ECE 5997
Hardware Accelerator Design & Automation
Fall 2021

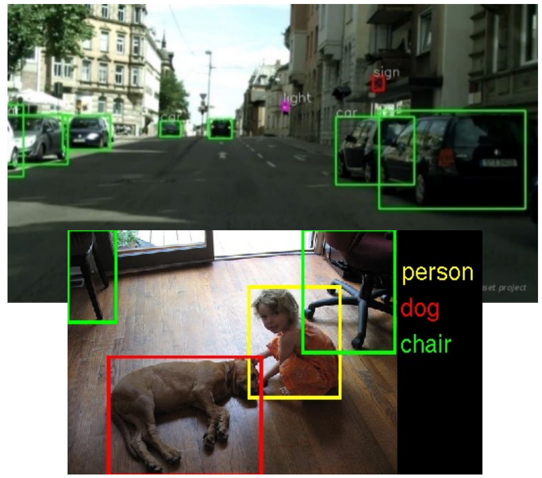
Introduction to Neural Networks

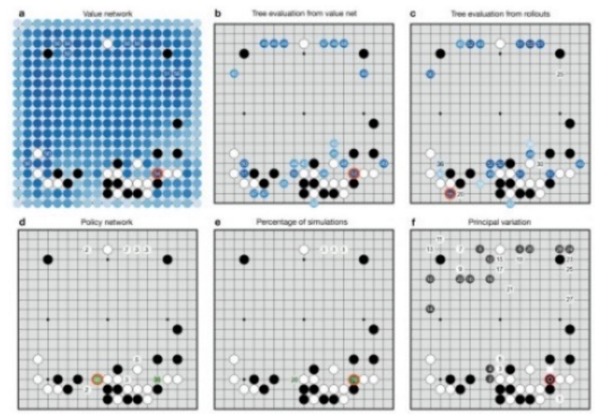
Yichi Zhang, Ritchie Zhao, Zhiru Zhang
School of Electrical and Computer Engineering

Cornell University

CSL

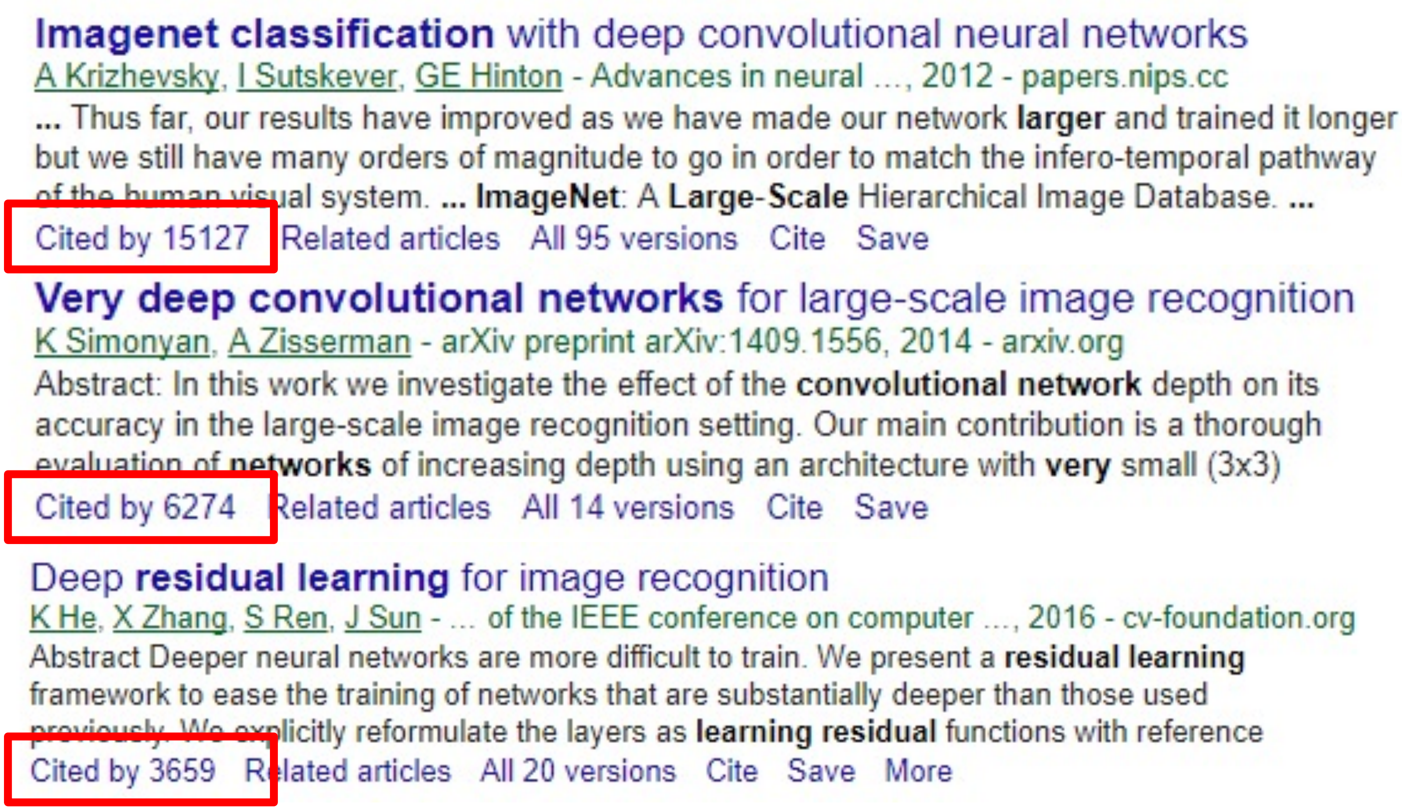
Rise of the Machines





- Neural networks have revolutionized the world
  - Self-driving vehicles
  - Advanced image recognition
  - Game AI

Rise of the Machines



Imagenet classification with deep convolutional neural networks
A Krizhevsky, I Sutskever, GE Hinton - Advances in neural ..., 2012 - papers.nips.cc
... Thus far, our results have improved as we have made our network **larger** and trained it longer but we still have many orders of magnitude to go in order to match the infero-temporal pathway of the human visual system. ... **ImageNet**: A **Large-Scale** Hierarchical Image Database. ...
Cited by 15127 Related articles All 95 versions Cite Save

Very deep convolutional networks for large-scale image recognition
K Simonyan, A Zisserman - arXiv preprint arXiv:1409.1556, 2014 - arxiv.org
Abstract: In this work we investigate the effect of the **convolutional network** depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of **networks** of increasing depth using an architecture with **very** small (3x3)
Cited by 6274 Related articles All 14 versions Cite Save

Deep **residual learning** for image recognition
K He, X Zhang, S Ren, J Sun - ... of the IEEE conference on computer ..., 2016 - cv-foundation.org
Abstract Deeper neural networks are more difficult to train. We present a **residual learning** framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as **learning residual** functions with reference
Cited by 3659 Related articles All 20 versions Cite Save More

- Neural networks have revolutionized research

A Brief History

- **1950's:** First artificial neural networks created based on biological structures in the human visual cortex
- **1980's – 2000's:** NNs considered inferior to other, simpler algorithms (e.g. SVM, logistic regression)
- **Mid 2000's**: NN research considered "dead", machine learning conferences outright reject most NN papers
- **2010 – 2012**: NNs begin winning large-scale image and document classification contests, beating other methods
- **2012 – Now**: NNs prove themselves in many industrial applications (web search, translation, image analysis)

Things Neural Networks Can Do…

- **Surpass human ability in image recognition**

Things Neural Networks Can Do…

- **Generate celebrities**

Things Neural Networks Can Do…

- **Art style transfer**

https://deepart.io/

Things Neural Networks Can Do…

- **Beat humans at DOTA 2** (>6.5k MMR)

https://openai.com/five/

Part 1

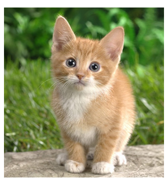
CLASSIFICATION WITH THE PERCEPTRON

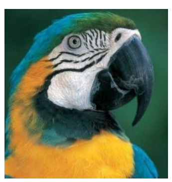
Classification Problems

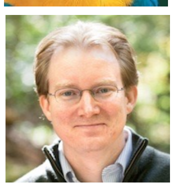
- We'll discuss neural networks for solving **supervised classification problems**

  - Given a training set consisting of labeled inputs
  - Learn a function which maps inputs to labels
  - This function is used to predict the labels of new inputs

Inputs | **Labels**


→ Cat


→ Bird


→ Human

Training Set

Inputs | **Predictions**


→ ???

Predict New Data

A Simple Classification Problem

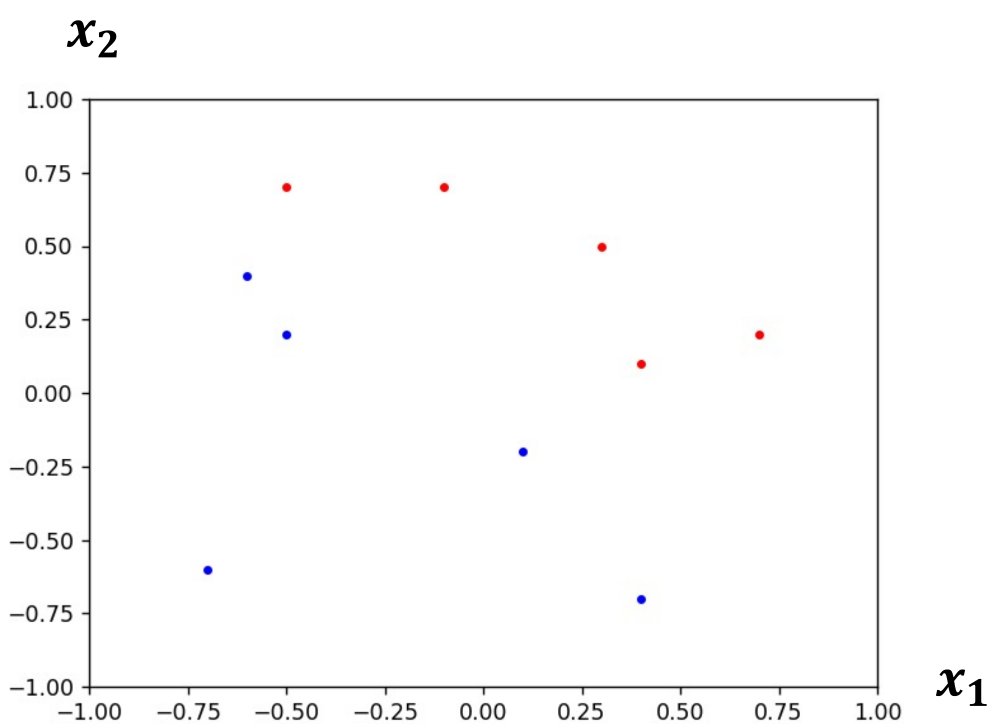
- **Inputs**: Pairs of numbers (x_1, x_2)
- **Labels**: 0 or 1
  - binary decision problem

- Real-life analogue:
  - Label = Raining or Not Raining
  - x_1 = relative humidity
  - x_2 = cloud coverage

x_1	x_2	Label
-0.7	-0.6	0
-0.6	0.4	0
-0.5	0.7	1
-0.5	-0.2	0
-0.1	0.7	1
0.1	-0.2	0
0.3	0.5	1
0.4	0.1	1
0.4	-0.7	0
0.7	0.2	1

Training Set

Visualizing the Data



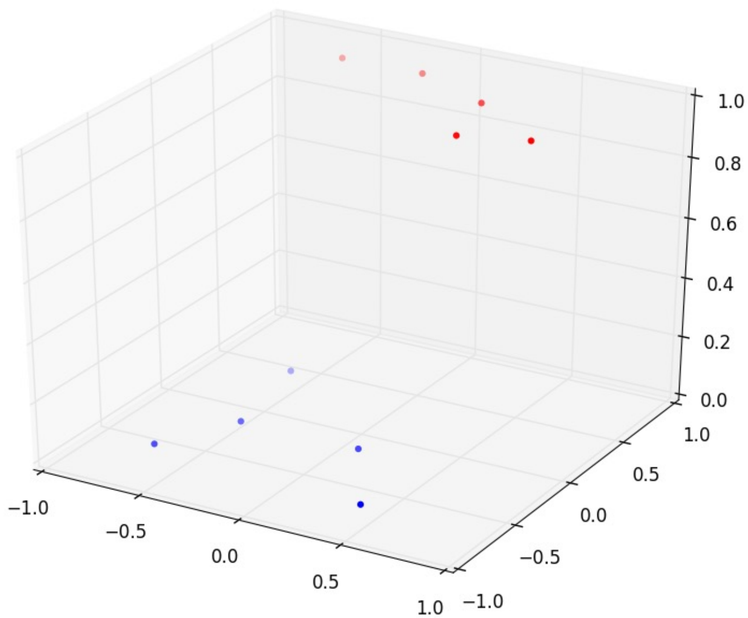
Plot of the data points

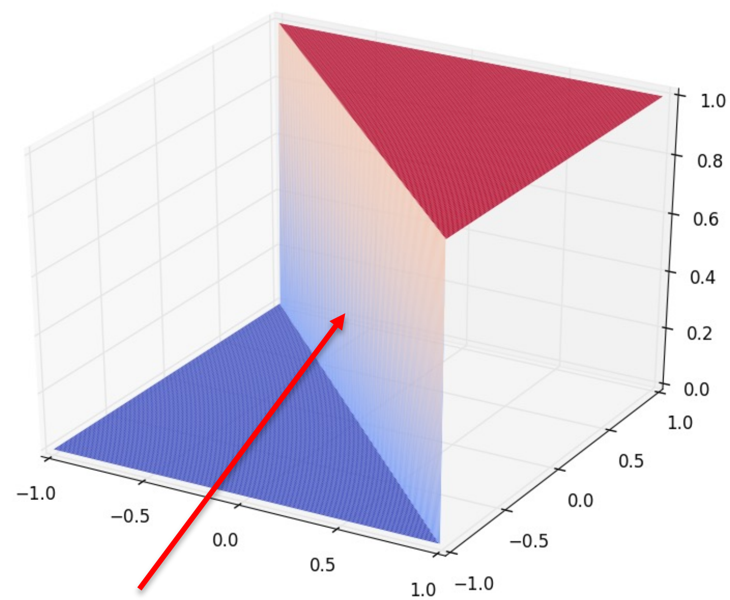
x_1	x_2	Label
-0.7	-0.6	0
-0.6	0.4	0
-0.5	0.7	1
-0.5	-0.2	0
-0.1	0.7	1
0.1	-0.2	0
0.3	0.5	1
0.4	0.1	1
0.4	-0.7	0
0.7	0.2	1

Training Set

Decision Function

- In this case, the data points can be classified with a **linear decision boundary**





Goal: learn this function

The Perceptron

- The perceptron is the simplest possible neural network, containing only one neuron

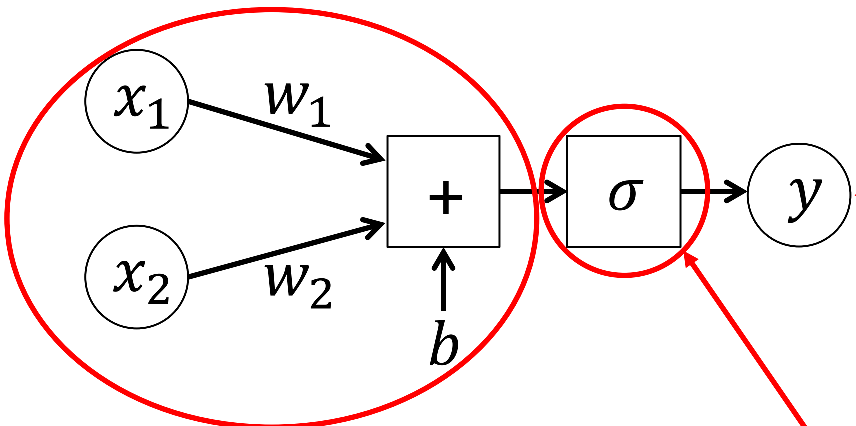
- Described by the following equation:

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + b)$$

w_i = weights
b = bias
σ = activation function

Breaking Down the Perceptron

- A 2-input, 1-output perceptron:



Output
Probability that the label is 1

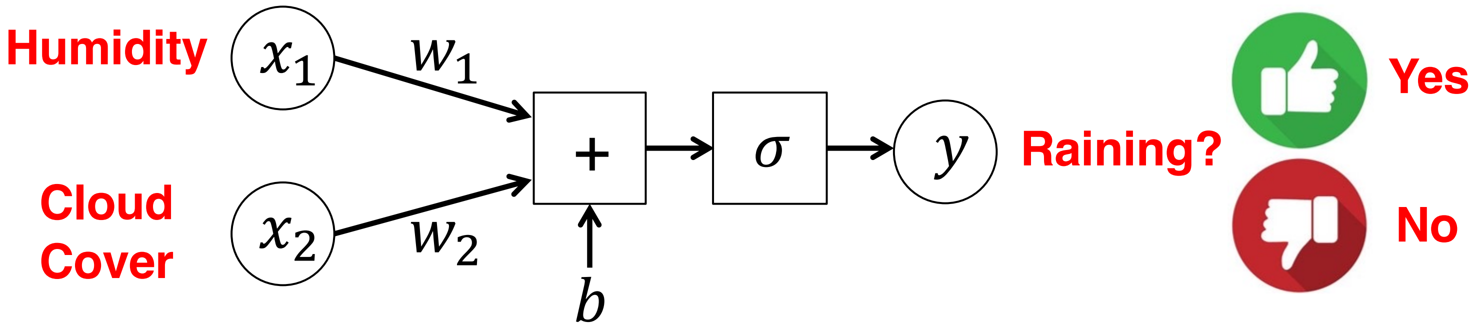
Non-linear function
"squashes" output to the interval (0,1)

Linear function of $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$Wx + b$$

Breaking Down the Perceptron

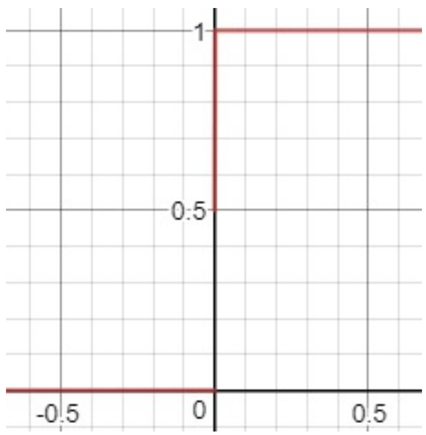
- A 2-input, 1-output perceptron:



Activation Function

- The activation function σ is non-linear:

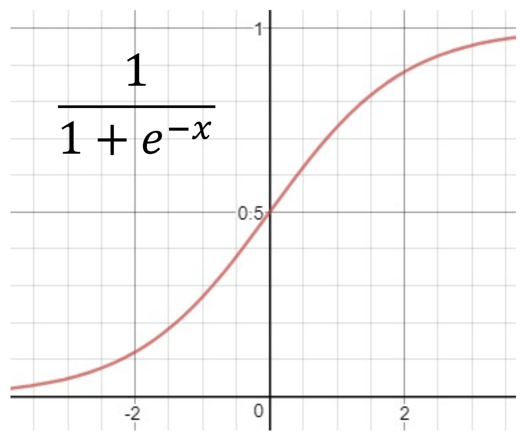
Unit Step



Hard Yes/No decision

Used in the first perceptrons

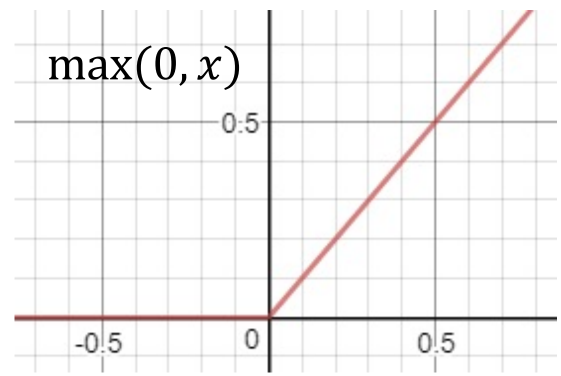
Sigmoid



Soft probability

Used in early neural nets

ReLU



Makes deep networks easier to train

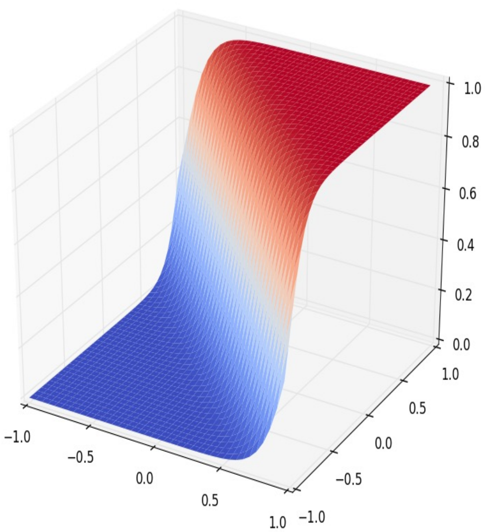
Used in modern deep nets

The Perceptron Decision Boundary

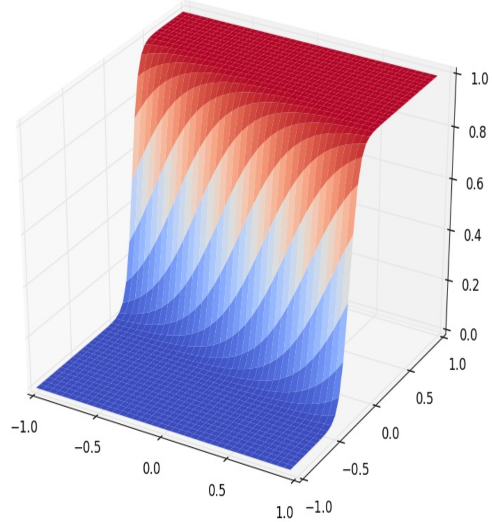
- Let’s plot the 2-input perceptron (sigmoid activation)

$$y = \sigma(w_1 x_1 + w_2 x_2 + b)$$

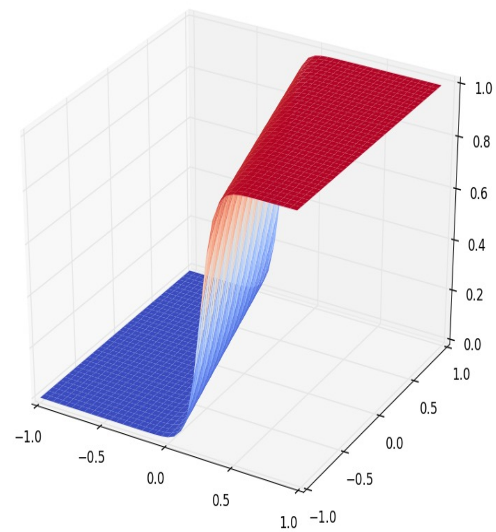
$w_1 = 10$
$w_2 = 10$



$w_1 = 5$
$w_2 = 20$



$w_1 = 20$
$w_2 = 5$



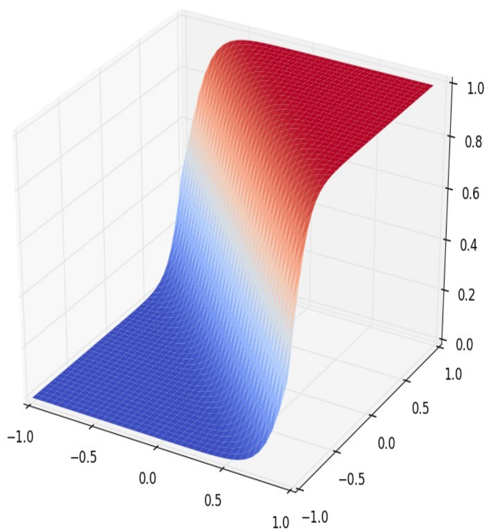
Ratio of weights change the direction of the decision boundary

The Perceptron Decision Boundary

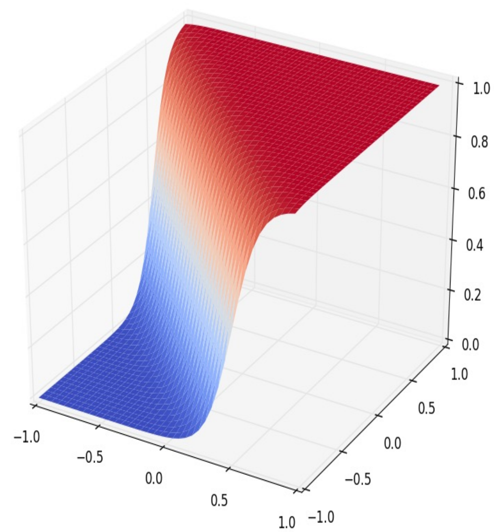
- Let's plot the 2-input perceptron (sigmoid activation)

$$y = \sigma(w_1 x_1 + w_2 x_2 + b)$$

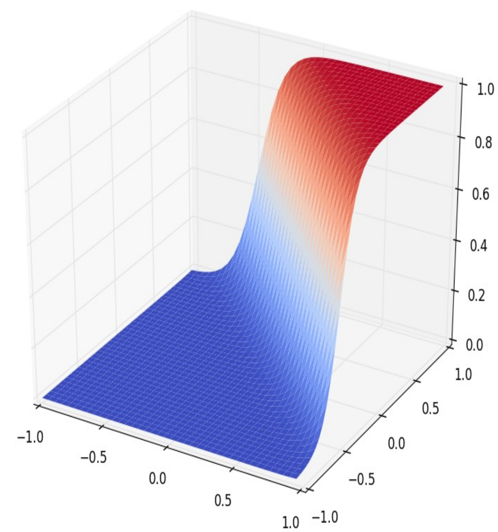
$b = 0$



$b = 5$



$b = -5$



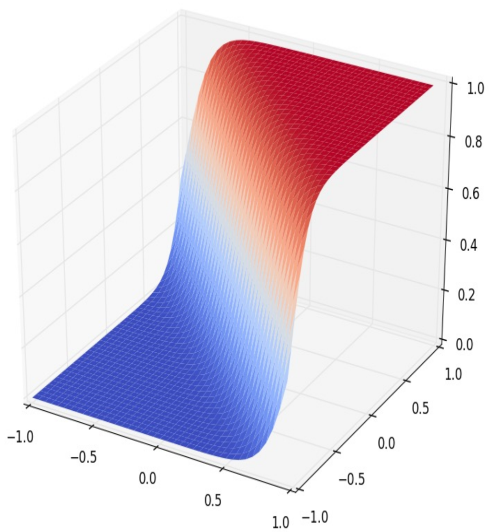
Bias moves boundary away the from origin

The Perceptron Decision Boundary

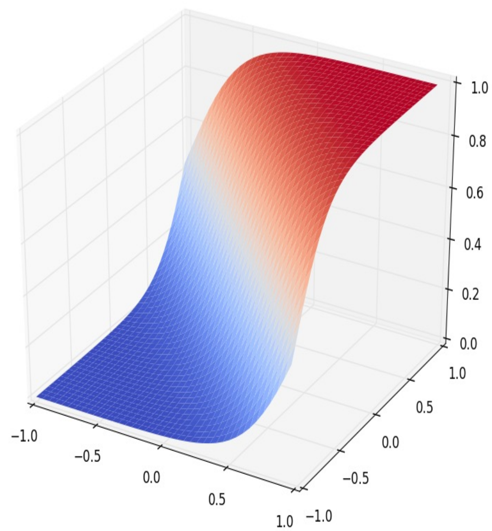
- Let's plot the 2-input perceptron (sigmoid activation)

$$y = \sigma(w_1 x_1 + w_2 x_2 + b)$$

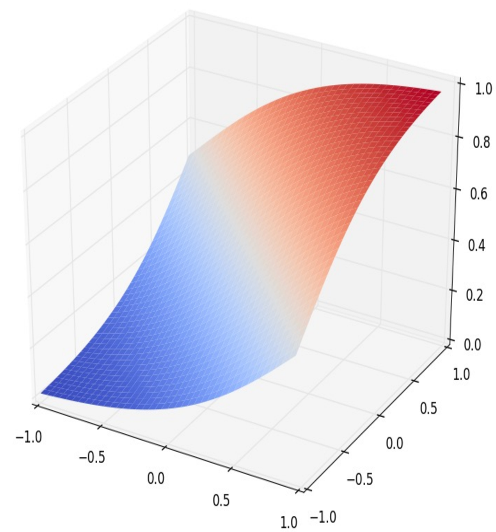
$w_1 = 10$
$w_2 = 10$



$w_1 = 5$
$w_2 = 5$



$w_1 = 2$
$w_2 = 2$



Magnitude of weights change the steepness of the decision boundary

Finding the Parameters

- The right parameters (weights and bias) will create any linear decision boundary we want

- **Training** = process of finding the parameters to solve our classification problem
  - Basic idea: iteratively modify the parameters to reduce the **training loss**
  - **Training loss**: measure of difference between predictions and labels on the training set

Gradient Descent

- **Loss function**
  - Measure of difference between predictions and true labels

$$L = \sum_{i=0}^{N} (y^{(i)} - t^{(i)})^2$$

Sum over training samples

$y^{(i)}$ = Prediction
$t^{(i)}$ = True label

- **Gradient Descent**:

$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

Gradient = direction of steepest descent in L

k = training step
η = learning rate or step size

Training a Neural Network

- At each step k:
  1. Classify each sample to get each $y^{(i)}$
  2. Compute the **loss** L
  3. Compute the **gradient** $\frac{\partial L}{\partial w_k}$
  4. Update the parameters using **gradient descent**

$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

Demo

- Perceptron training demo
  - No bias (bias = 0)
  - No test set (training samples only)

Part 2

DEEP NEURAL NETWORKS

Deep Neural Network

- A deep neural network (DNN) consists of many **layers** of neurons (perceptrons)
- Each connection indicates a weight w

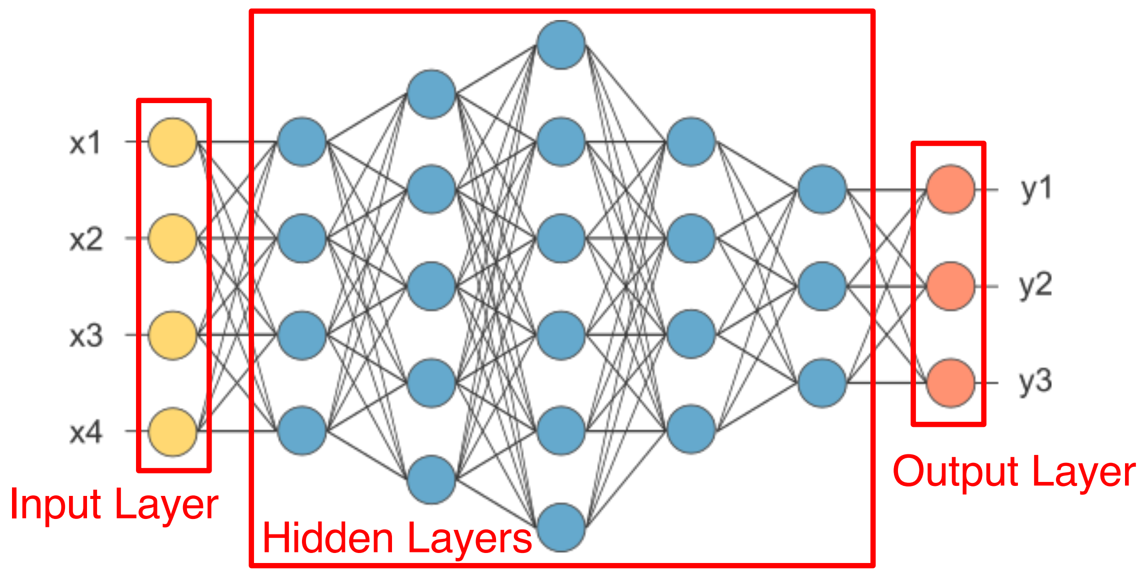


Image credit: http://www.opennn.net/

Combining Neurons

- A single neuron can only make a simple decision
- Feeding neurons into each other allows a DNN to learn **complex decision boundaries**

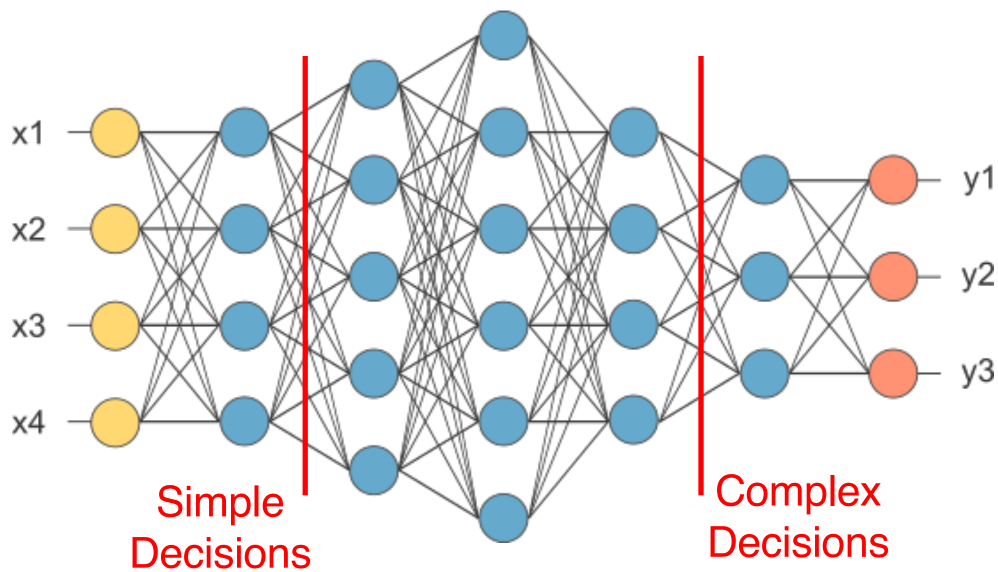


Image credit: http://www.opennn.net/

Complex Decision Boundaries

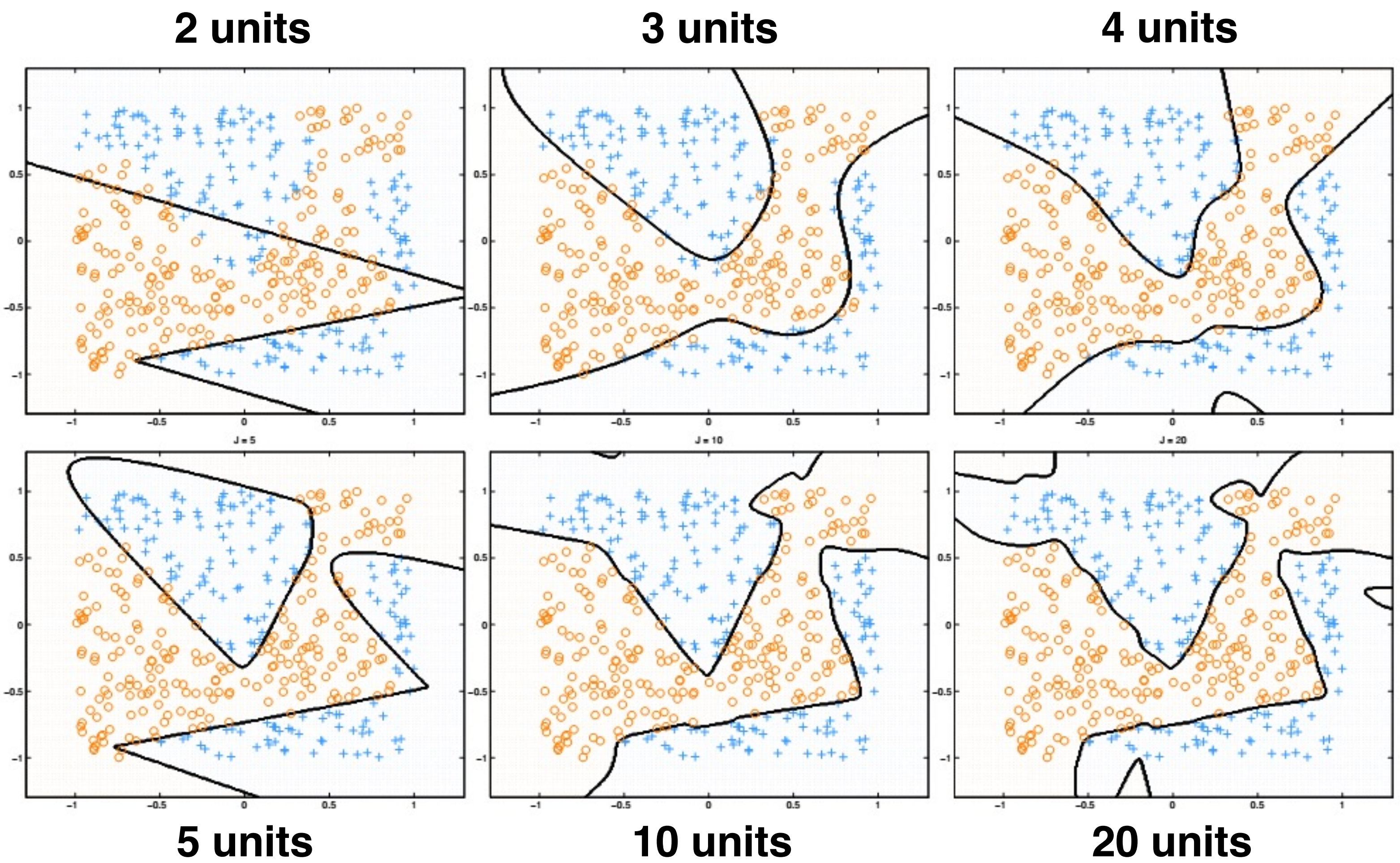


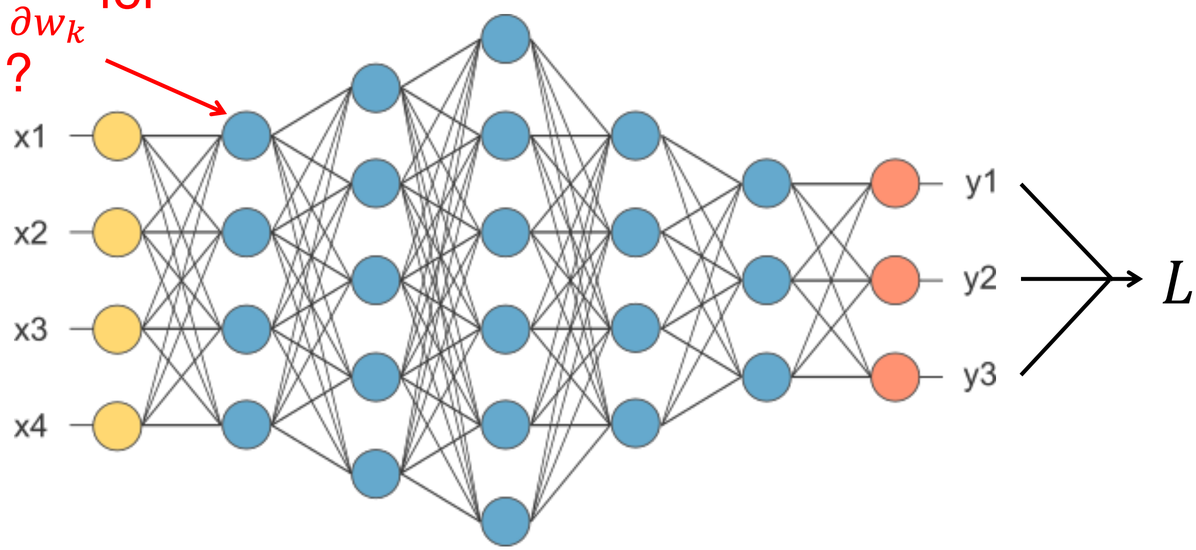
Image credit: https://www.carl-olsson.com/fall-semester-2013/

Learning a Deep Neural Network

- **Gradient Descent**:

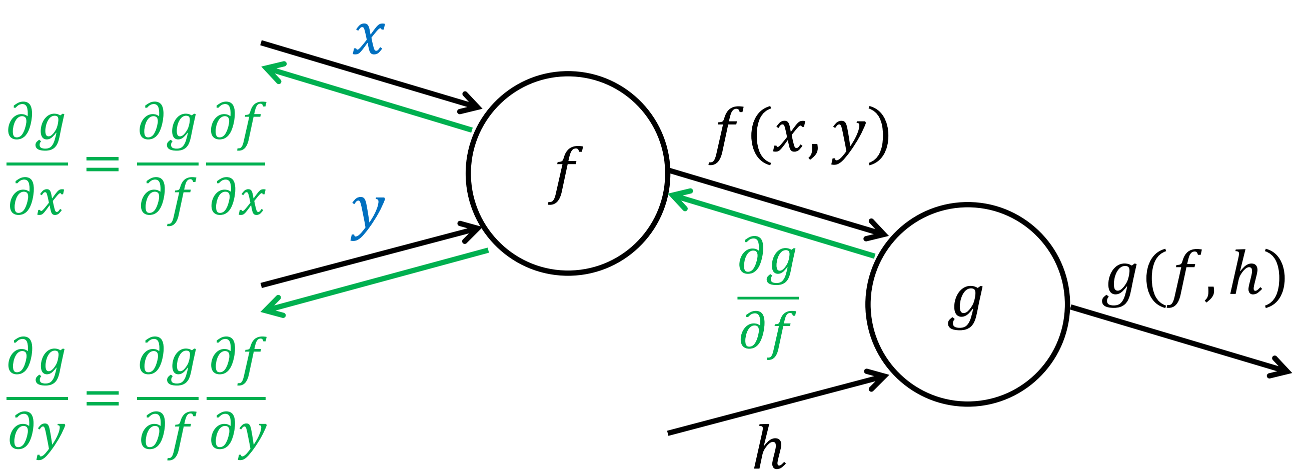
$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

How to get $\frac{\partial L}{\partial w_k}$ for this neuron?



Backpropagation

- **Backpropagation:** use the chain rule from calculus to propagate the gradients backwards through the network



Stochastic Gradient Descent

- Remember **Gradient Descent?**

$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

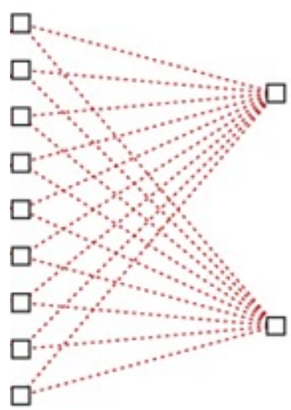
- L must be computed over the entire training set, which can be millions of samples!

- **Stochastic Gradient Descent:**
  - At each set, only compute L for a **minibatch** (a few samples randomly taken from the training set)
  - SGD is faster and **more accurate** than GD for DNNs!

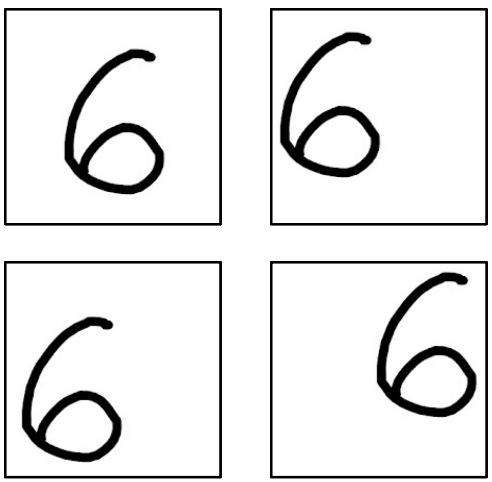
Part 3

CONVOLUTIONAL NEURAL NETWORKS

Neural Networks for Images

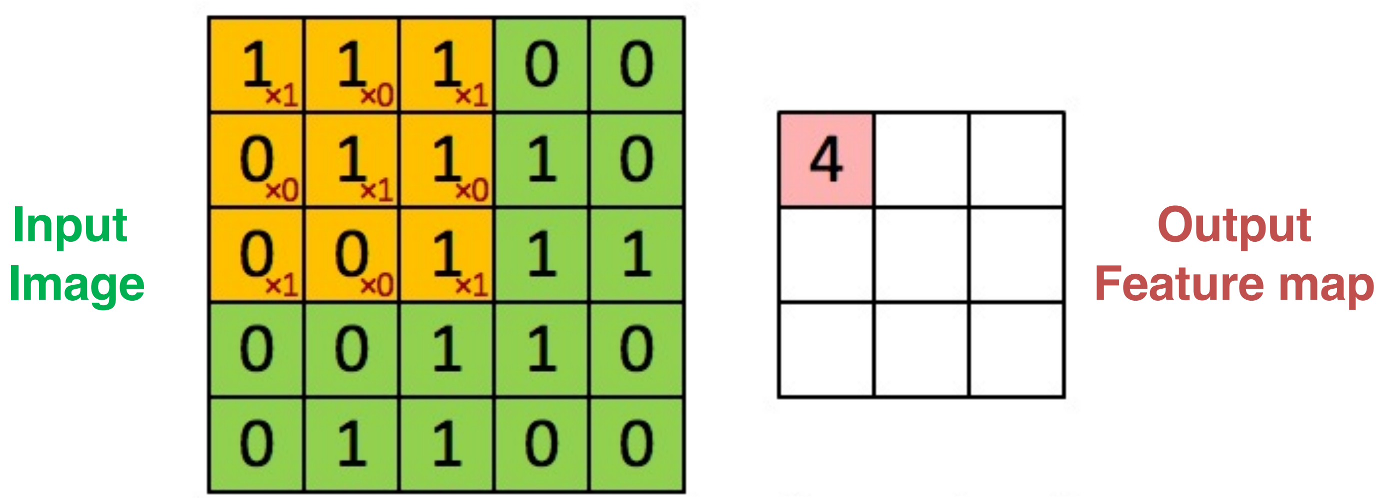


- So far, we've see networks built from **fully-connected layers**
- These networks don't work well for images. Why?



- Images are typically **shift-invariant** (i.e. a 6 is a 6 even when shifted)
- But a fully-connected neuron probably won't work when the input is shifted

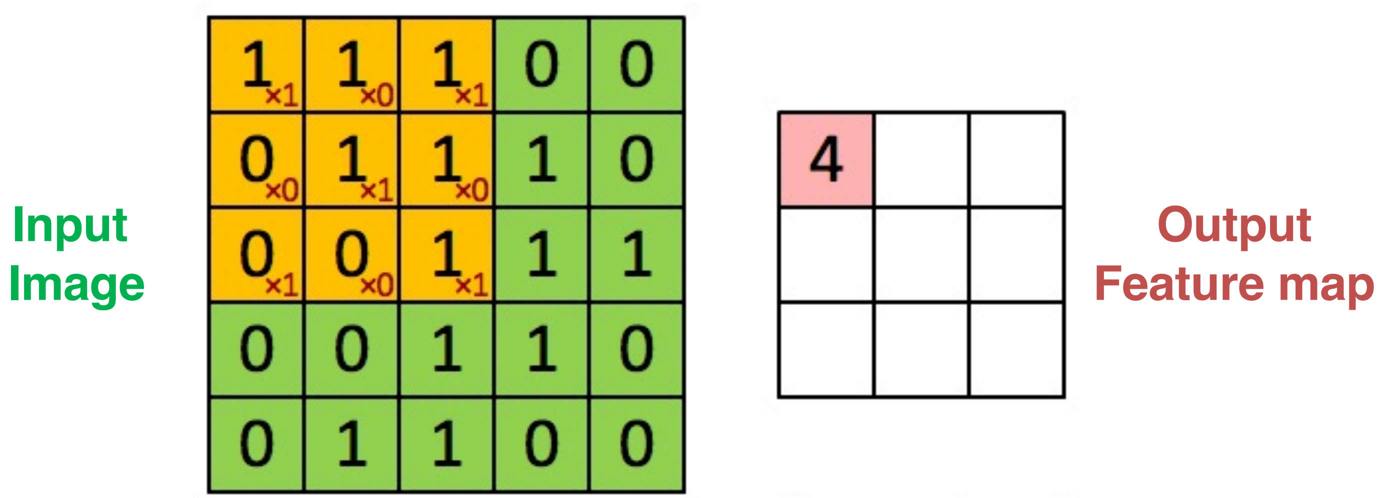
The Convolutional Filter



- Each neuron learns a **weight filter** and **convolves** the filter over the image
- Each neuron outputs a 2D **feature map** (basically an image of features)

Image credit: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

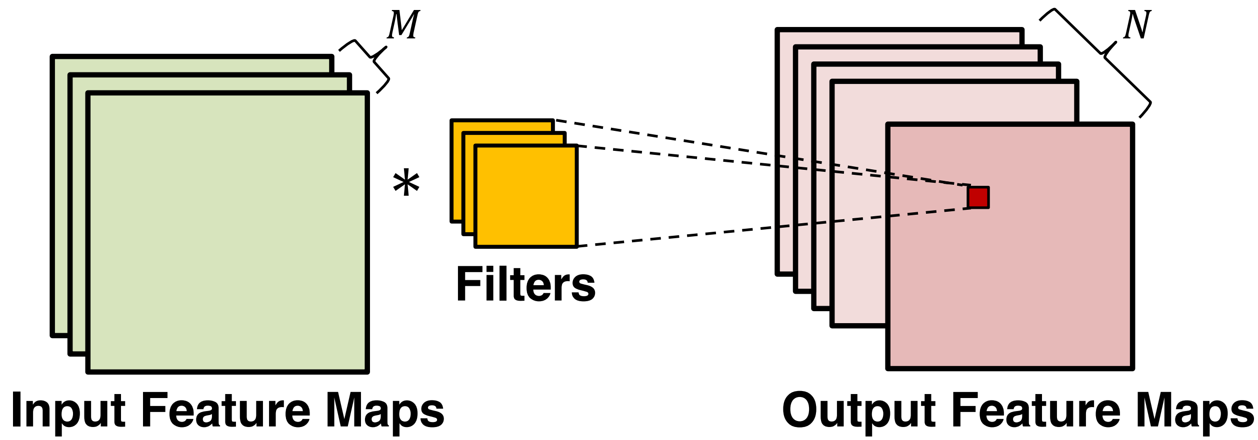
The Convolutional Filter



- Each point in the feature map encodes both a decision and its spatial location
- **Detects the pattern anywhere in the image!**

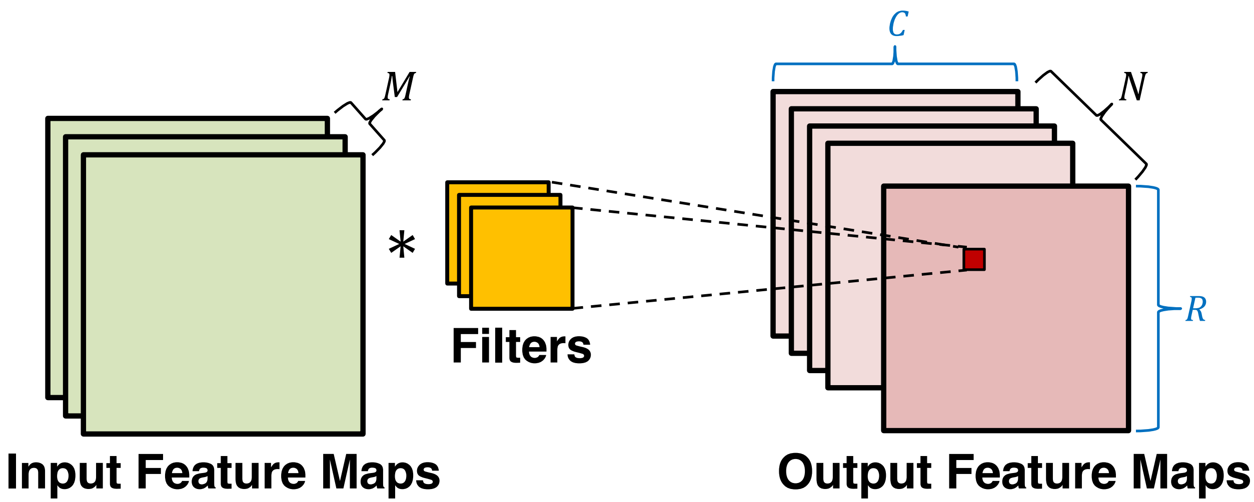
Image credit: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

The Convolutional Layer



- M input and N output feature maps
- Each output map uses M filters, 1 per input map
- $M \times N$ total filters

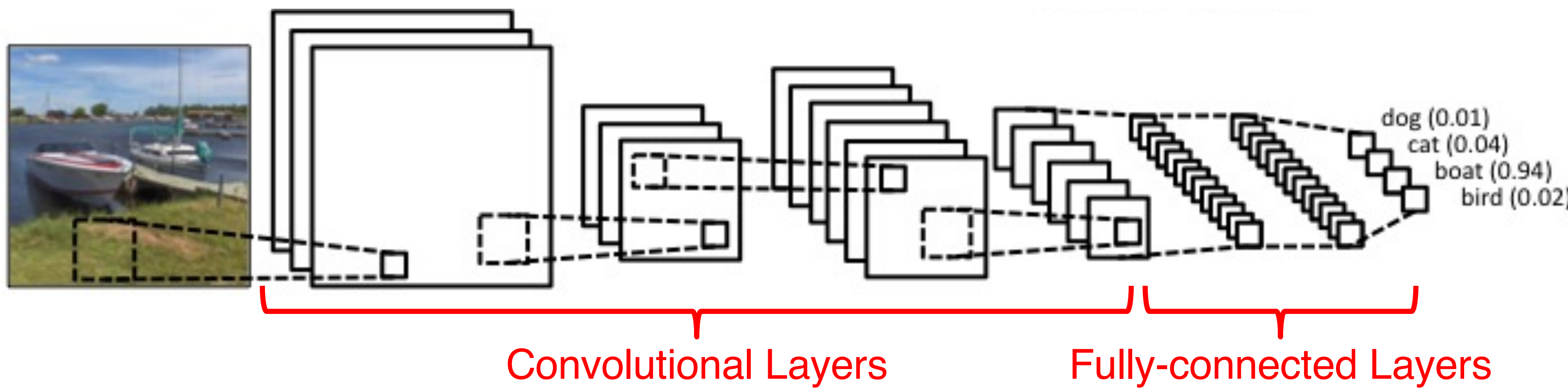
The Convolutional Layer




```
for(row=0; row<R; row++) {
    for(col=0; col<C; col++) {
        for(to=0; to<N; to++) {
            for(ti=0; ti<M; ti++) {
                for(i=0; i<K; i++) {
                    for(j=0; j<K; j++) {
                        output_fm[to][row][col] +=
                                weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

Huge amount of parallelism!

Convolutional Neural Network



- **Front**: convolutional layers learn visual features
- Feature maps get downsampled through the network
- **Back**: fully-connected layers perform classification using the visual features

Image credit: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Learning Complex Features

- Deep CNNs combine simple features into complex patterns
  - Early conv layers = edges, textures, ridges
  - Later conv layers = eyes, noses, mouths

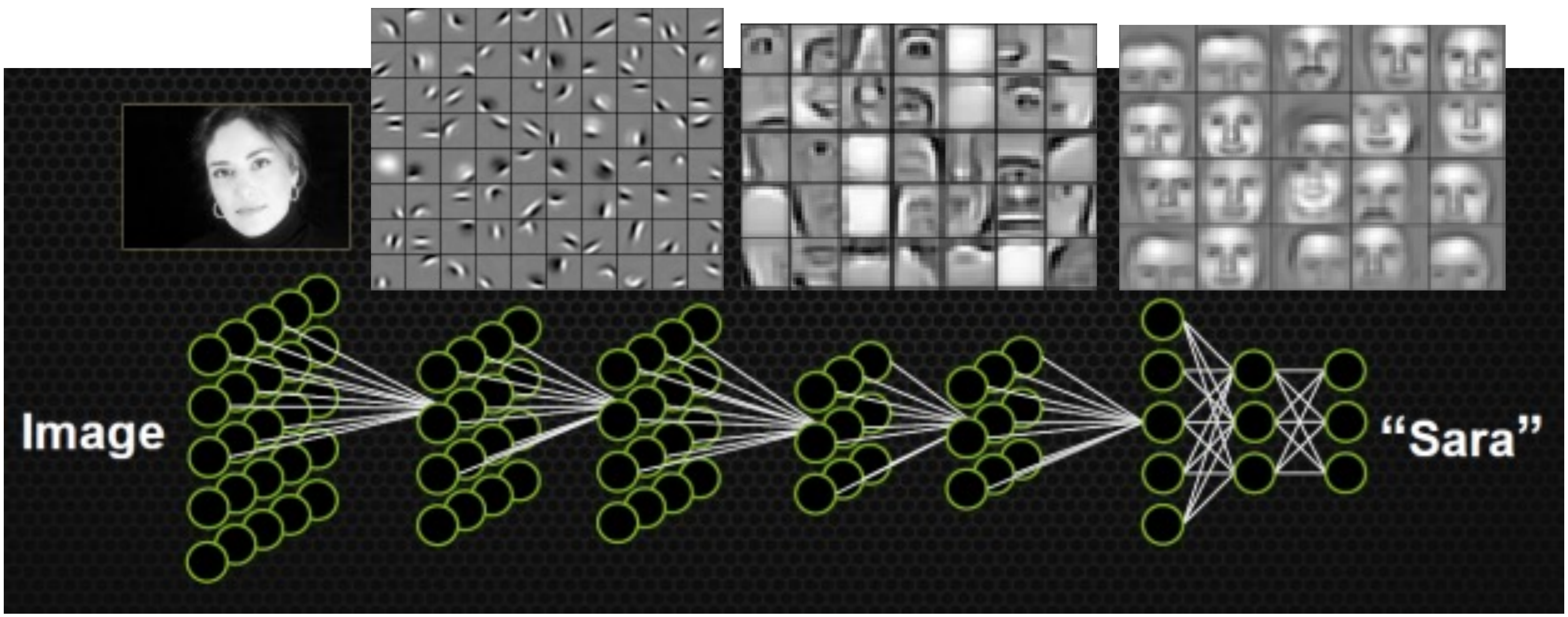


Image credit: https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/; H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks", CACM Oct 2011

FIN