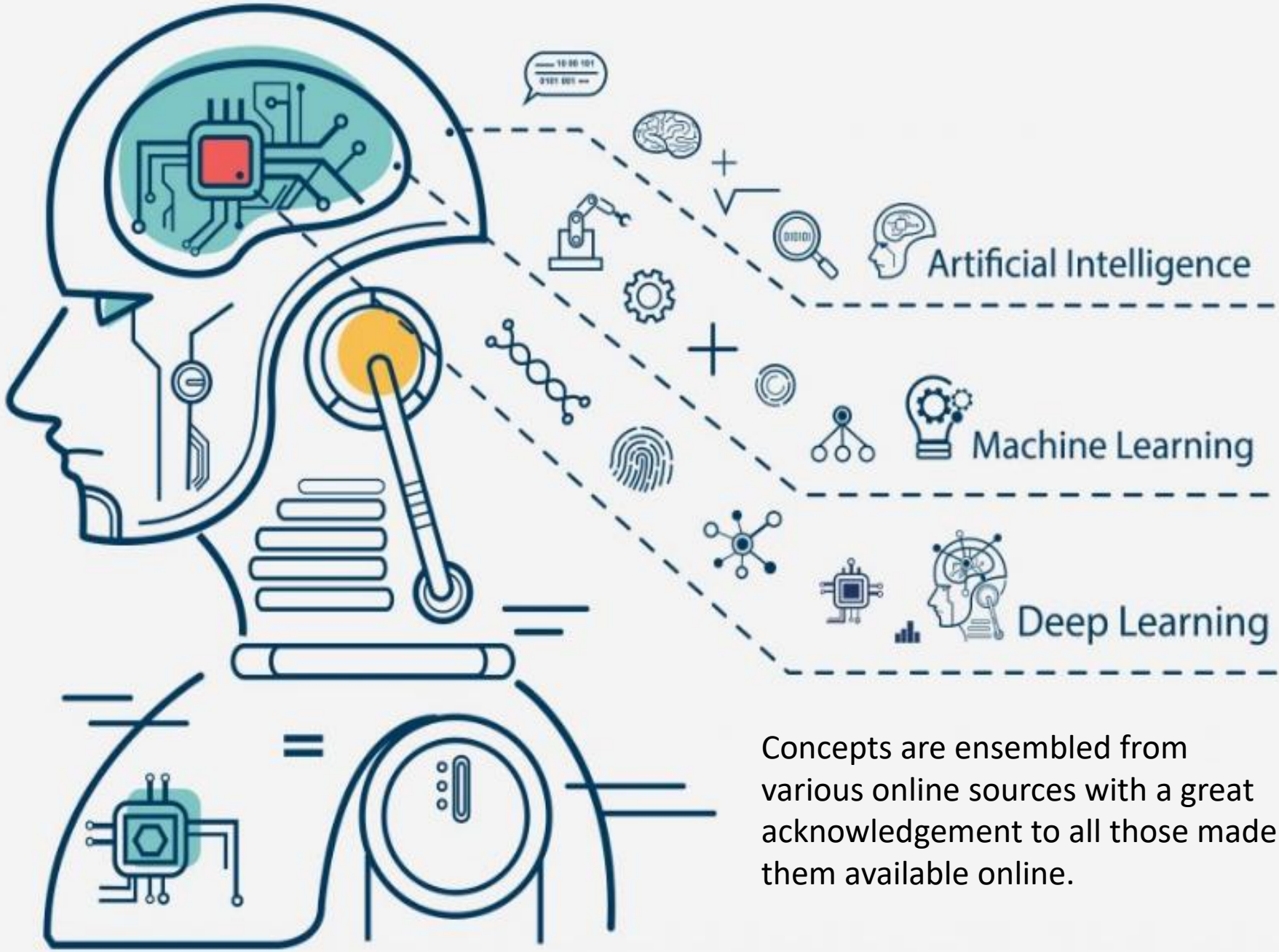
Introduction to Deep Learning and Neural Networks Basics

Venkateswara Rao Kagita
Asst. Professor
NIT Warangal



Concepts are ensembled from various online sources with a great acknowledgement to all those made them available online.

© Pantech Solutions

Overview of the talk

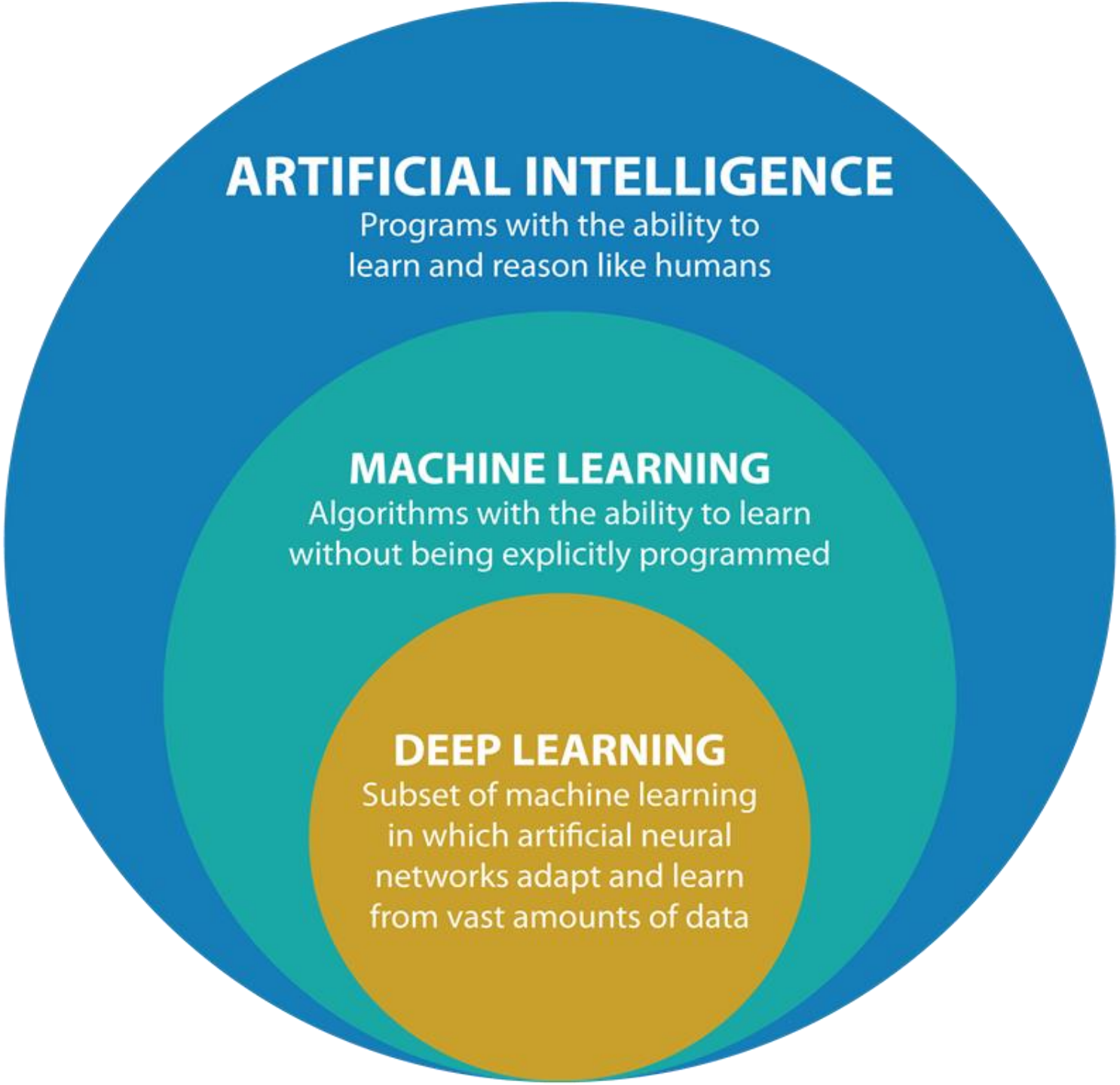
- Intelligent Decision Support Systems
- AI Vs. Machine Learning Vs. Deep Learning
- Introduction to Deep Learning
- Applications of Deep Learning
- Neural Networks basics
  - Machine Learning basics
    - Gradient descent
    - Regularization
  - Shallow neural networks
  - Deep neural networks
  - Forward propagation
  - Backward propagation

Overview of the talk

- Machine Learning models vs. CNN
- Drawbacks of Fully Connected Neural Networks !!
- Convolutional Neural Networks
  - —Convolution
  - —Padding
  - —Stride
  - —Pooling
- Classic Nets
- Application of CNN

Intelligent Decision Support Systems

- A decision support system (DSS) is an information system that supports business or organizational decision-making activities.
- An intelligent decision support system (IDSS) is a decision support system that makes extensive use of artificial intelligence (AI) techniques.
- A few examples
  - Crop-planning
  - Medical diagnosis
  - Business Intelligence
  - Weather monitoring
  - River monitoring


ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Traditional Machine Learning

- Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.

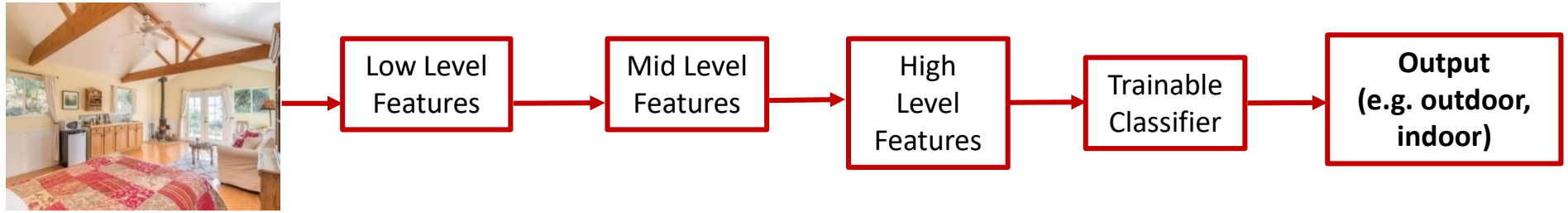
Extract Hand Crafted Features → **Trainable Classifier** (e.g. SVM, Random Forrest) → **Output** (e.g. Outdoor Yes or No)

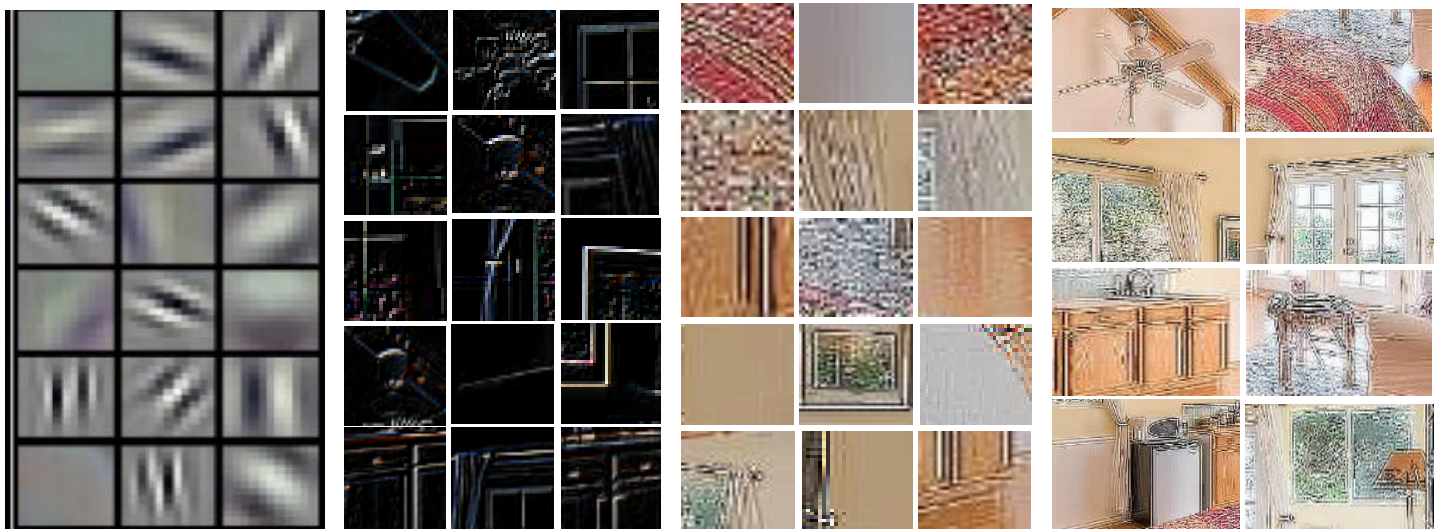
Limitations

- Very tedious and costly to develop hand crafted features.
- The hand-crafted features are usually highly depend on one application.

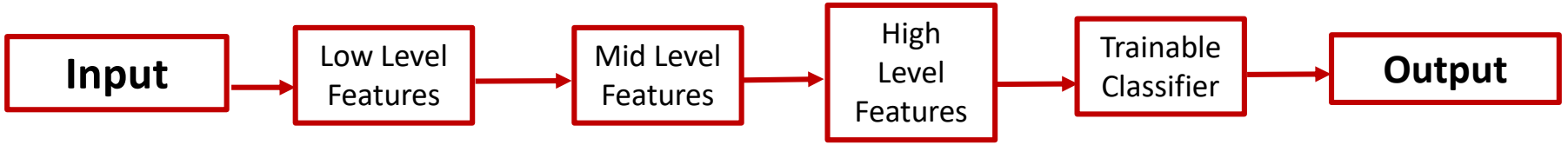
Deep Learning

- Deep learning has an **inbuilt automatic multi stage feature learning process** that learns rich hierarchical representations (i.e. features).





Deep Learning



- Image

  Pixel → Edge → Texture → Motif → Part → Object

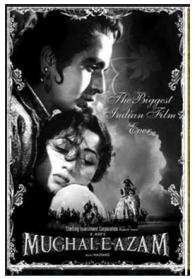
- Text

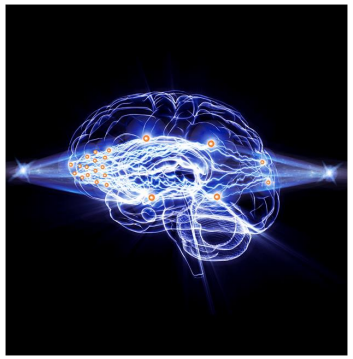
  Character → Word → Word-group → Clause → Sentence → Story

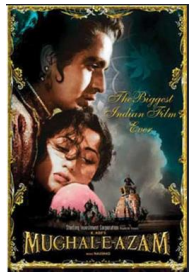
- Each module in Deep Learning transforms its input representation into a higher-level one, in a way similar to human cortex.

A Few Applications

Automatic Colorization of Black and White Images



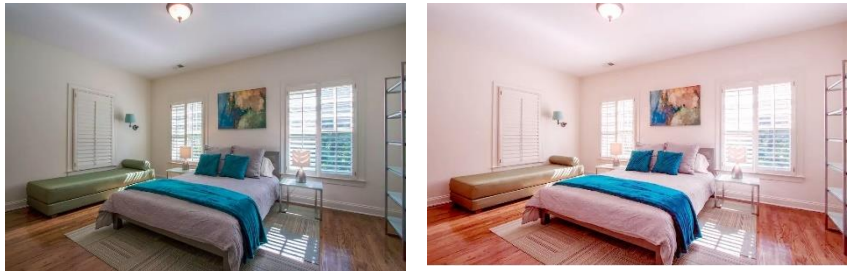




Optimizing Images

Post Processing Feature Optimization
(Color Curves and Details)

Post Processing Feature Optimization
(Illumination)

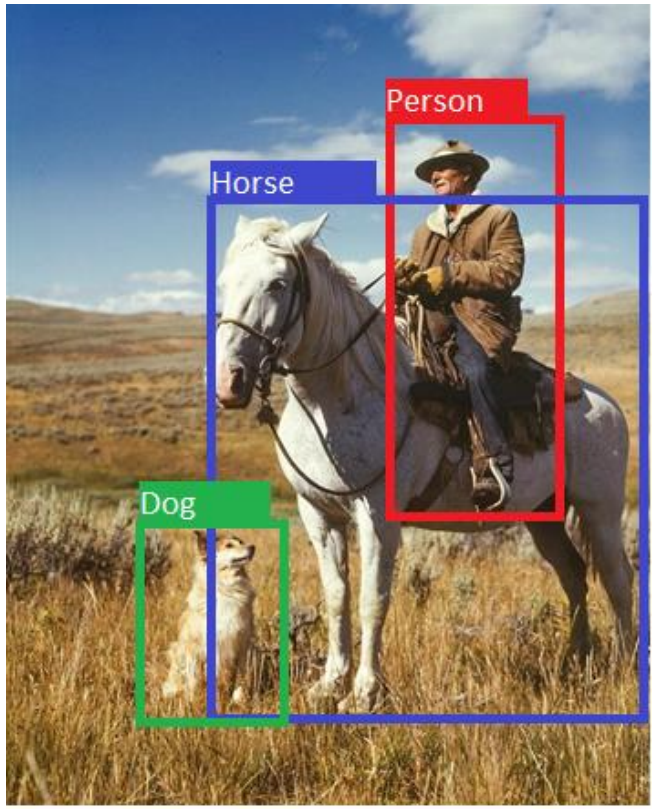


Post Processing Feature Optimization
(Color Tone: Warmness)

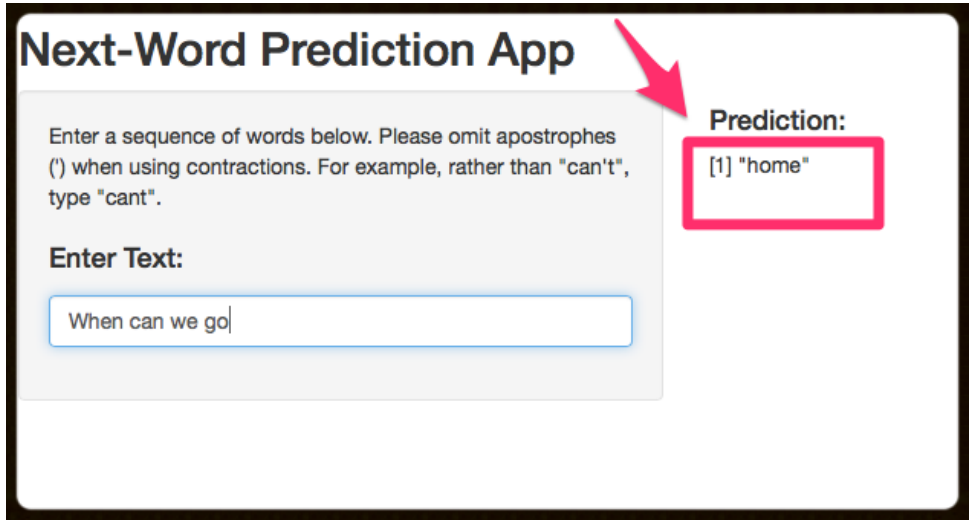
Captioning an image

Person riding a motor cycle on a dirt road

Object Detection Photographs

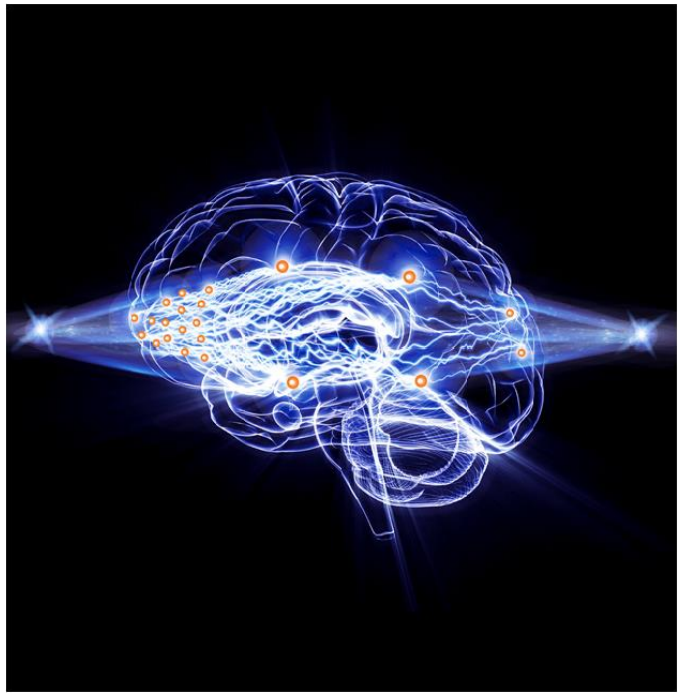


Next word prediction



Automatic Machine Translation

शुभ प्रभात

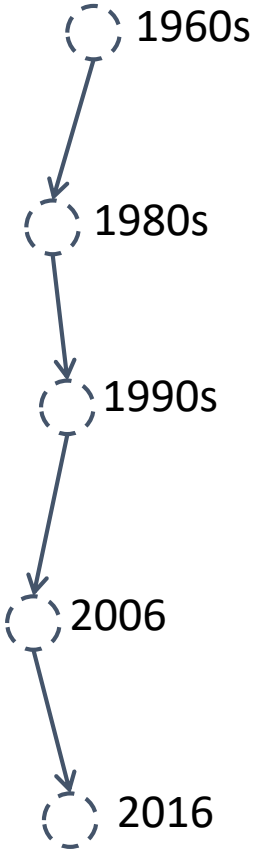


Good Morning

Why is everyone talking about Deep Learning?

Deep learning:

- Has won numerous pattern recognition competitions
- Requires minimal feature engineering



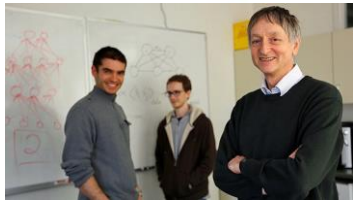
This wasn't always the case!

Since 1980s: Form of models hasn't changed much, but lots of new tricks…

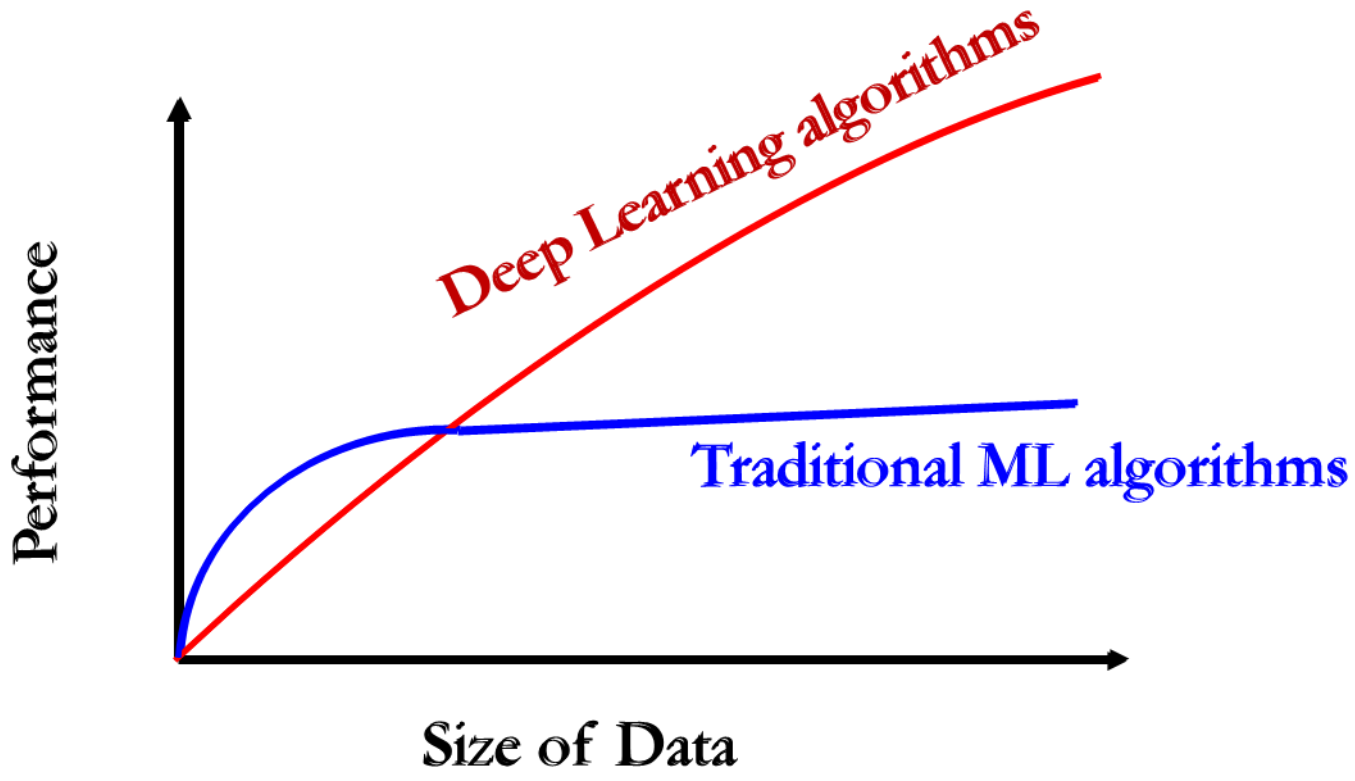
- More hidden units
- Better (online) optimization
- New nonlinear functions (ReLUs)
- Faster computers (CPUs and GPUs)

Why is everyone talking about Deep Learning?

- Because a lot of money is invested in it…
  - —DeepMind: Acquired by Google for **$400 million**
  - —DNNResearch: **Three person startup** (including Geoff Hinton) acquired by Google for unknown price tag
  - —Enlitic, Ersatz, MetaMind, Nervana, Skylab: Deep Learning startups commanding **millions of VC dollars**

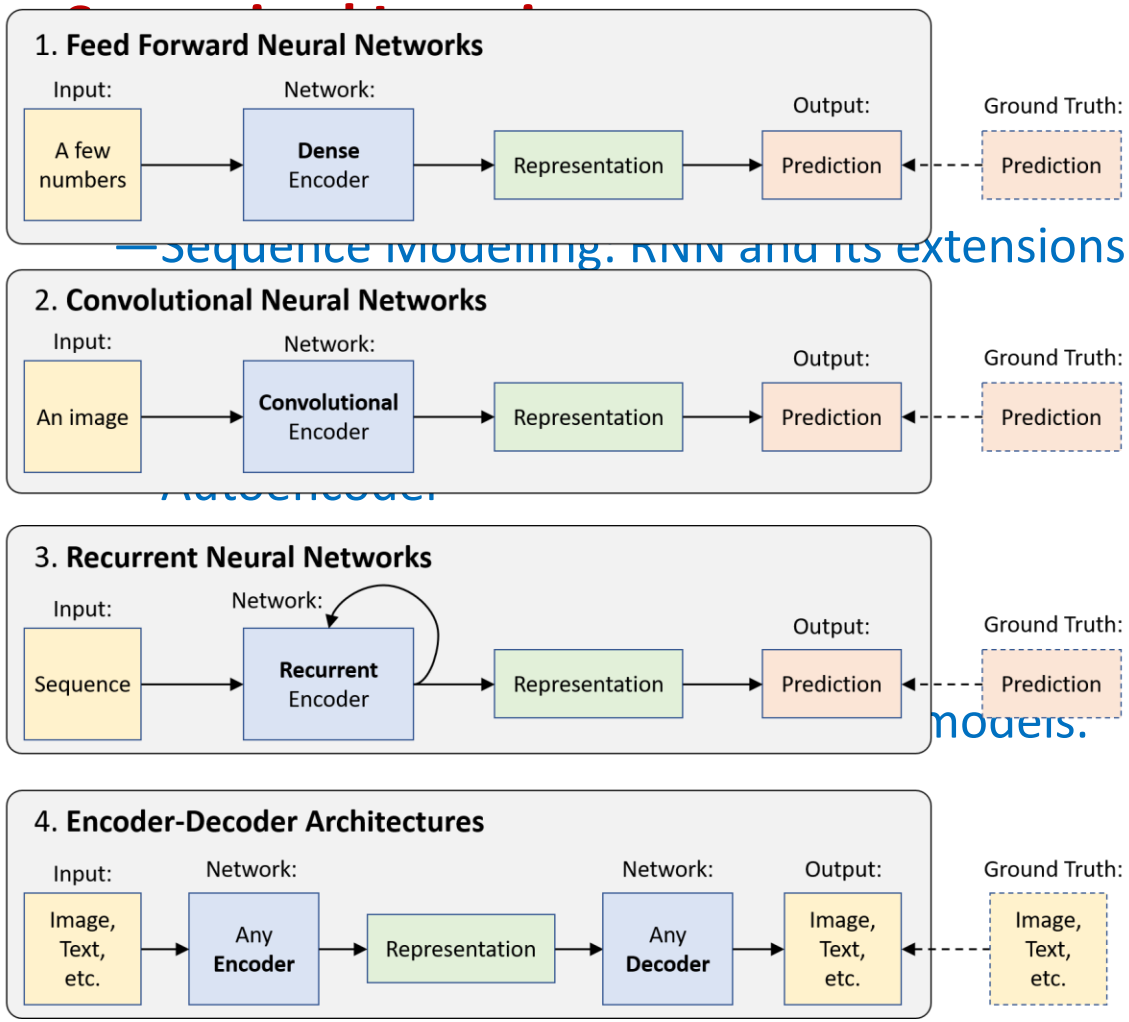


Performance vs Sample Size

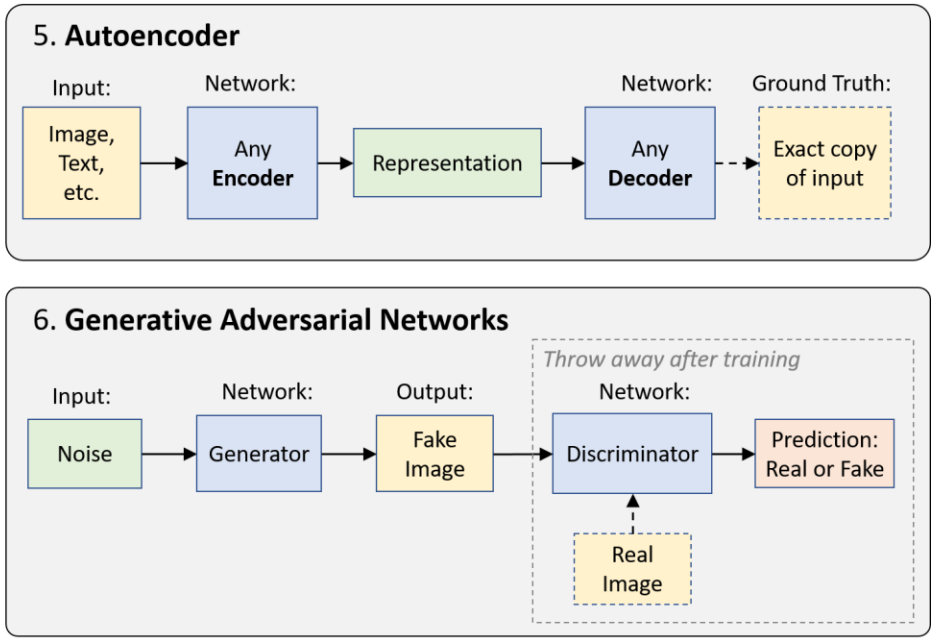


Different paradigms

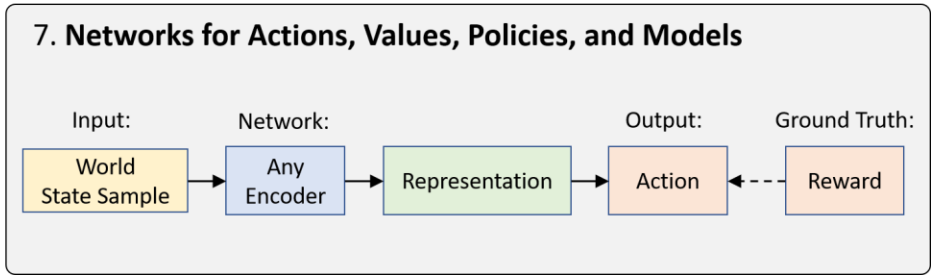
Supervised Learning



Unsupervised Learning



Reinforcement Learning



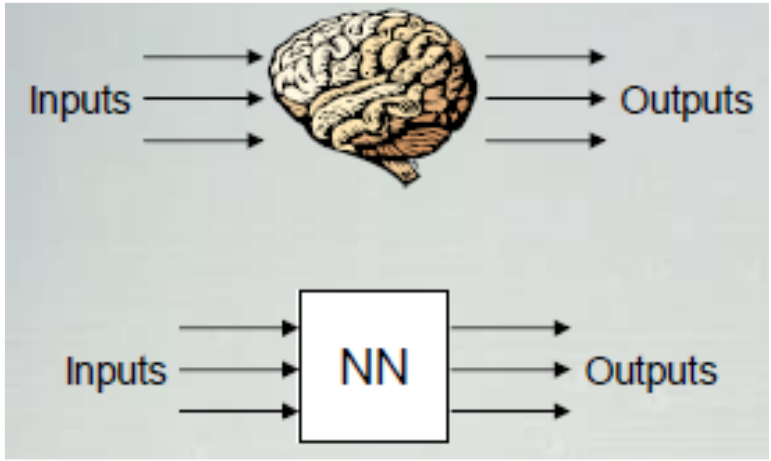
Precisely

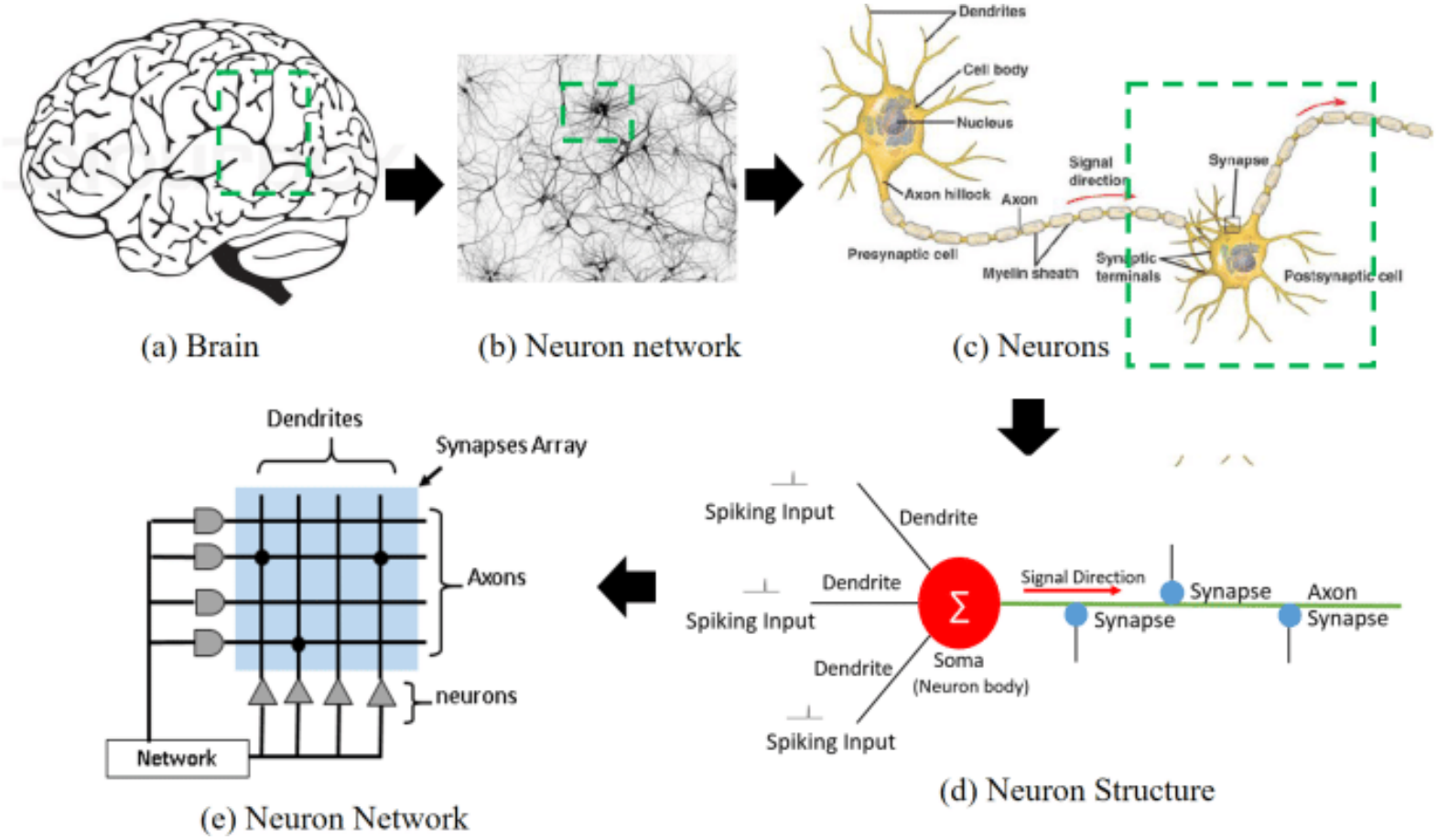
Deep learning is a branch of machine learning where **neural networks** learn from large amounts of data.

Artificial Neural Networks Basics

What Are Artificial Neural Networks?

- An **artificial neural network** is an interconnected group of nodes, inspired by a simplification of neurons in a brain.



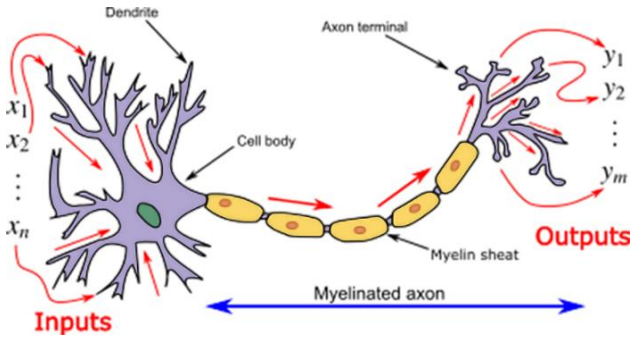


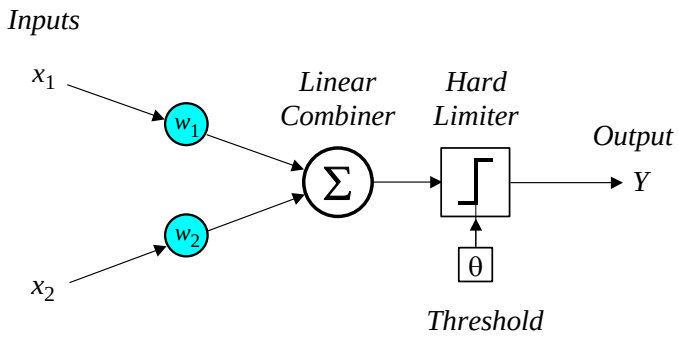
(a) Brain (b) Neuron network (c) Neurons (d) Neuron Structure (e) Neuron Network

- The original goal: to mimic human brain.
- Current attention: Performing specific tasks.
- Used on a variety of tasks,
  - Computer vision
  - Speech recognition
  - Machine translation
  - Social network filtering
  - Playing board and video games
  - Medical diagnosis, etc.

Perceptron and Neural Nets

- From biological neuron to artificial neuron (perceptron)

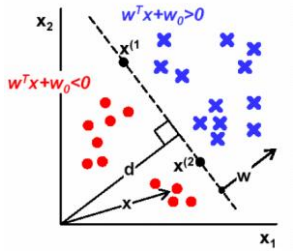




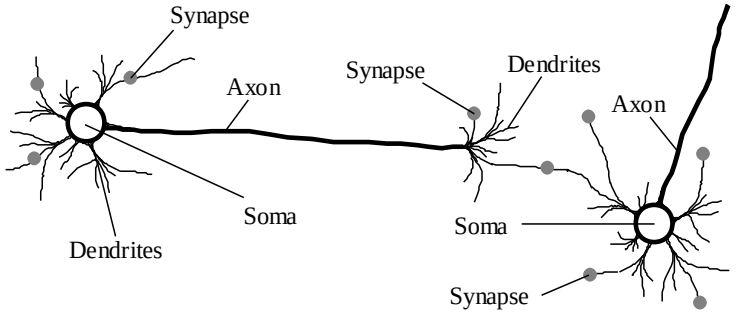
- Activation function

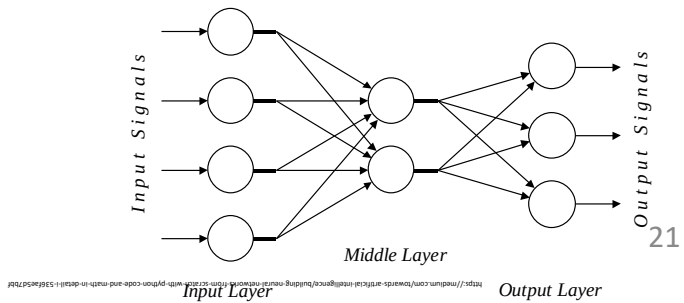
$$X = \sum_{i=1}^{n} x_i w_i$$

$$y = \begin{cases} +1, & \text{if } X \geq \omega_0 \\ -1, & \text{if } X < \omega_0 \end{cases}$$



- Neural Nets





https://medium.com/towards-artificial-intelligence/building-neural-network-from-scratch-with-python-code-and-math-in-detail-I-536e5d7bff

What are they used for ?

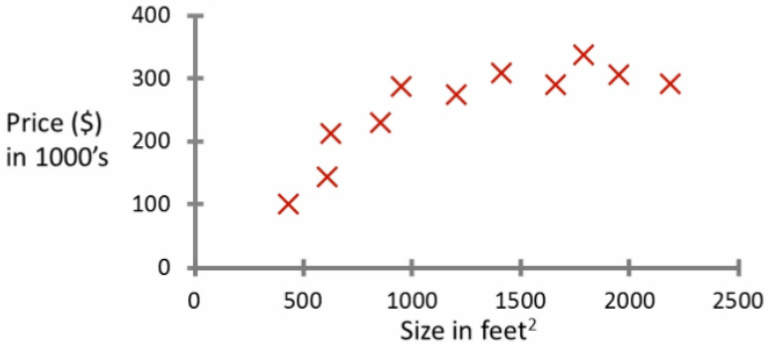
- Regression
- Classification
- Noise Reduction
  - —Recognize patterns in the inputs and produce noiseless outputs
- Prediction
  - —Extrapolation based on historical data
- …

Supervised Learning

A Machine Learning Recipe

- Given a training set of pairs $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$, where $x^{(i)} \in R^d$ and $y^{(i)}$ is a **continuous/discrete** target variable, the task is to predict for $x^{(m+j)}, j \geq 1$.
- How do we predict housing prices
  - Collect data regarding housing prices and how they relate to size in feet.

Housing price prediction.



- Choose each of these:
  - Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

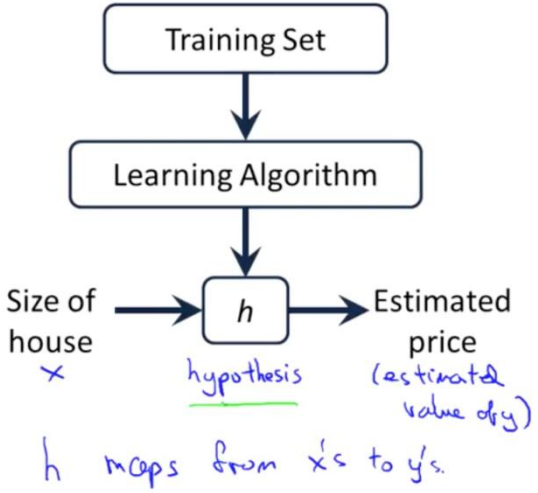
    - **Examples**: Linear regression, Logistic regression, Neural Network
  - Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

    - **Examples**: Mean-squared error, Cross Entropy
- Minimize the loss using optimization process (gradient descent)

Linear Regression

- With our training set defined - how do we use it?
  - Take training set
  - Pass into a learning algorithm
  - Algorithm outputs a function (denoted *h*)
  - This function takes an input (e.g. size of new house)
    - Tries to output the estimated value of Y
- How do we represent hypothesis *h* ?
  - Going to present *h* as $h_\theta(x) = \theta_0 + \theta_1 x$
  - Means Y is a linear function of x
  - θ_i are **parameters**

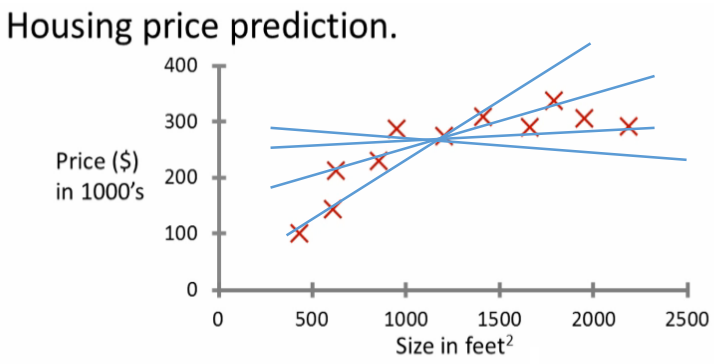


- A linear regression with one variable is also called **univariate linear regression**

- So in summary
  - A hypothesis takes in some variable
  - Uses parameters determined by a learning system
  - Outputs a prediction based on that input

Which line is best ?

- Many lines are possible !!
  Which is the best?
- A cost function lets us figure out how to fit the best straight line to our data.
- What makes a line different ?
  - Parameters θ_0, θ_1
- Which is the best line ?
  - The line that minimizes the difference between the actual and estimated prices.
- What is our objective ?
  - Choose these parameters θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples, i.e. minimize the difference between h(x) and y for each/any/every example.



Objective

- Loss function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right)^2$

- $\underset{\theta_0, \theta_1}{Minimize} \; J(\theta)$

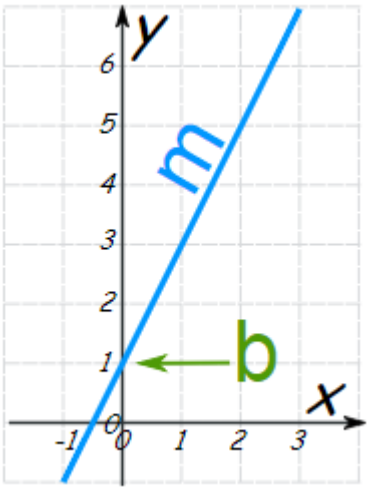
- How do we achieve this ?
  —That is where gradient descent helps us !

Gradient

$$y = mx + b$$

Slope or Gradient

y when x=0 (see Y Intercept)



y = how far up

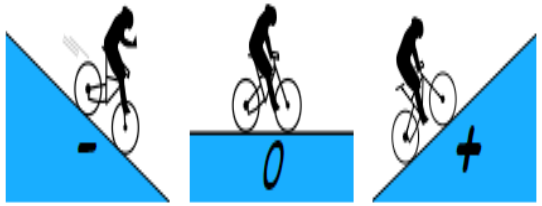
x = how far along

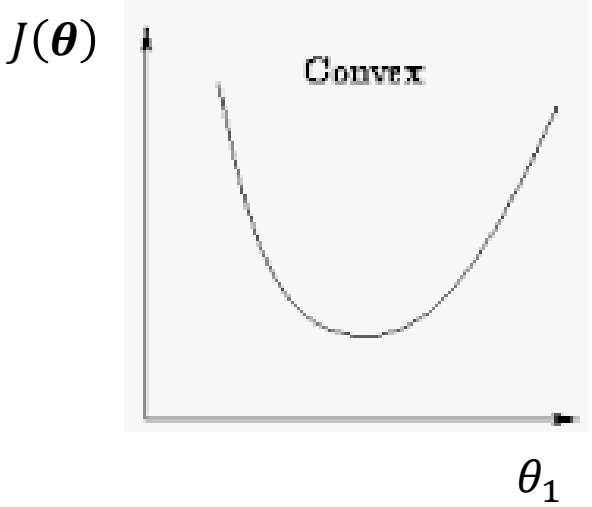
m = Slope or Gradient (how steep the line is)

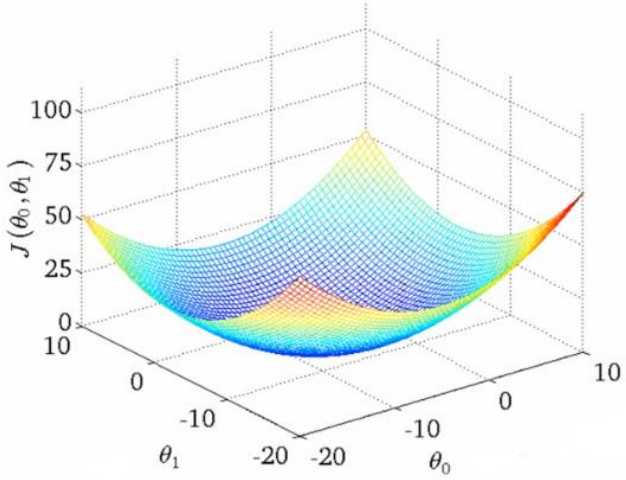
b = value of **y** when **x=0**

- The gradient is just the vector of partial derivatives

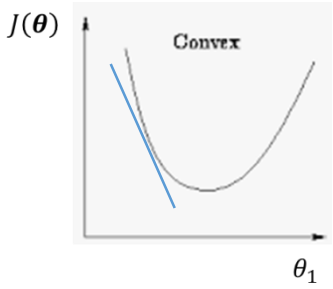
Graphical representation of a convex cost function



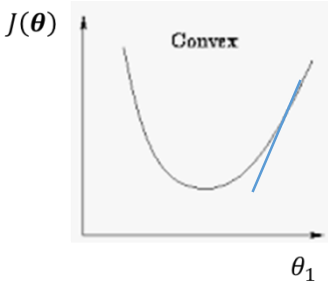




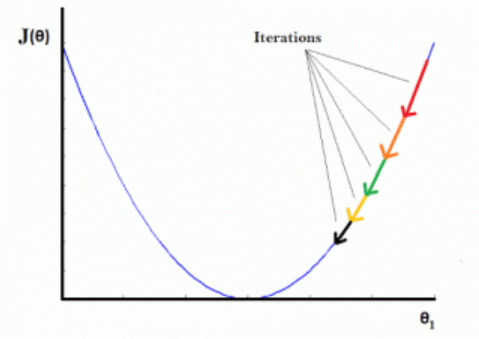
- If the slope is negative we have to increment θ_1 to reach to the minimum value of $J(\boldsymbol{\theta})$.



- If the slope is positive we have to decrement θ_1 to reach to the minimum value of $J(\boldsymbol{\theta})$.



Is this okay ?



Initialize θ_0, θ_1

- Repeat the following until convergence

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

Gradient Descent Algorithm

—That is when the gradient is negative we are incrementing θ_j and when the gradient is positive we are decrementing θ_j.

- j=0: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)$
- j=1: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) . x^i$

Multivariate Linear Regression

- Linear Regression with multiple input variables/features.

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

- Previously our hypothesis took the following form.
  - $h_\theta(x) = \theta_0 + \theta_1 x$
    - Here we have two parameters (theta 1 and theta 2) determined by our cost function
    - One variable x
- Now we have multiple features
  - $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

- Let us take $x_0 = 1$ for convenience of notation
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
  - $h_\theta(X) = \theta^T X$

Gradient descent for multiple variables

- Our cost function is

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

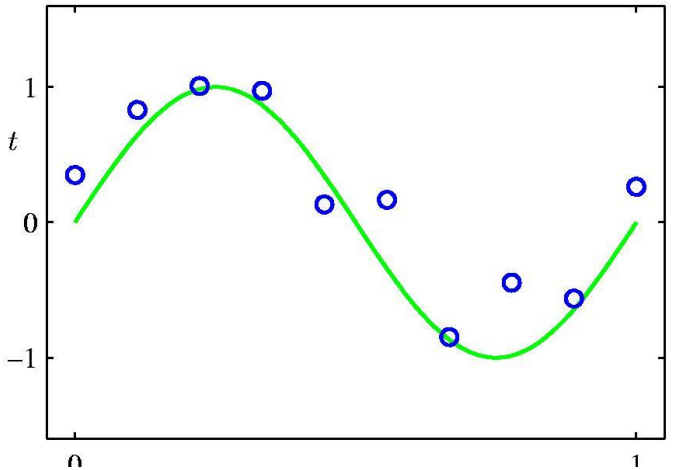
} (simultaneously update for every $j = 0, \ldots, n$)

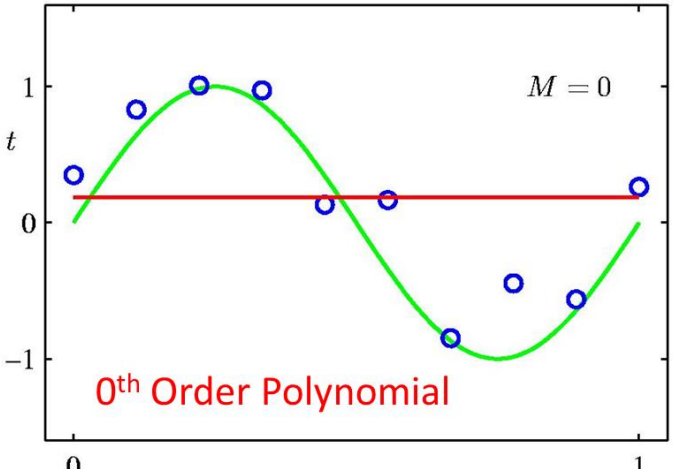
Regularization

- Penalize large coefficient values

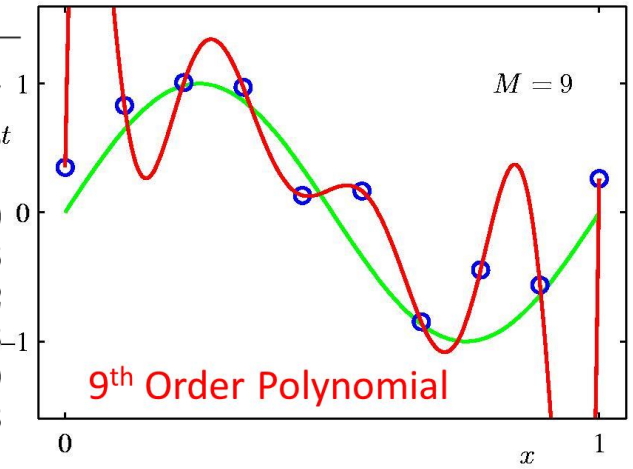
Loss function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda \, \|\theta\|_l$

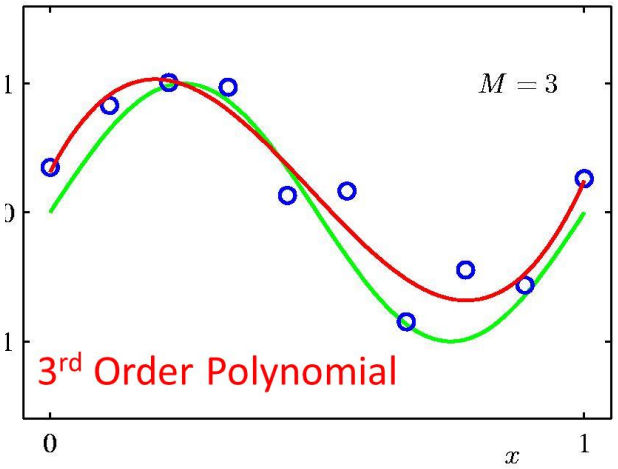
Idea: penalize high weights that contribute to high variance and sensitivity to outliers.

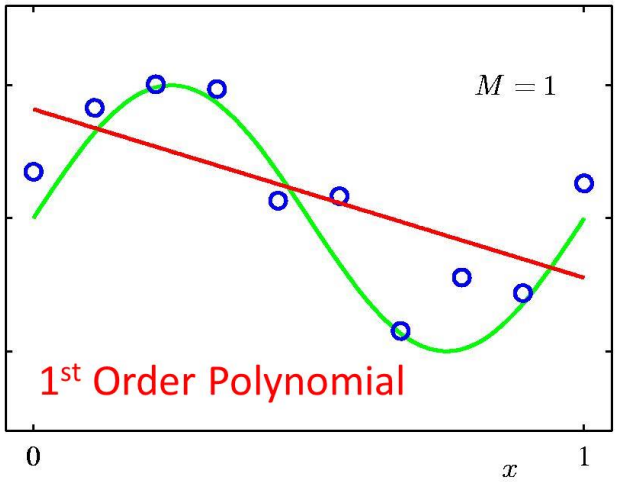




	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43





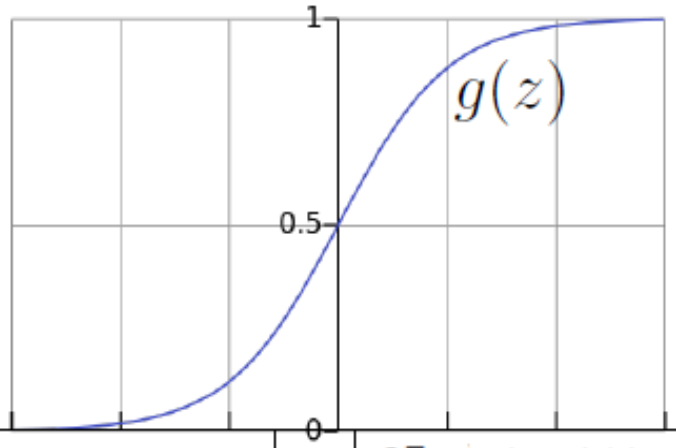


Logistic Regression for Classification

- A probabilistic approach to learn a classifier.
  - $h_\theta(x)$ should give $P(y = 1|x; \theta)$
  - $0 \leq h_\theta(x) \leq 1$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$



$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

- Assume a threshold and
  - Predict y=1 if $h_\theta(x) \geq 0.5$
  - Predict y=1 if $h_\theta(x) < 0.5$

Logistic Regression

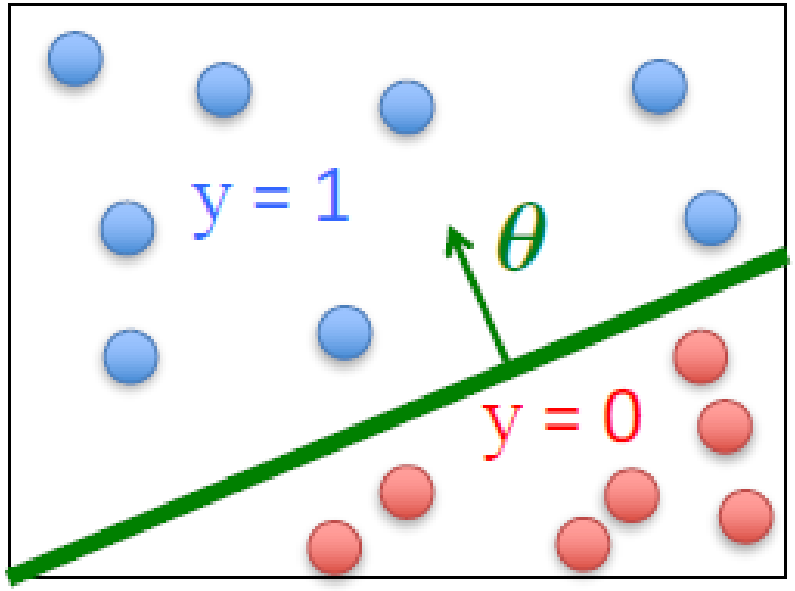
- Given a training set of pairs $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$, where $x^{(i)} \in R^d$ and $y^{(i)} \in \{0,1\}$.
- Model:

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\theta^T = [\theta_0 \ \theta_1 \ \ldots \theta_d]$$

$$\mathrm{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$



Logistic regression objective

- Can't just use squared loss as in linear regression:
  - $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right)^2$
  - Using the logistic regression model
    $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
    results in a non-convex optimization
- Logistic regression objective

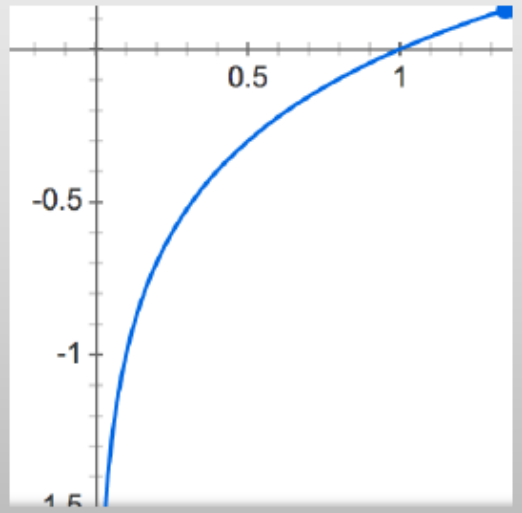
$$\min_\theta J(\theta)$$

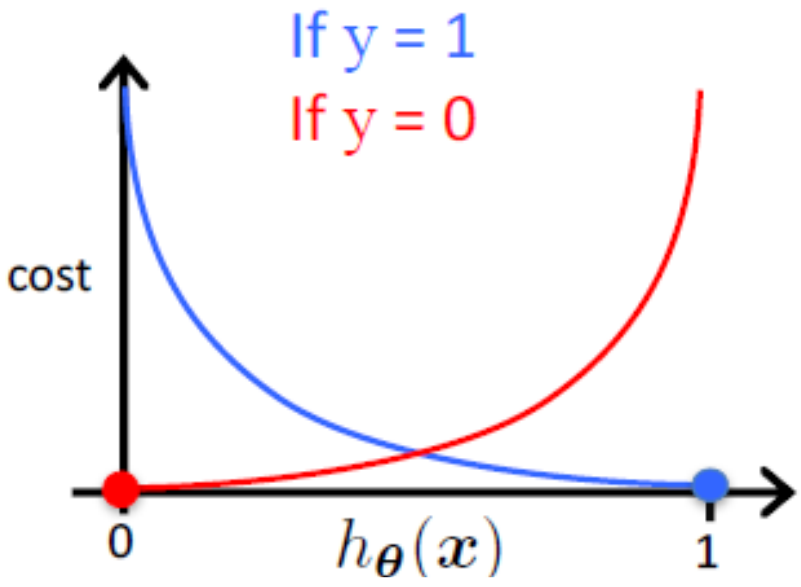
$$J(\theta) = \sum_{i=1}^{m} \left[y^i \log\, h_\theta\left(x^i\right) + \left(1 - y^i\right) \log\left(1 - h_\theta\left(x^i\right)\right)\right]$$

Intuition of the cost function

$$cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y\ = 1 \\ -\log(1 - h_\theta(x)) & if\ y\ = 0 \end{cases}$$

Aside: Recall the plot of log(z)





Regularization

- We can regularize logistic regression exactly as before:

$$J(\theta) = \sum_{i=1}^{m} \left[y^i \log\, h_\theta\left(x^i\right) + \left(1 - y^i\right) \log\left(1 - h_\theta\left(x^i\right)\right) \right] + \lambda \, \|\theta\|_l$$

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update for j = 0 ... d

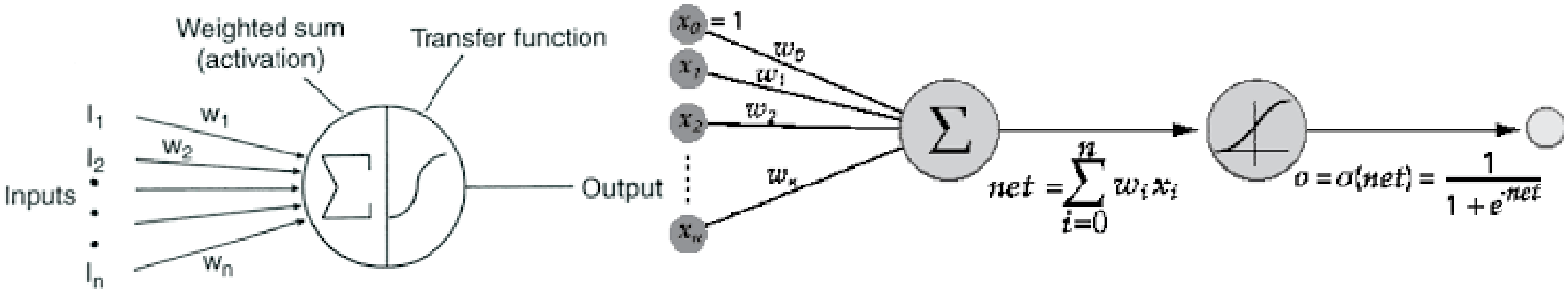
- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j \right]$$

Neuron or Perceptron

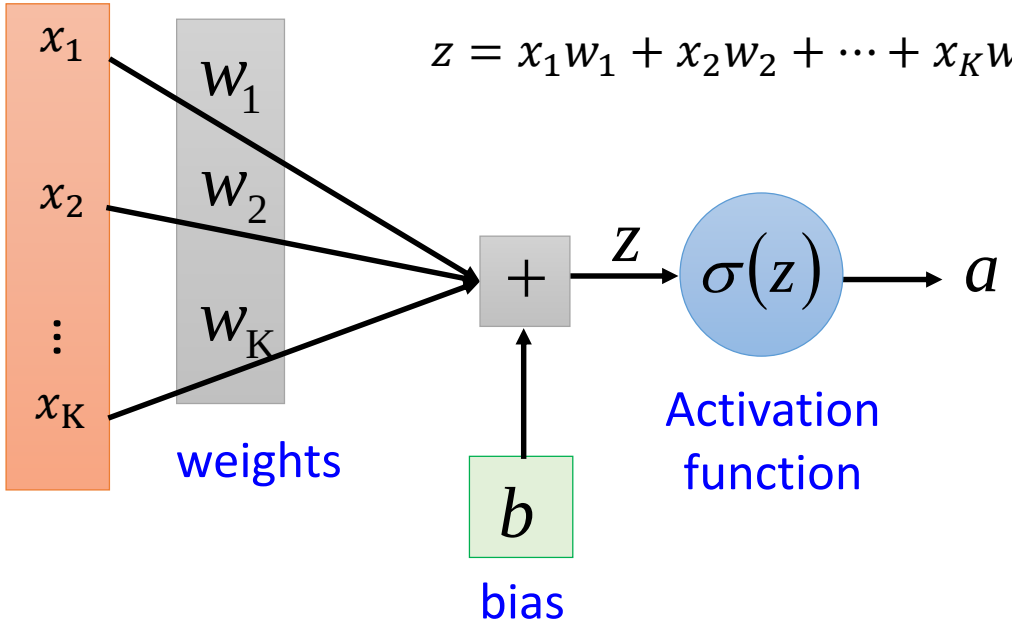
- ANN with single neuron cell is equivalent to logistic regression
- First, We will understand logistic regression with a neural network mind set.
- Then, we would explore the ANN with multiple layers and neurons.



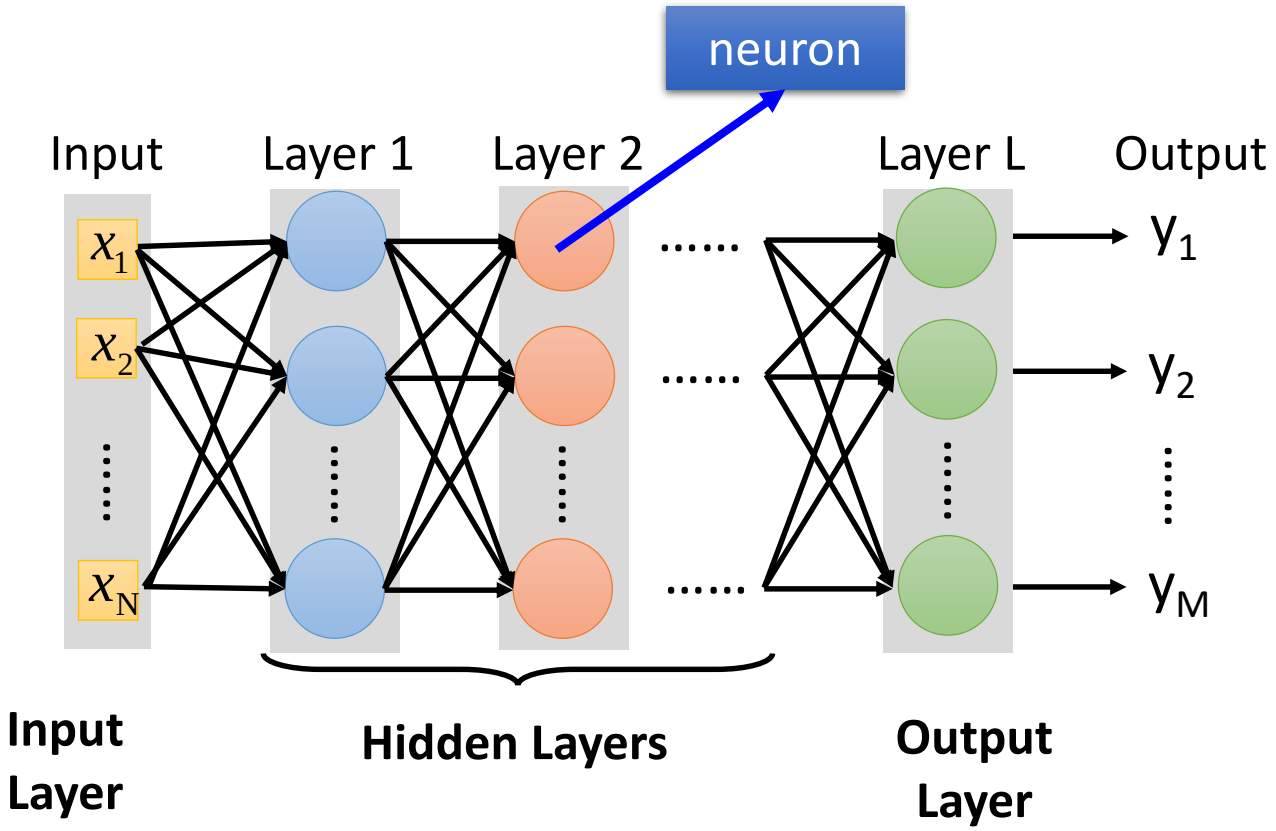
Element of Neural Network

Neuron $f: R^K \rightarrow R$

$$z = x_1 w_1 + x_2 w_2 + \cdots + x_K w_K + b$$

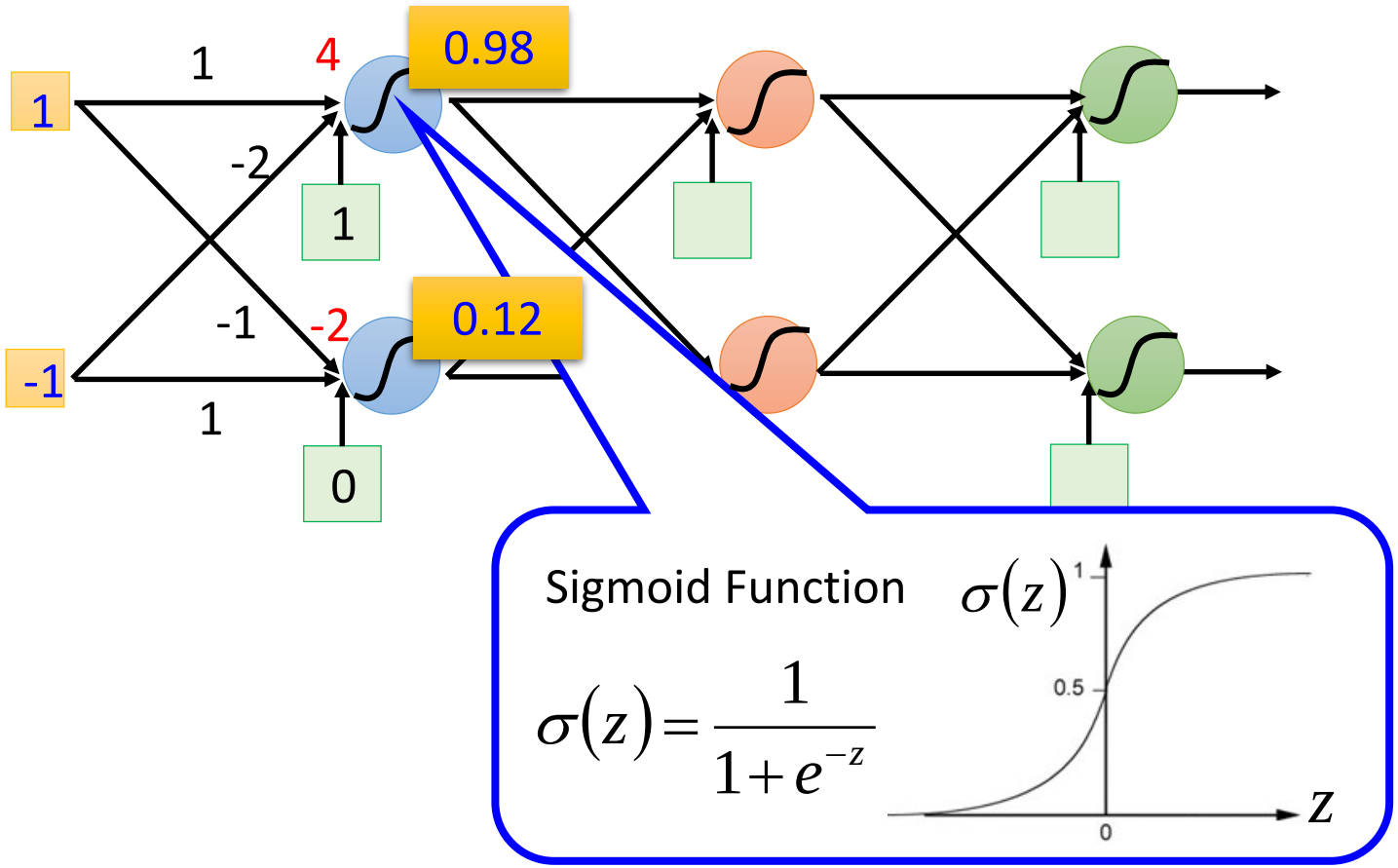


Neural Network

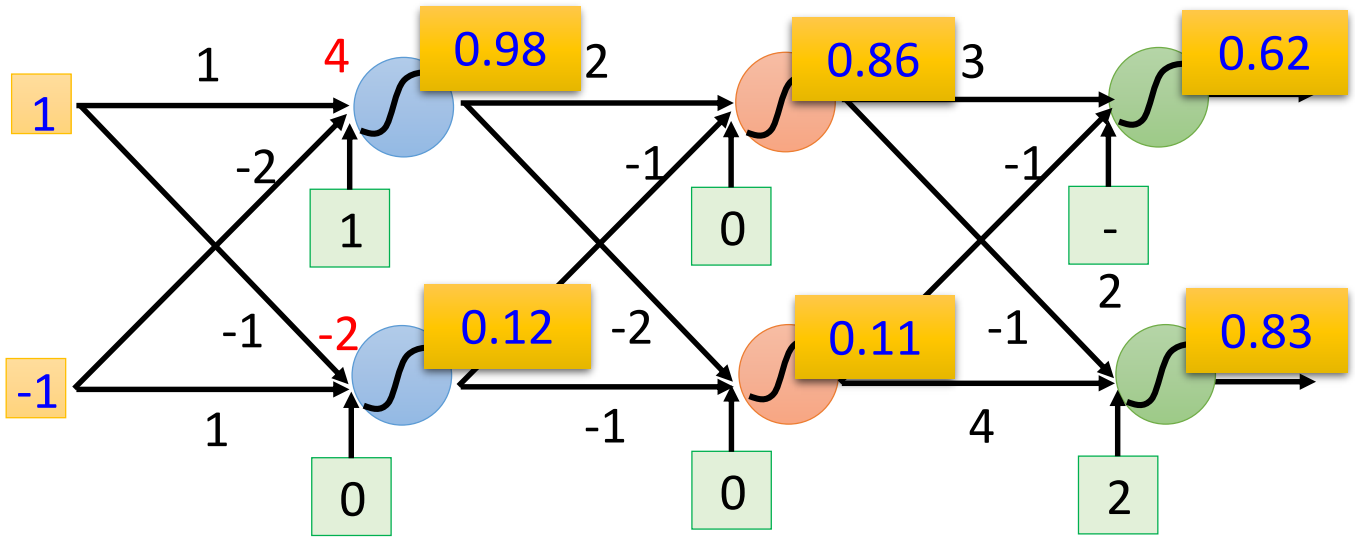


Deep means many hidden layers

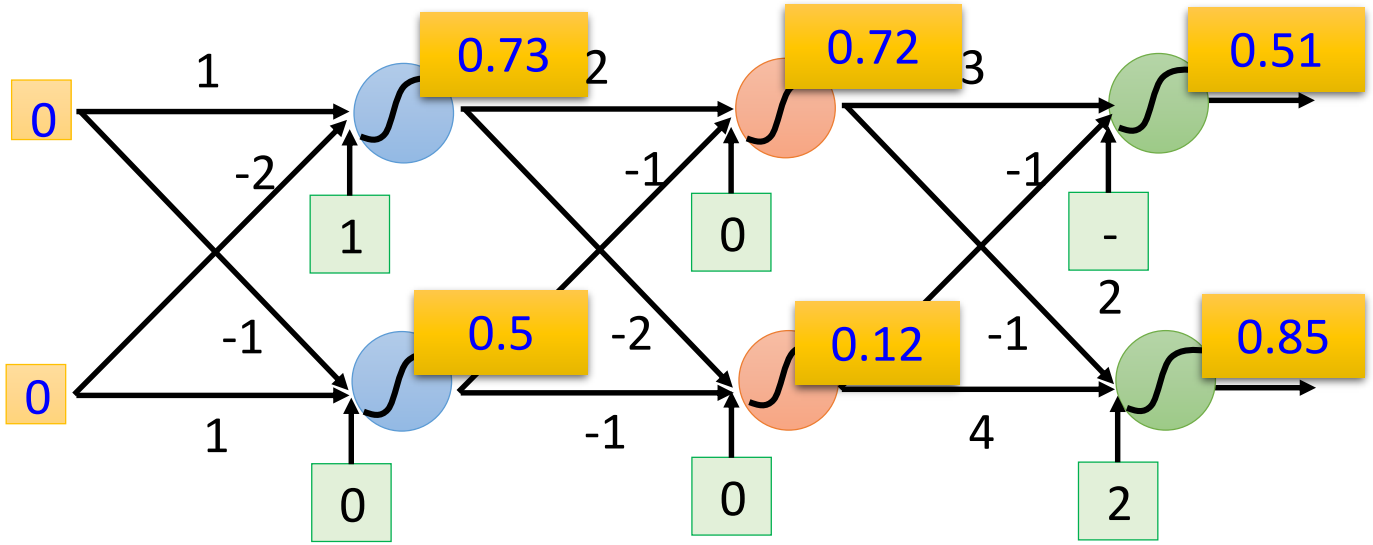
Example of Neural Network



Example of Neural Network



Example of Neural Network

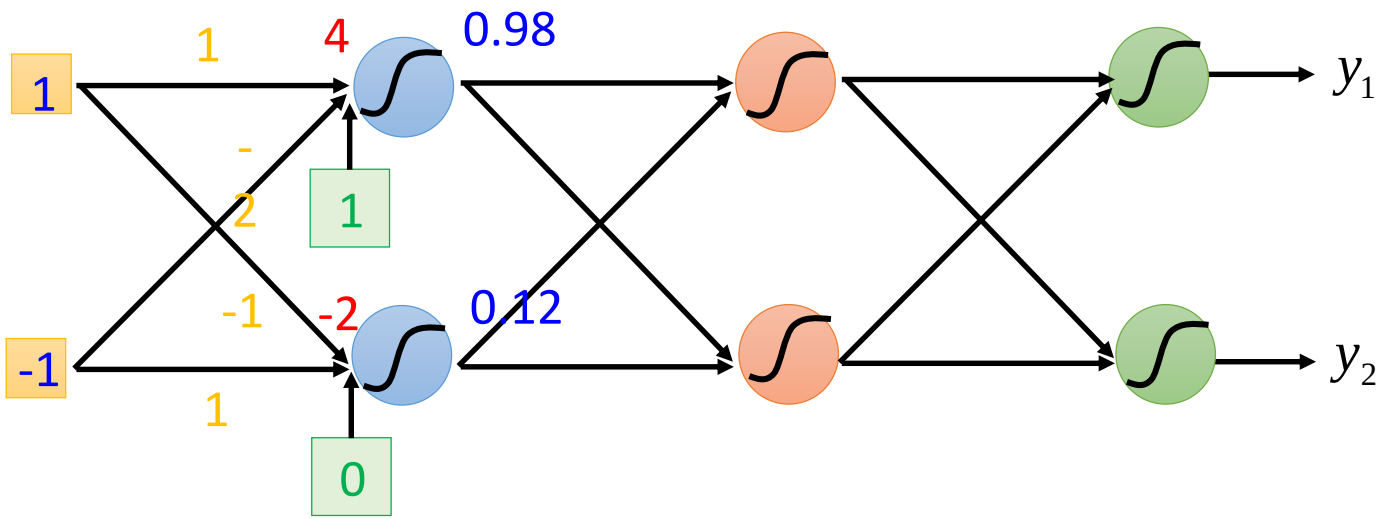


$f: R^2 \rightarrow R^2$

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}0.51\\0.85\end{bmatrix}$$

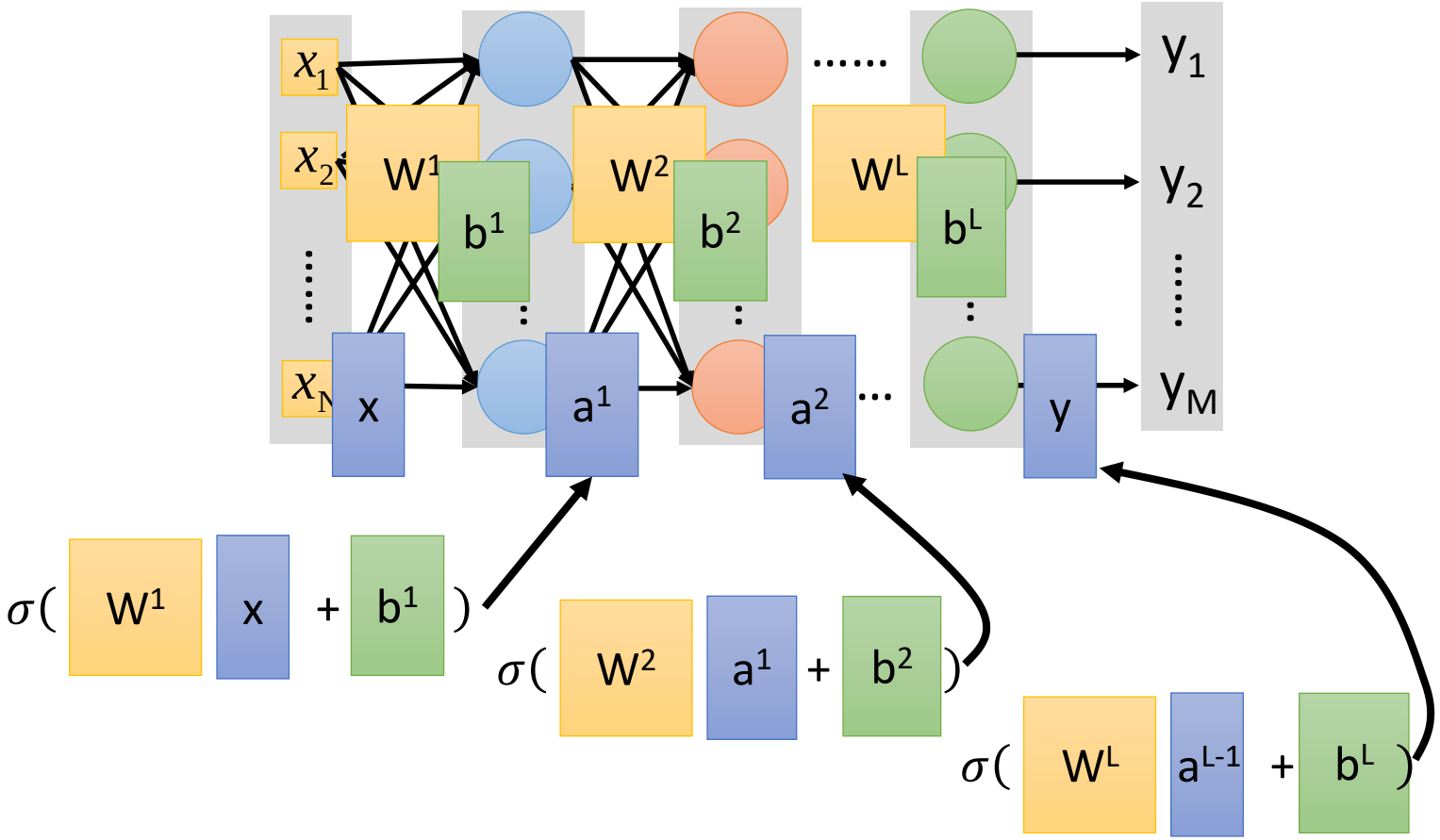
Different parameters define different function

Matrix Operation



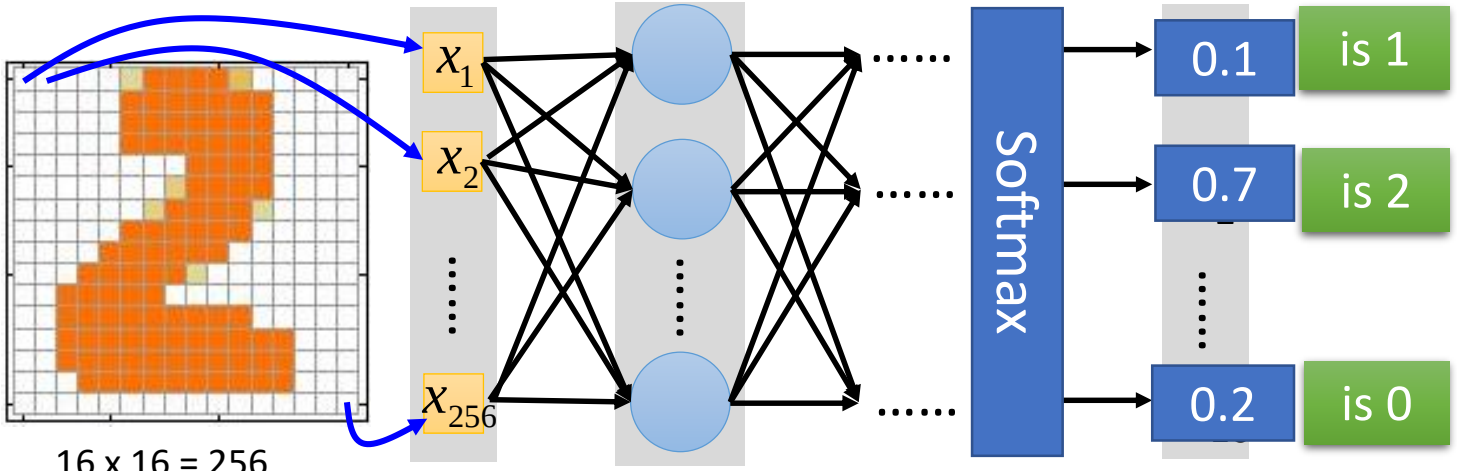
$$\sigma\Big(\underbrace{\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}}_{\begin{bmatrix}4\\ -2\end{bmatrix}}\Big) = \begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

Neural Network



How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

Ink → 1

No ink → 0

Set the network parameters θ such that .....

Input ... m value

Input ... m value

How to let the neural network achieve this

Why do we need neural networks ?

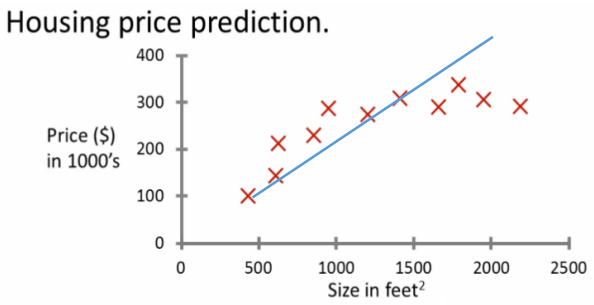
- We have linear regression and logistic regression, why do we need neural networks at all ?
- Lets look at how linear regression and logistic regression learns complex non linear hypothesis.

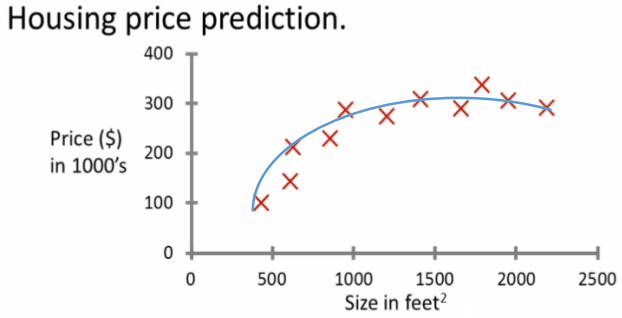
—Linear Regression: $h_\theta(x) = \theta_0 + \theta_1 x$

—Second order polynomial: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

—Third order polynomial:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$





Non Linear hypothesis

- Suppose if we have two features x_1 and x_2.
  - —Linear regression or first order polynomial:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

  - —Second order polynomial:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2$$

  - —Third order

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \theta_6 x_1^3 + \theta_7 x_2^3 + \theta_8 x_1^2 x_2 + \theta_9 x_1 x_2^2$$

- Similar is the case with logistic regression.
- What if we have 100 features ?
  - —How many features for second order polynomial ?

Non-linear classification example: XOR/XNOR

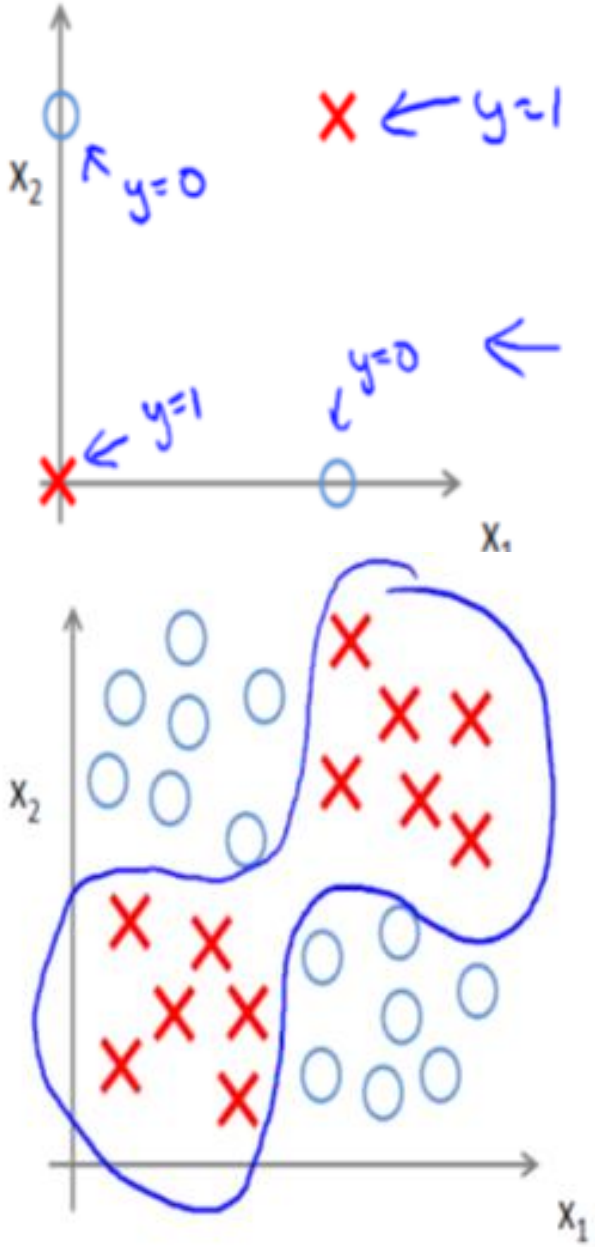
- X_1, X_2 are binary (0 or 1).

- XOR

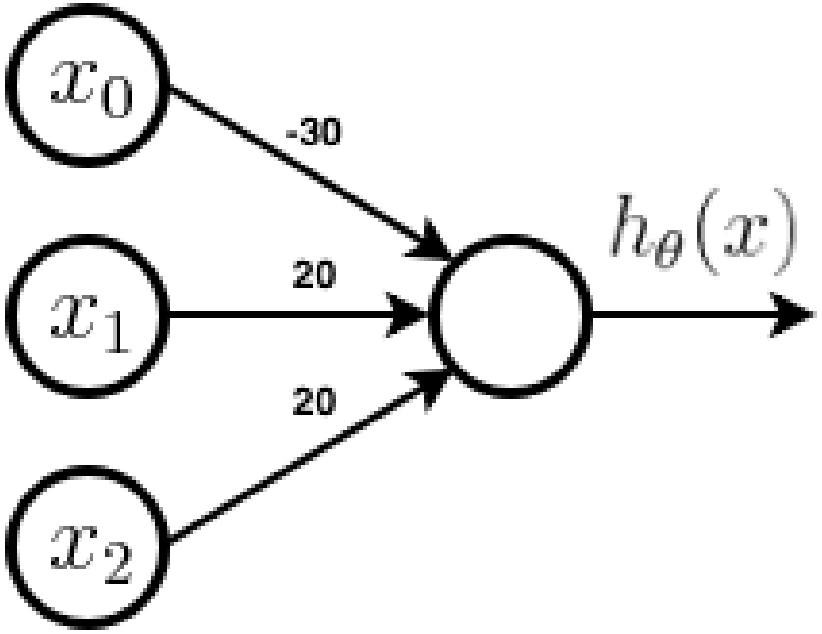
Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

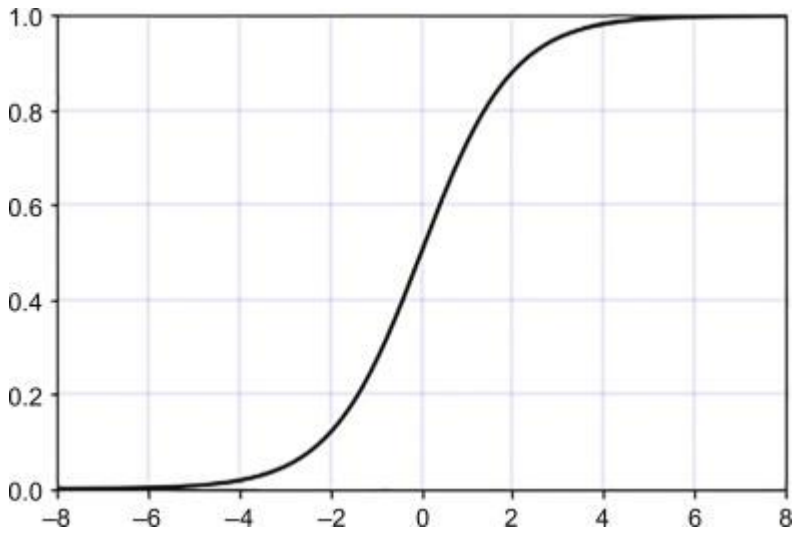
XNOR

INPUT		OUTPUT
IN1	IN2	OUT
0	0	1
0	1	0
1	0	0
1	1	1



Simple Example: AND

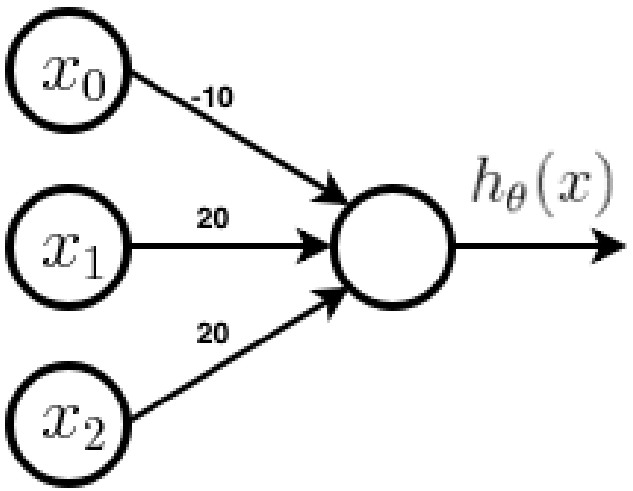


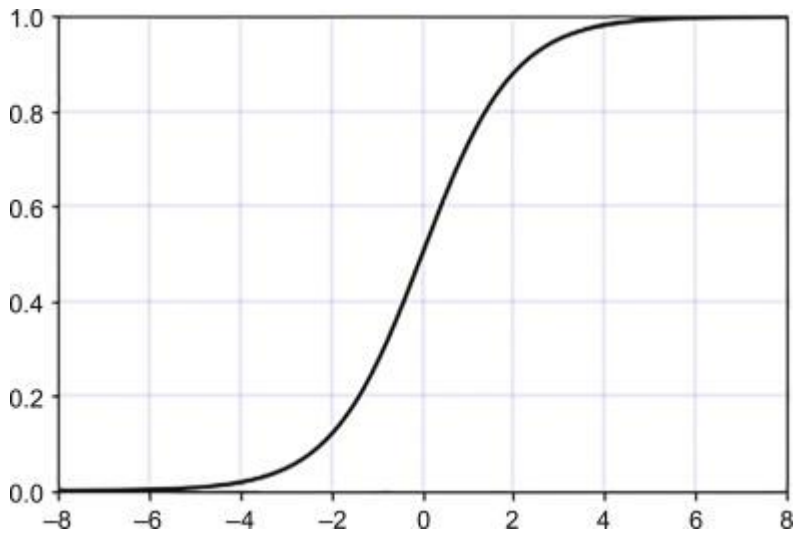


$$h_\theta(x) = g(-30 + 20\,x_1 + 20\,x_2)$$

x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

OR GATE with perceptron

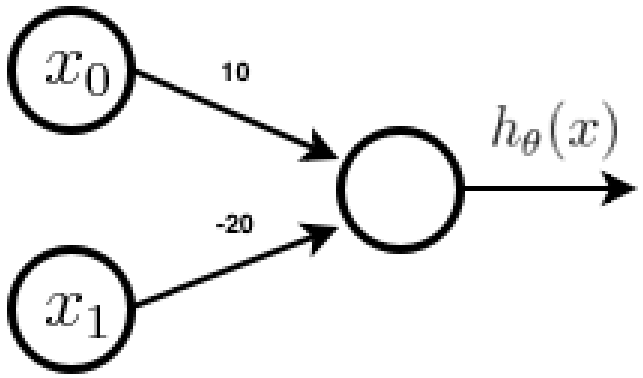




$$h_\theta(x) = g(-10 + 20\,x_1 + 20\,x_2)$$

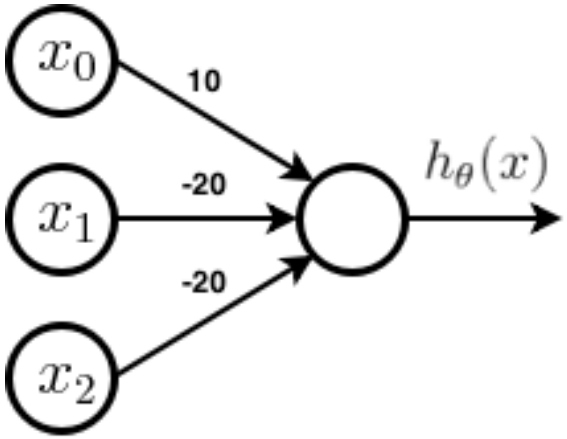
x_1	x_2	$h_\theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	$g(10) \approx 1$
1	1	$g(30) \approx 1$

NOT, (NOT x1) AND (NOT x2)



$$h_\theta(x) = g(10 - 20\,x_1)$$

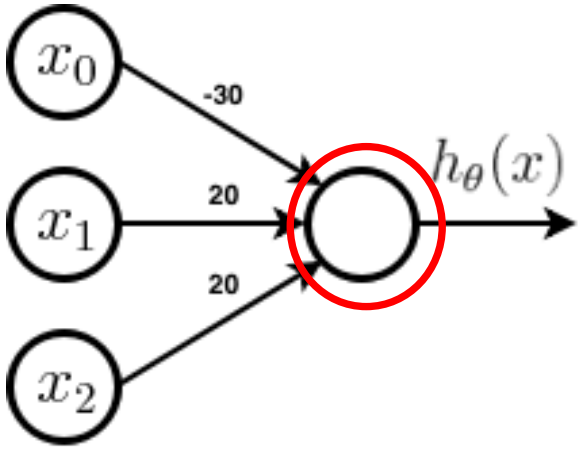
x_1	$h_\theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$



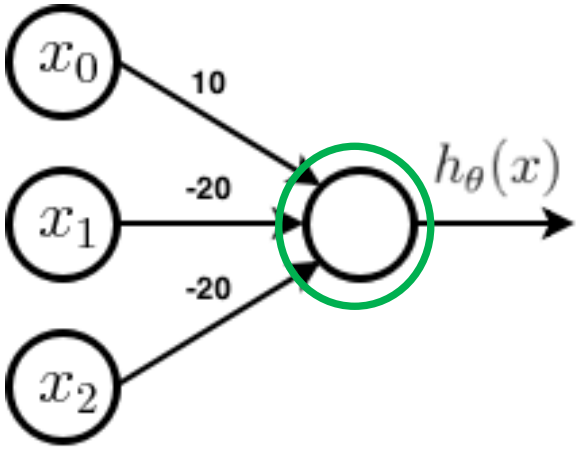
$$h_\theta(x) = g(10 - 20\,x_1 - 20\,x_2)$$

x_1	x_2	$h_\theta(x)$
0	0	$g(10) \approx 1$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(-30) \approx 0$

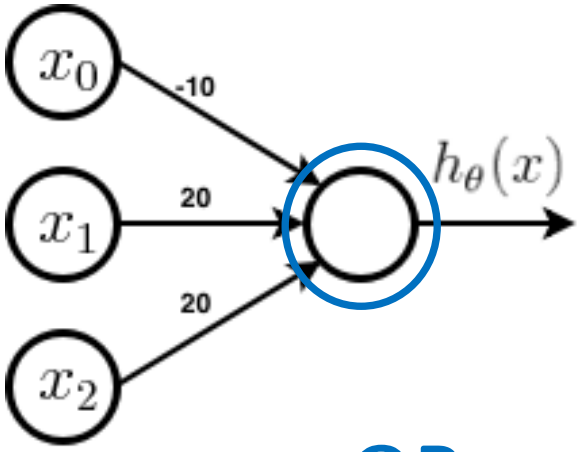
Putting it together: x1 XNOR x2



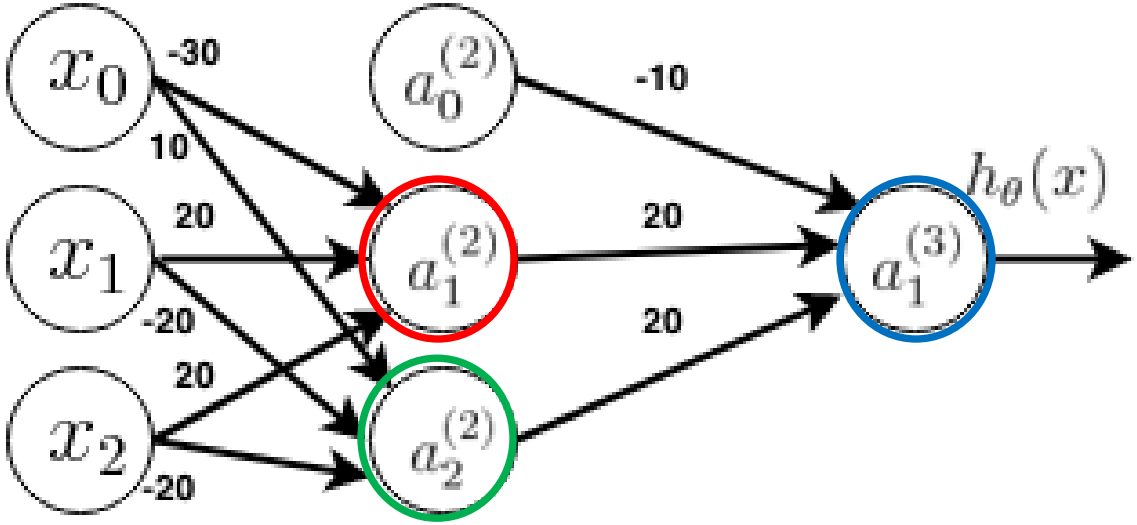
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

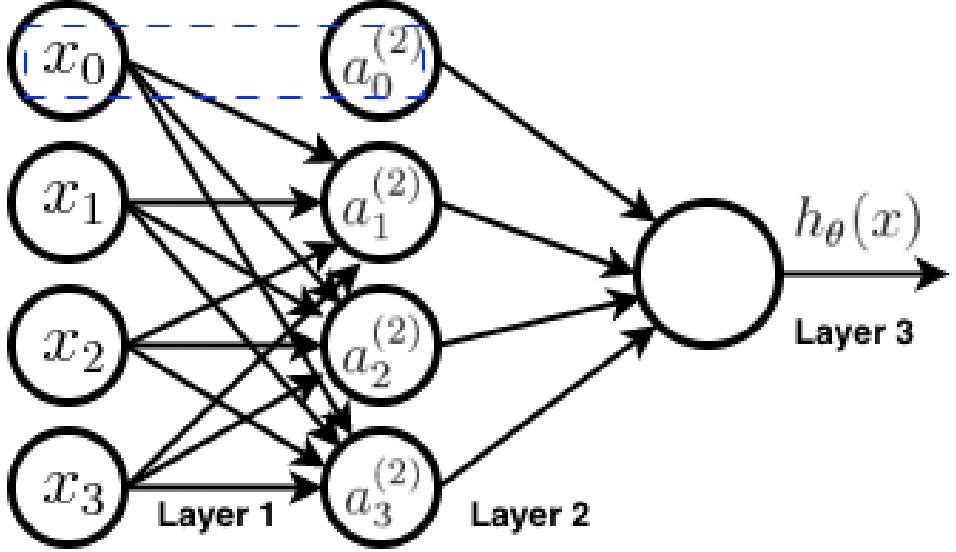


x_1 OR x_2



x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Notations



- Network has 3 layers
  - —Layer 1: Input and Layer 3: output
- x_0 and $a_0^{(2)}$ are the bias terms.
- Layer 1 and 2 have three units each.
- Notation:
  - —$a_i^{(j)}$ is the activation of unit i in layer j.
  - —$z_i^{(j)}$ is the net sum at unit i in layer j i.e. $a_i^{(j)} = g(z_i^{(j)})$
  - —$\Theta^{(j)}$ is the matrix of weights controlling function mapping from layer j to layer j+1.
  - —$\Theta_{ik}^{(j)}$ represents edge weight between unit ***i*** in layer (j+1) and unit k in layer (j).

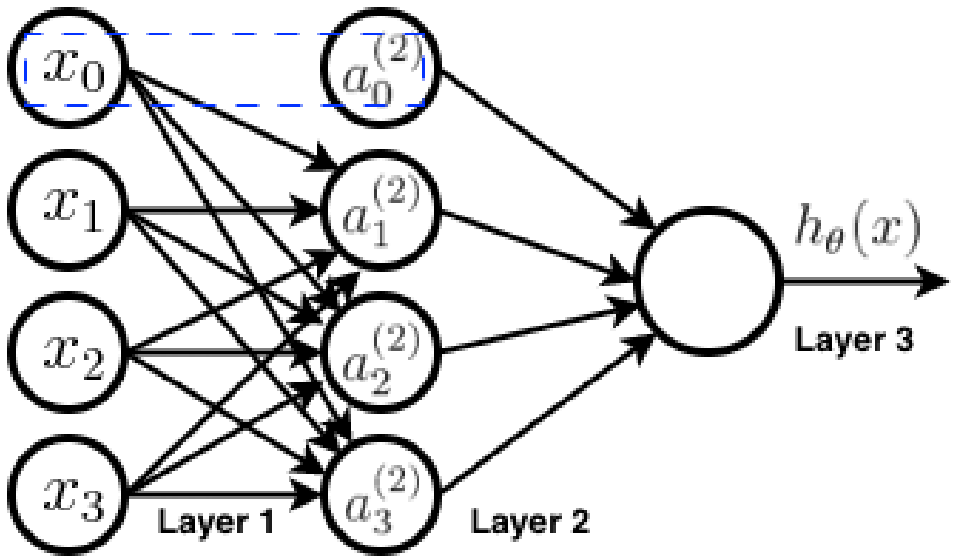
Forward Propagation

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$



- From the equation above one can generalize, **if a network has s_j units in layer j and s_{j+1} units in layer j+1, then $\Theta^{(j)}$ is a matrix of dimension $(s_{j+1} * (s_j+1))$.**

Forward propagation: Vectorization

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$\boxed{\begin{aligned} a_1^{(2)} &= g(z_1^{(2)}) \\ a_2^{(2)} &= g(z_2^{(2)}) \\ a_3^{(2)} &= g(z_3^{(2)}) \end{aligned}}$$

Where $z_i^{(j)} = \Theta_{i0}^{(j-1)} x_0 + \Theta_{i1}^{(j-1)} x_1 + \Theta_{i2}^{(j-1)} x_2 + \Theta_{i3}^{(j-1)} x_3$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ \vdots \\ z_n^{(j)} \end{bmatrix}$$

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$
$$a^{(j)} = g(z^{(j)})$$

$$a^{(1)} = x$$

$\Theta^{(j-1)}$ is a matrix of dimensions $(s_j * (s_{j-1} + 1))$

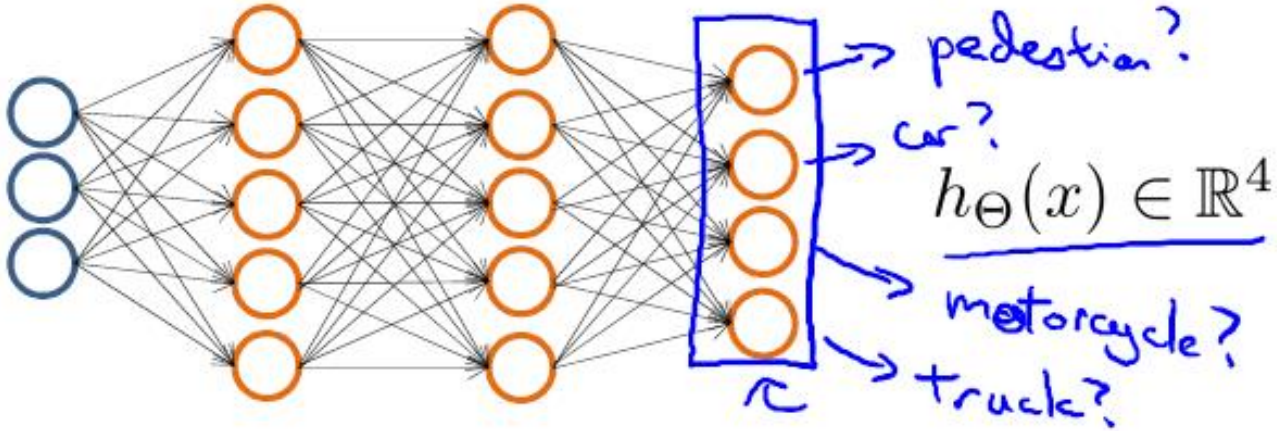
Multi-class classification

Pedestrian

Car

Motorcycle

Truck



$$h_\Theta(x) \in \mathbb{R}^4$$

Want $\boxed{h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix},}$ $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Cost Function

- Cost function of logistic regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\,log(h_\theta(x^{(i)})) + (1-y^{(i)})\,log(1-h_\theta(x^{(i)}))\right) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- Cost function of NN
  - —L no. of layers
  - —K no. of classes
  - —s_l no. of units in layer l

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\,log(h_\theta(x^{(i)}))_k + (1-y_k^{(i)})\,log(1-(h_\theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\theta_{ji}^{(l)})^2 \quad (2)$$

Learning weights

- Brute-force approach
  - —Compute gradient w.r.t. each weight
  - —Apply gradient descent to update the weights
  - —Time intensive
  - —Redundant calculations

- Solution: Back propagation Algorithm
  - —Kind of dynamic programming
  - —Avoids redundant computation

Forward propagation

$$
\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \qquad (add\, a_0^{(2)}) \\
&\vdots \\
a^{(L-1)} &= g(z^{(L-1)}) \qquad (add\, a_0^{(L-1)}) \\
z^{(L)} &= \Theta^{(L-1)} a^{(L-1)} \\
a^{(L)} &= h_\Theta(x) = g(z^{(L)})
\end{aligned}
$$

Back propagation

- The $h_\Theta(x)$ or a^L is the prediction.
- In order to reduce the error between the prediction and the actual value backpropagation is used.
- Say, $\delta_j^{(l)}$ is the **error of node j in the layer l** is associated with the prediction made at that node given by $a_j^{(l)}$.
- Backpropagation aims to calculate this error term propating backwards starting from the output unit in the last layer.
- So for each output unit in layer L, the error term is given by, $\delta_j^{(L)} = a_j^{(L)} - y_j$ which can be vectorized and written as,

$$\delta^{(L)} = a^L - y$$

Back propagation

- Error at layer L is $\delta^{(L)} = a^{(L)} - y$

- Now the error terms for the previous layers are calculated as follows.
  - $\delta^{(l)} = (\Theta^{(l)})^T \delta^{(l+1)} .* g'(z^{(l)})$
  - $where\ g'\left(z^{(l)}\right) = \mathrm{a}^{(l)} .* \left(1 - \mathrm{a}^{(l)}\right)$
- ***This backward propagation stops at l = 2. why ?***
- Now the gradient for the cost function which is needed for the minimization of the cost function is given by

$$\frac{\partial J}{\partial \Theta_{ij}^{(l)}} = \mathrm{a}_j^{(l)} \delta_i^{(l+1)}$$

Back propagation Algorithm

Given training set $(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})$

- Set $\Delta_{ij}^{l} = 0$ for all (i, j, l)
- For i = 1 to m:
  - Set $a^{(1)} = x^{(i)}$
  - Perform forward propagation to compute $a^{(l)}$ for l = 1, ..., L
  - Using $y^{(i)}$ compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
  - Compute $\delta^{(L-1)}, \cdots, \delta^{(2)}$ using backpropagation
  - $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Activ
Go to I

$$\boxed{\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} \left(a^{(l)}\right)^T}$$

Vectorized implementation

Back propagation Algorithm

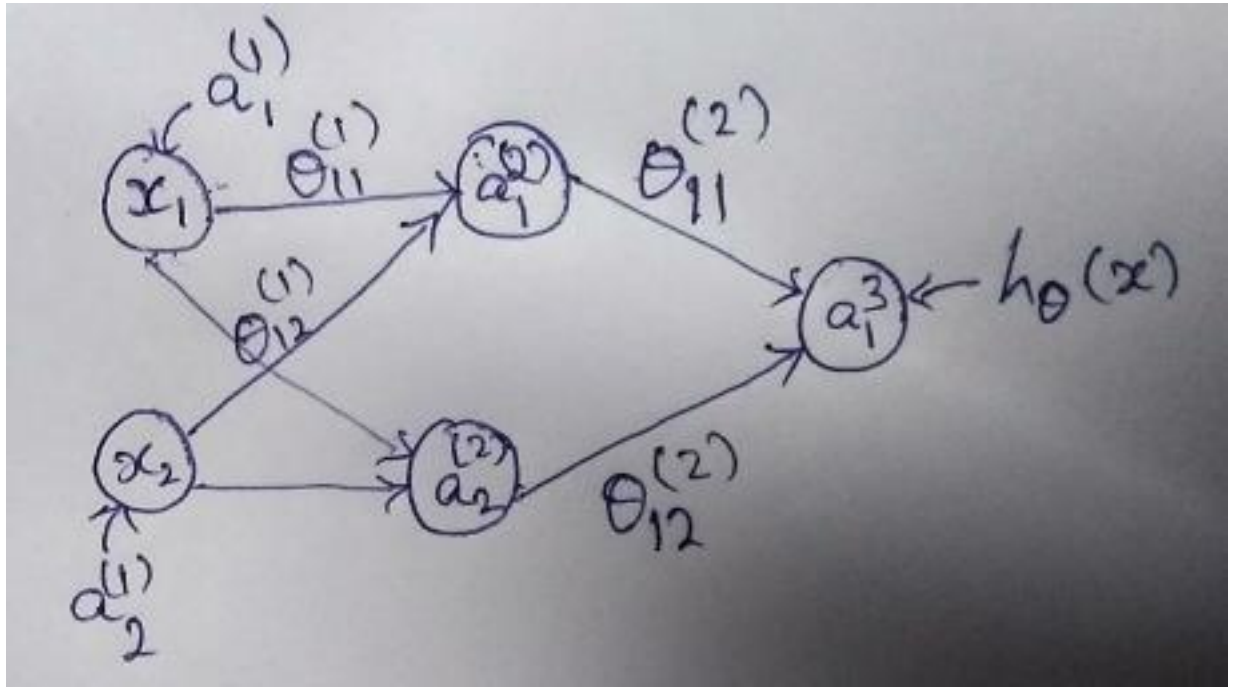
$$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\,\Theta_{ij}^{(l)} \text{ if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} \text{ if } j = 0$$

And finally, $\frac{\partial}{\partial\Theta_{ij}^{(l)}} = D_{ij}^{(l)}$

- Update each weight using the gradient descent algorithm.

Demonstration using example



$$z_1^{(2)} = \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2$$

$$a_1^{(2)} = g(z_1^{(2)})$$

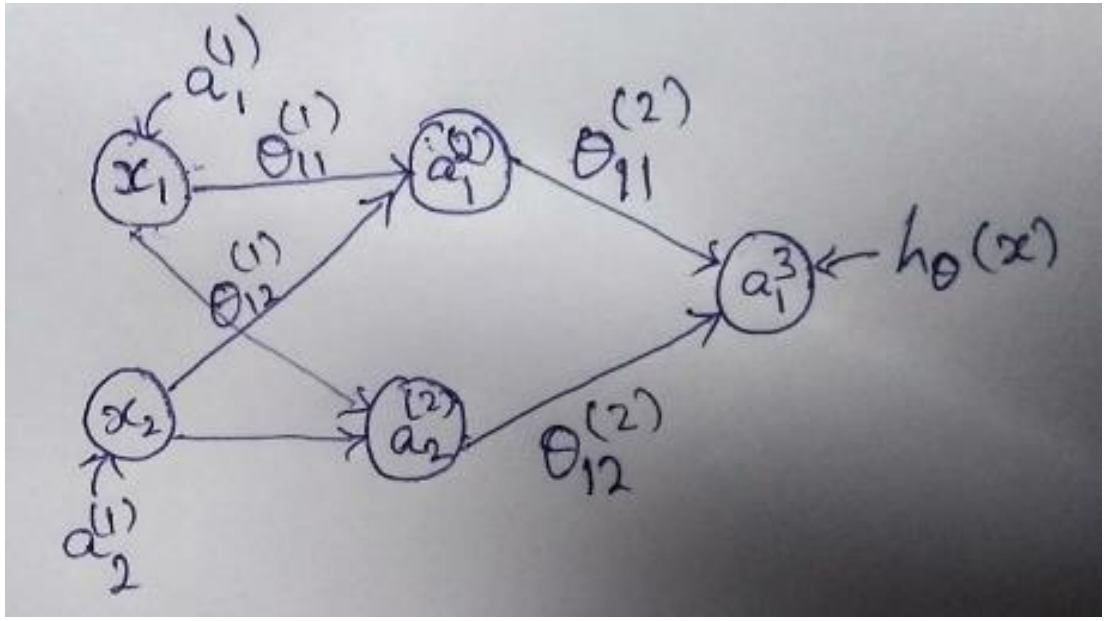
$$z_2^{(2)} = \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2$$

$$a_2^{(2)} = g(z_2^{(2)})$$

$$z_1^{(3)} = \theta_{11}^{(2)} \cdot a_1^{(2)} + \theta_{12}^{(2)} \cdot a_2^{(2)}$$

$$h_\theta(x) = a_1^{(3)} = g(z_1^{(3)})$$

Demonstration using example



$$\frac{\partial J}{\partial \theta_{11}^{2}} = \frac{\partial J}{\partial h_\theta(x)} \cdot \frac{\partial h_\theta(x)}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial \theta_{11}^{(2)}}$$

$$\frac{\partial J}{\partial h_\theta(x)} = \sum_i y_i \cdot \frac{1}{h_\theta(x_i)} + (1-y_i)\frac{1}{1-h_\theta(x_i)} \cdot (-1)$$

$$= \sum_i \frac{y_i - h_\theta(x_i)}{h_\theta(x_i)\left(1-h_\theta(x_i)\right)}$$

- $J(\theta) = \sum_{i=1}^{m} \left[y^i \log h_\theta(x^i) + (1-y^i) \log\left(1 - h_\theta(x^i)\right)\right]$

$$\frac{\partial J}{\partial \Theta_{ij}^{(l)}} = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial J}{\theta_{11}^{(2)}} = a_1^{(2)} \cdot \delta_1^{3}$$

$$z_1^{(3)} = \theta_{11}^{(2)} \cdot a_1^{(2)} + \theta_{12}^{(2)} \cdot a_2^{(2)}$$

$$\frac{\partial z_1^{(3)}}{\partial \theta_{11}^{(2)}} = a_1^{(2)}$$

Demonstration using example

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

$$\sigma'(x) = -\frac{e^{-x}}{(1+e^{-x})^2} \cdot -1$$

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{1+e^{-x}-1}{1+e^{-x}}$$

$$= \sigma(x) \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}} \right)$$

$$\sigma'(x) = \sigma(x) \cdot (1-\sigma(x))$$

$$h_\theta(x) = g(z_1^3)$$

$$\frac{\partial h_\theta(x)}{\partial z_1^3} = h_\theta(x)\,(1-h_\theta(x))$$

- Combining all of them together.

$$\frac{\partial J}{\partial \theta_{11}^2} = \underbrace{(y - h_\theta(x))}_{\delta^L} \cdot \underbrace{a_1^{(2)}}_{a_j^{(l)}}$$

$$[\delta_i^{(l+1)}]$$

Demonstration using example

$$\frac{\partial J}{\partial \theta_{11}^{(1)}} = \boxed{\frac{\partial J}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}}} \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}$$

$$(y_i - a_1^3) \cdot \theta_{11}^{(2)} \cdot a_1^{(2)}(1 - a_1^{(2)}) \cdot a_1^1$$

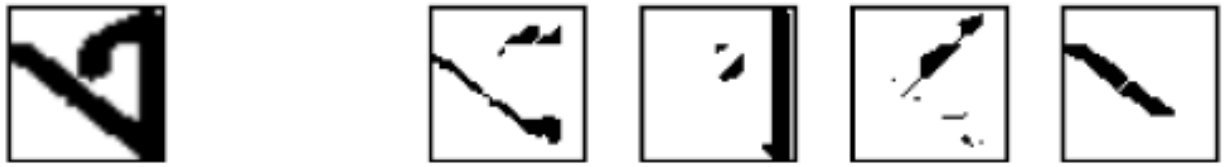
$$\delta^{(l+1)} \qquad \theta^{(l)} \qquad g'(z^{(l)})$$

Applications

- Classification of data
- Anomaly detection
- Speech recognition
- Audio generation
- Time series analysis
- Spell checking
- Character recognition
- Machine translation
- Healthcare
- Weather monitoring
- And many other, Almost in every domain.

Character Recognition

Fig. 7. Fixed Mesh- Five regions as outlined



a. Original image b. hor. c. ver. d. right e. left

Directional decomposition in 4 directions

- 5 regions, 4 directions, 20 features.
- 133 classes
- Around 10000 instances
- About 95% accuracy

Foundational Concepts of Convolutional Neural Networks

Traditional Machine Learning

- Traditional pattern recognition models work with hand crafted features and relatively simple trainable classifiers.

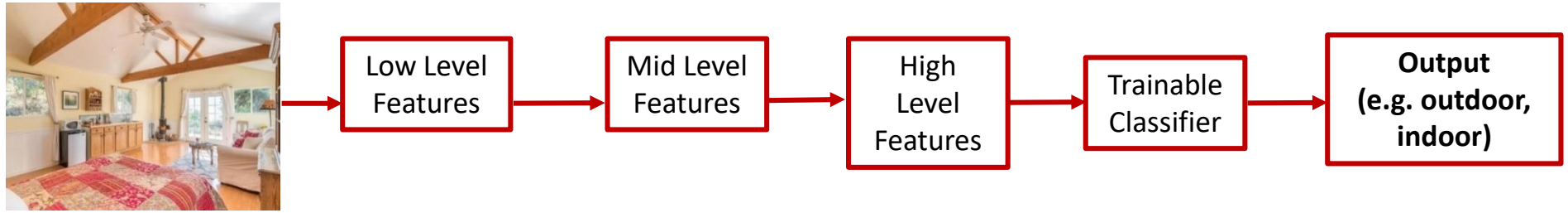
Extract Hand Crafted Features → **Trainable Classifier** (e.g. SVM, Random Forrest) → **Output** (e.g. Outdoor Yes or No)

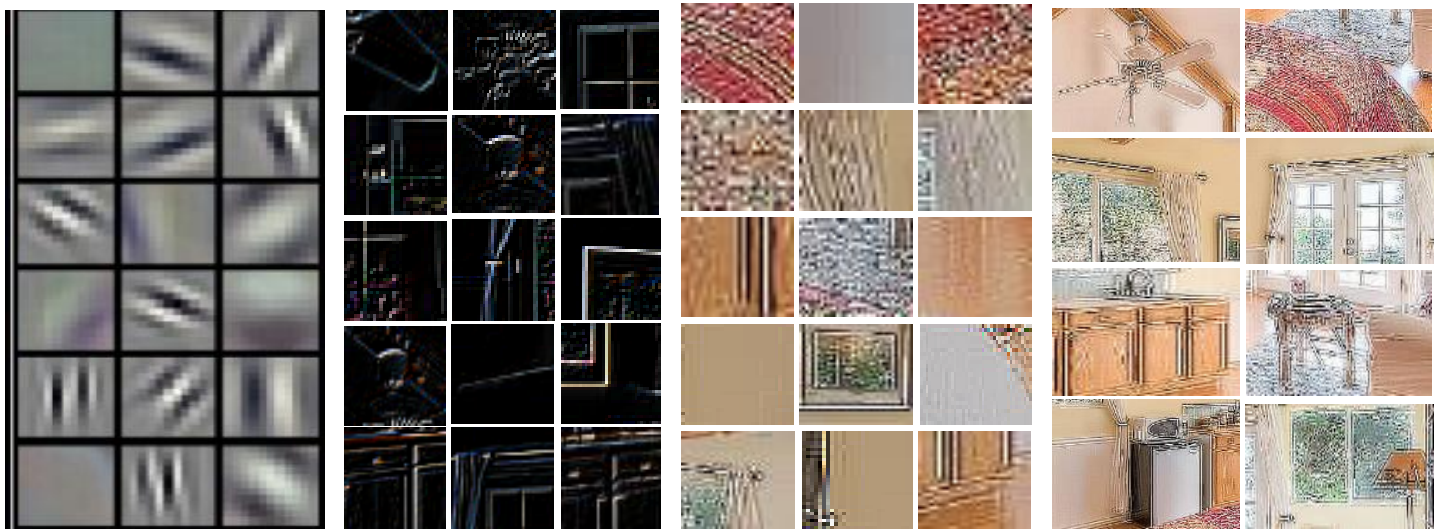
Limitations

- Very tedious and costly to develop hand crafted features.
- The hand-crafted features are usually highly depend on one application.

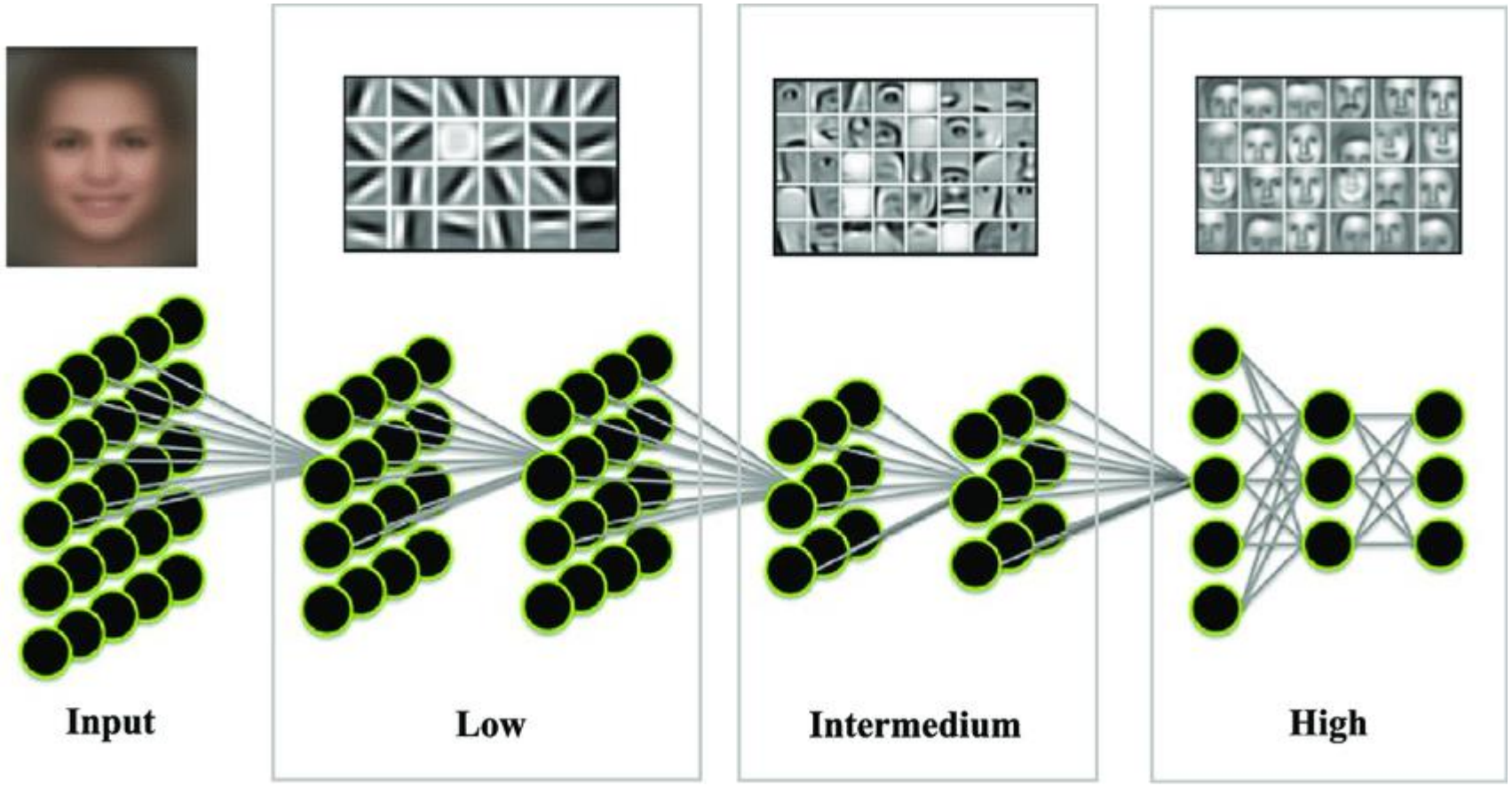
Deep Learning

- Deep learning has an **inbuilt automatic multi stage feature learning process** that learns rich hierarchical representations (i.e. features).





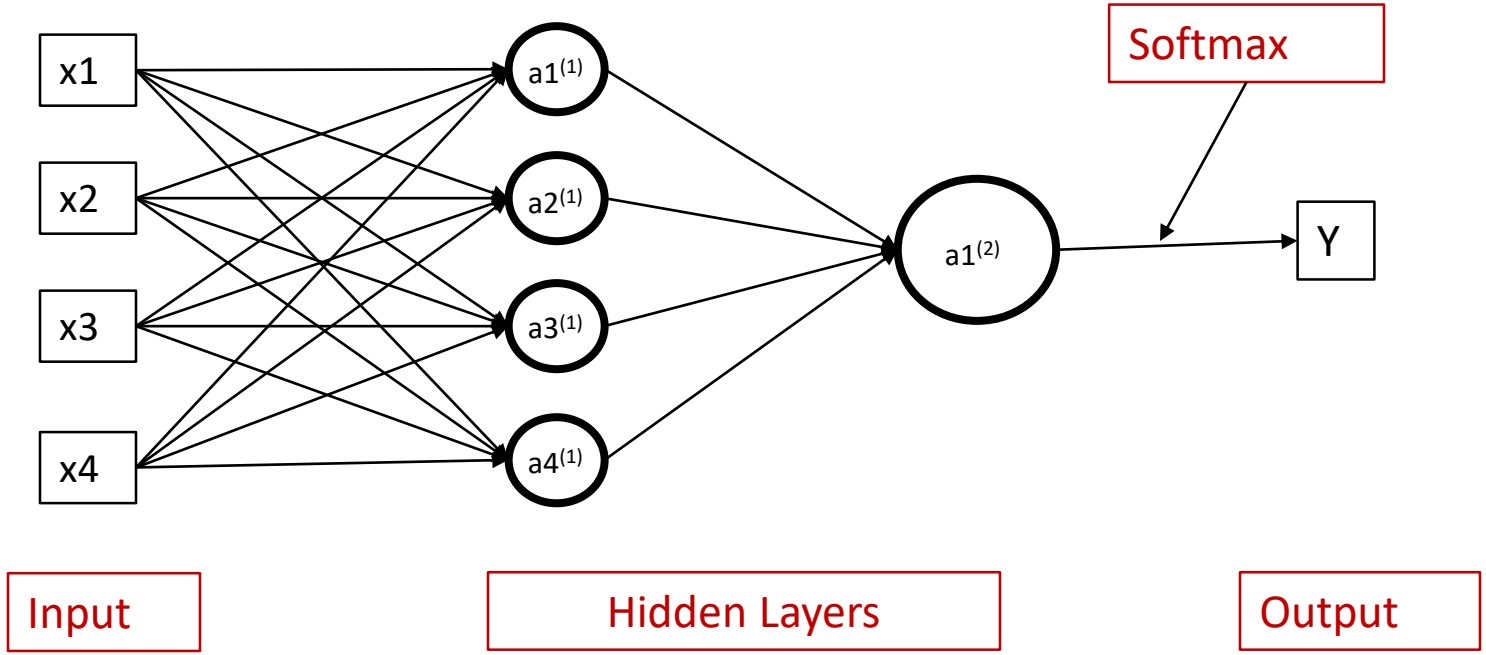
Face Recognition Problem



Research gate

Drawbacks of Fully Connected Neural Networks !!

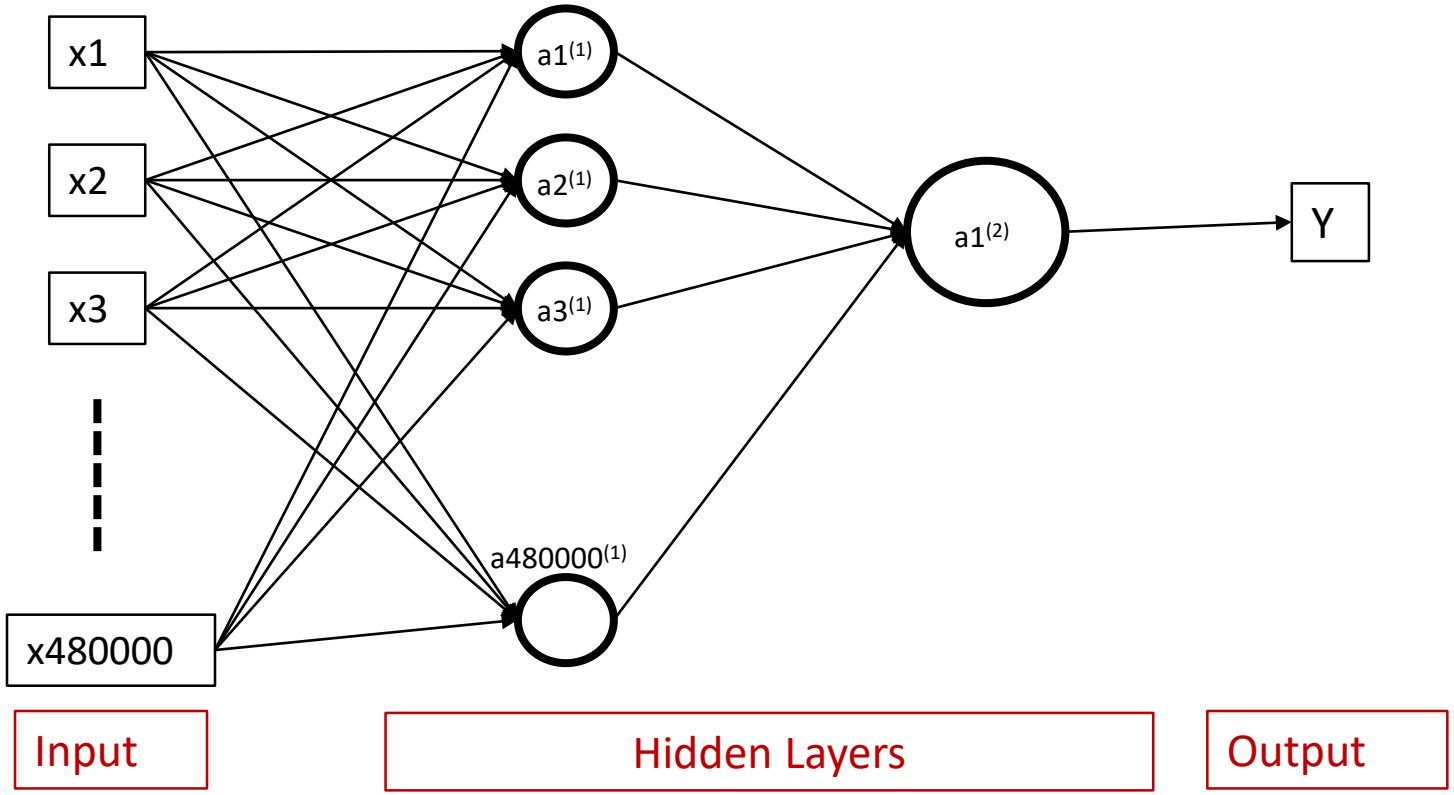
Simple Neural Network



Number of parameters = 4*4 + 4 +1 = 21

If the input is an Image?

400 X 400 X 3

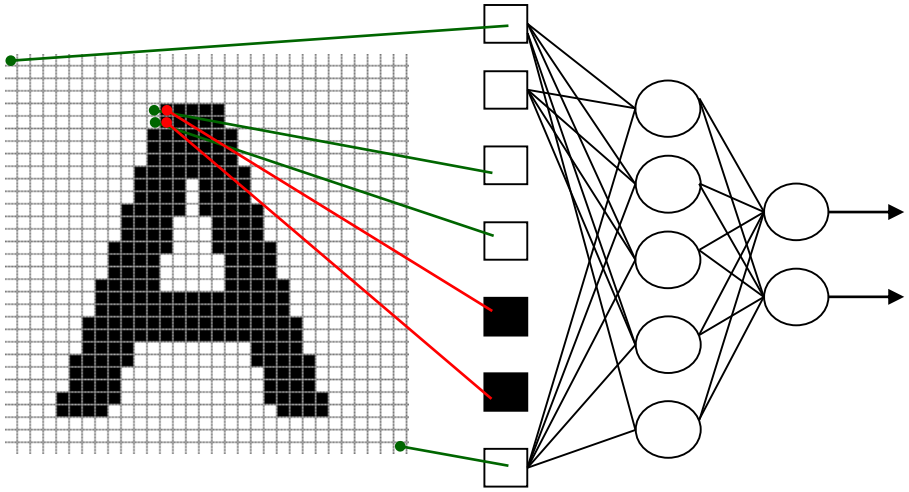


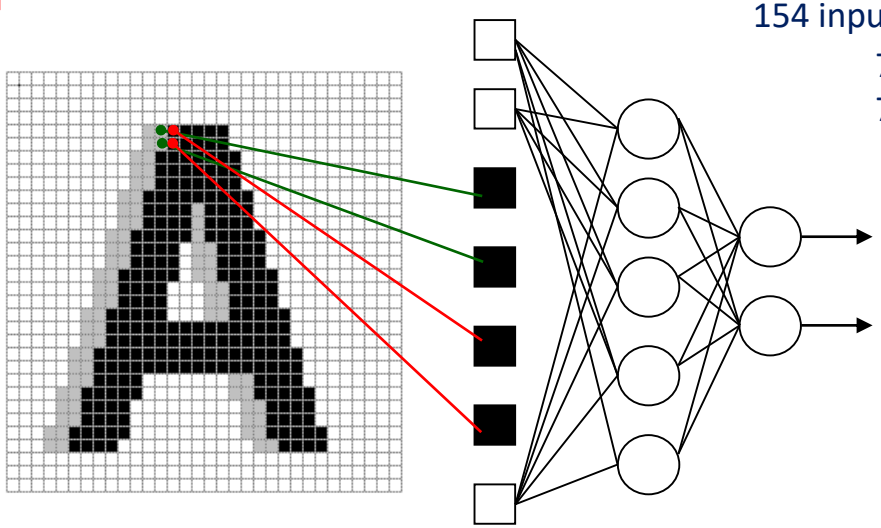
Number of Parameters

480000*480000 + 480000 +1 = **approximately 230 Billion !!!**

480000*1000 + 1000 +1 = approximately **480 million !!!**

Little or no invariance to shifting, scaling, and other forms of distortion





154 input change from 2 shift left

77 : black to white

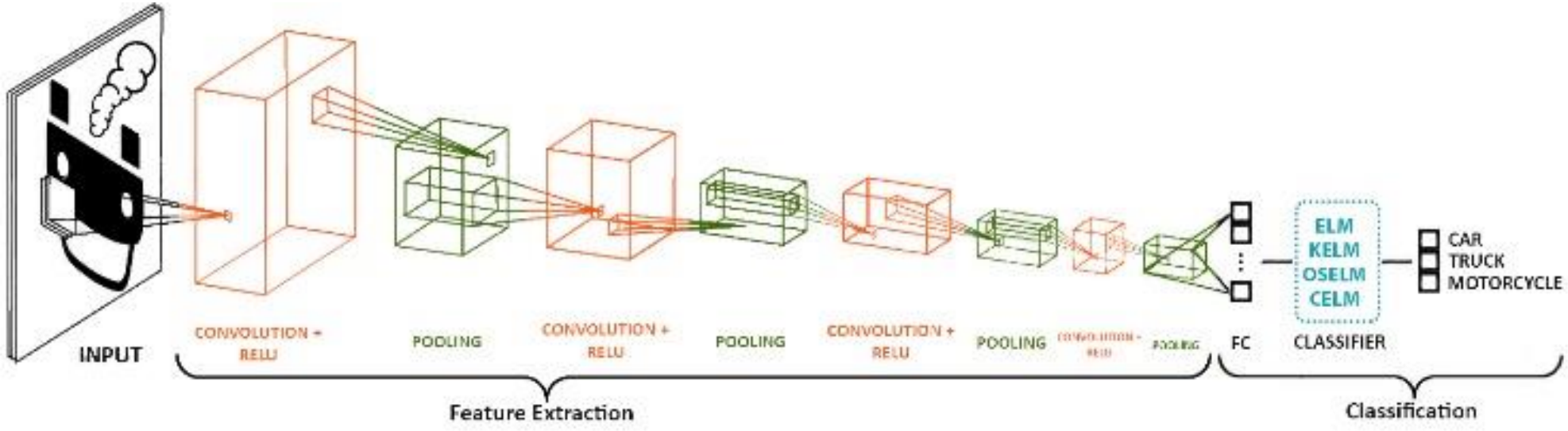
77 : white to black

Scaling and other forms of distortion

How Convolutional Neural Networks (CNN) Helps !

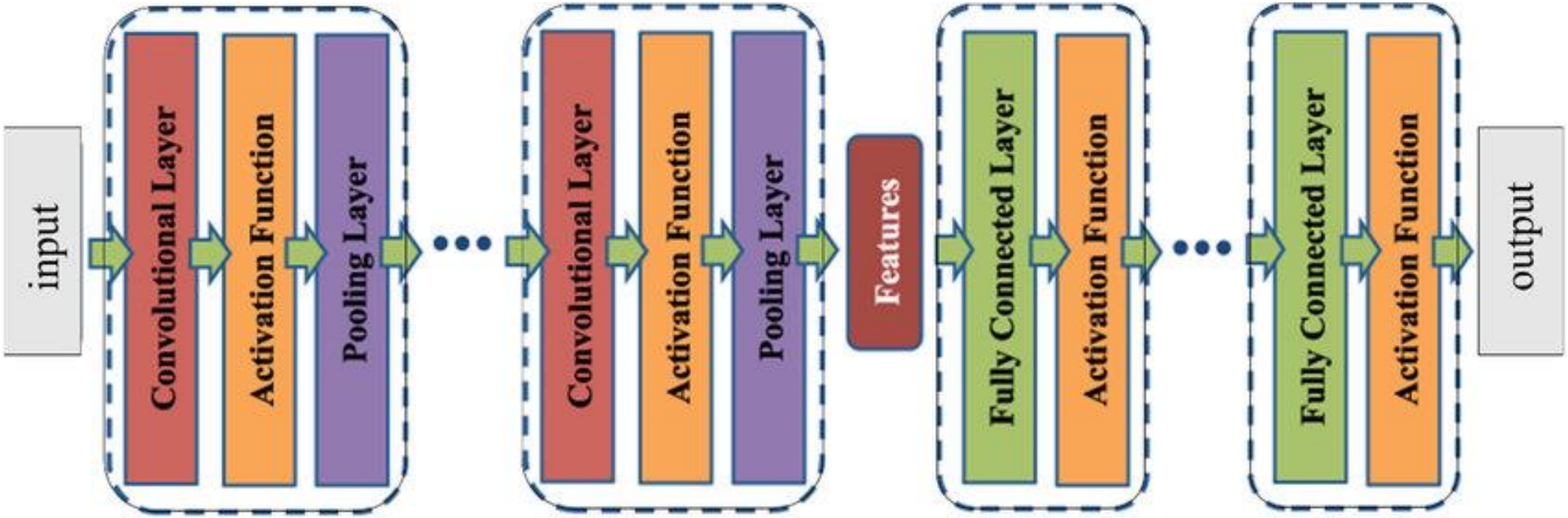
What is CNN ?

- A sub-category of neural networks
- Outperforms other models on image data.
- Architecture is composed of two main blocks.
  - —Feature extraction phase
  - —Classification phase

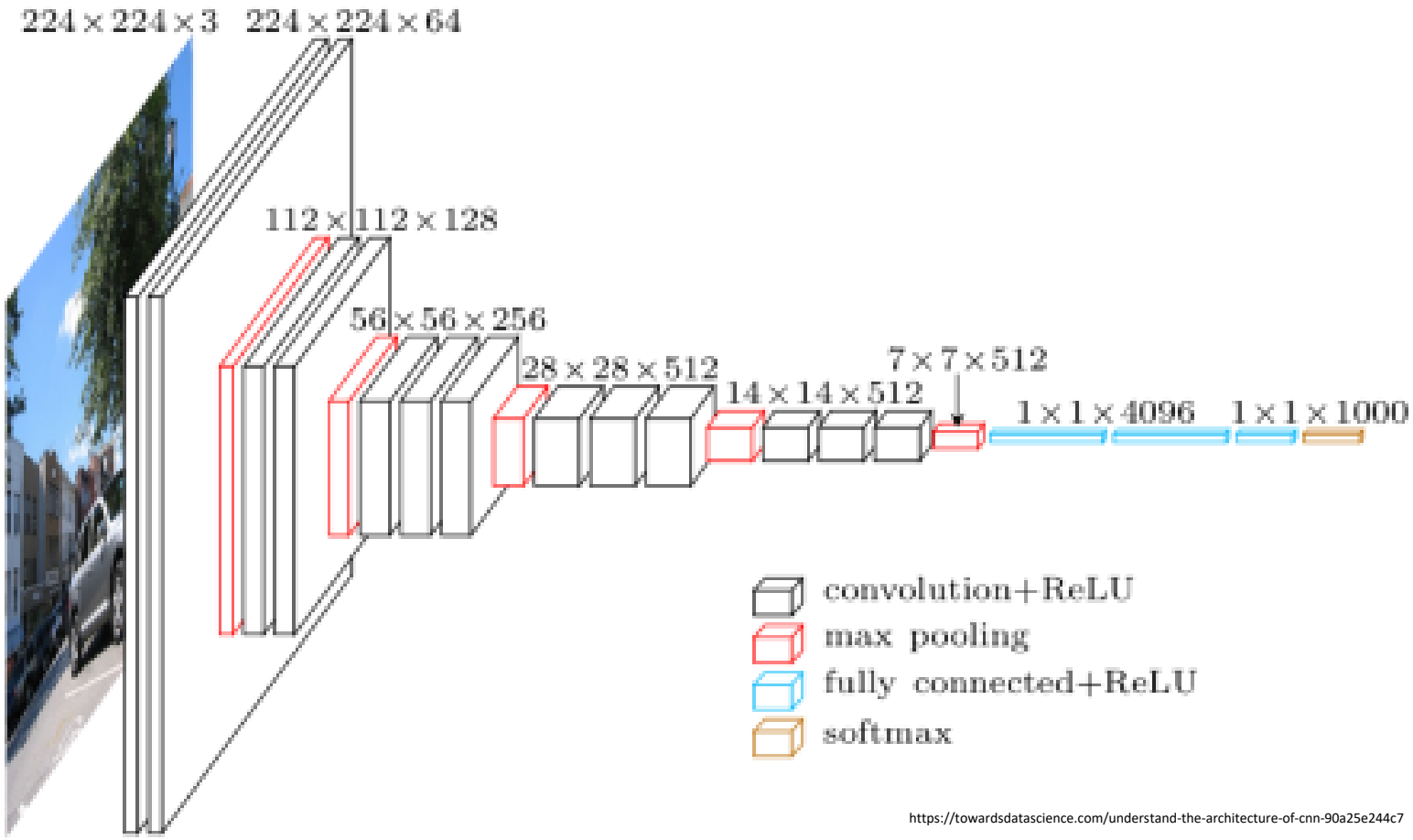


http://norc.aut.ac.ir/convolution-neural-network-joint-with-mixture-of-extreme-learning-machines-for-feature-extraction-and-classification-of-accident-images/

CNN Architecture



Understanding CNN Architecture



https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

Convolutional Layers

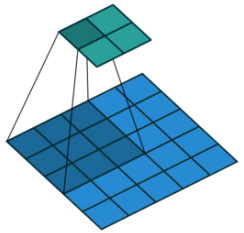
- Filter

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Input Image

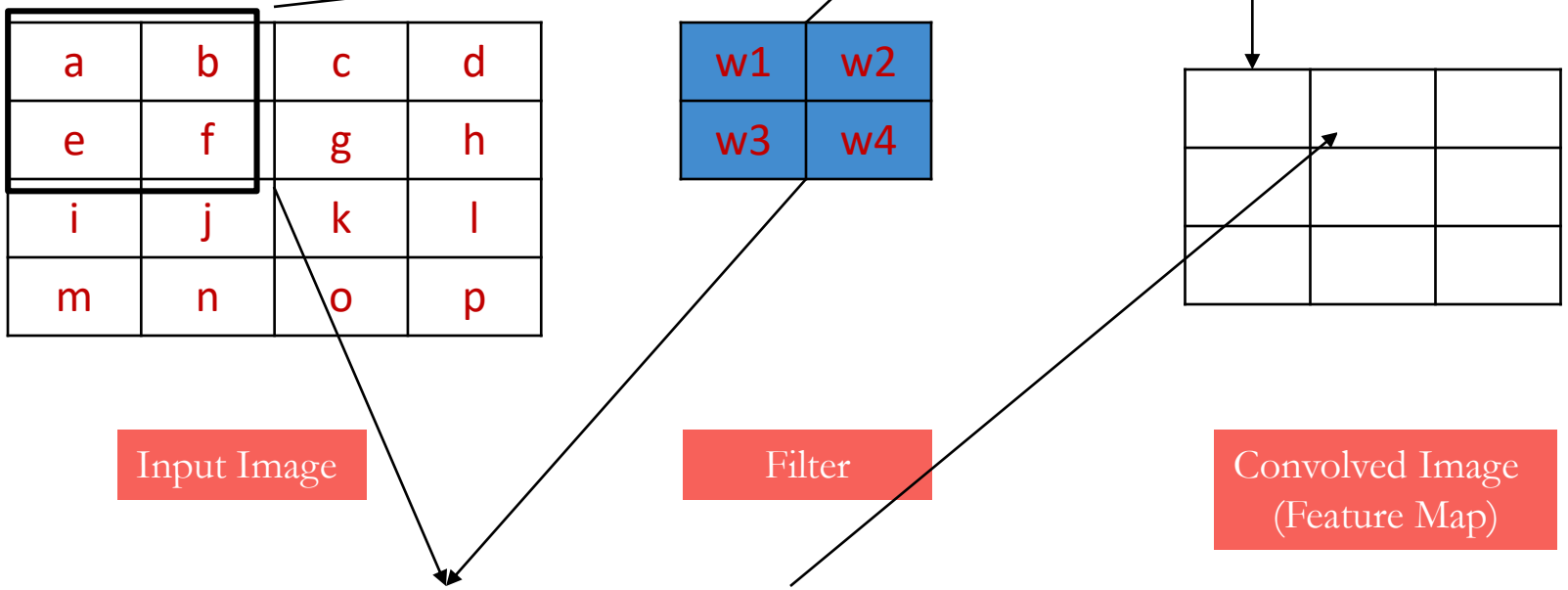
Convoluted Image

Convolutional Layers



- What is Convolution?

$$h1 = f(a * w1 + b * w2 + e * w3 + f * w4)$$

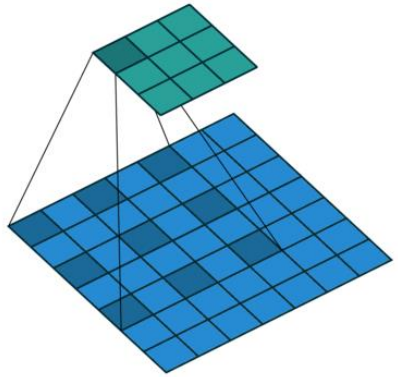


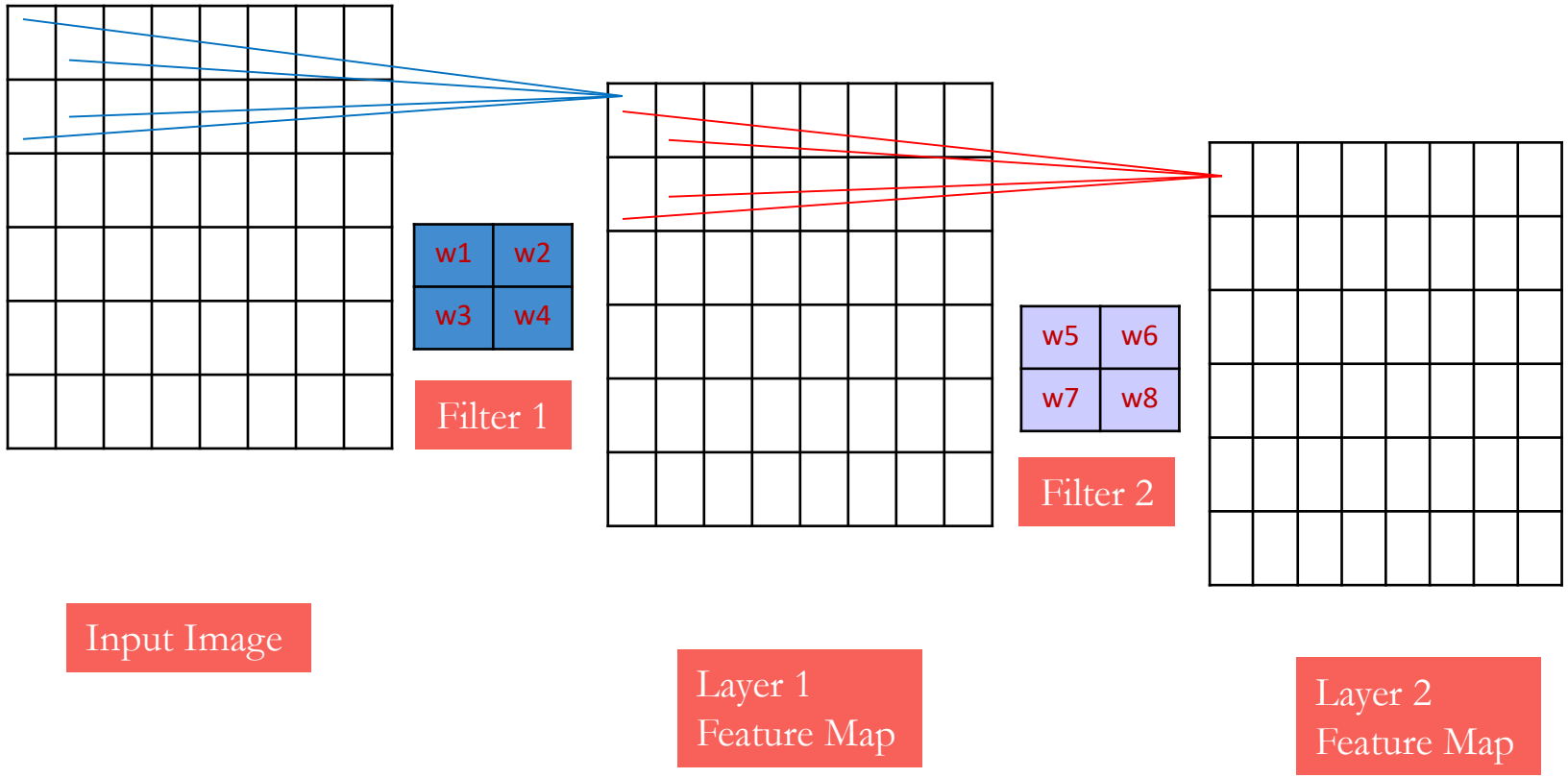
$$h2 = f(b * w1 + c * w2 + f * w3 + g * w4)$$

Number of Parameters for one feature map = 4

Number of Parameters for 100 feature map = 4*100

Lower Level to More Complex Features





- In Convolutional neural networks, hidden units are only connected to local receptive field.

Understanding Convolutional Filters

- Convolutions have been used in image and signal processing for a long time.
- Convolutions in machine learning are different than those in image processing.
- Instead of using these filters, we can create our own as well and treat them as a parameter which the model will learn using backpropagation.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Blur

0	-1	0
-1	5	-1
0	-1	0

Sharpen

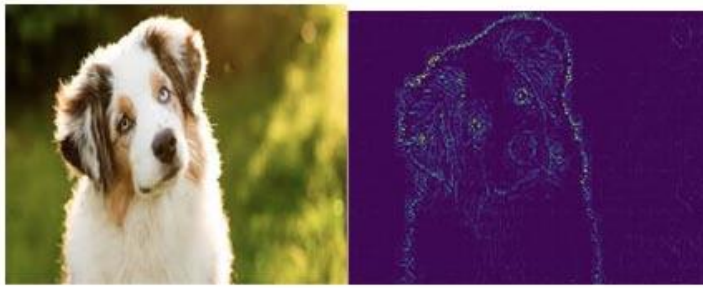
-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical

Edges



https://www.saama.com/different-kinds-convolutional-filters/

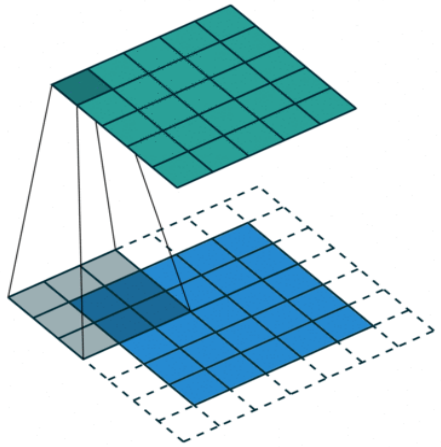
- Question1: convolving an input of 6 X 6 dimension with a 3 X 3 filter results in _____output.

  — Answer: 4 X 4

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

- Question 2: If the input is **n X n** and the filter size is **f X f**, What will be the output size ?
  - **Input:** n X n
  - **Filter size:** f X f
  - **Output:** (n-f+1) X (n-f+1)

Padding



- Without padding
  - **Input:** n X n
  - **Filter size:** f X f
  - **Output:** (n-f+1) X (n-f+1)
- There are primarily two disadvantages here:
  - Every time we apply a convolutional operation, the size of the image shrinks
  - Pixels present in the corner of the image are used only a few number of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss
- To overcome these issues, we can pad the image with an additional border, i.e., we add one pixel all around the edges.

Padding

- If we have 6 X 6 input matrix, after zero-padding it will become 8 X 8 matrix.
- Applying convolution of 3 X 3 on it will result in a 6 X 6 matrix which is the original shape of the image.
  - **Input:** n X n
  - **Padding:** p
  - **Filter size:** f X f
  - **Output:** (n+2p-f+1) X (n+2p-f+1)

0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0
0	0	0	0	0	0	0	0

WEIGHT

1	0	1
0	1	0
1	0	1

139

- There are two common choices for padding:
  - **Valid:** It means no padding. If we are using valid padding, the output will be (n-f+1) X (n-f+1)
  - **Same:** Here, we apply padding so that the output size is the same as the input size, i.e., n+2p-f+1 = n
    So, p = (f-1)/2

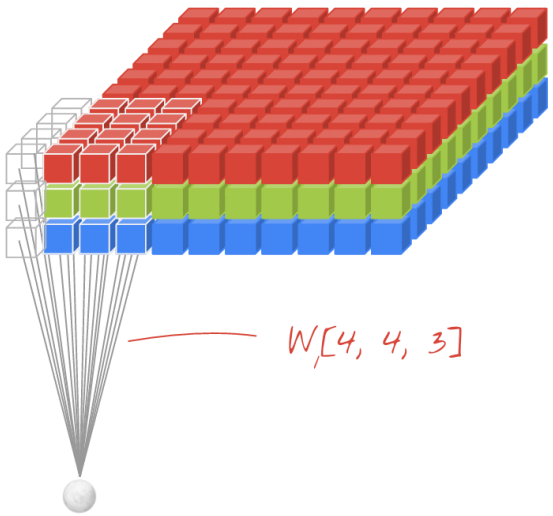
Strided Convolutions

- Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately. The dimensions for stride *s* will be:
  - **Input:** n X n
  - **Padding:** p
  - **Stride:** s
  - **Filter size:** f X f
  - **Output:** [(n+2p-f)/s+1] X [(n+2p-f)/s+1]
- Stride helps to reduce the size of the image, a particularly useful feature.

Convolutions Over Volume

- Suppose, instead of a 2-D image, we have a 3-D input image

https://obaydakov.github.io/post/2018/what-do-you-mean-by-1d-2d-and-3d-convolutions-in-cnn/

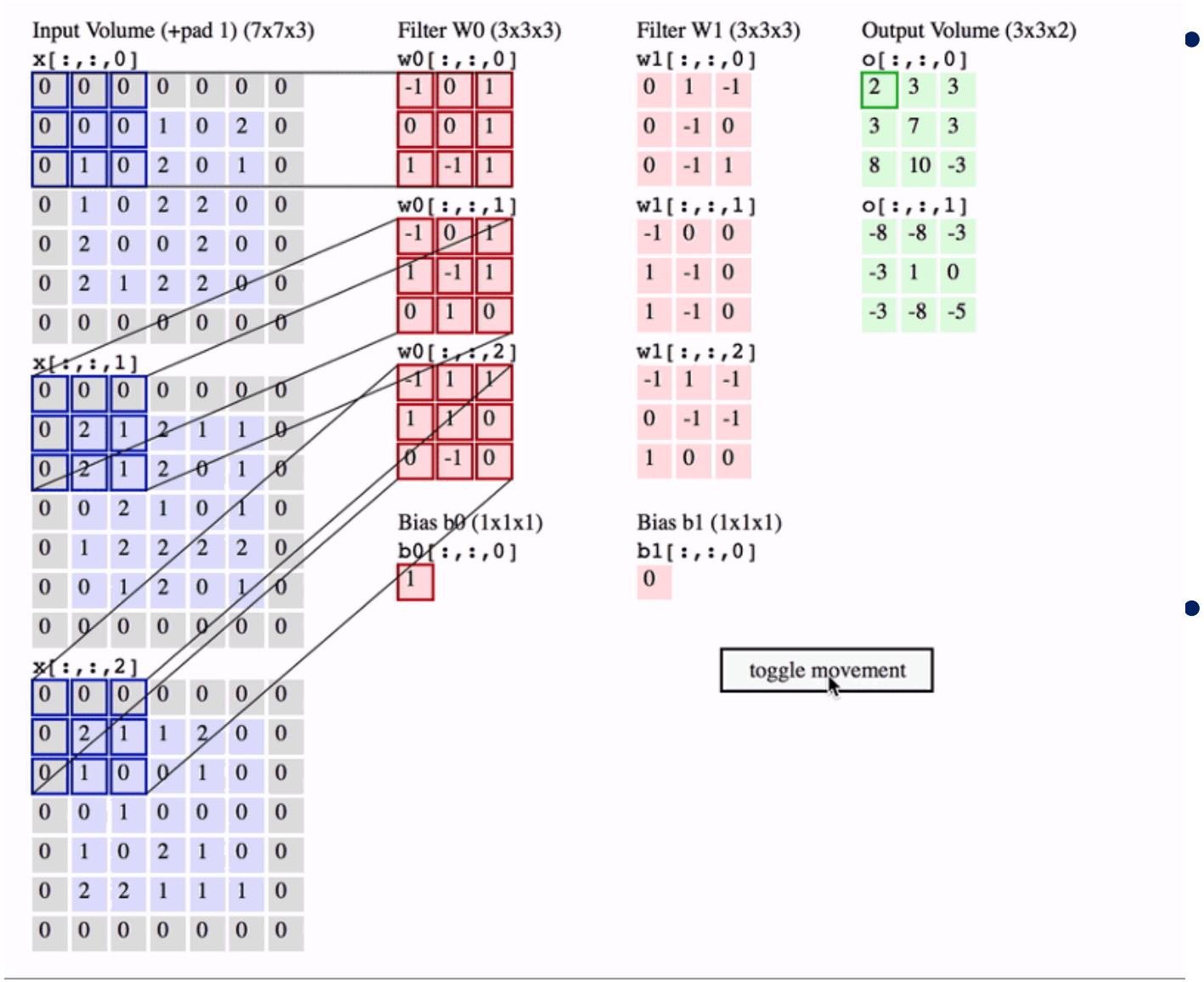


- How will we apply convolution on this image?
  - —We will use a 3D filter instead of a 2D filter.

- The dimensions In the figure represent the height, width and channels in the filter.

- ***Keep in mind that the number of channels in the input and filter should be same.***

Convolutions Over Volume: Example



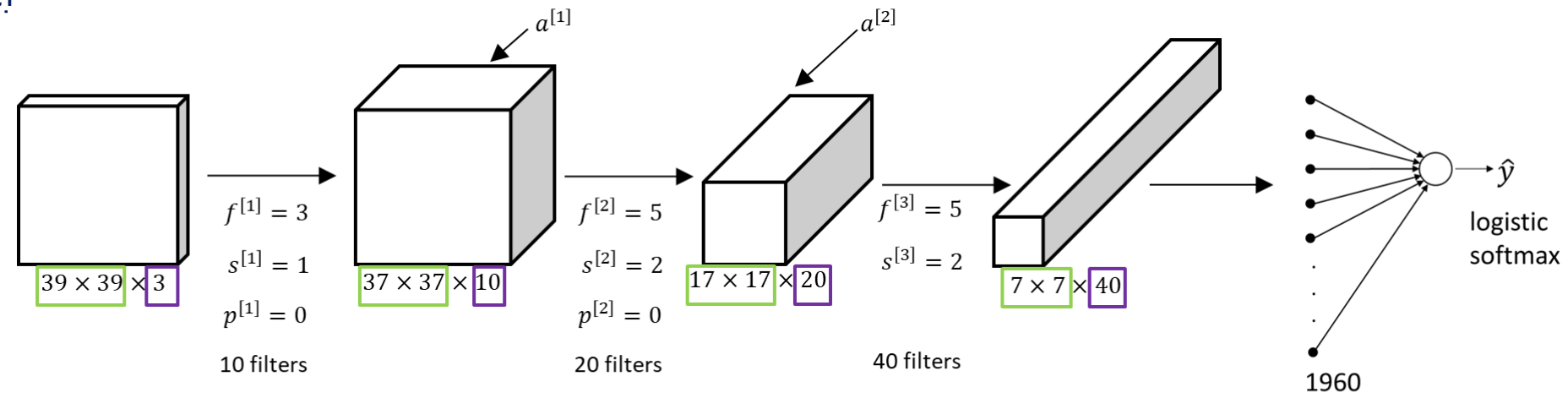
- Generalized dimensions can be given as:
  - **Input:** n X n X n_c
  - **Filter:** f X f X n_c
  - **Padding:** p
  - **Stride:** s
  - **Output:** [(n+2p-f)/s+1] X [(n+2p-f)/s+1] X n_c'
- Here, n_c is the number of channels in the input and filter, while n_c' is the number of filters.

One Layer of a Convolutional Network

- Convolving over the entire image using a filter
- Add a bias term to convolution outputs
- Finally apply an activation function to generate activations.
- *This is one layer of a convolutional network*.
- Question: Let the input size is (6 X 6 X 3) and filters size is (3 X 3 X 3). Suppose we have 10 filters, what will be the number of parameters in that layer?
  - —Number of parameters for each filter = 3*3*3 = 27
  - —There will be a bias term for each filter, so total parameters per filter = 28
  - —As there are 10 filters, the total parameters for that layer = 28*10 = 280
- Clearly, the number of parameters in case of convolutional neural networks is independent of the size of the image. It essentially depends on the filter size.
- No matter how big the image is, the parameters only depend on the filter size. Awesome, isn't it?

Simple Convolutional Network Example

- $f^{[l]}$ = filter size
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_{[c]}^{[l]}$ = number of filters



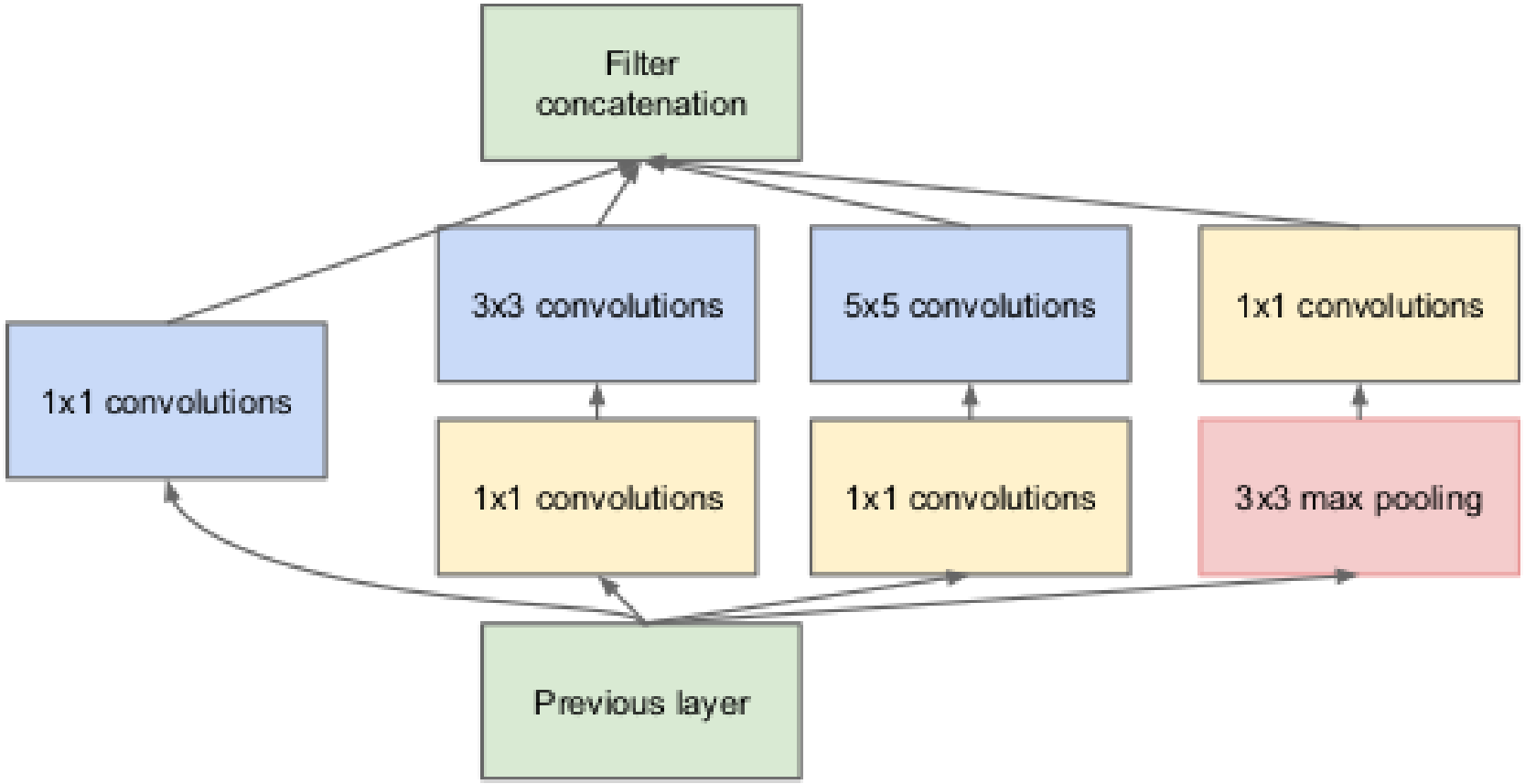
$n_H^{[0]} = n_W^{[0]} = 39$ $\quad$ $n_H^{[1]} = n_W^{[1]} = 37$ $\quad$ $n_H^{[2]} = n_W^{[2]} = 17$

$n_C^{[0]} = 3$ $\quad$ $n_C^{[1]} = 10$ $\quad$ $n_C^{[2]} = 20$

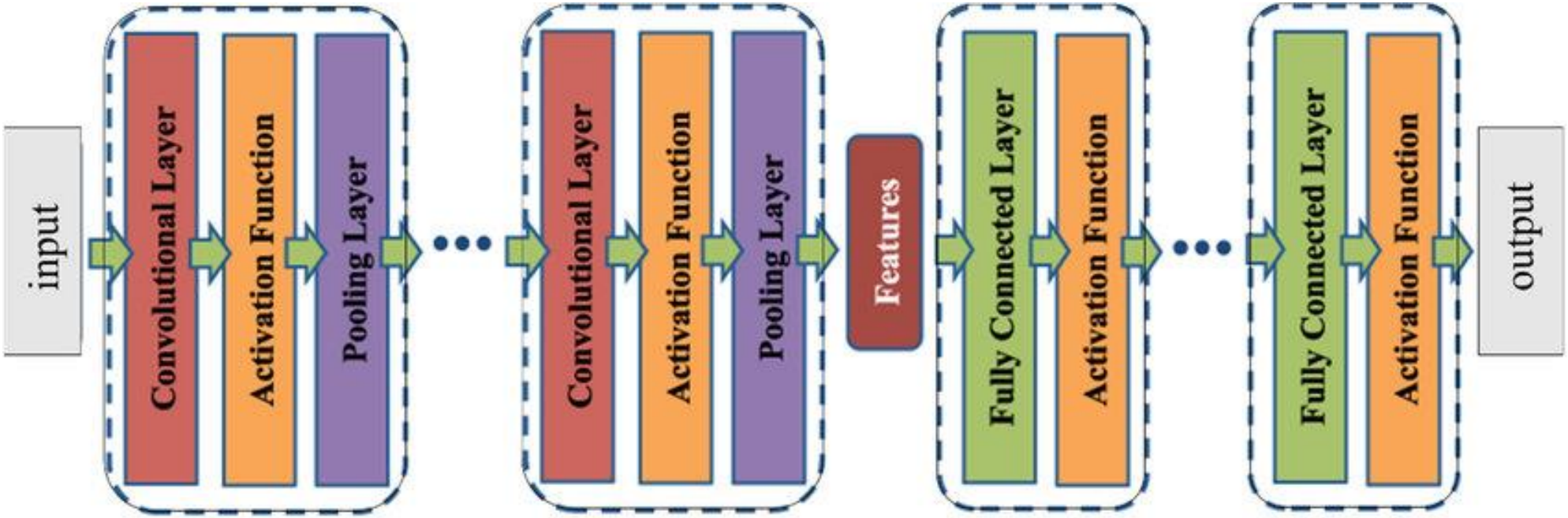
- Layer 1: 10 comes from the fact that we use 10 filters and 37 comes from this formula (n+2p−f)/s+1.

https://datahacker.rs/example-of-convolutional-neural-network/

Parallel CNN



CNN Architecture



Activation Functions

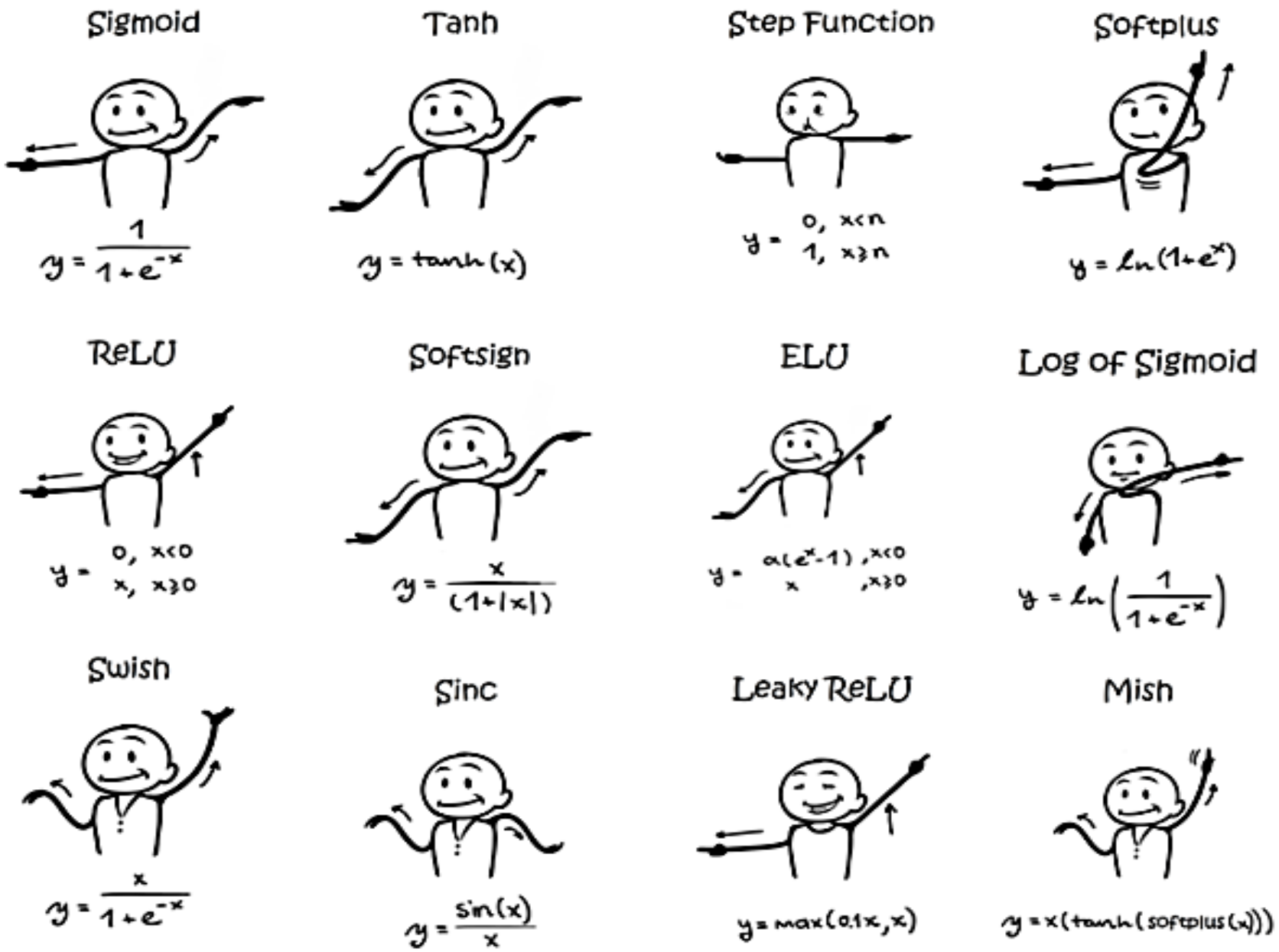


Image Source: https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions/

Pooling Layers

- Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation.
- Retains the most important information.
  - —Max pooling: reports the maximum output within a rectangular neighborhood.
  - —Average pooling: reports the average output of a rectangular neighborhood.

1	3	5	3
4	2	3	1
3	1	1	3
0	1	0	4

Input Matrix

MaxPool with 2X2 filter with stride of 2

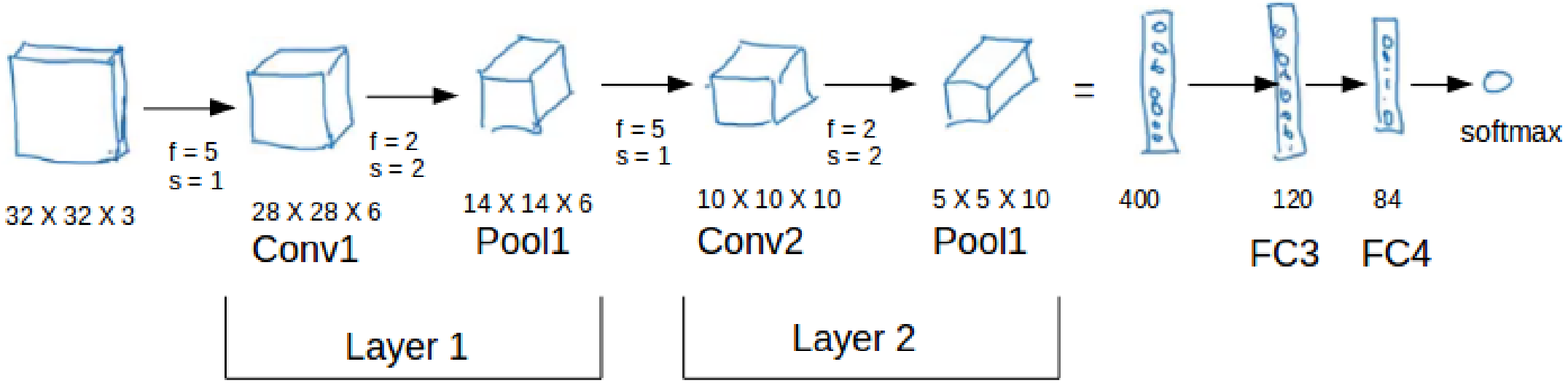
4	5
3	4

Output Matrix

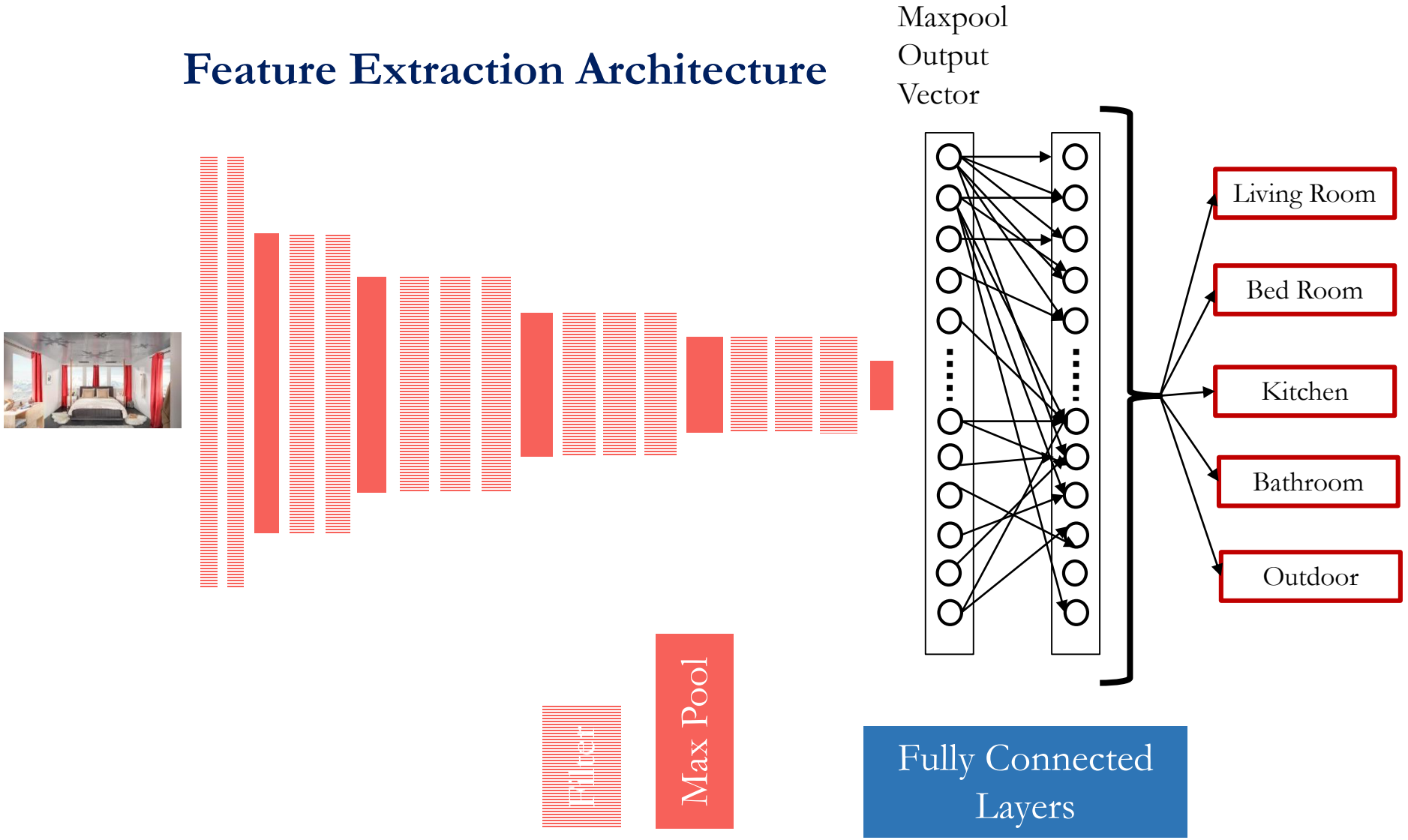
—Input of the pooling layer: n_h X n_w X n_c

—Output: $((n_h - f) / s + 1)$ X $((n_w - f) / s + 1)$ X n_c.

Example

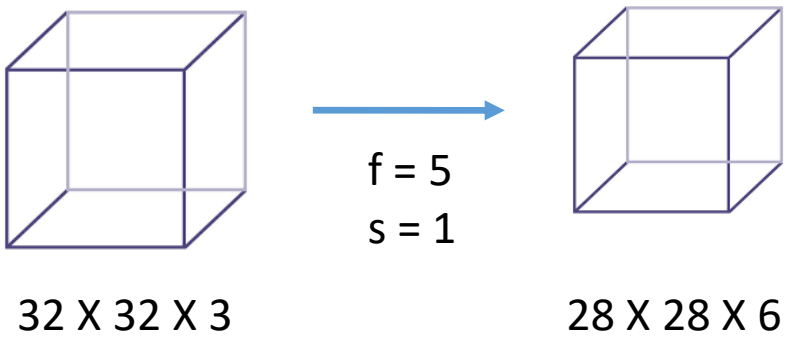

f = 5
s = 1
f = 2
s = 2
f = 5
s = 1
f = 2
s = 2
=
softmax
32 X 32 X 3
28 X 28 X 6
Conv1
14 X 14 X 6
Pool1
10 X 10 X 10
Conv2
5 X 5 X 10
Pool1
400
120
FC3
84
FC4
Layer 1
Layer 2

Convolutional Neural Network



Why convolutions ?

- There are primarily two major advantages of using convolutional layers over using just fully connected layers:
  - —Parameter sharing
  - —Sparsity of connections

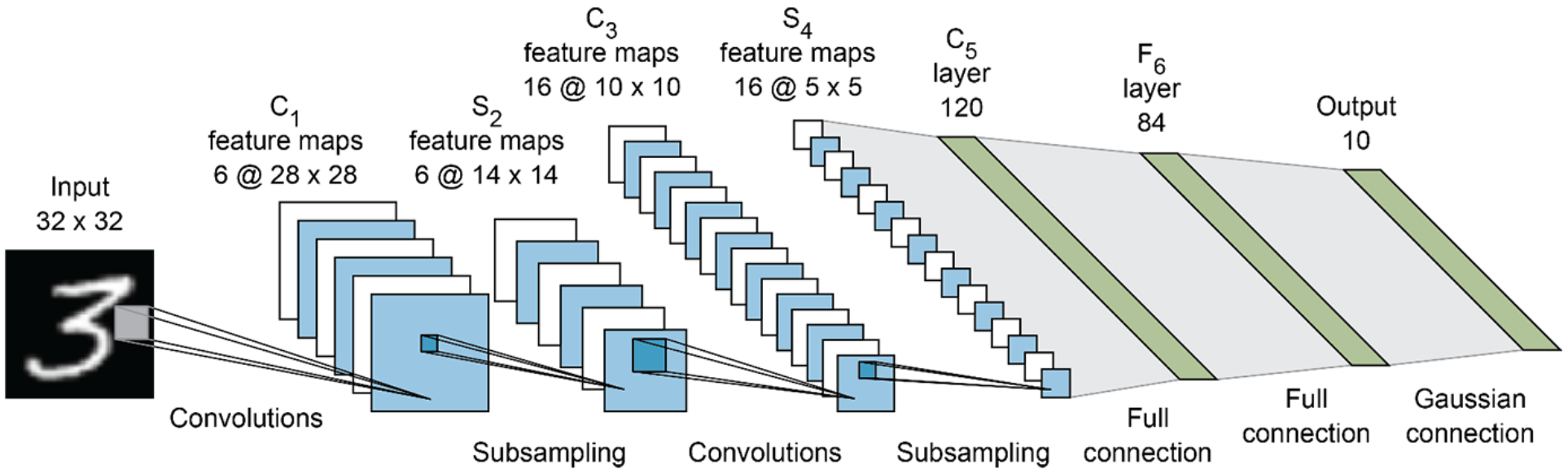


- How many parameters do we require if we use fully connected network for the above example ?
  - —32*32*3*28*28*6, which is nearly equal to 14 million!
  - —Makes no sense, right?
- How many parameters if we use convolutional layer ?
  - — (5*5 + 1) * 6 = 156

Classic Networks

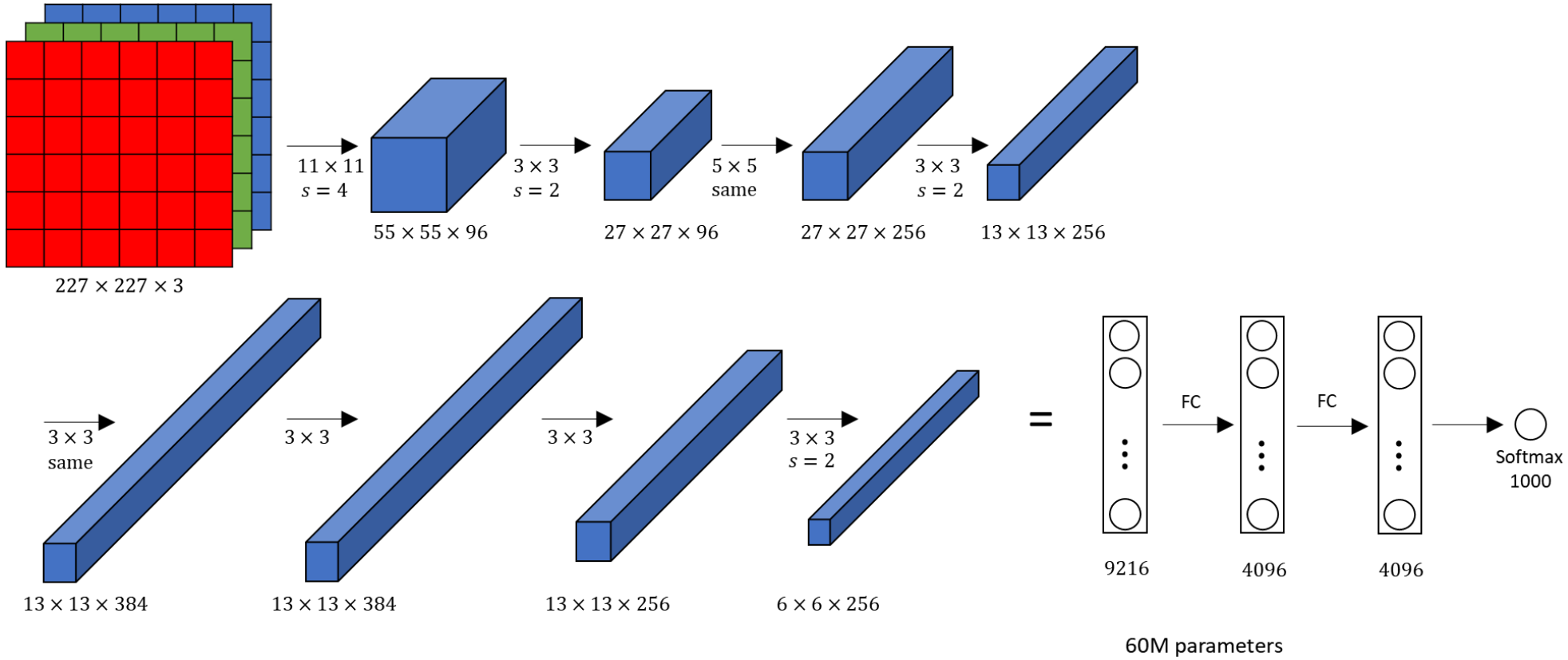
LeNet-5

- **Parameters:** 60k
- **Layers flow:** Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
- **Activation functions:** Sigmoid/tanh and ReLu



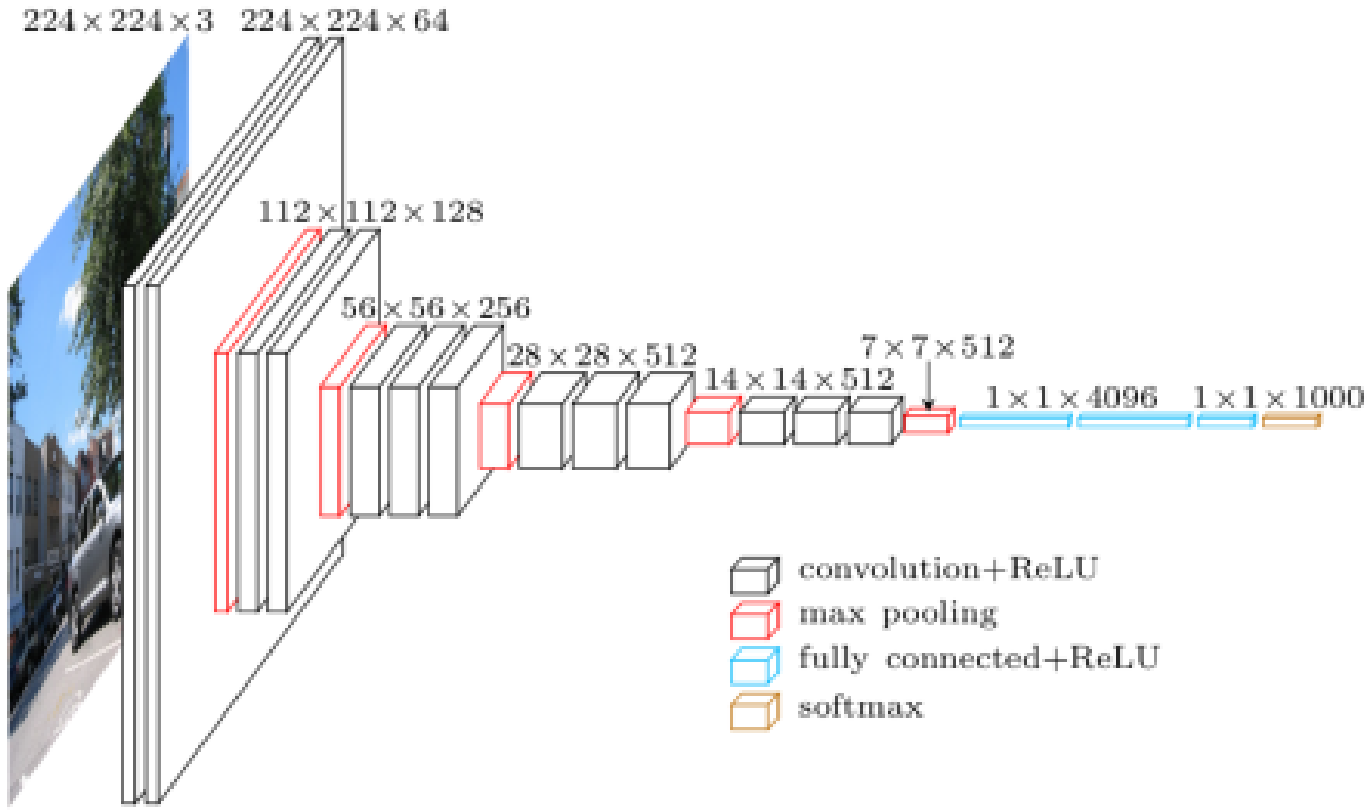
Alex Net

- **Parameters:** 60 million
- **Activation function:** ReLu



VGG-16

- **Parameters:** 138 million



https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

Many more …

- Inception
- ResNet
- ResNeXt
- DenseNet
- …

Applications

- **Image Classification**
  - **Search Engines, Recommender Systems, Social Media**
- **Face Recognition**
  - Social media, entertainment, security
- **Optical Character Recognition**
  - **Legal, Banking, Insurance, Document digitization**

- **Medical Image Computing**
  - Risk analysis, anomaly detection, drug discovery
- And many other

Thank you !

Happy Learning !!

References

- Pattern recognition and machine leanring, Christopher M. Bishop.
- https://machinelearningmedium.com/2017/09/21/neural-networks/
- https://machinelearningmedium.com/2017/10/03/neural-networks-cost-function-and-back-propagation/
- https://machinelearningmedium.com/2018/03/20/backpropagation-derivation/
- https://machinelearningmedium.com/2017/09/27/neural-network-intuition/
- Machine Learning: Coursera.org
- Directional Decomposition for *Odia* Character Recognition, Chandana Mitra and Arun K. Pujari.
- https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
- Figures and examples are taken from various online sources.
- https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441#:~:text=ReLU%20activation%20function%20is%20widely,used%20in%20the%20hidden%20layers.