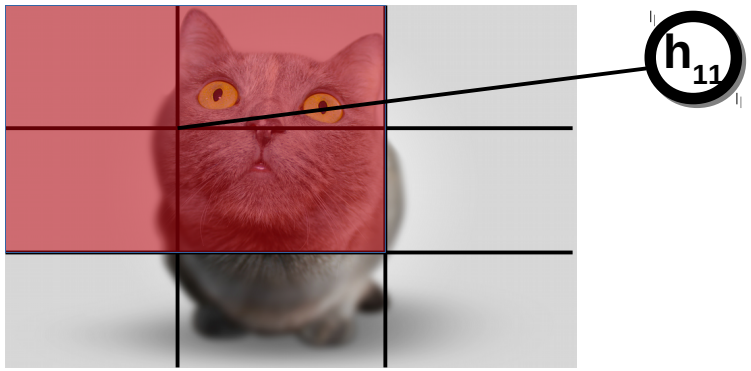
$$h_{ij} = \sum_{l,m=1}^{H} \sum_{c=1}^{C} x_{i+l-1\, j+m-1\, c} W_{lmc} + b_c$$

  - ***W*** and ***b*** are the learnable parameters
  - ***W*** has size HxHxC, **b** has size C
  - C is the number of channels (e.g. RGB image, C=3)

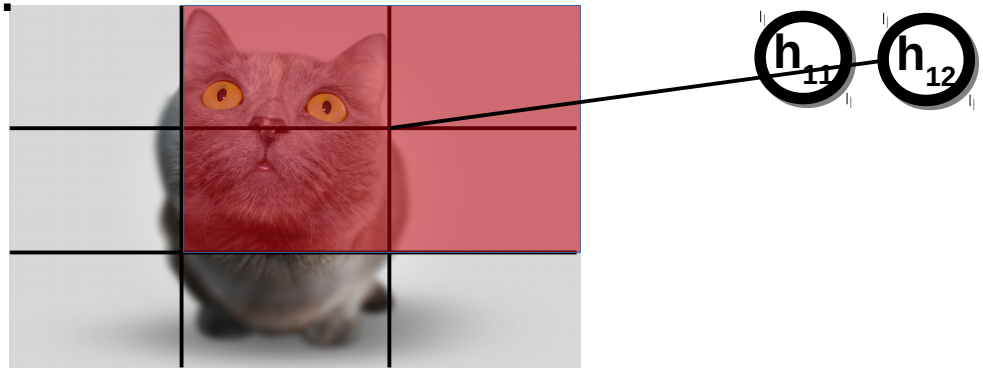
Neural Network Layers

- Convolutional Layers

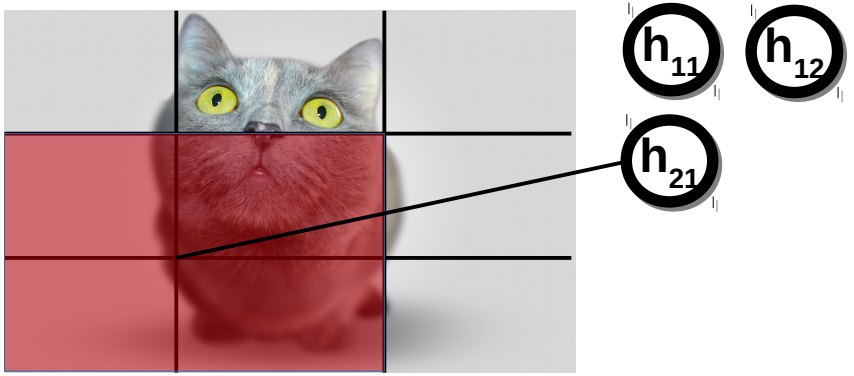
1.



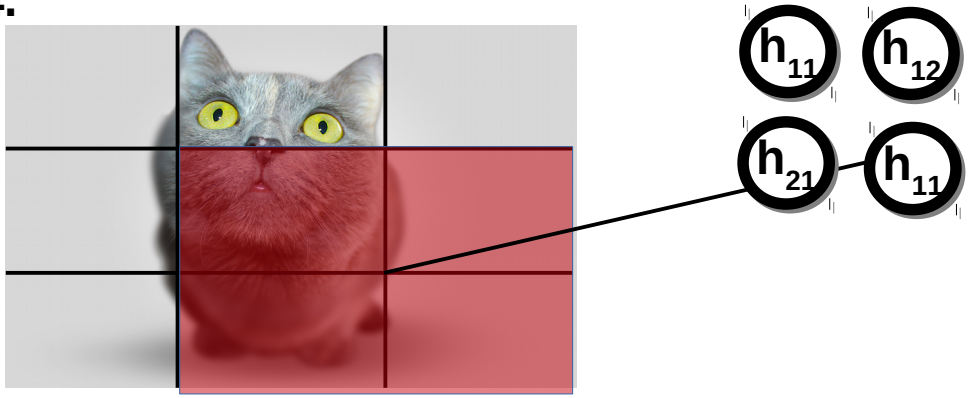
2.



3.



4.

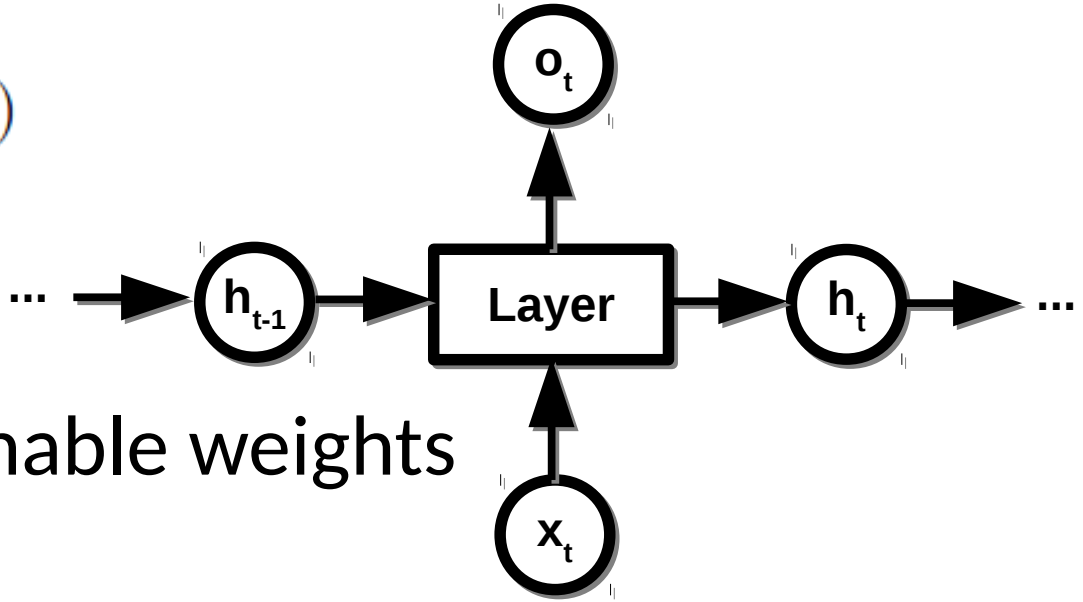


Neural Network Layers

- Recurrent Layers
  - Used for sequential data, e.g. text, time-series
  - Use same weights across each timestep

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = \sigma(Vh_t + b_o)$$

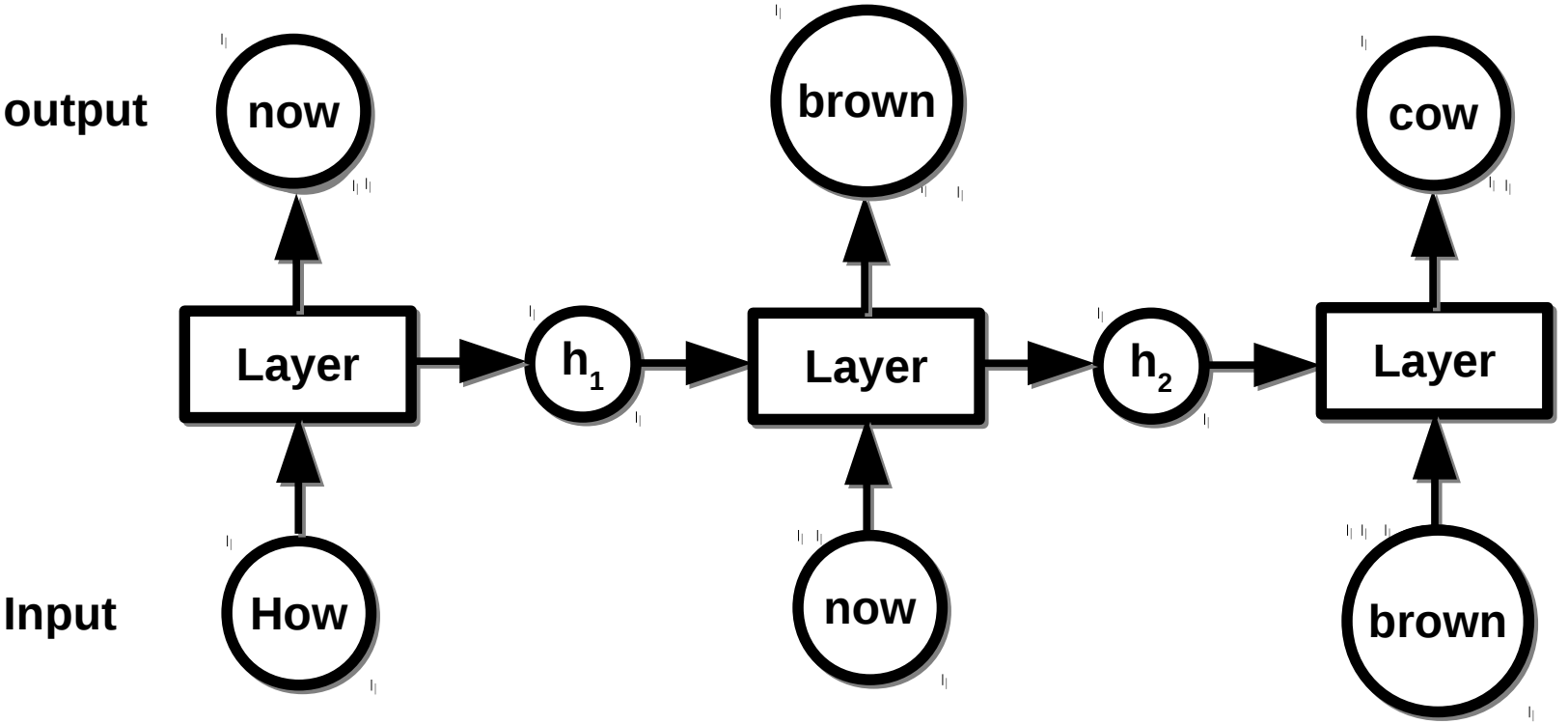


- **W,U,V,b_h,b_o** are the learnable weights
- Other recurrent layers
  - Long short-term memory
  - Gated recurrent unit

ICME

Neural Network Layers

- Recurrent layers



How Do We Train Neural Networks?

Training Neural Networks

- Neural Networks typically used for supervised learning
- **Supervised Learning:**
  - Have dataset of inputs $x_1,\ldots,x_N$ and outputs $y_1,\ldots,y_N$
  - Given input **x** we want our neural network to predict **y**
  - Find neural network weights such that $f(x_i)\approx y_i$ for all data points

ICME

Training Neural Networks

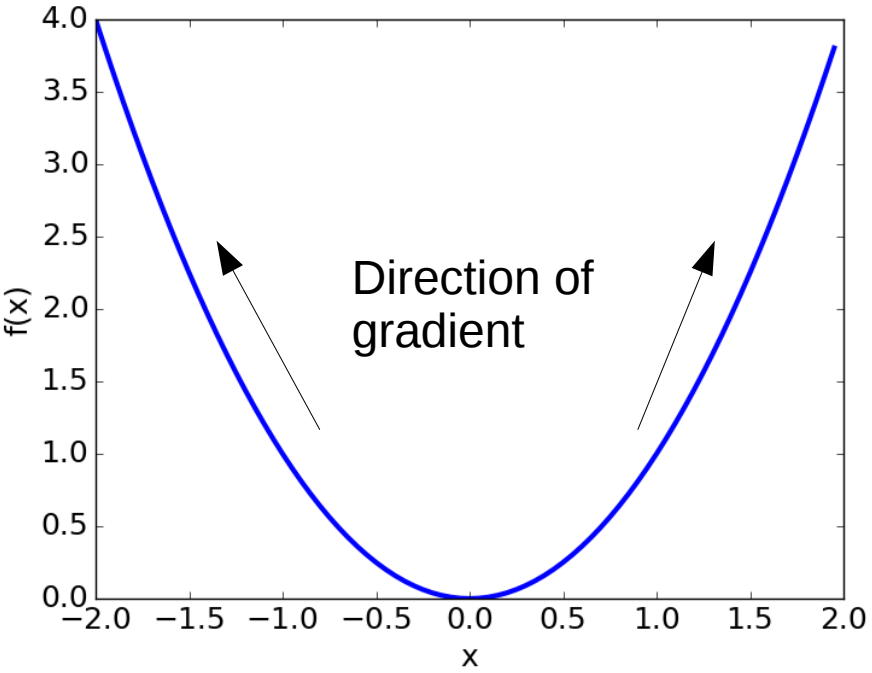
- How do we find the weights?
  - Gradient Descent algorithm, algorithm for finding minimum of a function
- Function gradient points in direction of increase
- Take small steps in direction of **negative** gradient

Iterate:

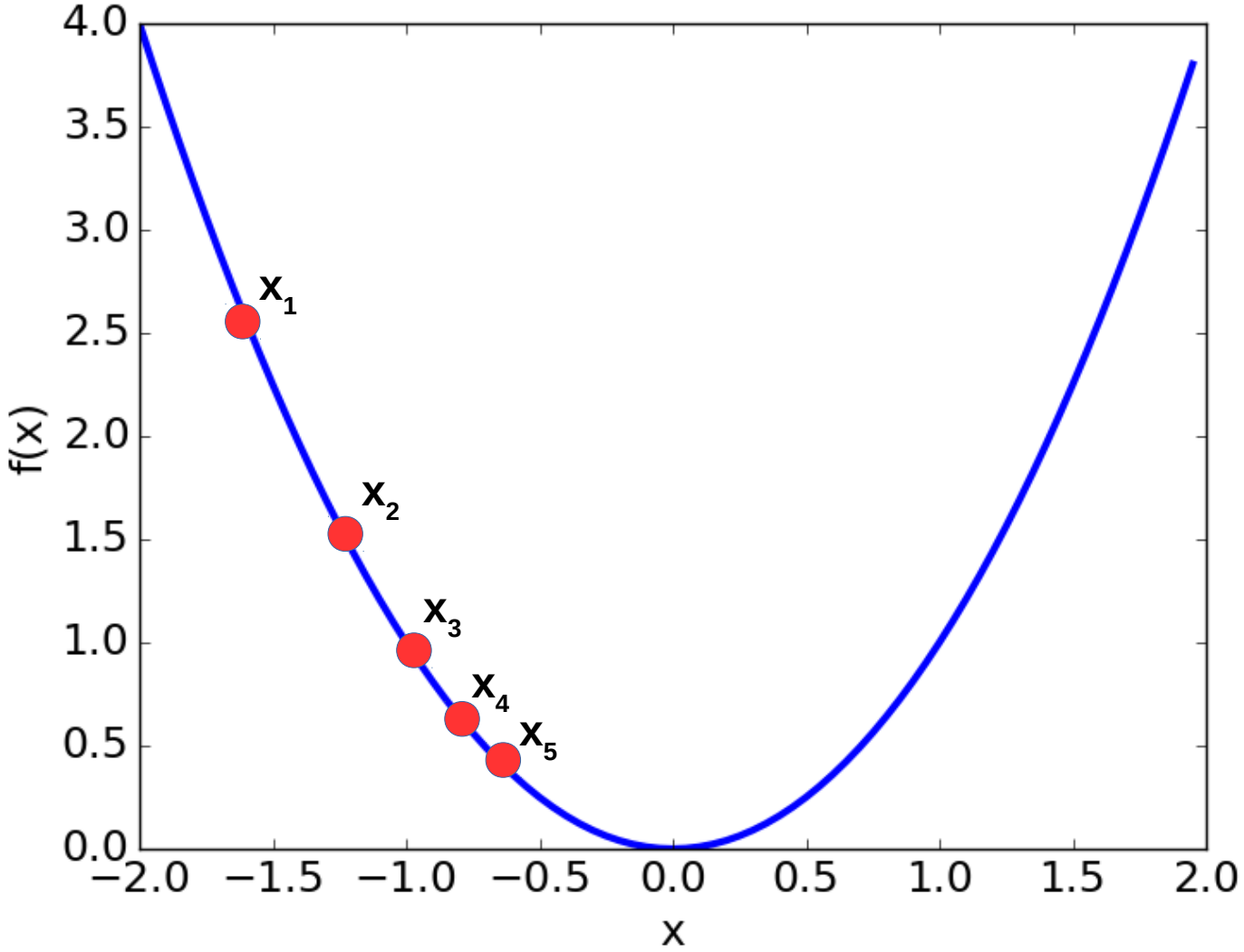
for t = 1,…

$$x_t = x_{t-1} - \alpha \nabla f(x_{t-1})$$

- α = learning rate, typically<1



Gradient Descent



ICME

Training Neural Networks

- How do we use gradient descent to find Neural Network weights?
  - Specify cost function that measures prediction error
  - Treat the network weights **W** as **x** in gradient descent
    - Taking steps in weight space!
  - Requires calculating the gradient of the cost function, with respect to **W**

Training Neural Networks

- Cost functions:
  - f(x) is neural network prediction
  - Regression:
    - L_2/Mean-Squared error $$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$
  - Classification:
    - Cross-entropy (2 classes, y=0 or 1)
    $$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - 1)\log(1 - f(x_i)) - y_i\log(f(x_i))$$
    - And many more: categorical cross-entropy, hinge-loss, …

ICME

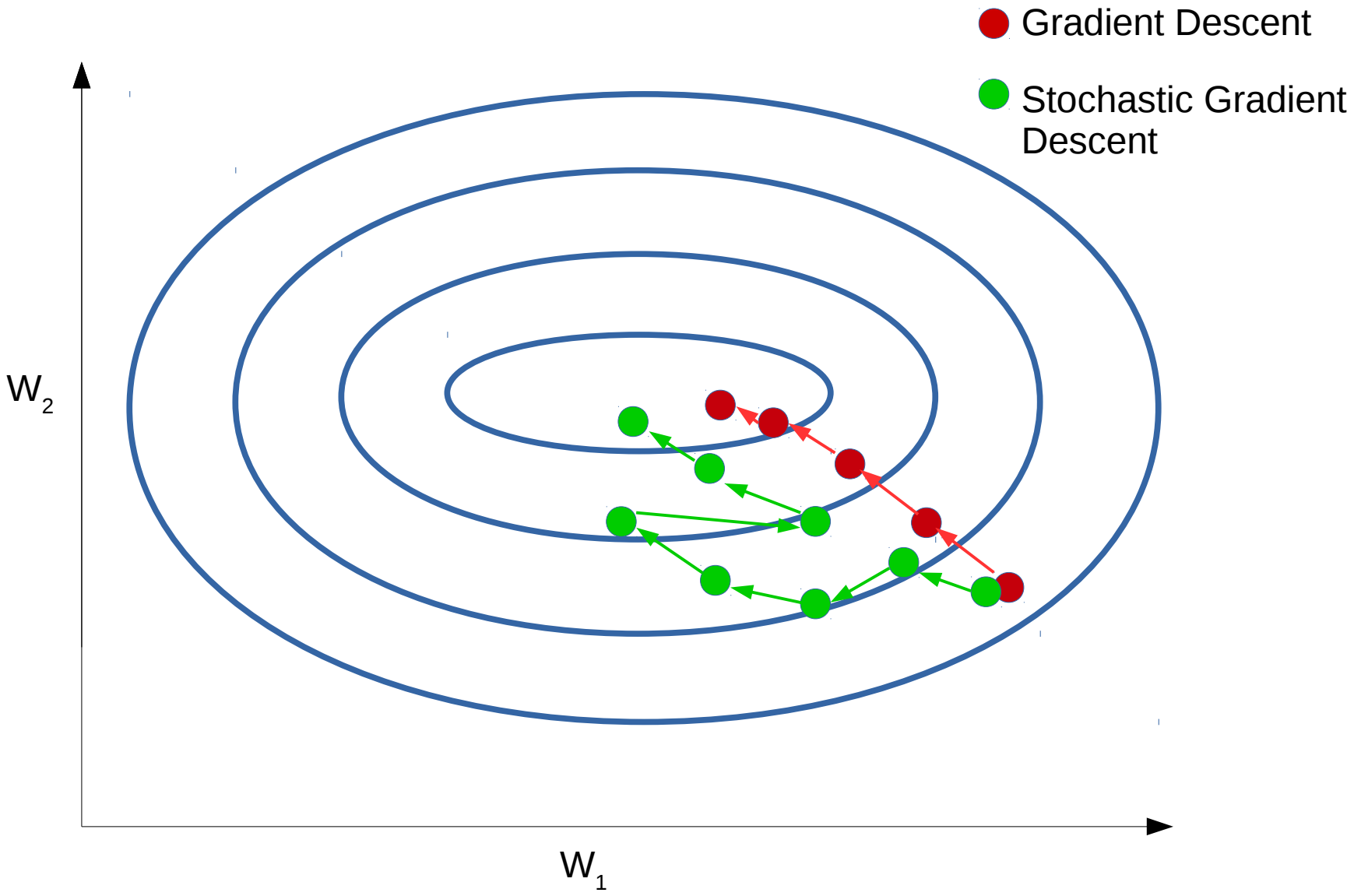
Stochastic Gradient Descent

- Gradient Descent Update for Neural Networks:

$$W_t = W_{t-1} - \alpha \nabla_W L(W_{t-1})$$

  - Every time we update **W**, we have to recompute the gradient
  - Large network, lots of data = very expensive!
- **Stochastic Gradient Descent**
  - Sample a small batch of data points randomly
  - Calculate gradient only using sampled points

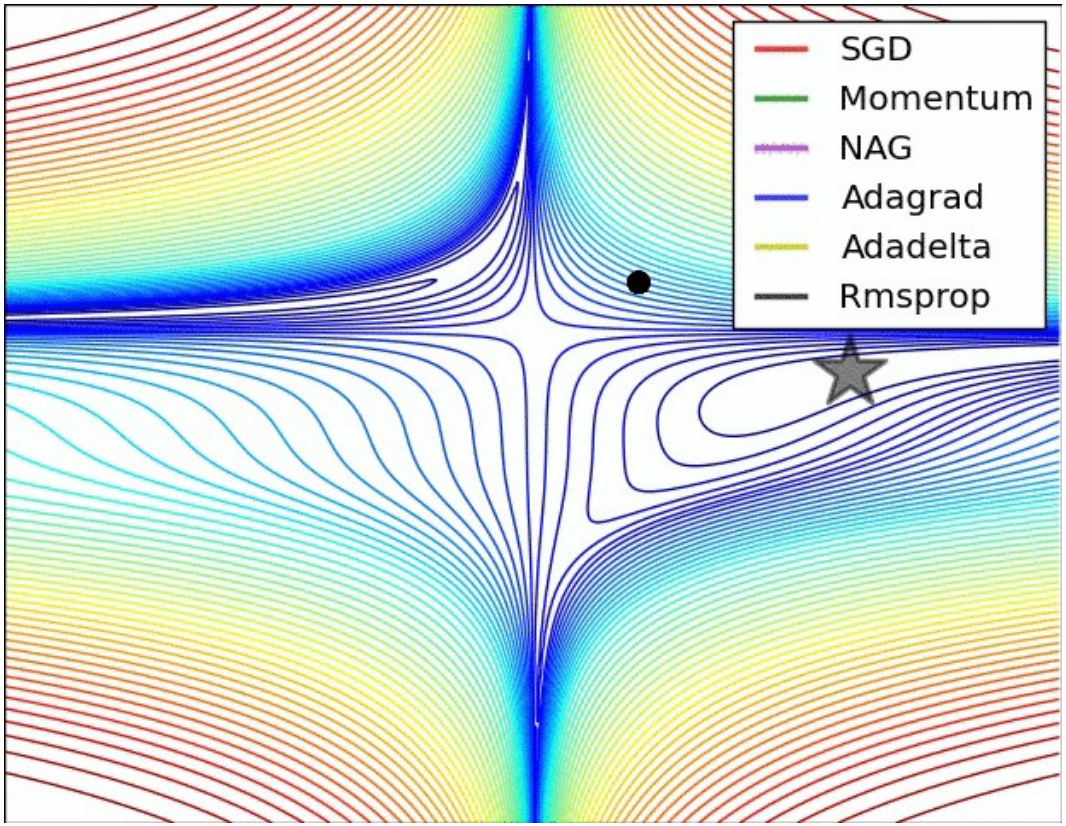
Stochastic Gradient Descent



ICME

Stochastic Gradient Descent

- Many variants of SGD have been proposed to improve training speed
  - Momentum SGD
  - Nesterov SGD
  - RMSProp
  - ADAGrad
  - ADAM
  - …



Taken from CS231N at Stanford

Training Neural Networks

- How do we calculate the gradient of the cost function with respect to the network weights?
- Chain Rule!

Layer 2 derivative:

$$\frac{\partial}{\partial W_{ij}^{(2)}} L(y, f(x)) = \frac{\partial L}{\partial f} \frac{\partial f}{\partial W_{ij}^{(2)}}$$

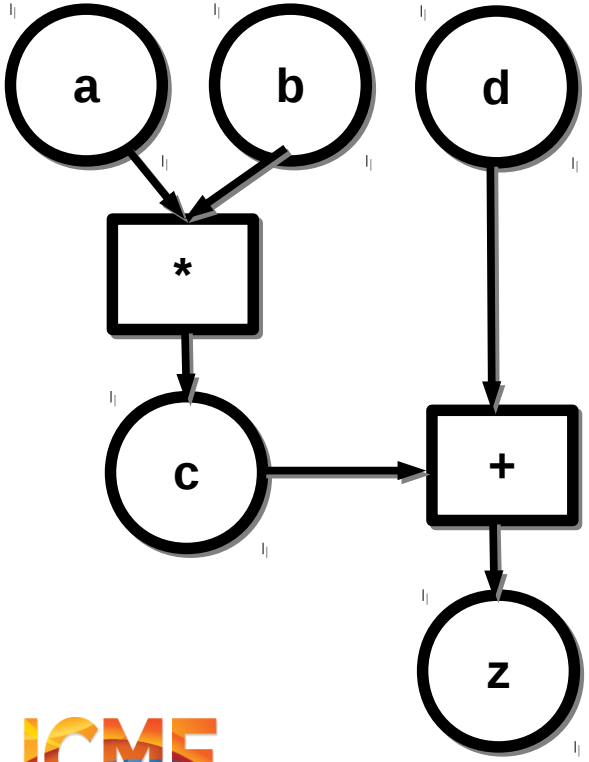
Layer 1 derivative:

$$\frac{\partial}{\partial W_{ij}^{(1)}} L(y, f(x)) = \frac{\partial L}{\partial f} \frac{\partial f}{\partial a_1} \frac{\partial a_1}{\partial h_1} \frac{\partial h_1}{\partial W_{ij}^{(1)}}$$

ICME

Backpropagation

- Applying the chain rule directly is cumbersome
- **Backpropagation:** method for recursively calculating the gradient of computational graphs



$$\frac{\partial z}{\partial a} = \frac{\partial z}{\partial c}\frac{\partial c}{\partial a} = d\frac{\partial c}{\partial a} = db$$

For our Neural Network:

$$\frac{\partial}{\partial W_{ij}^{(1)}} L(y, f(x)) = \delta_2 \frac{\partial h_1}{\partial W_{ij}^{(1)}}$$

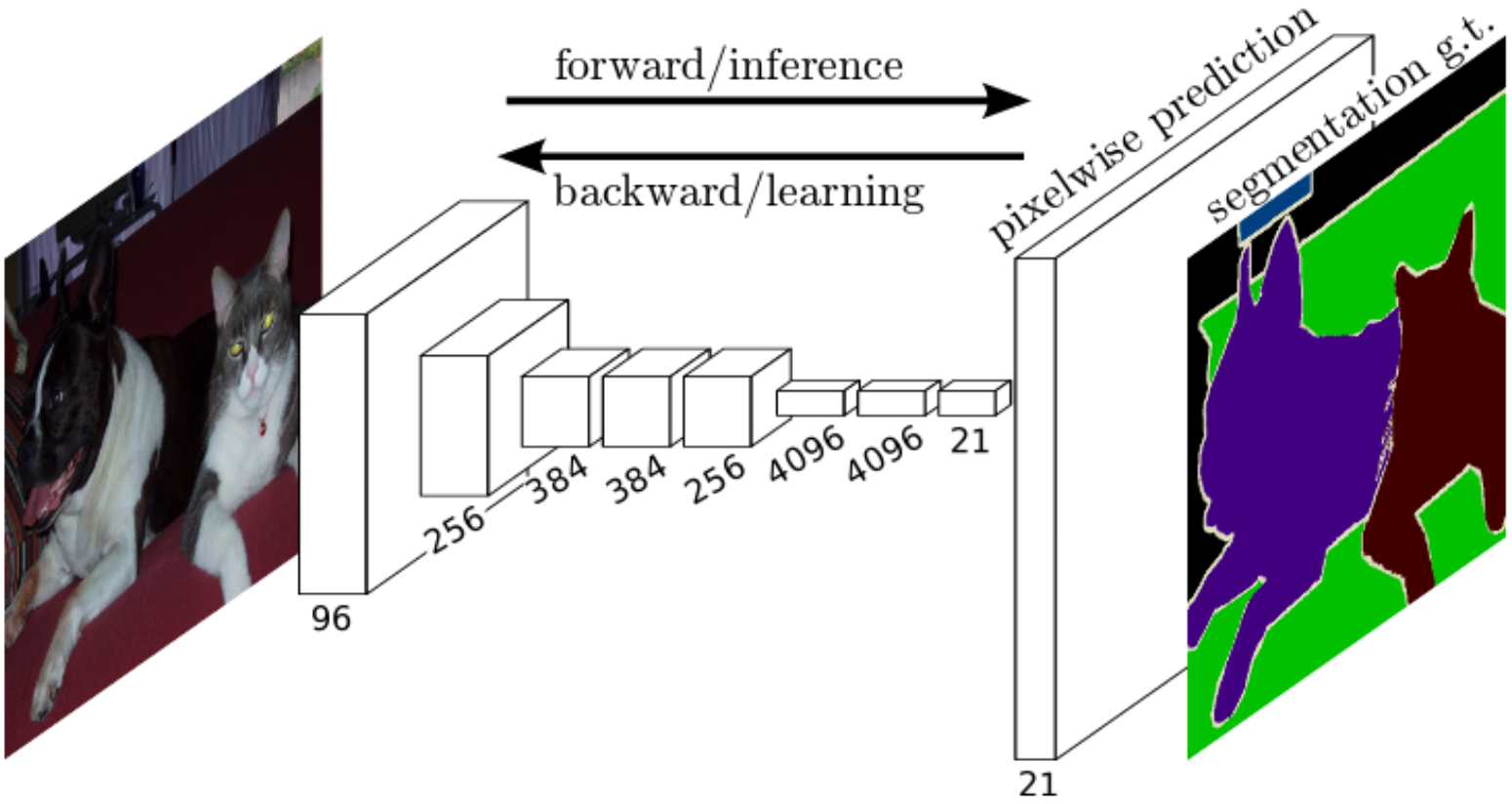
δ_i is the upstream derivative

ICME

Applications

Applications – Semantic Segmentation

- Classify all pixels in an image into one of several classes

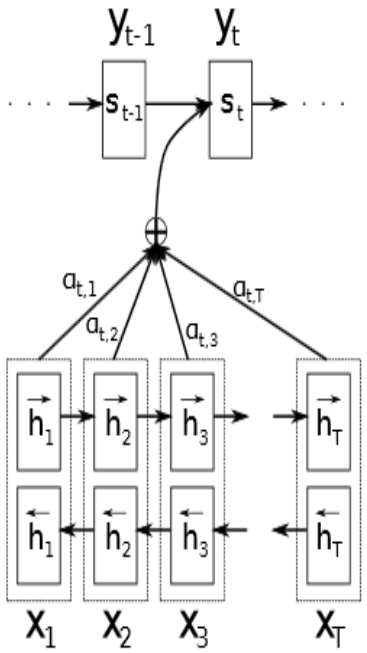


Fully Convolutional Networks for Semantic Segmentation
Long, Shelhamer and Darrell, 2014

ICME

Applications – Machine Translation

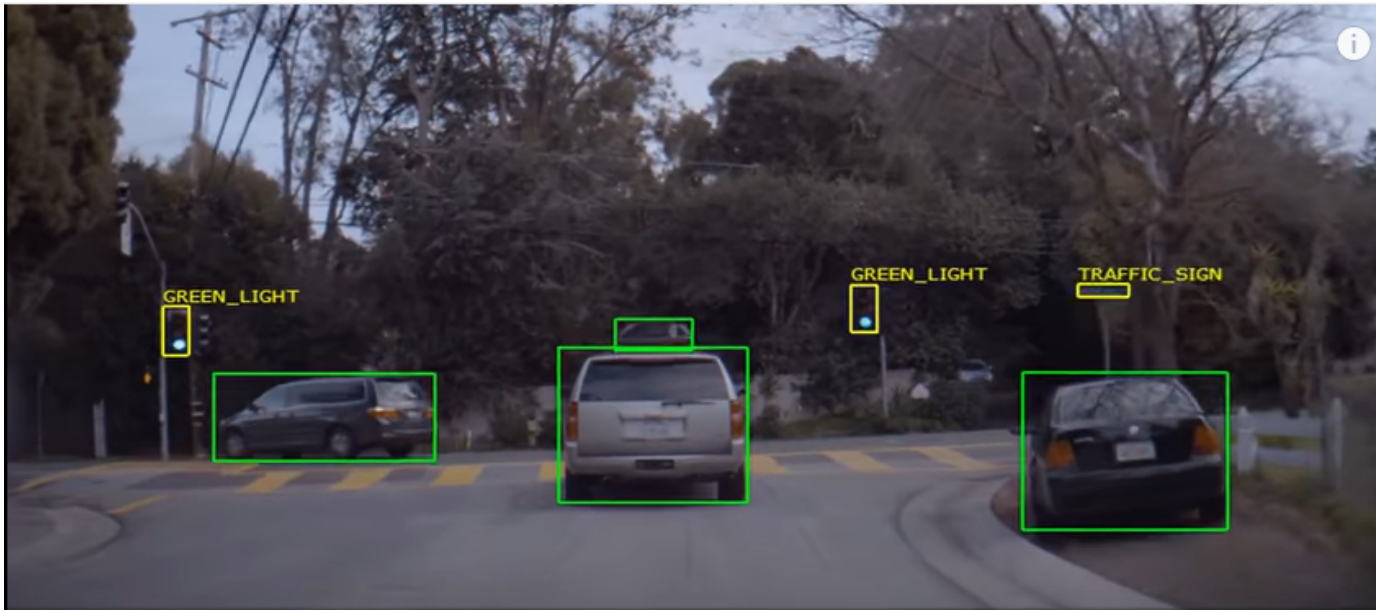
- Language data inherently sequential
- Recurrent Neural Networks used extensively



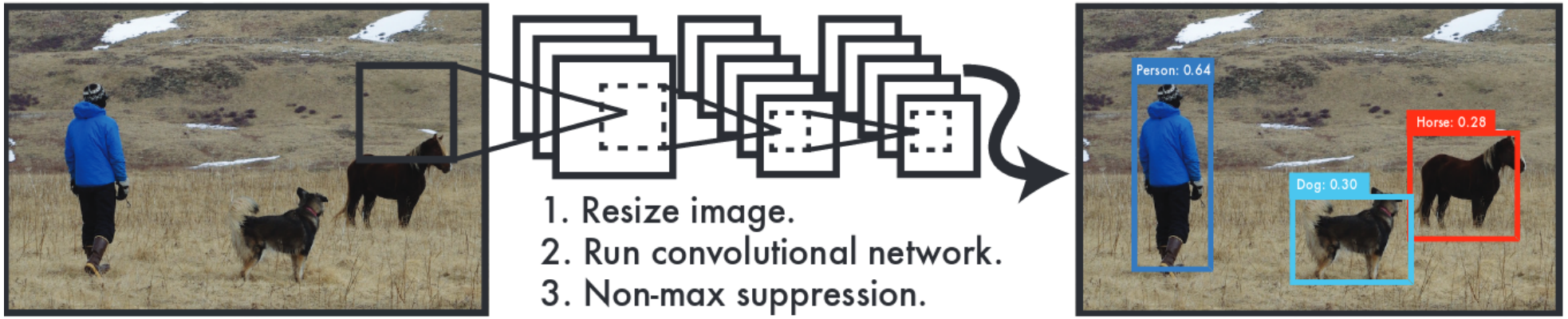
Source	An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.
RNNsearch-50	Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.

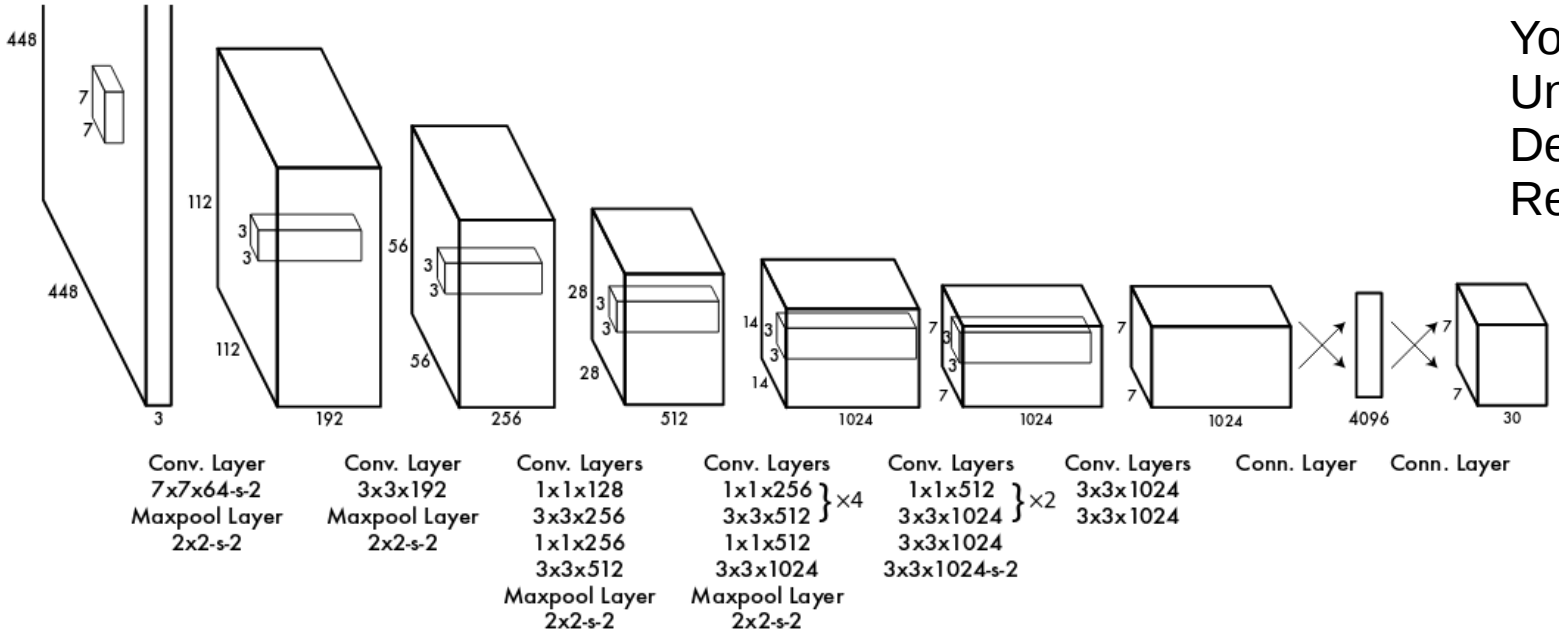
Source: D. Bahdanau et al, ICLR 2015

Applications – Autonomous Cars



Applications – Bounding Box Detection



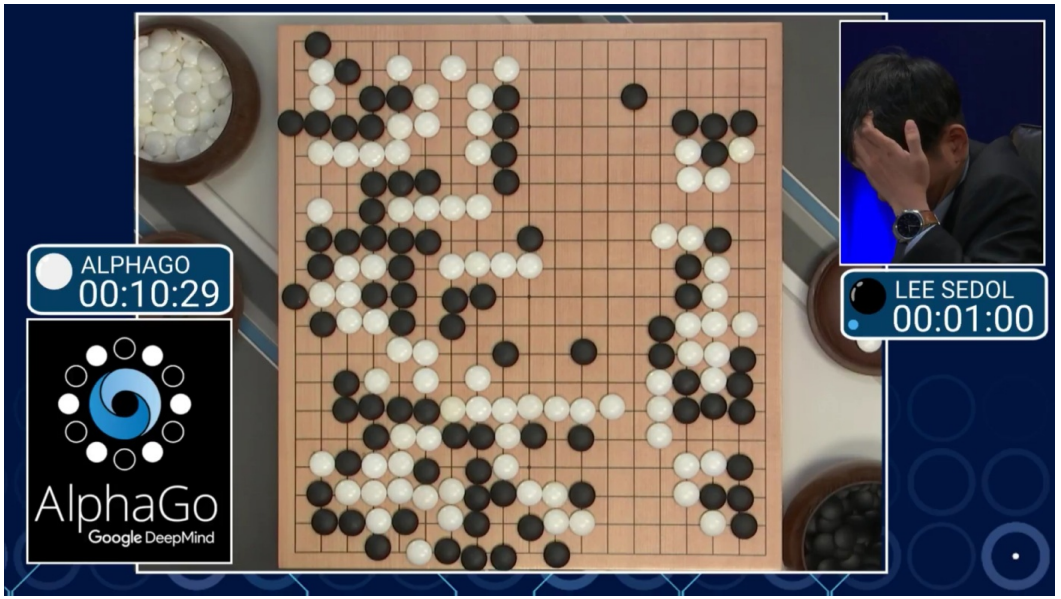


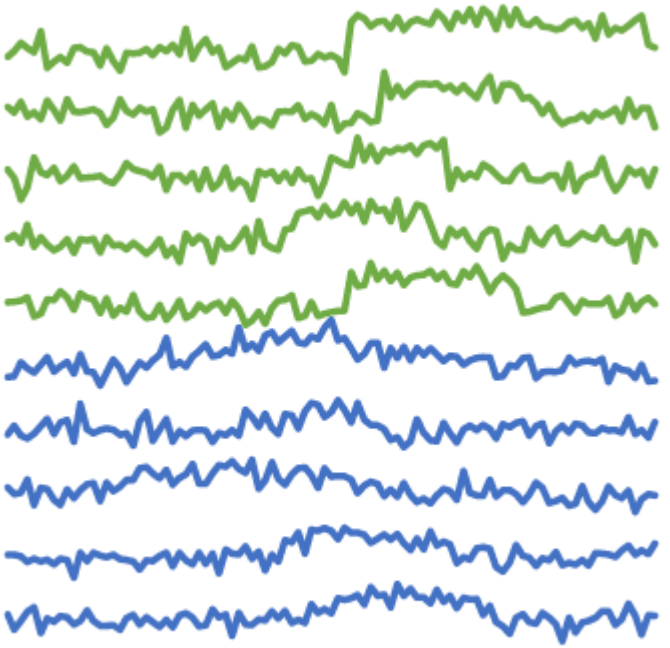
You Only Look Once:
Unified Real-Time Object Detection
Redmon et al.

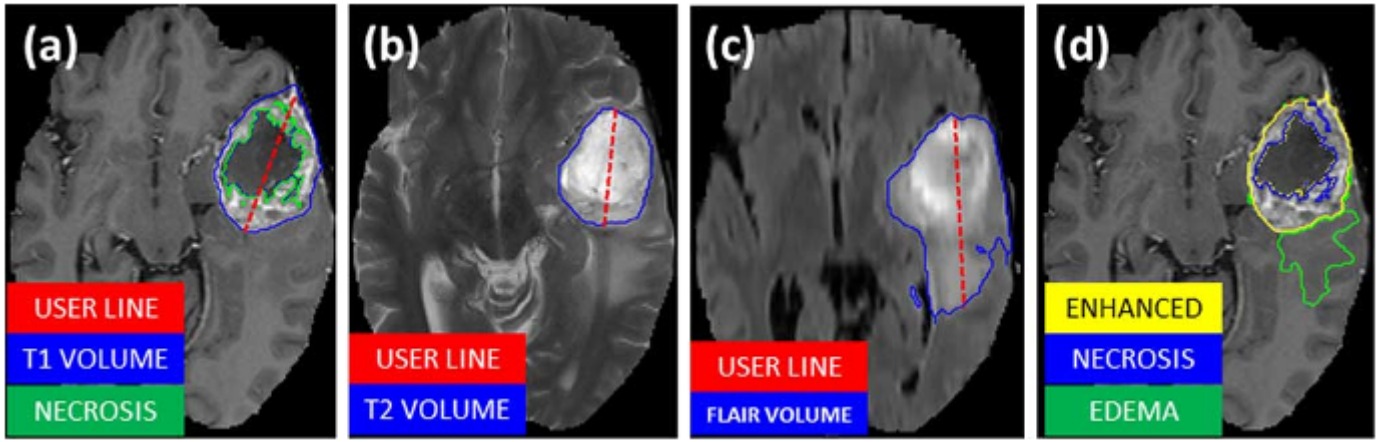
ICME

Applications – Bounding Box Detection

And many more!







ICME

Software Libraries

- Implement yourself as practice, then use a library
- Python: Tensorflow, Theano, Caffe, MxNet (C++ too)
- Lua: Torch
- Many more coming out frequently

References

- Many excellent resources online
  - www.deeplearningbook.org
  - http://cs231n.stanford.edu/
- However most deep learning still active research
  - Many techniques only in conference/journal papers

- **I'm writing a book**
  - "Practical Deep Learning with Tensorflow"
  - Quick and rigorous introduction to deep learning with examples in Tensorflow, aimed at industry/professionals
  - Help me by being a reviewer!
    (email me: gdmaher@stanford.edu)