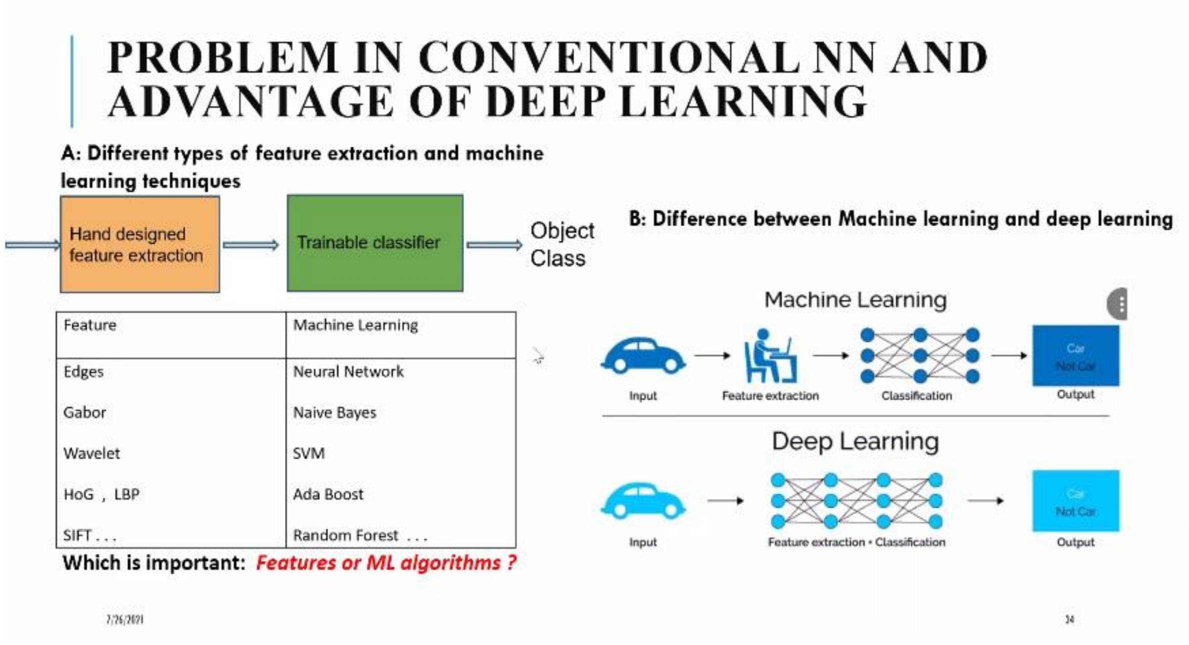
Deep Learning

- When learning is based on increasing levels of abstraction, then that is Deep Learning process. Reference is taken from how brain works and deep learning methodology basically replicates the architecture of brain in the computer. Like, brain first extracts edges, then patches, then surfaces, then objects, etc.
- Deep learning is defined as the machine learning techniques that use supervised and / or unsupervised methods to automatically learn hierarchical / multiple levels of representations (multi-layered models of inputs – neural networks) in deep architectures for classification.
- Deep learning is a special approach in machine learning which seeks to extend "supervised, unsupervised and reinforcement learning" to address other problems in AI which are not usually included in machine learning such as *knowledge representation, reasoning, planning etc.*
- Deep learning, is deep-hierarchical levelled learning architecture, based on biological neurons within human brain.
- Deep learning is a greedy layer wise unsupervised pre-training that is to learn a hierarchy of features one level at a time.
- Deep learning is a subset of artificial intelligence field called as machine learning which is predicted on the basic idea of learning from example.
- Deep learning provides big data and web data analytics.
- Deep learning harnesses the power of deep neural networks in order to train models on large data sets.
- DL has very low trade-off between training time and computation error.



PROBLEM IN CONVENTIONAL NN AND ADVANTAGE OF DEEP LEARNING

A: Different types of feature extraction and machine learning techniques

Hand designed feature extraction → Trainable classifier → Object Class

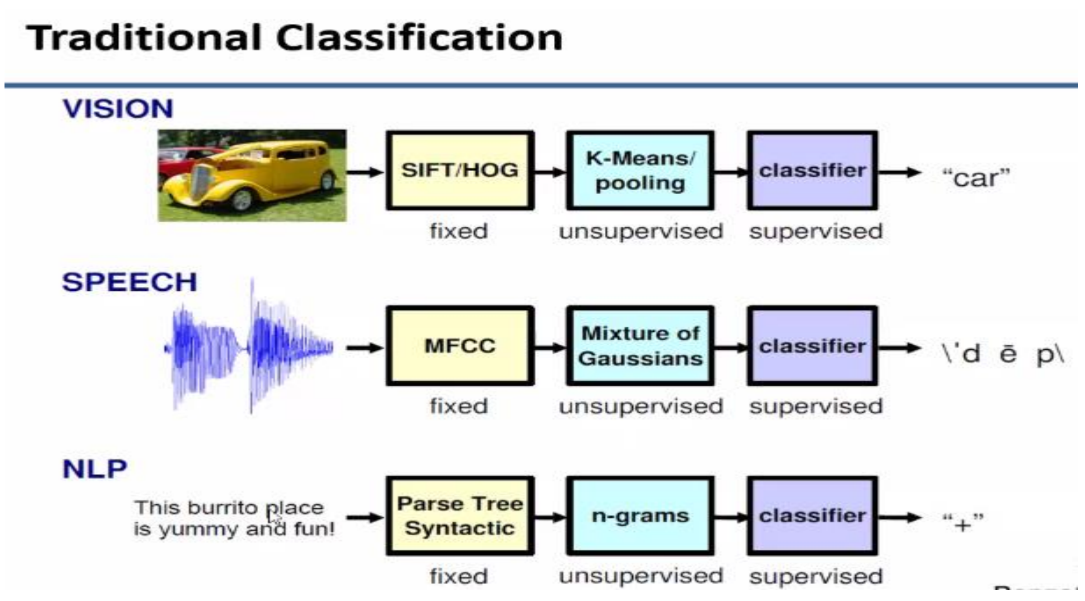
Feature	Machine Learning
Edges	Neural Network
Gabor	Naive Bayes
Wavelet	SVM
HoG , LBP	Ada Boost
SIFT . . .	Random Forest . . .

Which is important: ***Features or ML algorithms ?***

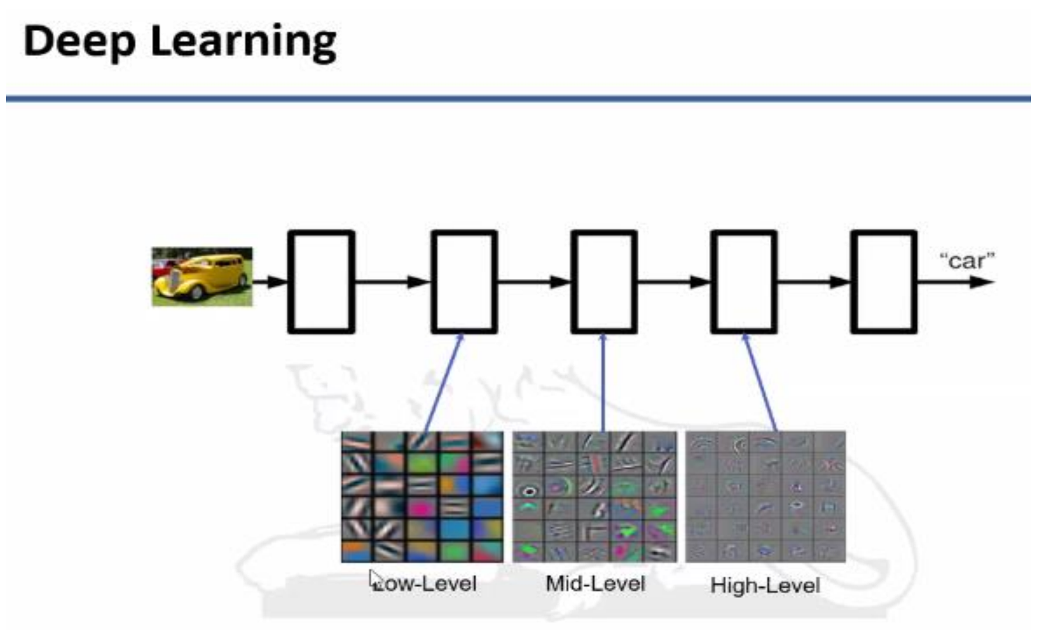
B: Difference between Machine learning and deep learning

7/26/2021

34



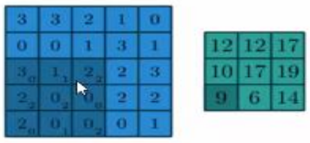
Traditional Classification



Deep Learning

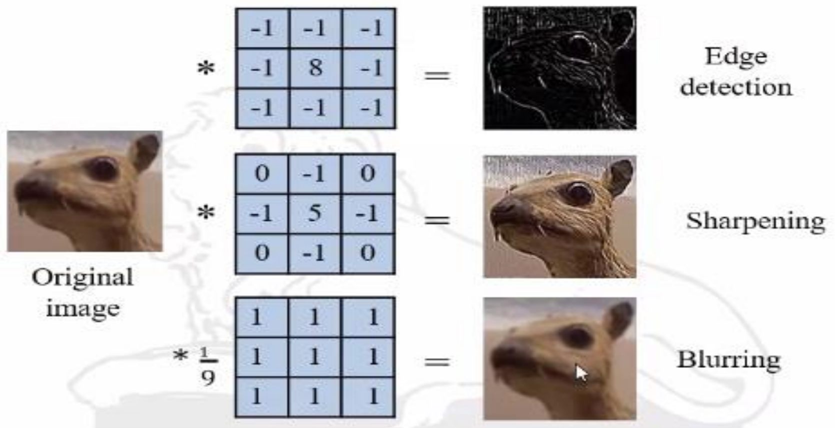
Convolutional Layer

- A discrete convolution operation is a linear transformation that preserves the notion of ordering (spatial and temporal).
- It is sparse as there are only a few input units contribute to a given output unit.
- It reuses parameters as the weights (kernel) are applied to multiple locations in the input.



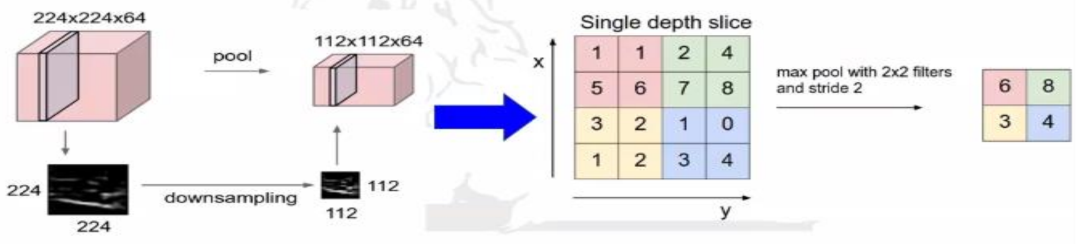
Here the light blue grid is the *input feature map* and shaded area is a kernel of value $\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$

Image Convolution: Different Kernels

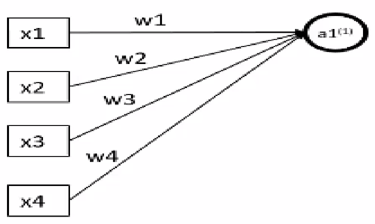


Pooling Layer

- Role of an aggregator.
- Invariance to Image transformation and increases compactness to representation.
- Pooling types: Max, Average,



A Simple Neural Network



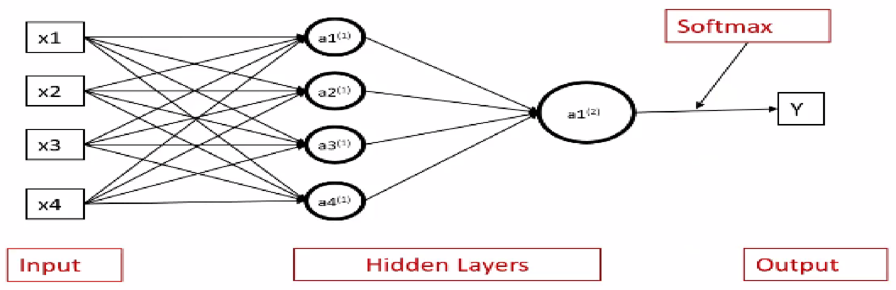
$$a1^{(1)} = f(w1 * x1 + w2 * x2 + w3 * x3 + w4 * x4)$$

$f(\,)$ is activation function: Relu or sigmoid

$$Relu: \max(0, x)$$

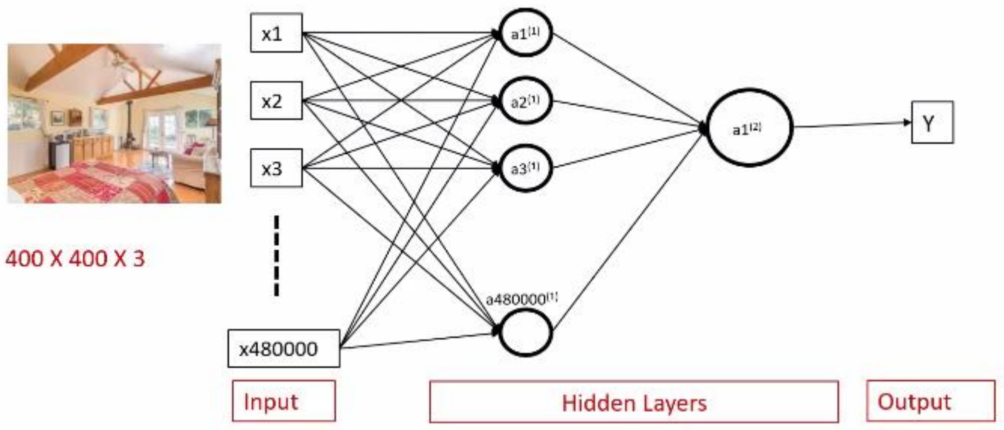
$$a1^{(1)} = max(0, w1 * x1 + w2 * x2 + w3 * x3 + w4 * x4)$$

Number of Parameters



4*4 + 4 +1

If the input is an Image?

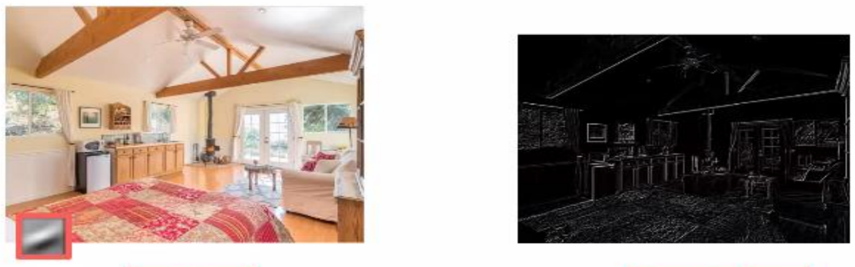


Number of Parameters

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

Convolutional Layers

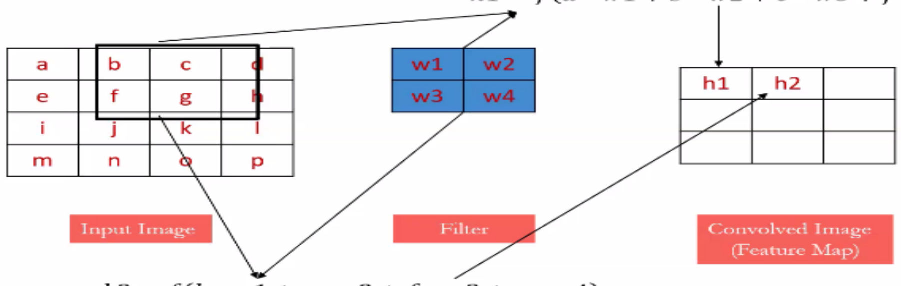
- Filter $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$



Convolutional Layers

- What is Convolution?

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



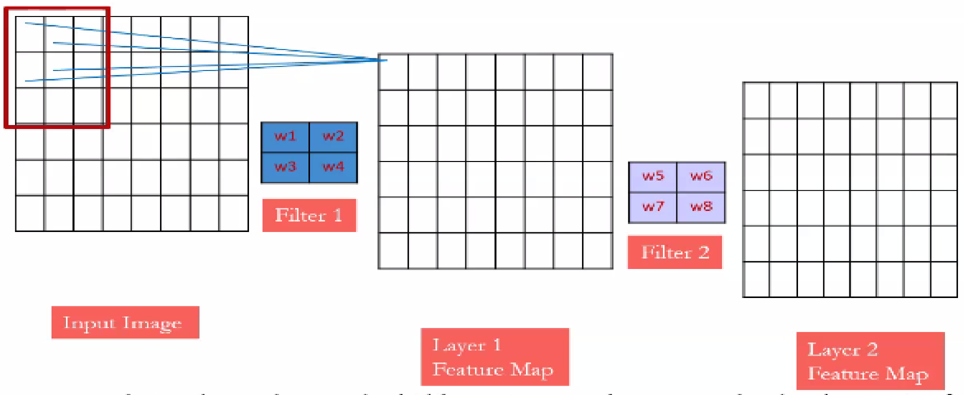
$$h2 = f(b * w1 + c * w2 + f * w3 + g * w4)$$

Number of Parameters for one feature map = 4

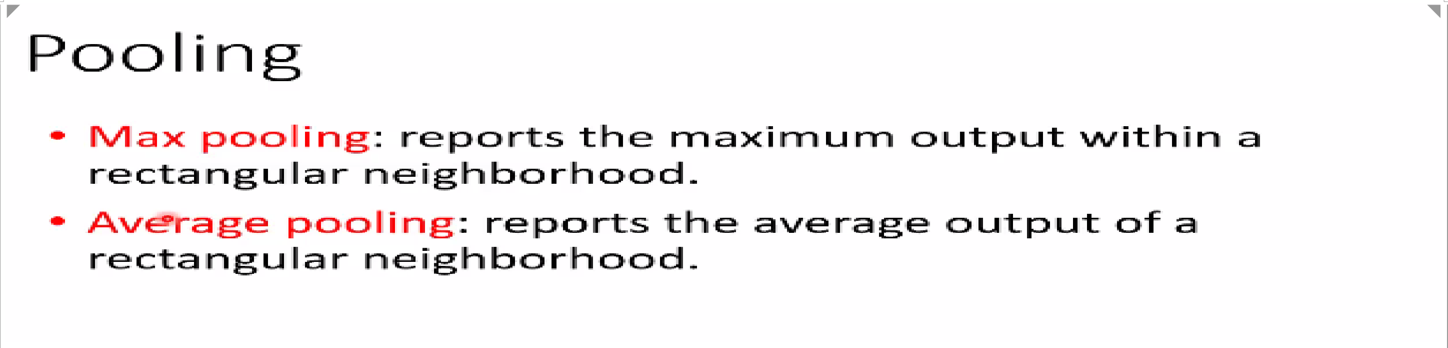
Number of Parameters for 100 feature map = 4*100

2021 ATAL FDP on AI for Computer Vision, IIT Roorkee, 16 July 2021

Lower Level to More Complex Features



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



Pooling

- Max pooling: reports the maximum output within a rectangular neighborhood.
- Average pooling: reports the average output of a rectangular neighborhood.

Advantages

(1) Non-linear functions can be represented in a more efficient manner.

(2) Deep representations may allow for hierarchical representations that allow for combinatorial sharing of statistical strength.

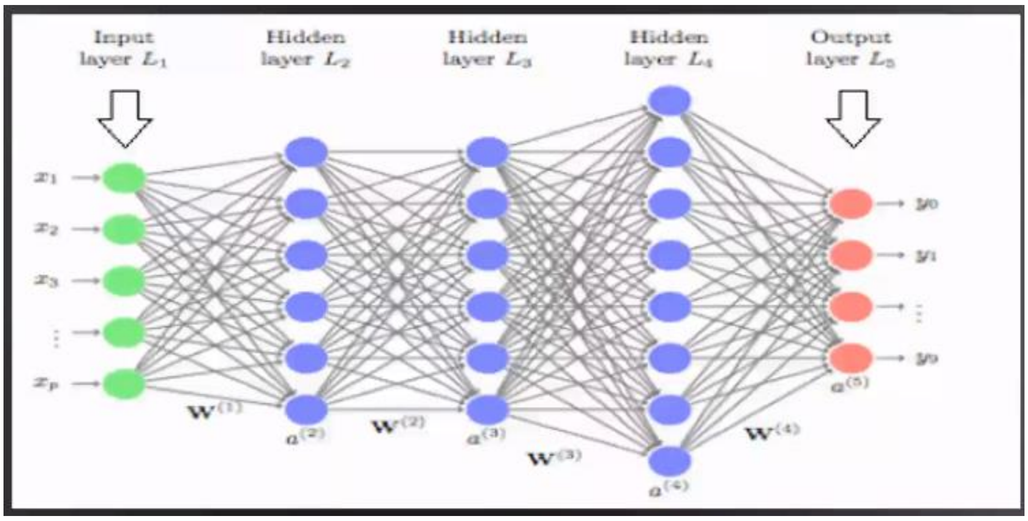
Hyper-parameters of Deep Learning

Learning rate, number of hidden layers, number of hidden units, number of iterations to get the minimas, activation function, momentum, batch size, Regularization etc.

Challenges created by BIG DATA to Deep Learning

- Large Scale data
- Heterogeneity of data
- Noisy labels
- Non-stationary distribution

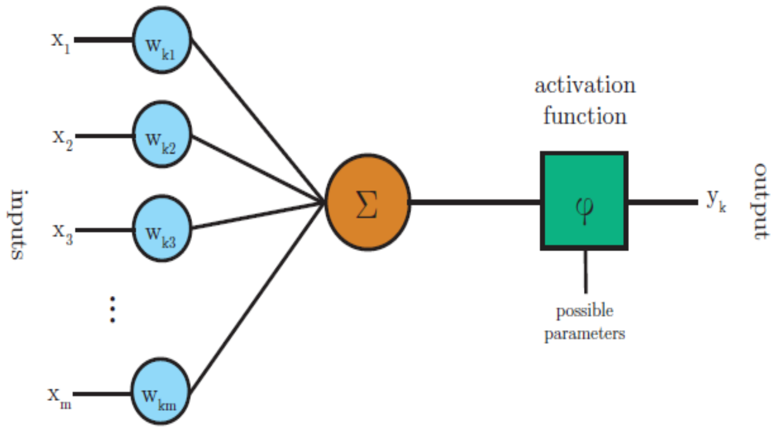
Deep Learning (based on human neuron structure)

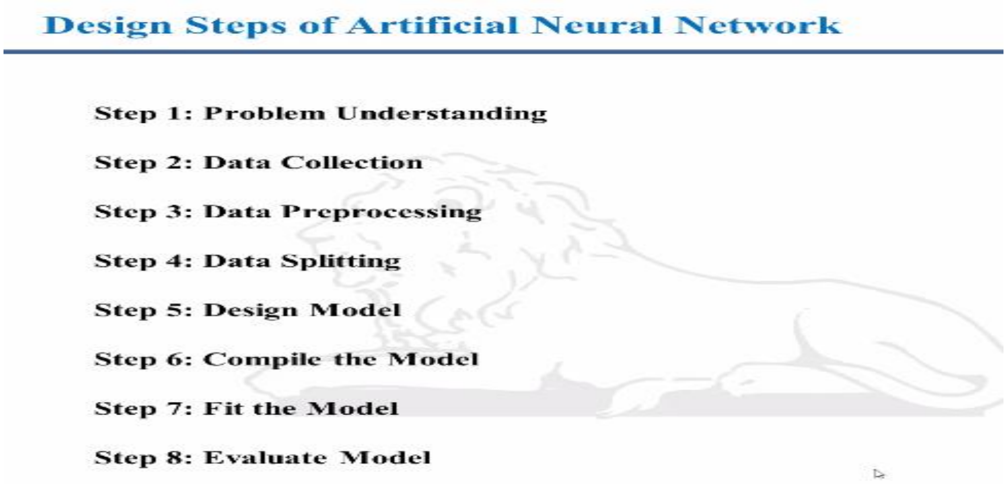


Artificial Neuron Structure

- It is seen in the diagram; neuron k receives m input parameters x_j. The neuron also has m weight parameters w_{kj}. The inputs and weights are linearly combined and summed. The sum is then fed to an activation function φ which produces the output y_k of the neuron:

$$\boldsymbol{y_k = \varphi(s_k) = \varphi(\sum_{j=0}^{m} w_{kj}x_j)}$$





Design Steps of Artificial Neural Network

Step 1: Problem Understanding

Step 2: Data Collection

Step 3: Data Preprocessing

Step 4: Data Splitting

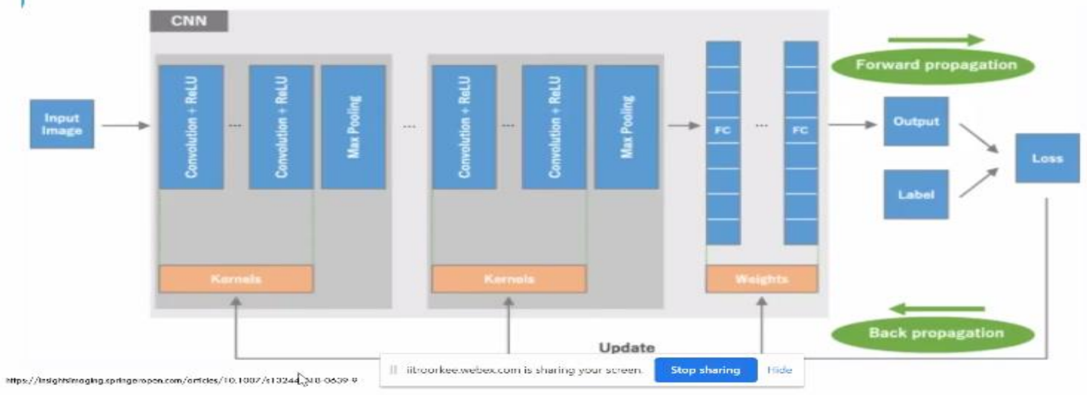
Step 5: Design Model

Step 6: Compile the Model

Step 7: Fit the Model

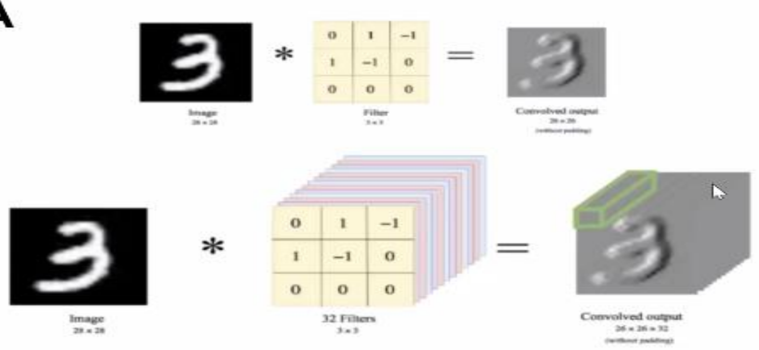
Step 8: Evaluate Model

CNN ARCHITECTURE AND LEARNING

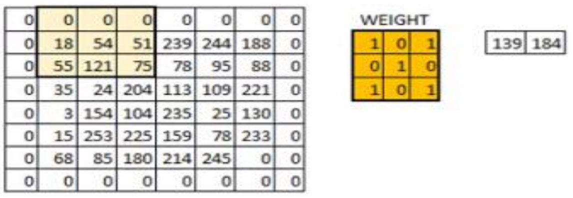


CNN-1. FEATURE EXTRACTION (EDGE DETECTION)

A



B



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	184

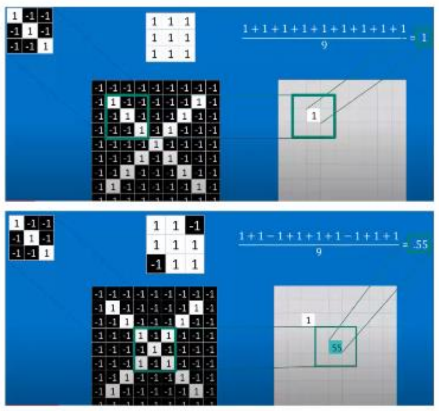
https://www.youtube.com/watch?v=FmpDIaiMIeA&ab_channel=BrandonRohrer
https://www.programmersought.com/article/59363490490/
7/26/2021

40

CNN-1. FEATURE EXTRACTION(HOW IT WORKS?)

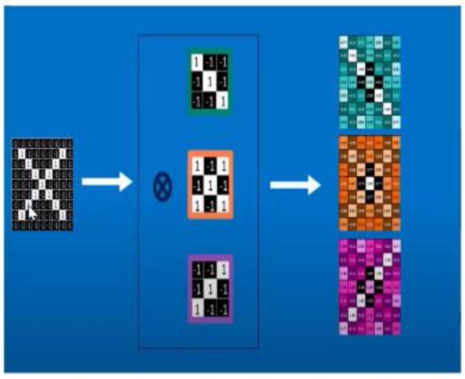
A

B



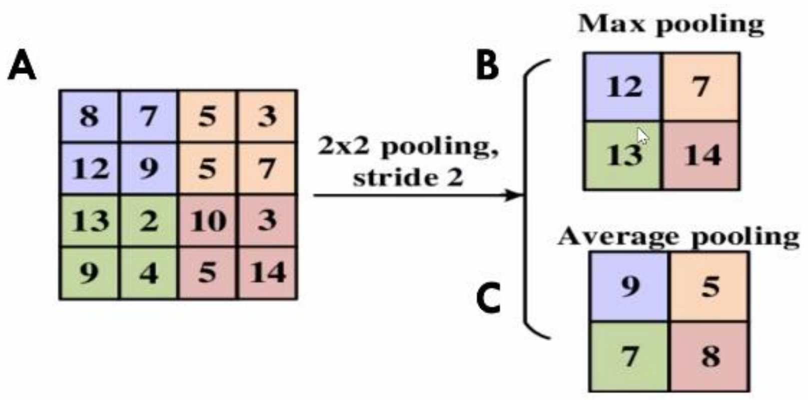
$\frac{1+1+1+1+1+1+1+1+1}{9} = 1$
$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$

C



https://www.youtube.com/watch?v=FmpDIaiMIeA&t=2059s&ab_channel=BrandonRohrer

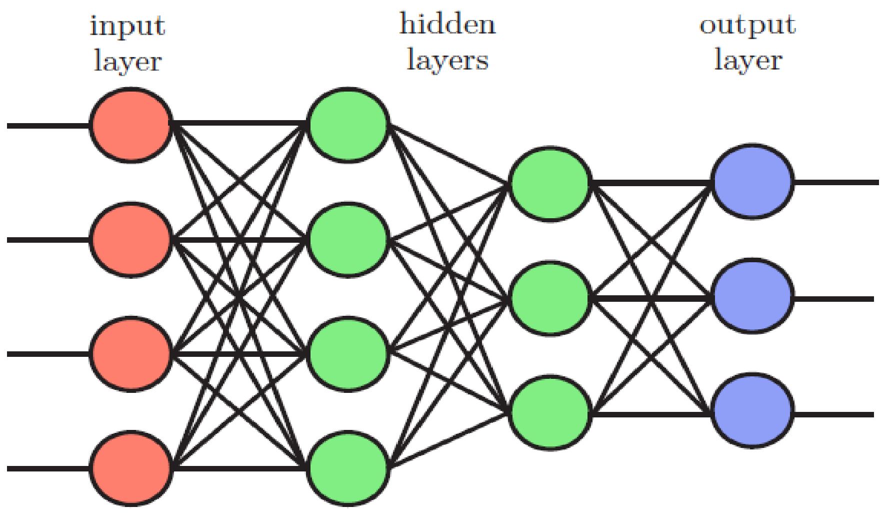
CNN-1. FEATURE EXTRACTION (POOLING)

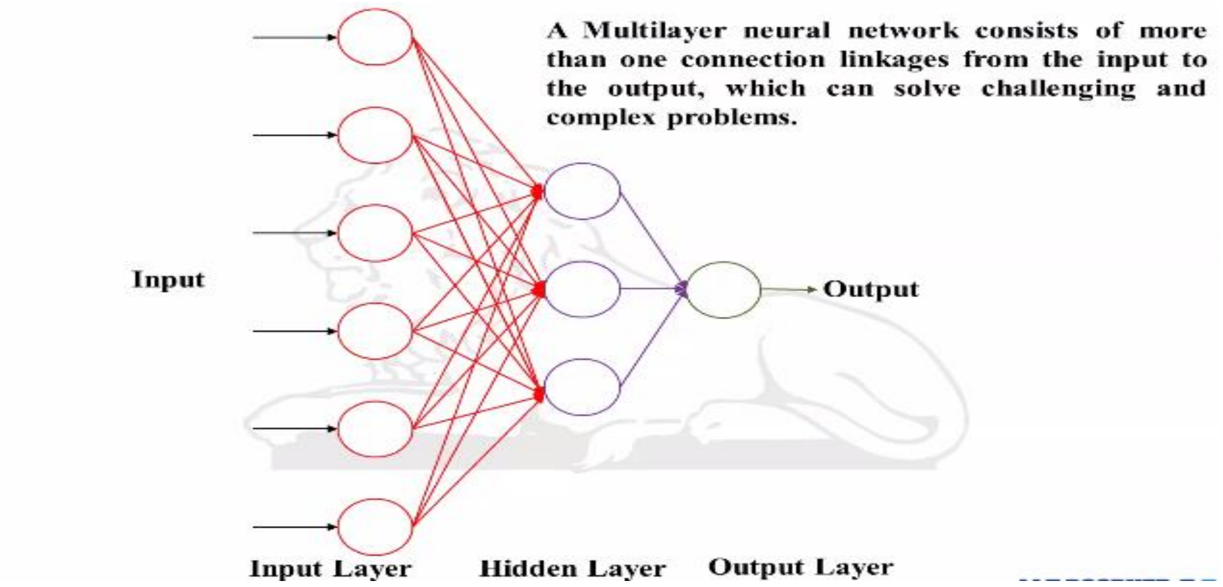


https://www.youtube.com/watch?v=FmpDIaiMIeA&ab_channel=BrandonRohrer

Multi-layer Networks

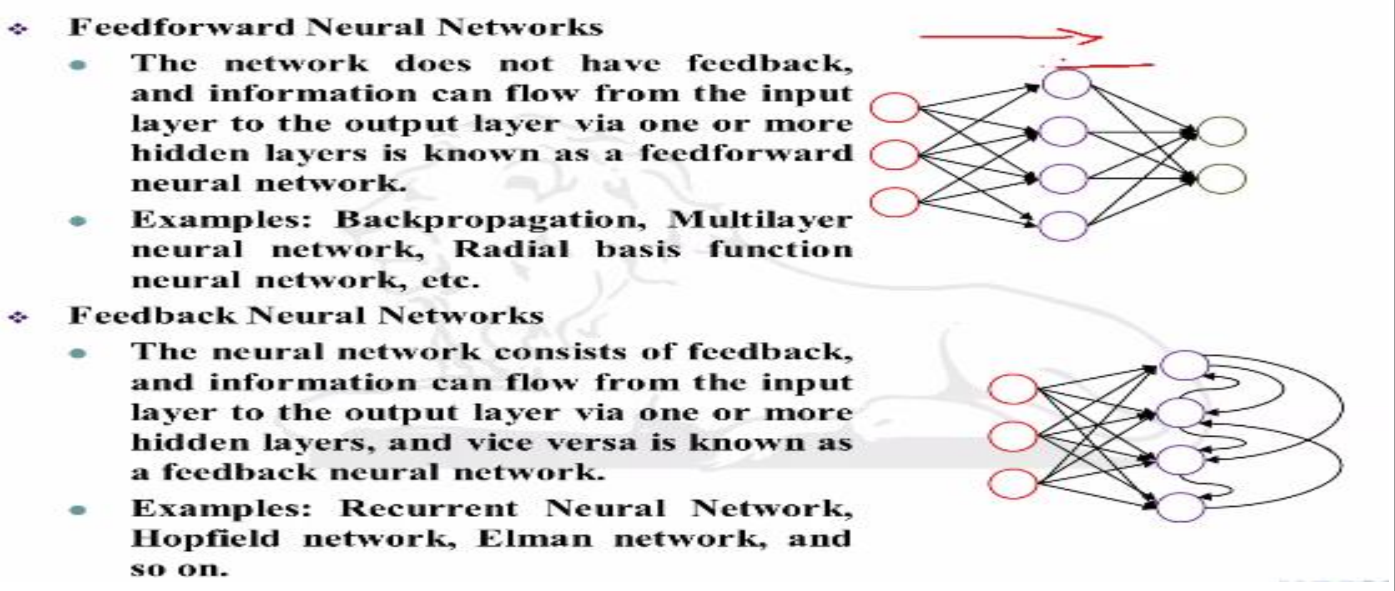
- A fully connected feed-forward multi-layer network is shown in the network, where neurons are typically grouped into layers and each output of a layer of neurons is fed as input to each neuron of the next layer. Some layers process original input data and some process data received from other neurons. Each neuron has a number of weights equal to the number of neurons in the previous layer.
- A multi-layer network includes three types of layers: an input layer, one or more hidden layers and an output layer.
- Input layer passes data only without modifying that.
- Computations are carried out in the hidden layers.
- Output layer converts the hidden layer activations to an output (like classification)



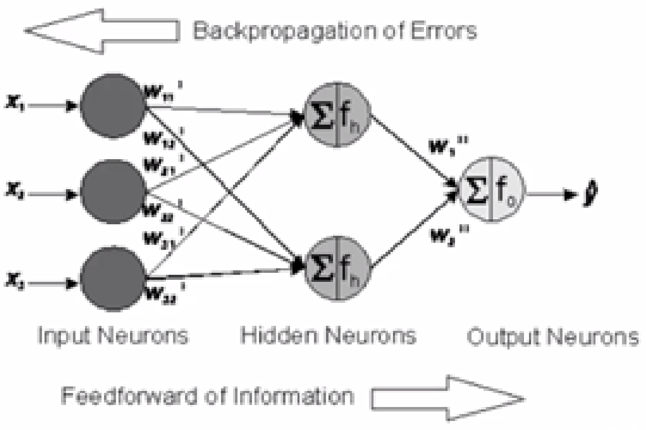

A Multilayer neural network consists of more than one connection linkages from the input to the output, which can solve challenging and complex problems.

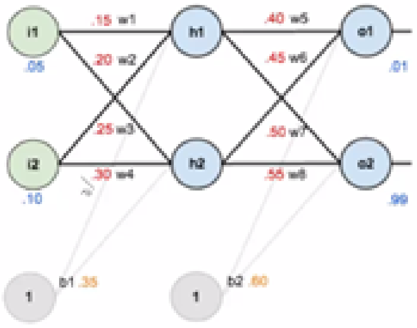
Back Propagation

- A neural network is trained by selecting the weights of all neurons so that the network learns to approximate target outputs from known inputs, but analytically it is difficult to calculate the neuron weights.
- Back-propagation algorithm does weight calculation iteratively.


- Feedforward Neural Networks
  - The network does not have feedback, and information can flow from the input layer to the output layer via one or more hidden layers is known as a feedforward neural network.
  - Examples: Backpropagation, Multilayer neural network, Radial basis function neural network, etc.
- Feedback Neural Networks
  - The neural network consists of feedback, and information can flow from the input layer to the output layer via one or more hidden layers, and vice versa is known as a feedback neural network.
  - Examples: Recurrent Neural Network, Hopfield network, Elman network, and so on.

TRAINING: BACK PROPAGATION





For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

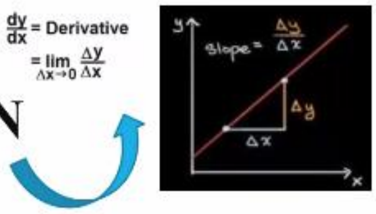
Calculating the Total Error $E_{total} = \sum \frac{1}{2}(target - output)^2$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

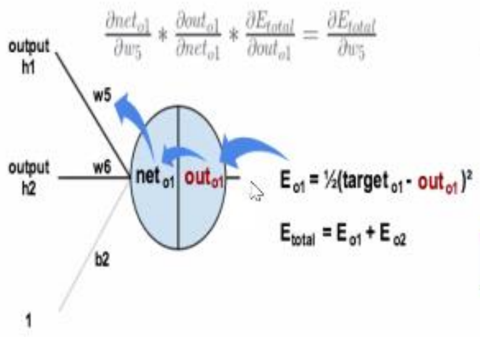
$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

7/26/2020


$\frac{dy}{dx}$ = Derivative
$= \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$
Slope = $\frac{\Delta y}{\Delta x}$
Δy
Δx
y
x

TRAINING: BACK PROPAGATION CALCULATION (1)

First, how much does the total error change with respect to the output?


$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
output h2
w5
w6
b2
1
net_{o1}
out_{o1}
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$x^2 \longrightarrow n = 2$$

$$\frac{d}{dx} x^n = nx^{n-1}$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

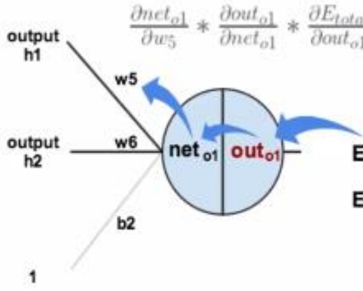
$$\boxed{\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507}$$

Eq-1

TRAINING: BACK PROPAGATION CALCULATION (2)

Next, how much does the output of o_1 change with respect to its total net input?

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

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

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

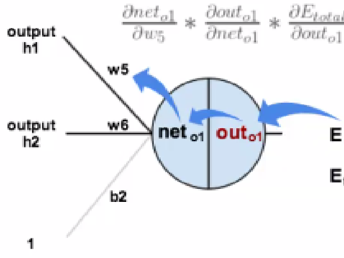
$$\boxed{\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602}$$

Eq-2

TRAINING: BACK PROPAGATION CALCULATION (3)

Finally, how much does the total net input of $o1$ change with respect to w_5?

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\boxed{\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992}$$

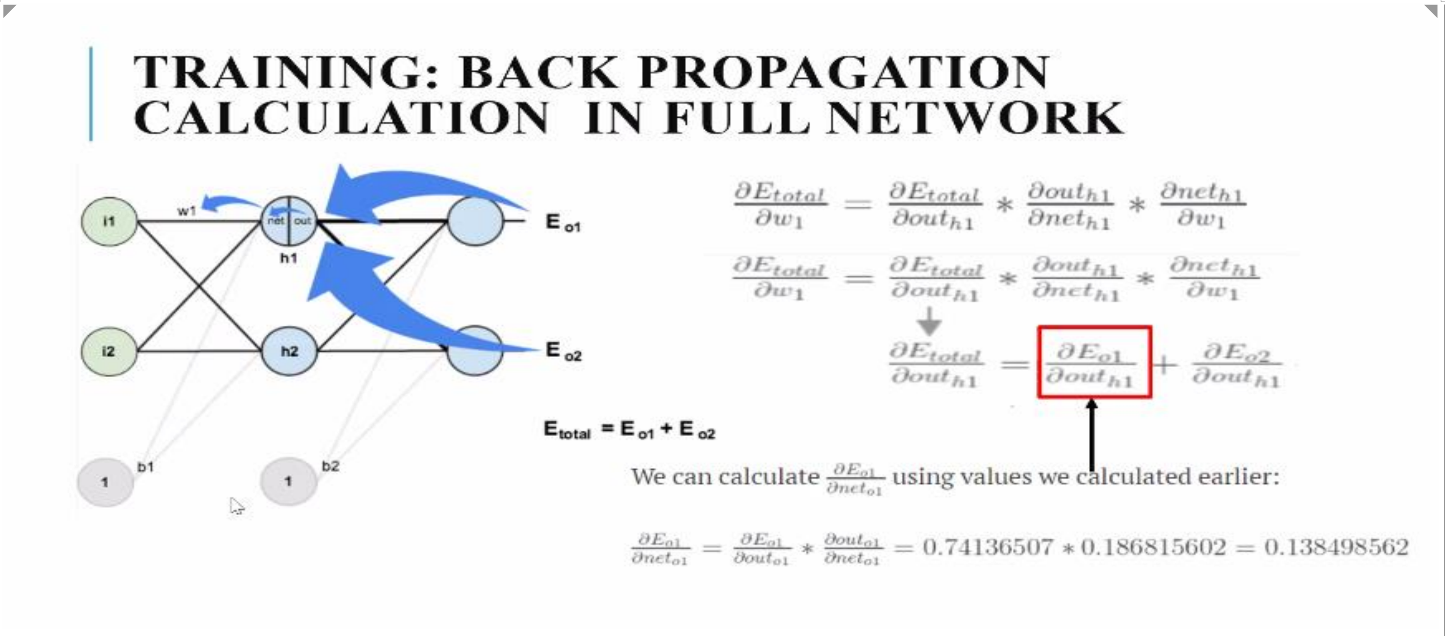
Eq-3

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\boxed{\frac{\partial E_{total}}{\partial w_5} = \underbrace{0.74136507}_{\text{Eq-1}} * \underbrace{0.186815602}_{\text{Eq-2}} * \underbrace{0.593269992}_{\text{Eq-3}} = 0.082167041}$$

New weight of w5:

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$



TRAINING: BACK PROPAGATION CALCULATION IN FULL NETWORK

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \boxed{\frac{\partial E_{o1}}{\partial out_{h1}}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$E_{total} = E_{o1} + E_{o2}$

We can calculate $\frac{\partial E_{o1}}{\partial net_{o1}}$ using values we calculated earlier:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

Deep Auto-Encoder

An auto-encoder is an unsupervised feed-forward neural network which is trained to predict the input itself. The objective is to learn a compressed representation / encoding for a data set. This means that

It was being used for dimensionality reduction and feature discovery.

Deep Learning Libraries:

- caffe
- theano
- torch
- deeplearning4j (dl4j)
- mocha

Deep Learning CASE TOOLS:

- Apache mahout
- Scikit-learn
- Openai
- Tensor Flow
- Char-mn
- Paddle Paddle
- Cntx

- Apache Singa
- Deeplearning4j
- H2O, etc.

Tensor Flow – Python based open source software library released by Google in 2015 to design, build and train deep learning models.

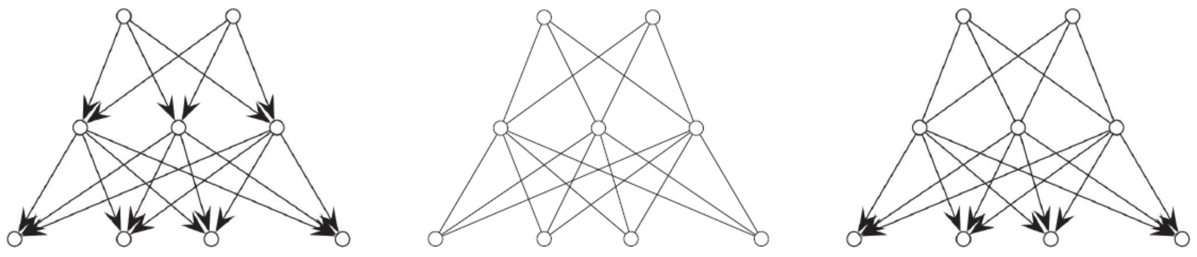
Deep Generative Models

Deep generative models are the replacement of Deep models. Because, in deep models, huge amount of labelled data can be acquired and must be trained, which increases the difficulty. Like for hand-written character recognition, lots of labelled data can be generated by modifying labelled training set manually. To overcome this, three deep generative models are formed: directed, undirected and mixed.

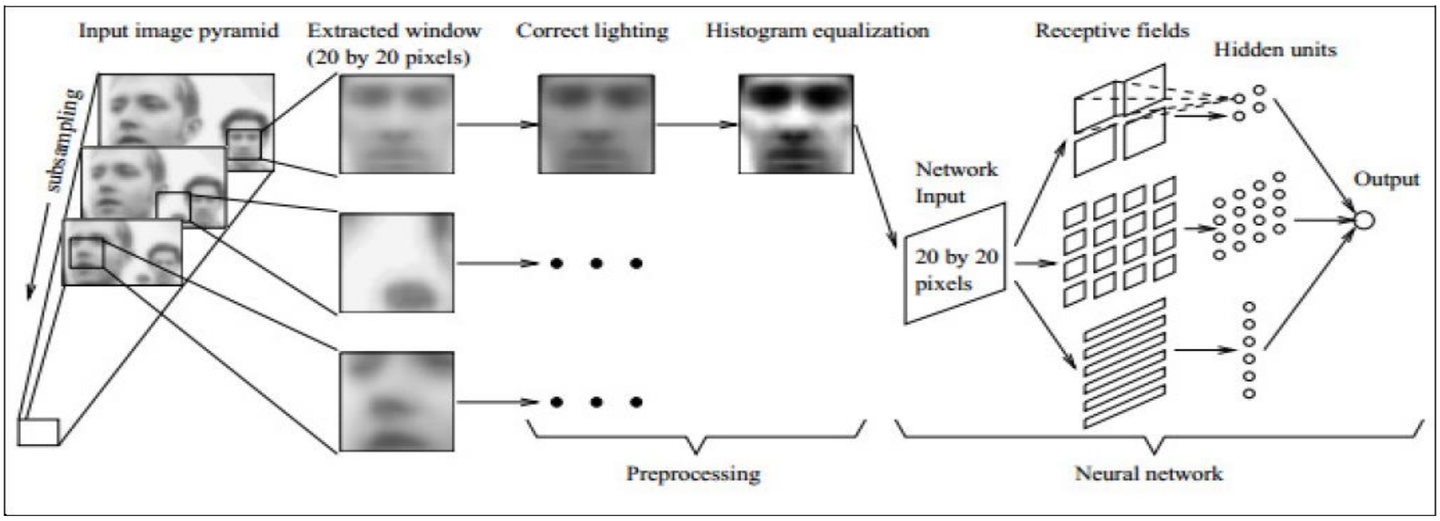
For the directed model (**Deep Directed Network – DDN**), bottom level contains the observed pixels or data and remaining layers are hidden.

Undirected model (**Deep Boltzmann Machines – DBM**) has certain edge over directed model, as in undirected model, all the nodes in each layer are conditionally independent of each other given the layers above and below. Disadvantage of undirected model is, for having **partition function,** it is difficult to train it.

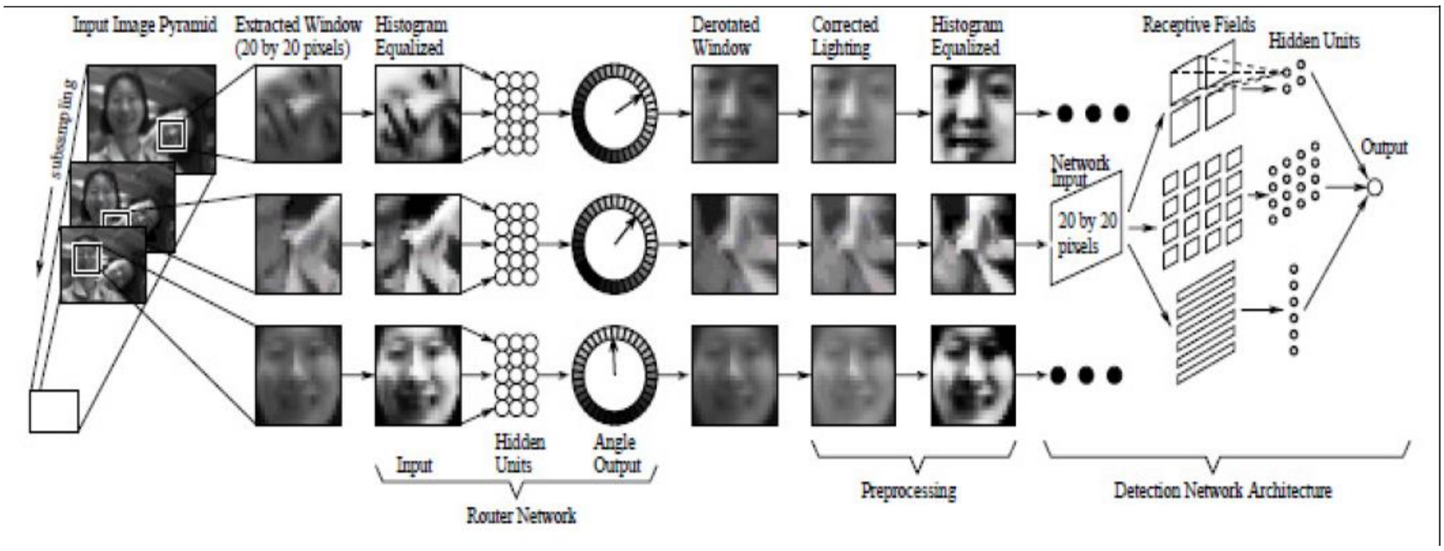
For learning deep undirected models, greedy layer-wise model can be used which is partially directed and partially undirected (**Deep Belief Network – DBN).** The layered model can have directed arrows, except at top, where is undirected bipartite graph. In this model, top 2 layers' act as an associative memory and the remaining layers then generate the output. In this model, hidden states can be inferred fast and bottom-up fashion.



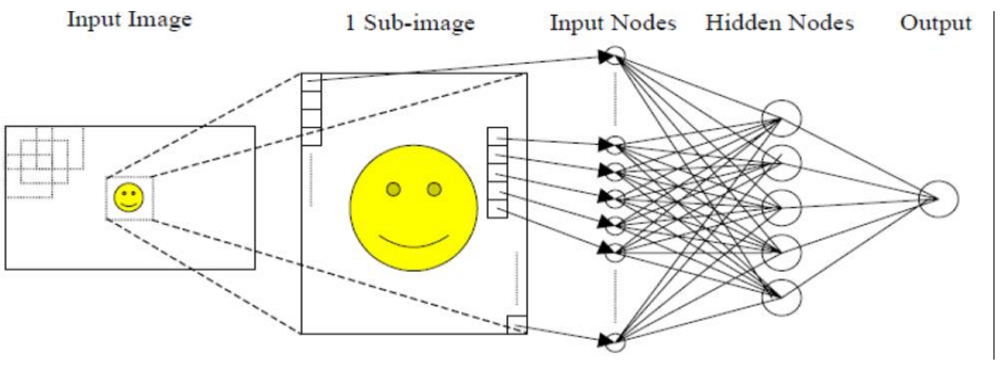
Retinal Connected Neural Network



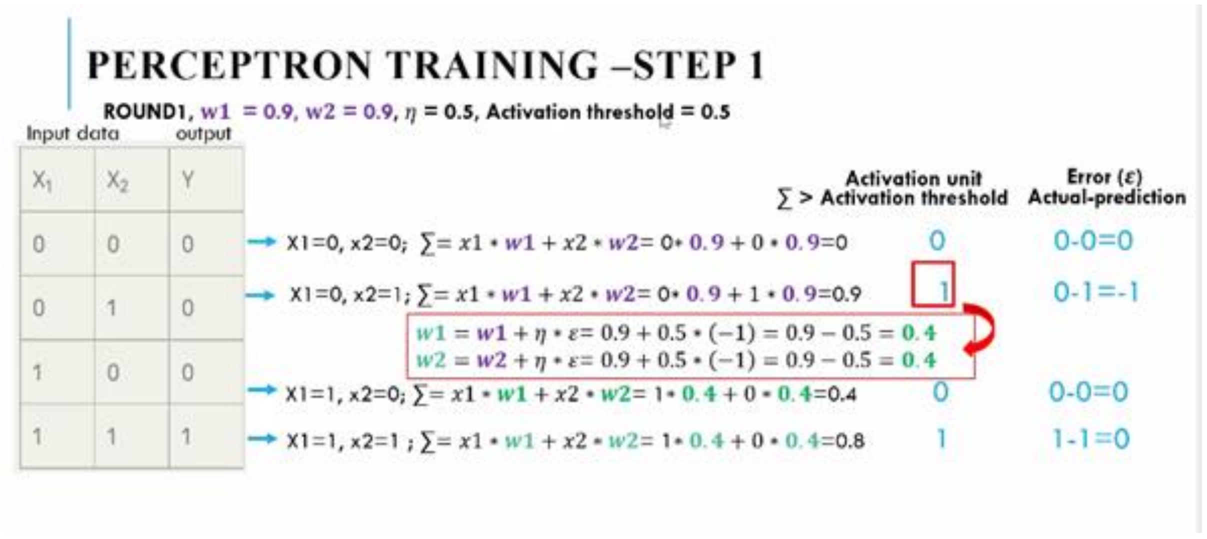
Rotation Invariant Neural Network



Principal Component Analysis and Artificial Neural Network



PERCEPTRON TRAINING



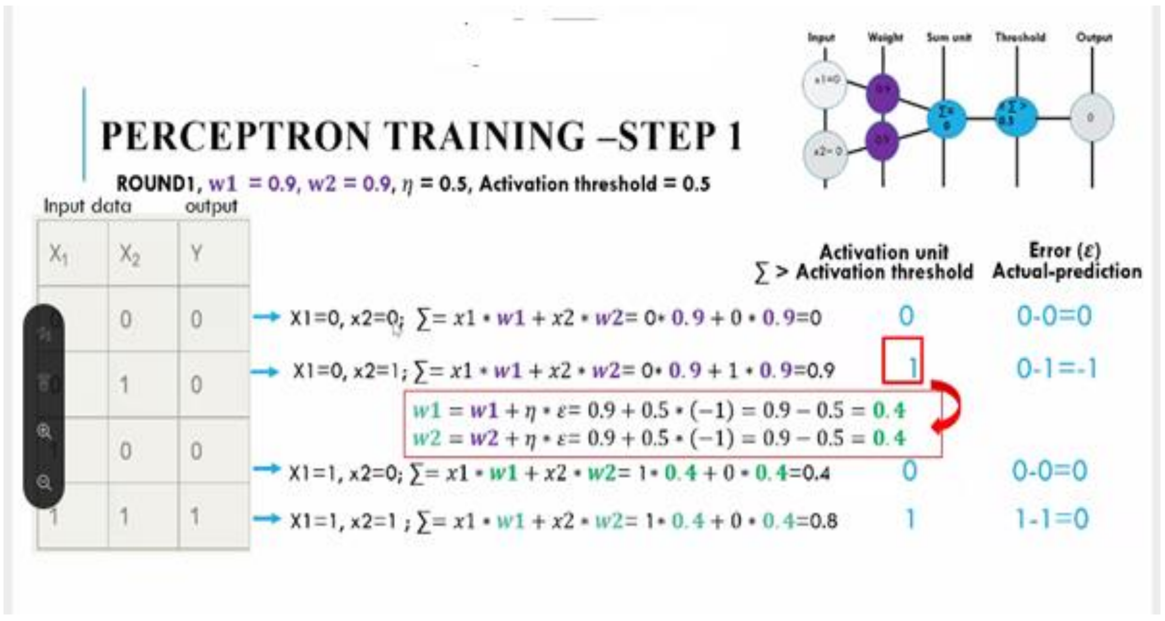
PERCEPTRON TRAINING –STEP 1

ROUND1, $w1 = 0.9$, $w2 = 0.9$, $\eta = 0.5$, **Activation threshold = 0.5**

Input data / output

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

	Activation unit $\sum >$ Activation threshold	Error (ε) Actual-prediction
→ X1=0, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 0 \cdot 0.9 = 0$	0	0-0=0
→ X1=0, x2=1; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 1 \cdot 0.9 = 0.9$	$\boxed{1}$	0-1=-1
$w1 = w1 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
$w2 = w2 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
→ X1=1, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.4$	0	0-0=0
→ X1=1, x2=1 ; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.8$	1	1-1=0



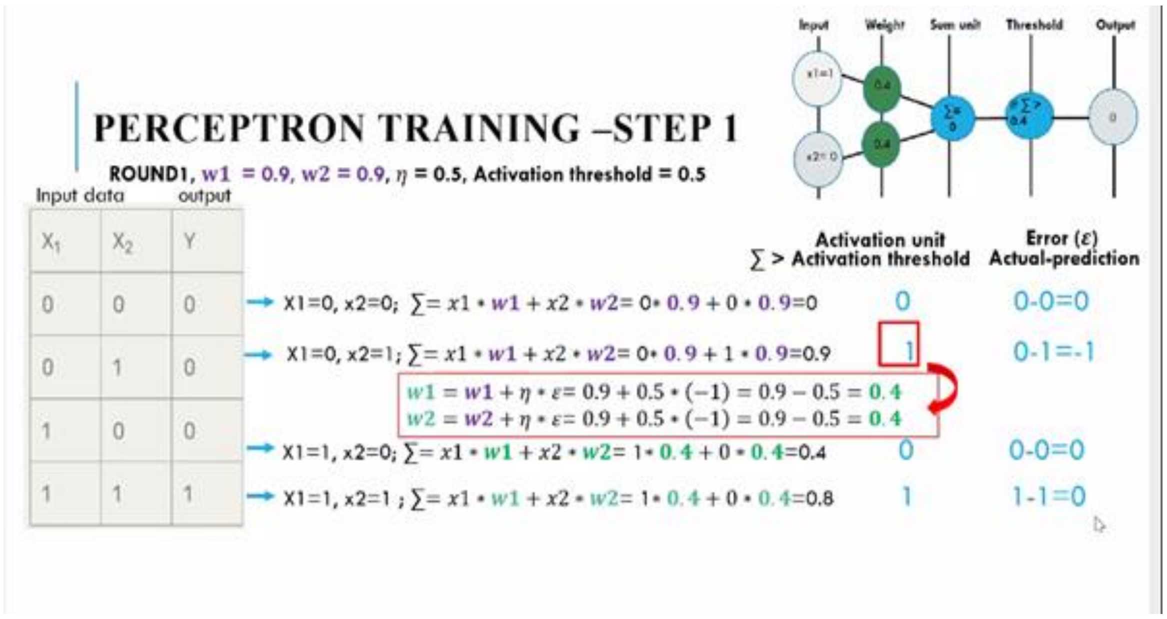
PERCEPTRON TRAINING –STEP 1

ROUND1, $w1 = 0.9$, $w2 = 0.9$, $\eta = 0.5$, **Activation threshold = 0.5**

Input data / output

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

	Activation unit $\sum >$ Activation threshold	Error (ε) Actual-prediction
→ X1=0, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 0 \cdot 0.9 = 0$	0	0-0=0
→ X1=0, x2=1; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 1 \cdot 0.9 = 0.9$	$\boxed{1}$	0-1=-1
$w1 = w1 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
$w2 = w2 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
→ X1=1, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.4$	0	0-0=0
→ X1=1, x2=1 ; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.8$	1	1-1=0



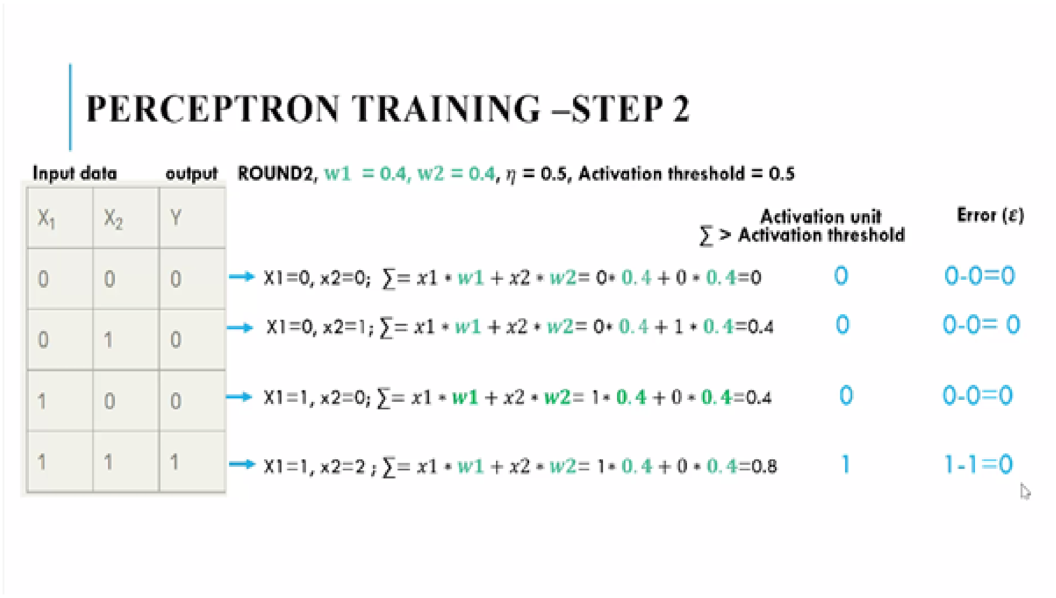
PERCEPTRON TRAINING –STEP 1

ROUND1, $w1 = 0.9$, $w2 = 0.9$, $\eta = 0.5$, **Activation threshold = 0.5**

Input data / output

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

	Activation unit $\sum >$ Activation threshold	Error (ε) Actual-prediction
→ X1=0, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 0 \cdot 0.9 = 0$	0	0-0=0
→ X1=0, x2=1; $\sum = x1 \cdot w1 + x2 \cdot w2 = 0 \cdot 0.9 + 1 \cdot 0.9 = 0.9$	$\boxed{1}$	0-1=-1
$w1 = w1 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
$w2 = w2 + \eta \cdot \varepsilon = 0.9 + 0.5 \cdot (-1) = 0.9 - 0.5 = 0.4$		
→ X1=1, x2=0; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.4$	0	0-0=0
→ X1=1, x2=1 ; $\sum = x1 \cdot w1 + x2 \cdot w2 = 1 \cdot 0.4 + 0 \cdot 0.4 = 0.8$	1	1-1=0



PERCEPTRON TRAINING –STEP 2

Input data output ROUND2, w1 = 0.4, w2 = 0.4, η = 0.5, Activation threshold = 0.5

X_1	X_2	Y		Activation unit $\sum$ > Activation threshold	Error (ε)
0	0	0	X1=0, x2=0; $\sum = x1 * w1 + x2 * w2 = 0 * 0.4 + 0 * 0.4 = 0$	0	0-0=0
0	1	0	X1=0, x2=1; $\sum = x1 * w1 + x2 * w2 = 0 * 0.4 + 1 * 0.4 = 0.4$	0	0-0= 0
1	0	0	X1=1, x2=0; $\sum = x1 * w1 + x2 * w2 = 1 * 0.4 + 0 * 0.4 = 0.4$	0	0-0=0
1	1	1	X1=1, x2=2 ; $\sum = x1 * w1 + x2 * w2 = 1 * 0.4 + 0 * 0.4 = 0.8$	1	1-1=0