ECE 6775
High-Level Digital Design Automation
Fall 2023

Introduction to Neural Networks

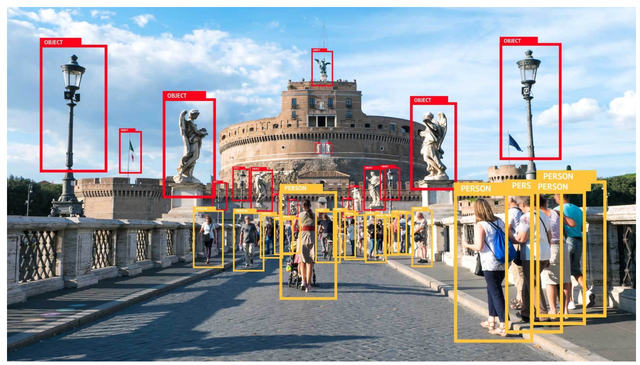
Jordan Dotzel, Ritchie Zhao, Zhiru Zhang
School of Electrical and Computer Engineering

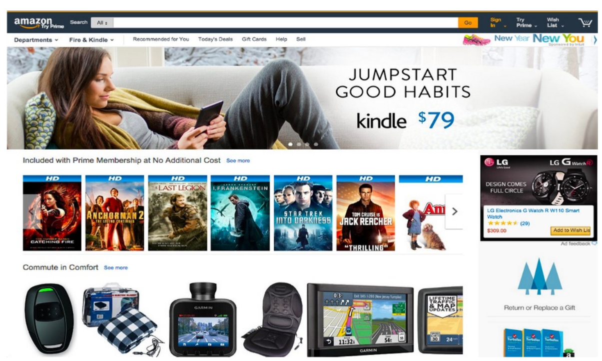
Cornell University

CSL

Rise of Deep Neural Networks (DNNs)

- DNNs have revolutionized information technology
  - computer vision, e-commerce, finance, game AI, healthcare, machine transcription & translation, robots, web search, and many more (to come)



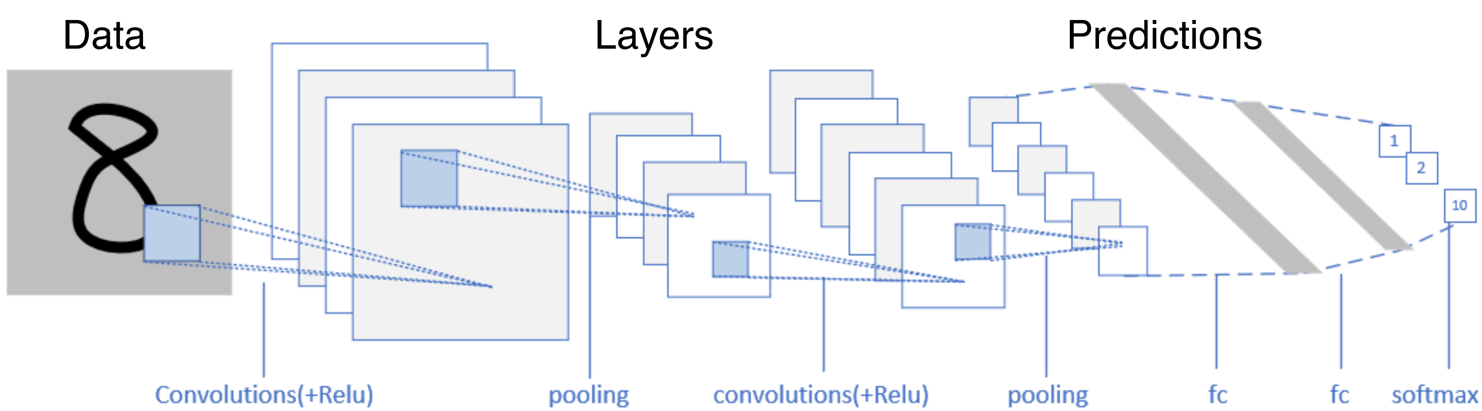


A Brief History

- **1940's – 1950's:** First artificial neural networks proposed based on biological structures in the human visual cortex
- **1980's – 2000's:** Neural networks (NNs) considered inferior to other simpler algorithms (e.g., SVM)
- **Mid 2000's**: NN research considered "dead", machine learning conferences outright reject most NN papers
- **2010 – 2012**: DNNs begin winning large-scale image and document classification contests
- **2012 – Now**: DNNs prove themselves in many industrial applications (web search, translation, image analysis)

Neural Network (NN) in a Nutshell

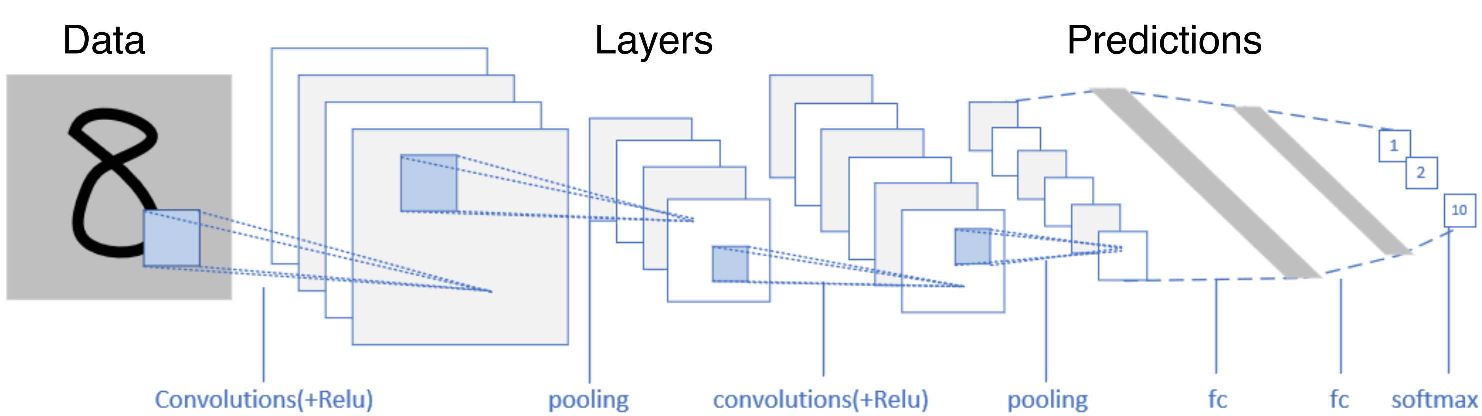
- Depends on a large volume of data & mostly supervised
- **Learns a function** that encodes a hierarchy of abstract *features*
  - Consists of a stack of connected *layers*, such as convolutional, pooling, full connected, attention



NN Training and Inference

- **Training** refers to the process of building an NN model to accomplish a specific AI task by “learning” from a predetermined dataset

- **Inference** refers to the use of a trained NN model to make a prediction (or decision) on unseen data



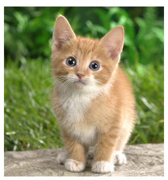
Part 1

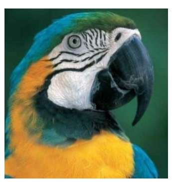
CLASSIFICATION WITH THE PERCEPTRON

Classification Problems

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

Inputs | **Labels**

 → Cat

 → Bird

 → Human

Training Set

- 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 | **Predictions**

 → ???

Predict New Data

Artificial Neuron

- The simplest possible neural network contains only one "neuron", which is described by the following equation:

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

w_i = weights
b = bias
σ = activation function

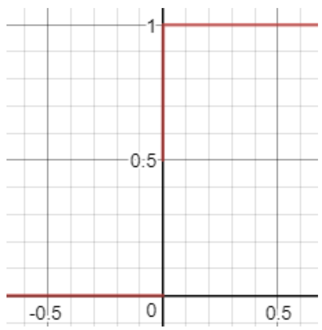
- When σ is the unit step (or sign) function, we have a **perceptron** *
  - Invented in 1957 by Frank Rosenblatt

* Perceptron is often used as a synonym for artificial neuron, where σ could be any activation function

Activation Function

- The activation function σ is non-linear

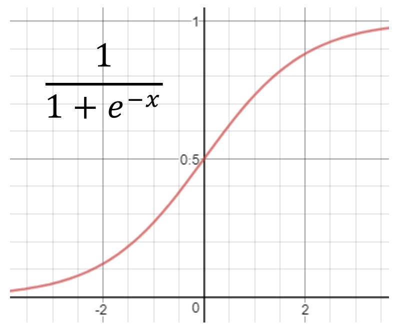
Unit Step



Hard Yes/No decision

Used in the initial perceptrons

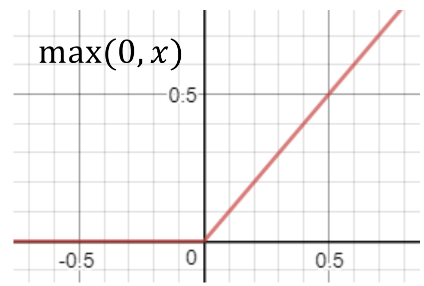
Sigmoid



Soft probability

Used in early neural nets

ReLU

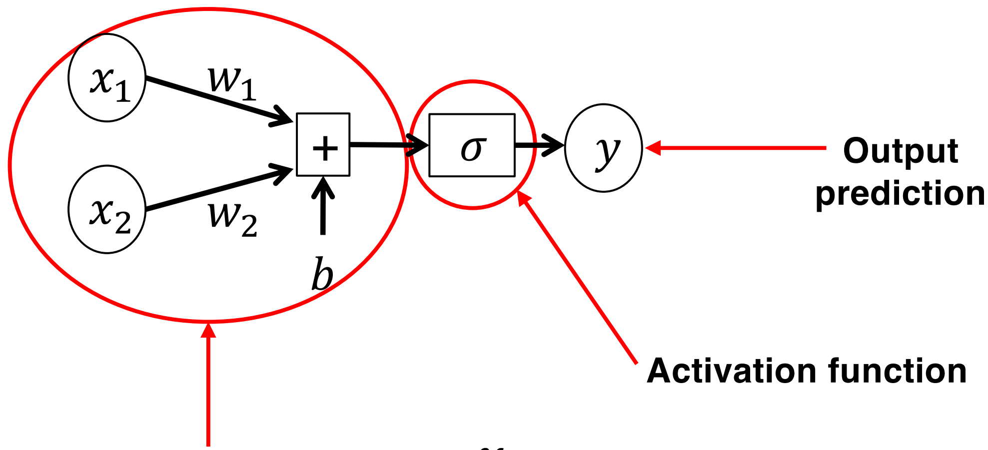


Makes deep networks easier to train

Used in modern deep nets

Breaking Down the "Neuron"

A 2-input, 1-output neuron

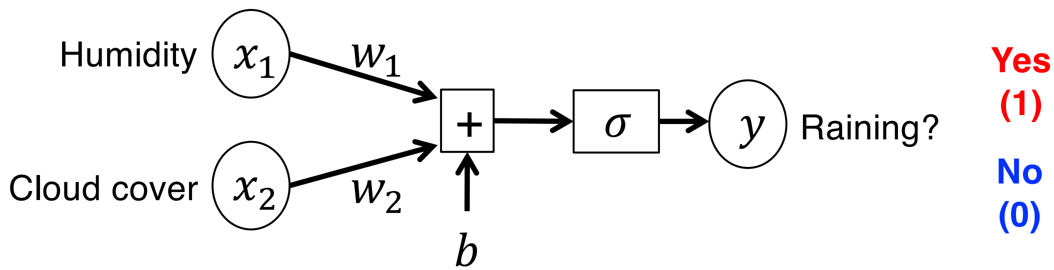


Linear function of $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$$

Breaking Down the "Neuron"

A 2-input, 1-output neuron



A real-life analogue

A Real-life Classification Problem

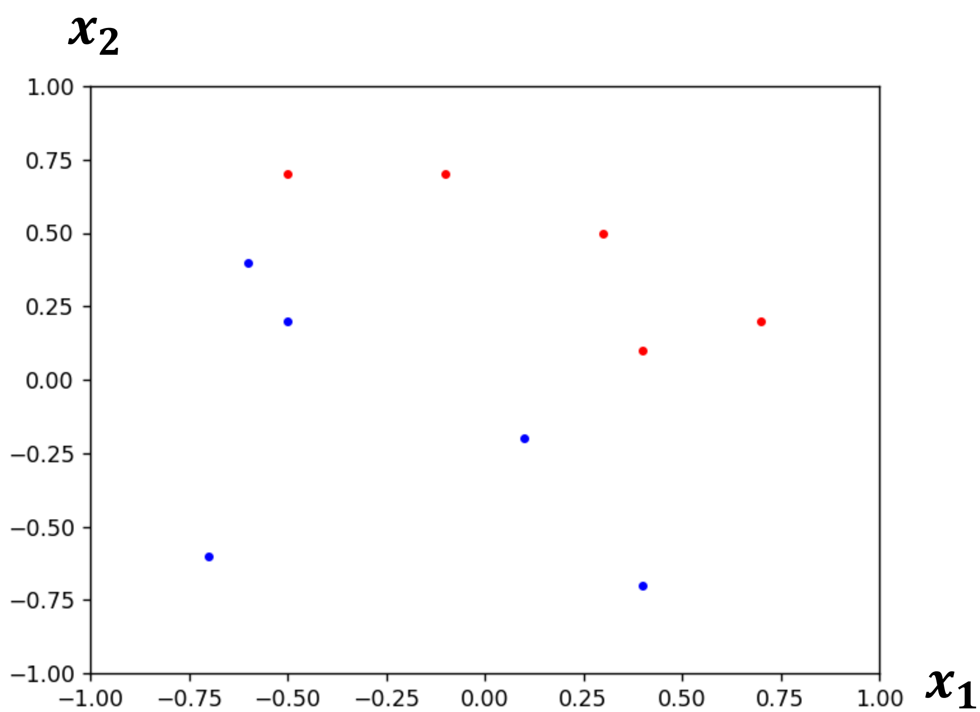
- **Inputs**: Pairs of numbers (x_1, x_2)
- **Labels**: 0 or 1 (binary decision problem)

- 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

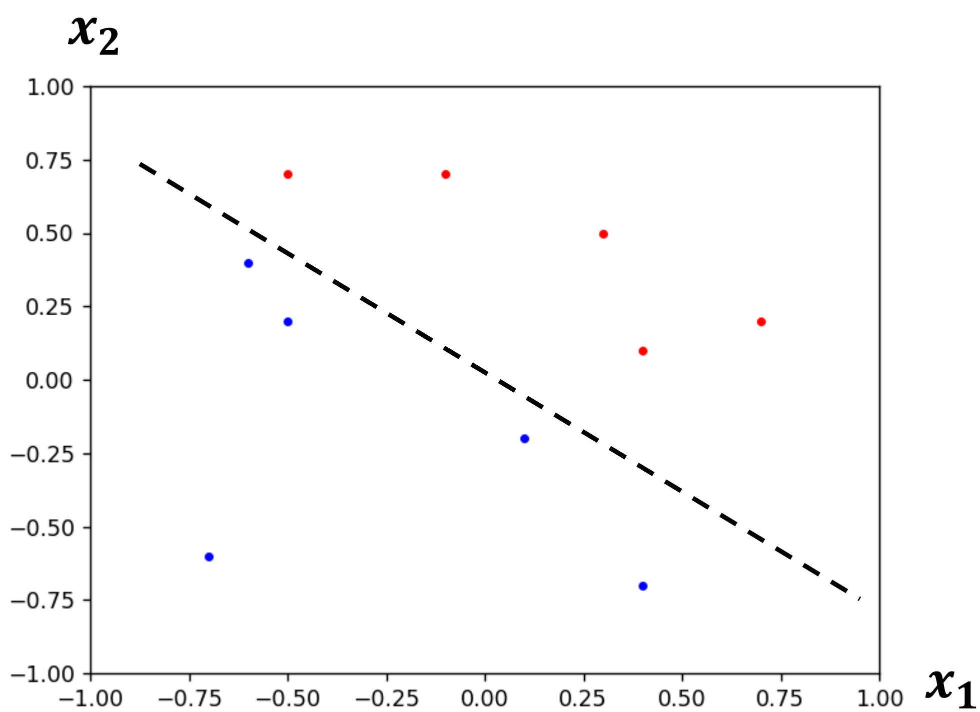


Plot of the data points

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 Boundary



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

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

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**

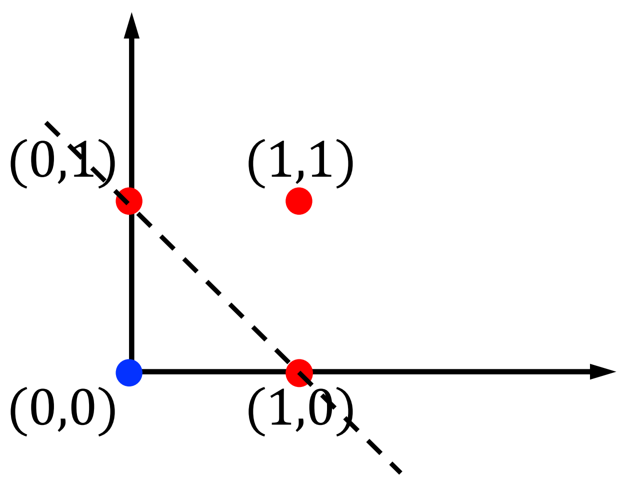
$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

Demo

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

Another Example

x_2	x_1	OR		
0	0	0	$w_1x_1 + w_2x_2 + b < 0$	$\Rightarrow b < 0$
0	1	1	$w_1x_1 + w_2x_2 + b \geq 0$	$\Rightarrow w_1 \geq -b$
1	0	1	$w_1x_1 + w_2x_2 + b \geq 0$	$\Rightarrow w_2 \geq -b$
1	1	1	$w_1x_1 + w_2x_2 + b < 0$	$\Rightarrow w_1 + w_2 \geq -b$

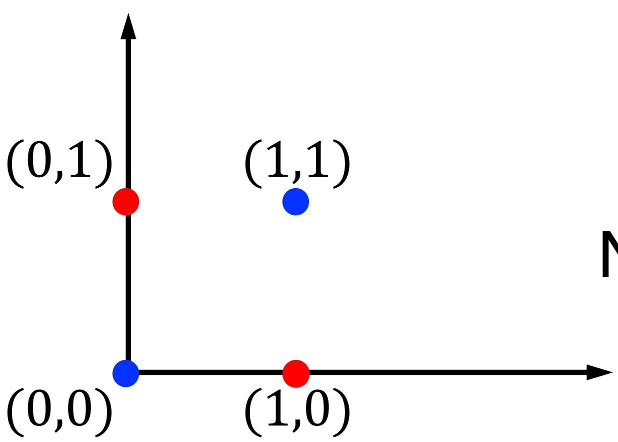


One solution:

$w_1 = 1;\ w_2 = 1;\ b = -1$

Yet Another Example (The XOR Problem)

x_2	x_1	XOR		
0	0	0	$w_1x_1 + w_2x_2 + b < 0$	$\Rightarrow$ $b < 0$
0	1	1	$w_1x_1 + w_2x_2 + b \geq 0$	$\Rightarrow$ $w_1 \geq -b$
1	0	1	$w_1x_1 + w_2x_2 + b \geq 0$	$\Rightarrow$ $w_2 \geq -b$
1	1	0	$w_1x_1 + w_2x_2 + b < 0$	$\Rightarrow$ $w_1 + w_2 < -b$



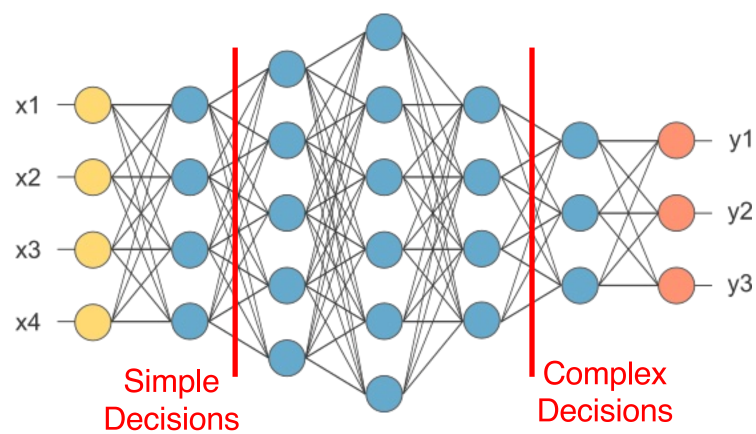
Not linearly separable

Part 2

DEEP NEURAL NETWORKS

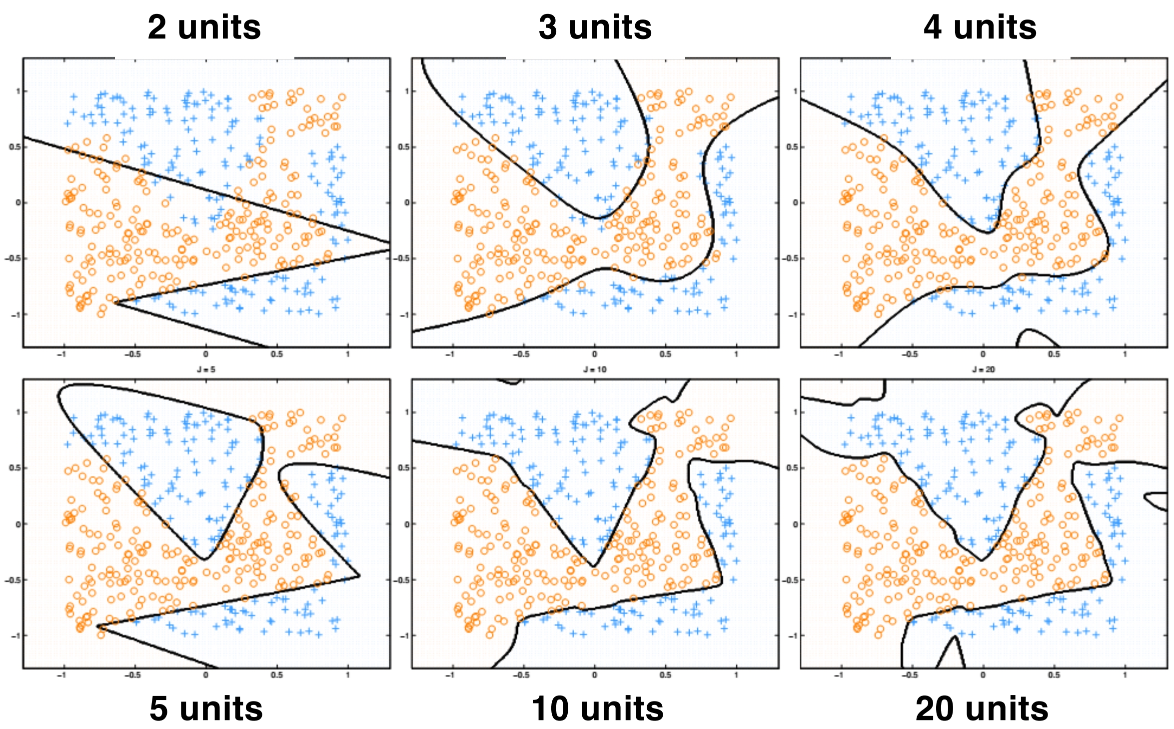
Combining Neurons

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



Source: http://www.opennn.net/

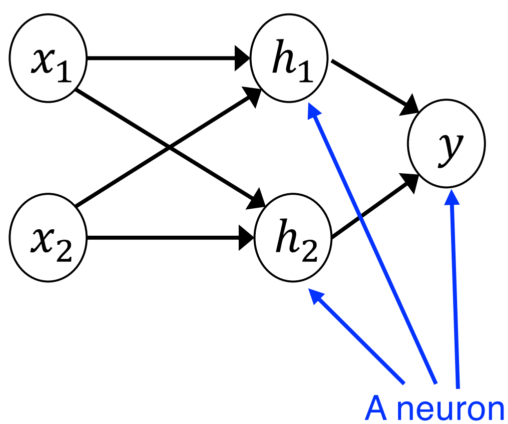
Complex Decision Boundaries



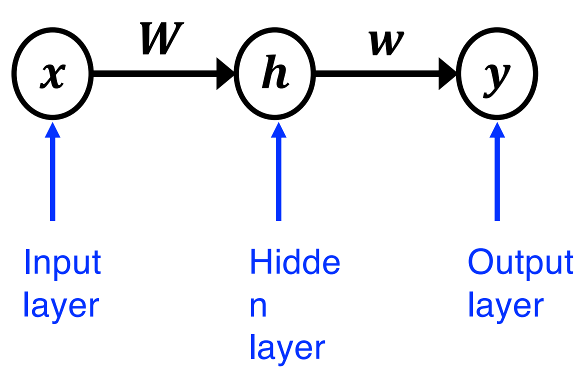
Source: https://www.carl-olsson.com/fall-semester-2013/

Multi-Layer Perceptron (MLP)

- MLP is a fully connected class of feedforward artificial neural network (ANN)
  - An MLP model consists of multiple **layers** of neurons
  - Often referred to as "vanilla" ANNs, especially with a single hidden layer



A vectorized (tensorized) view of MLP



Deep Neural Network (DNN)

- MLPs are neural networks with at least three layers, while DNNs typically have even more (hidden) layers

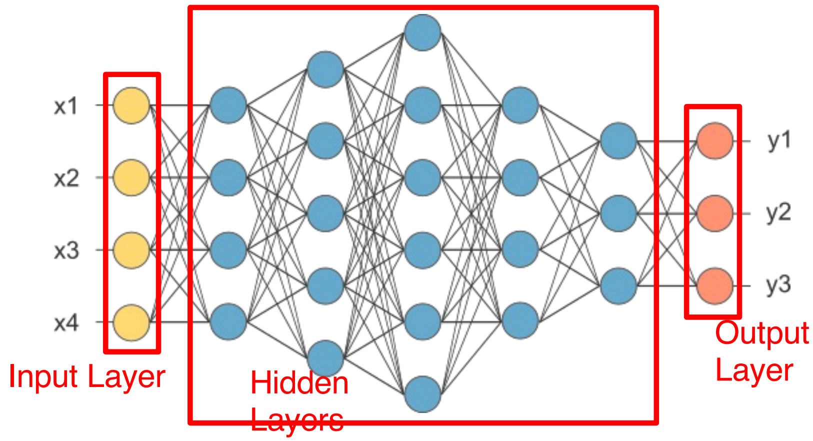


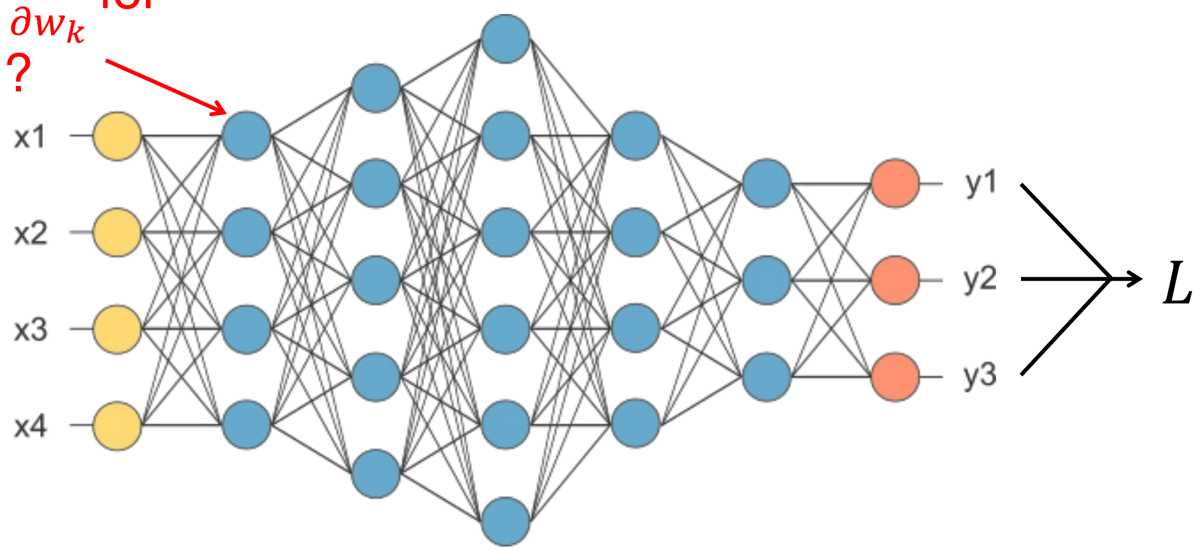
Image credit: http://www.opennn.net/

Learning a Deep Neural Network

- **Gradient Descent**:

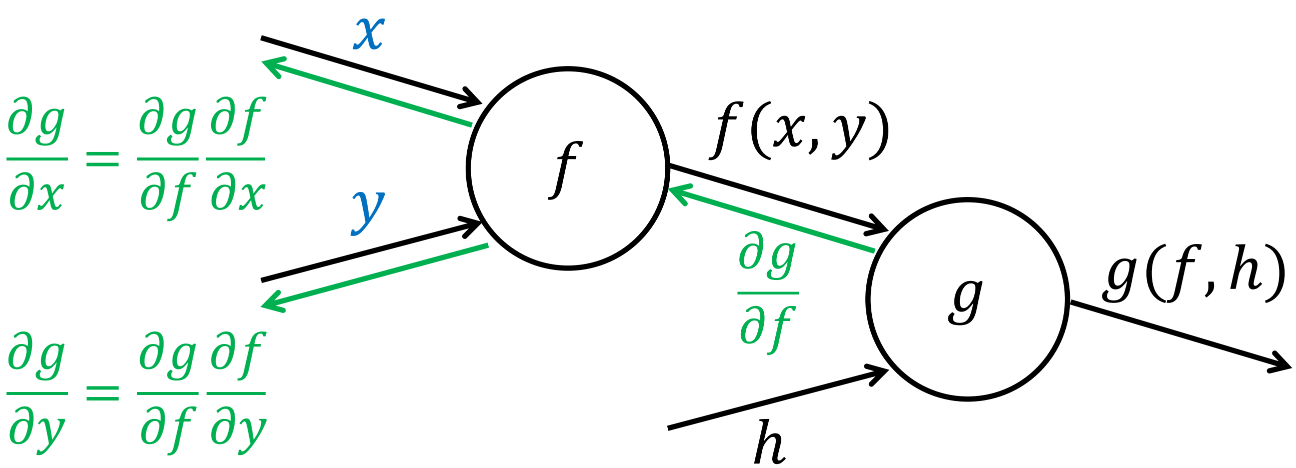
$$w_{k+1} = w_k - \eta \frac{\partial L}{\partial w_k}$$

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!

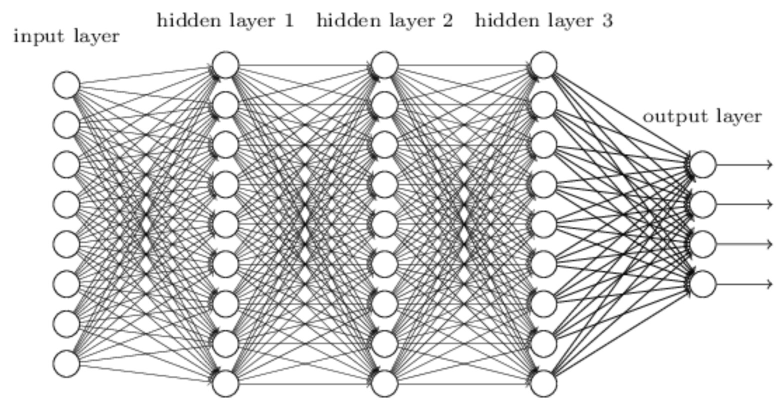
- **Stochastic Gradient Descent:**
  - At each set, only compute L for a **minibatch** (a few samples randomly taken from the training set)
  - SGD is faster and **more accurate** than GD for DNNs!

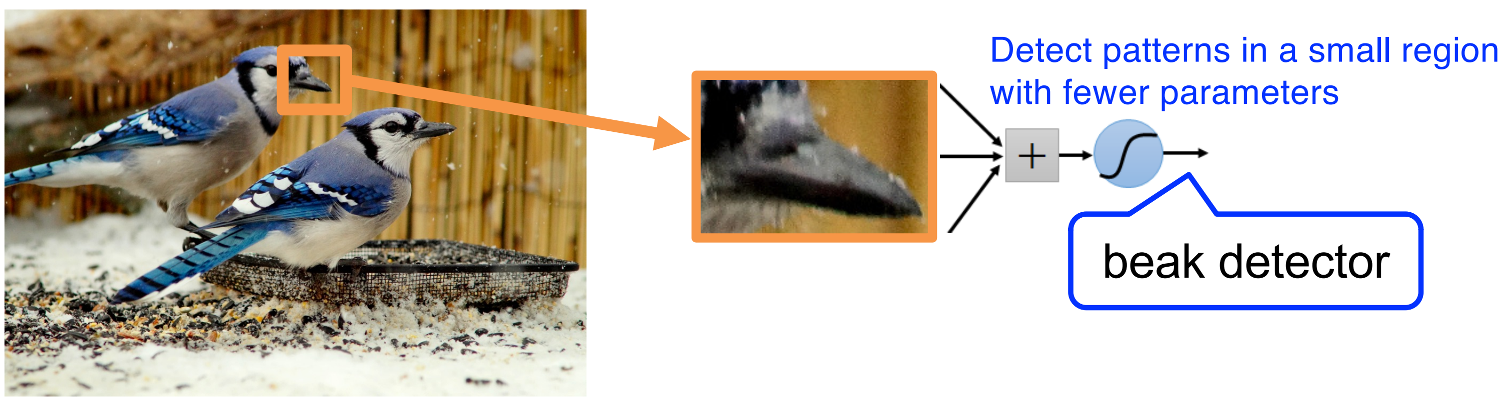
Part 3

CONVOLUTIONAL NEURAL NETWORKS

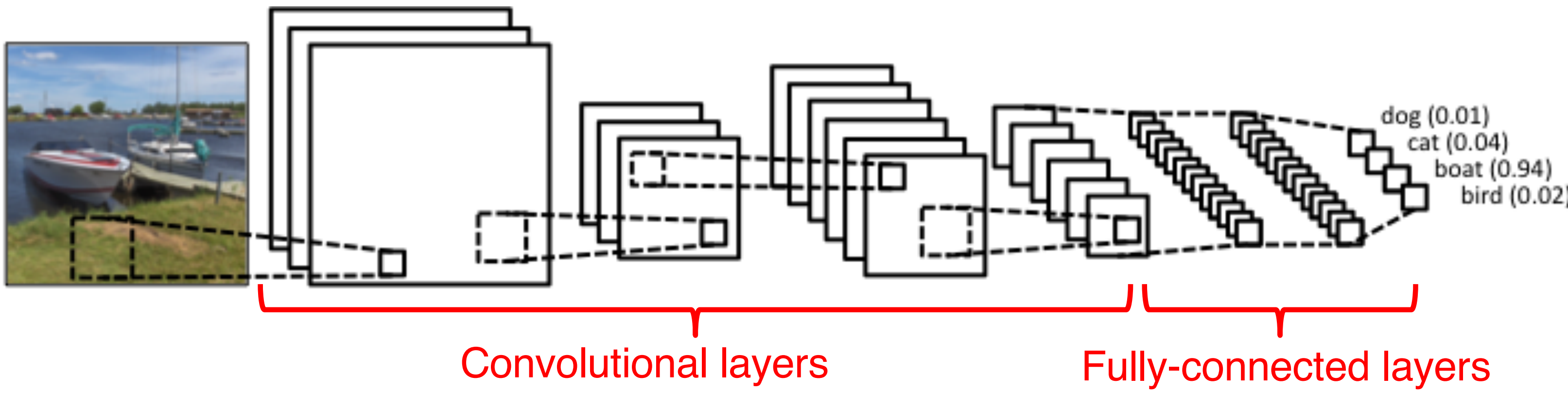
Neural Networks for Images

- So far, we've seen neural networks built from **fully-connected layers**
- Do we really need all the edges for learning an image?
  - Important patterns are typically much smaller than the whole image
  - Images are also shift-invariant (e.g., a bird is a bird even when shifted)





Convolutional Neural Network (CNN)

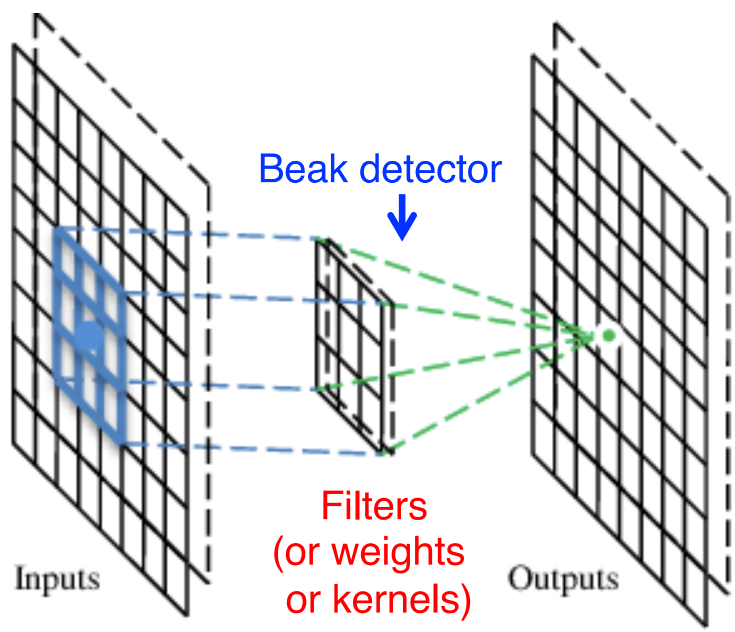


- **Front**: convolutional layers learn visual features
- Feature maps get downsampled through the network
- **Back**: fully-connected layers perform classification using the visual features

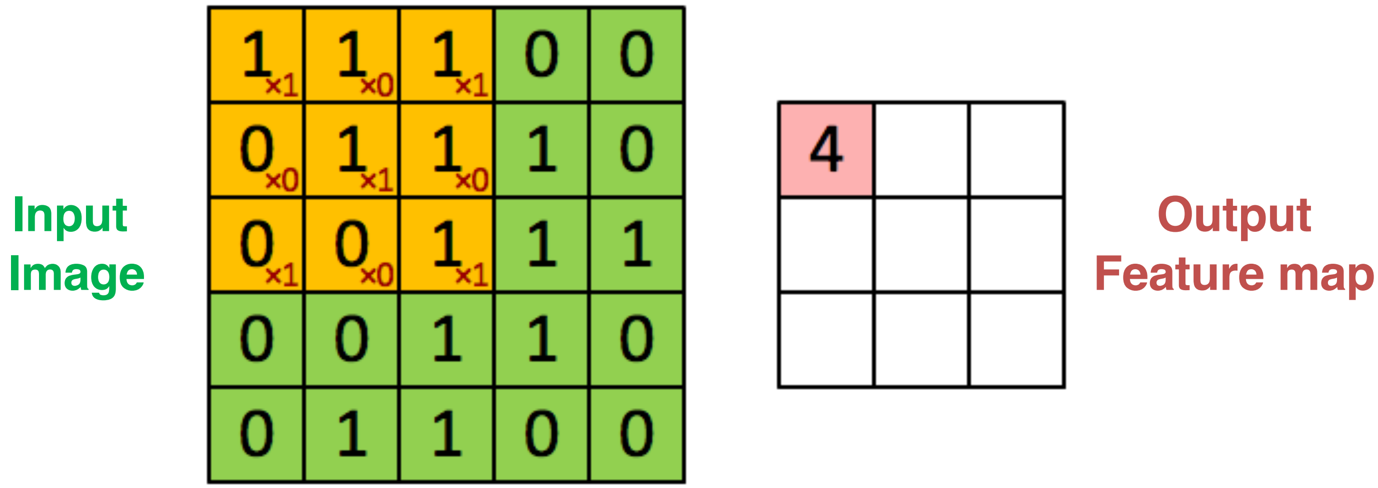
Image credit: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

A Convolutional (conv) Layer

- A CNN stacks multiple conv layers (and some other layers)
- A conv layer has a set of learnable filters that perform convolution operation



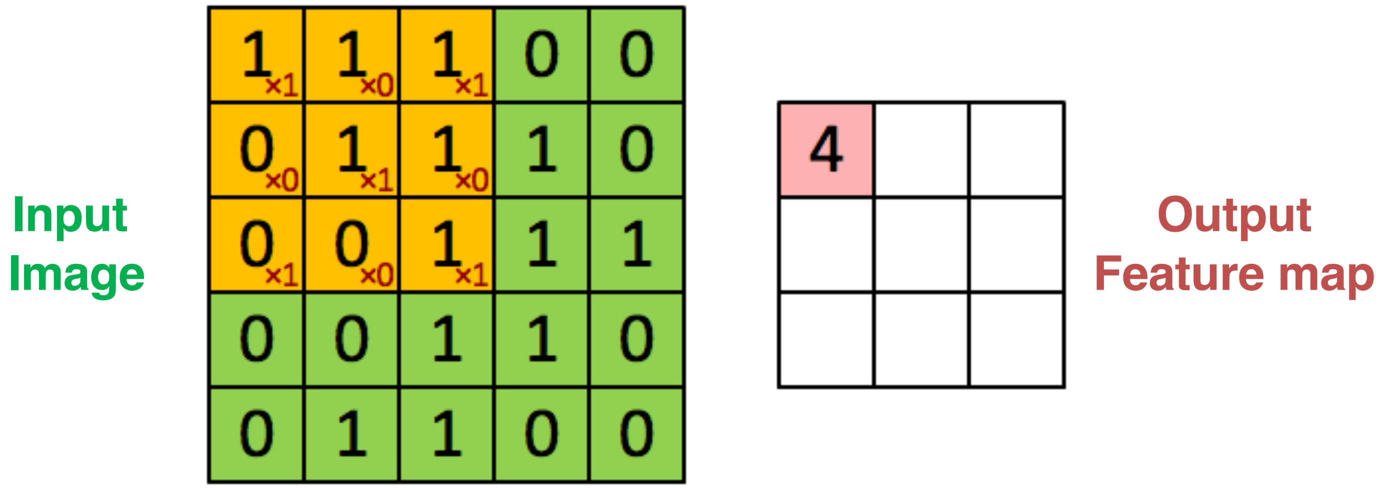
The Convolutional Filter



- 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

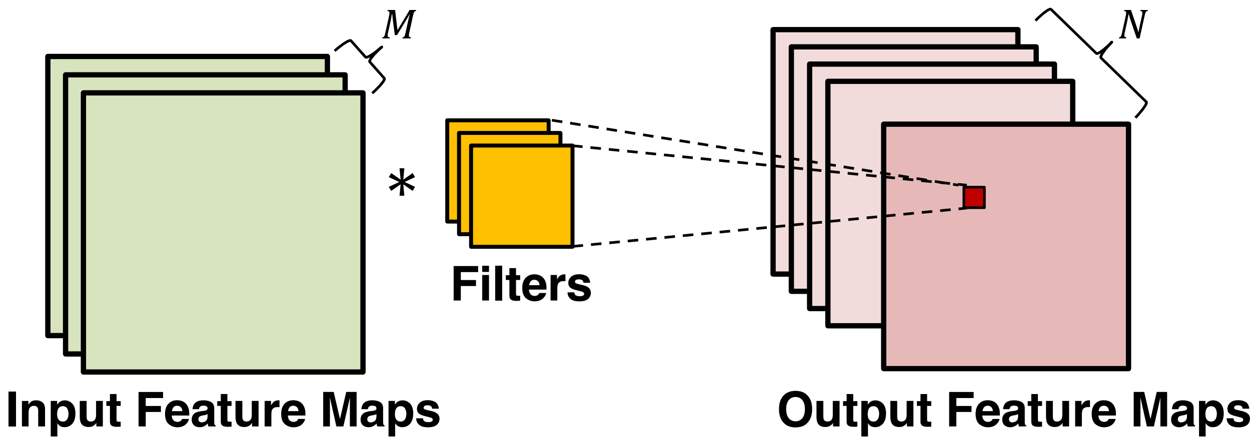
The Convolutional Filter



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

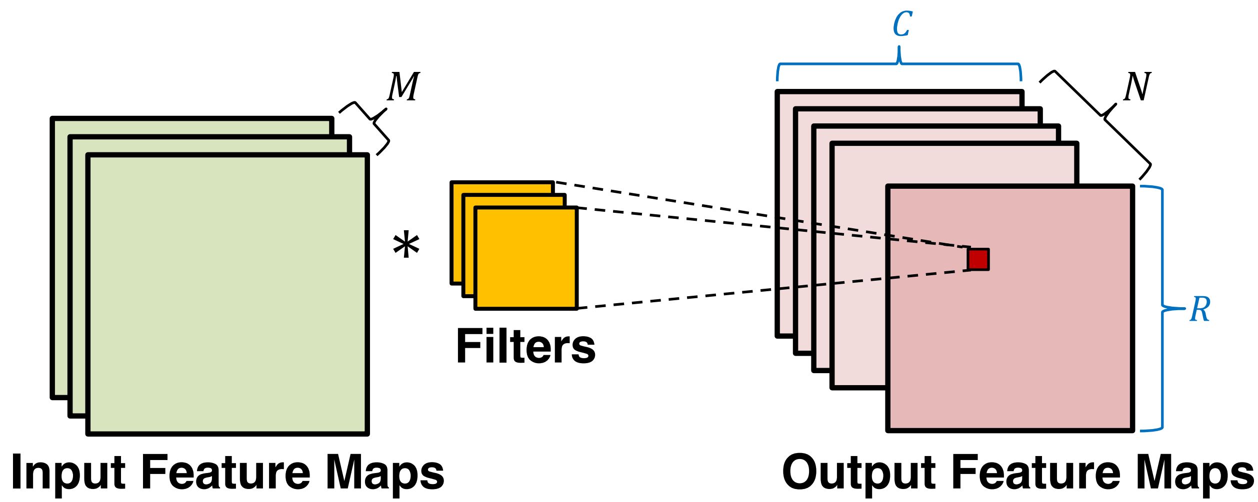
Image credit: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

The Convolutional Layer: A Closer Look



- 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: A Closer Look




```
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!

Learning Complex Features with CNNs

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

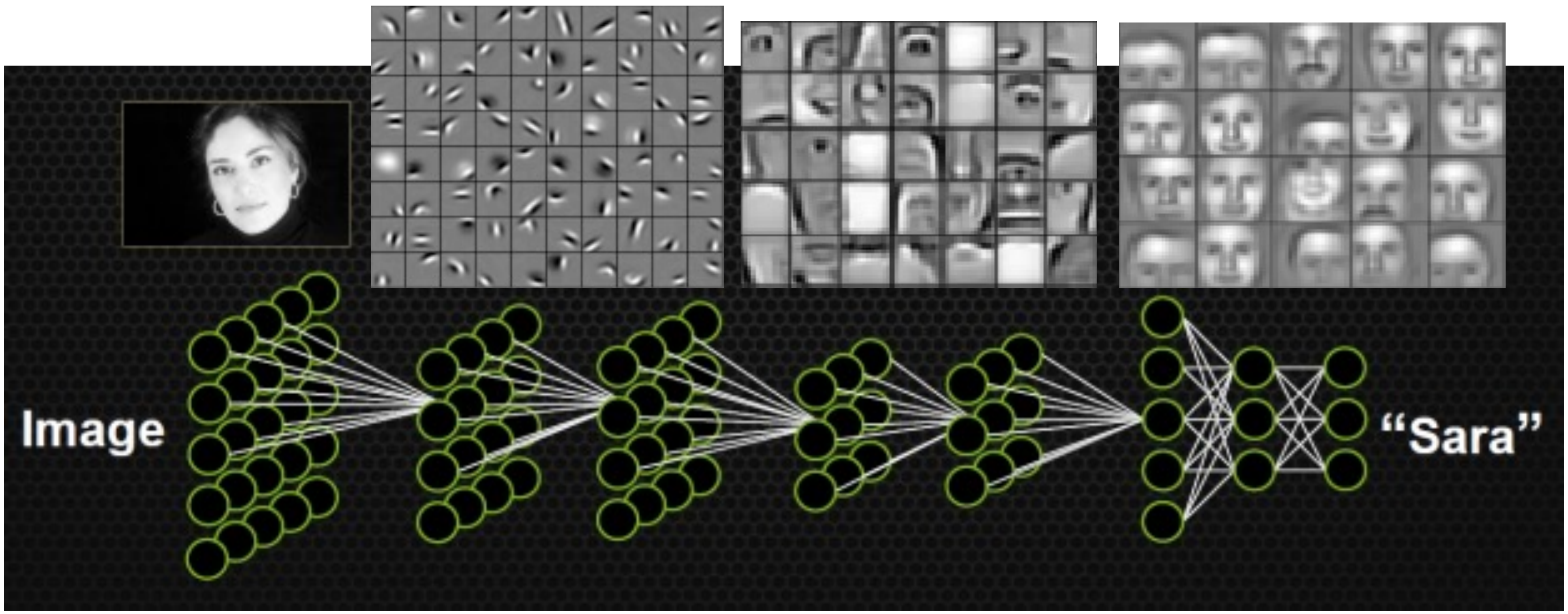


Image credit: https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/; H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks", CACM Oct 2011

Acknowledgements

- The lecture slides contain/adapt materials from
  - Dr. Ritchie Zhao (Microsoft)
  - ECE 5545 by Prof. Mohamed Abdelfattah (Cornell Tech)
  - CS898 by Prof. Ming Li (University of Waterloo)
  - System for AI Education Resource by Microsoft Research