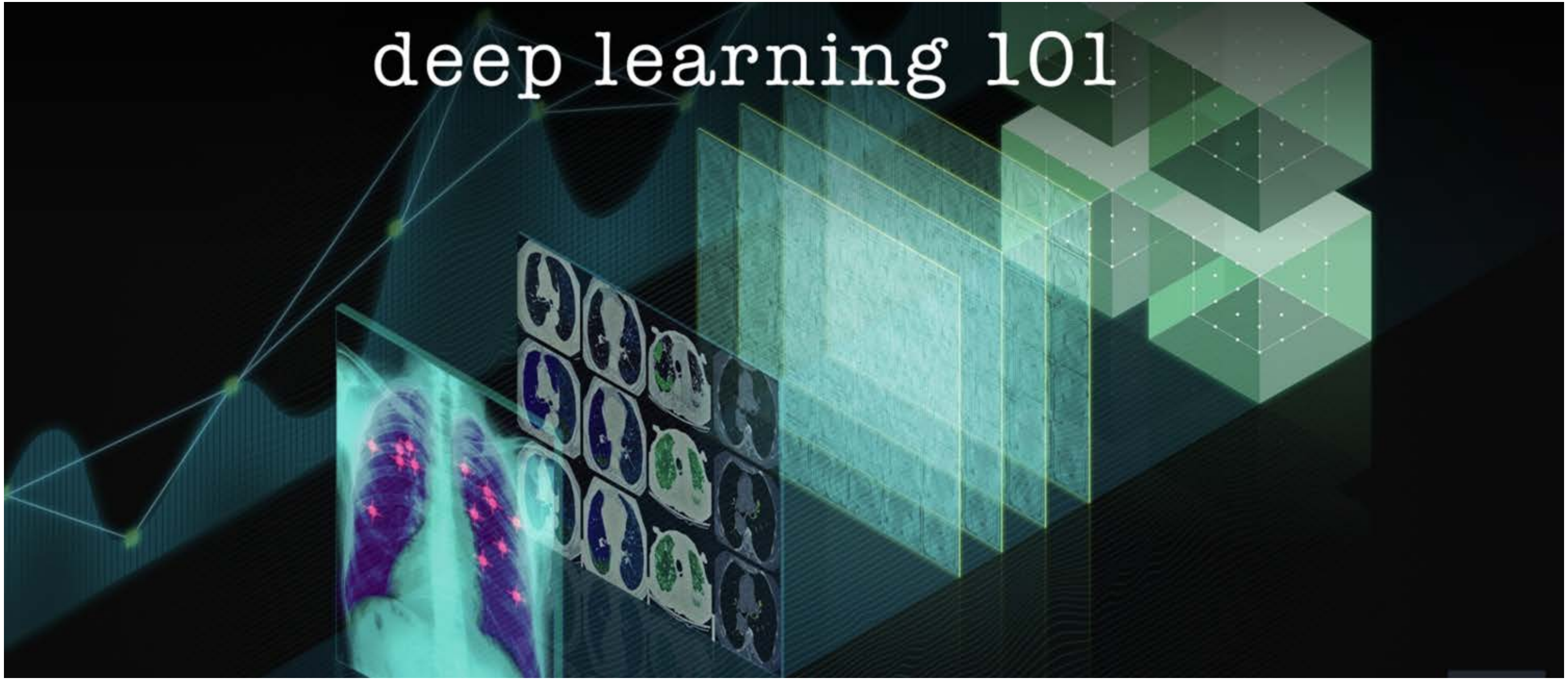
Introduction to Neural Networks and Deep Learning



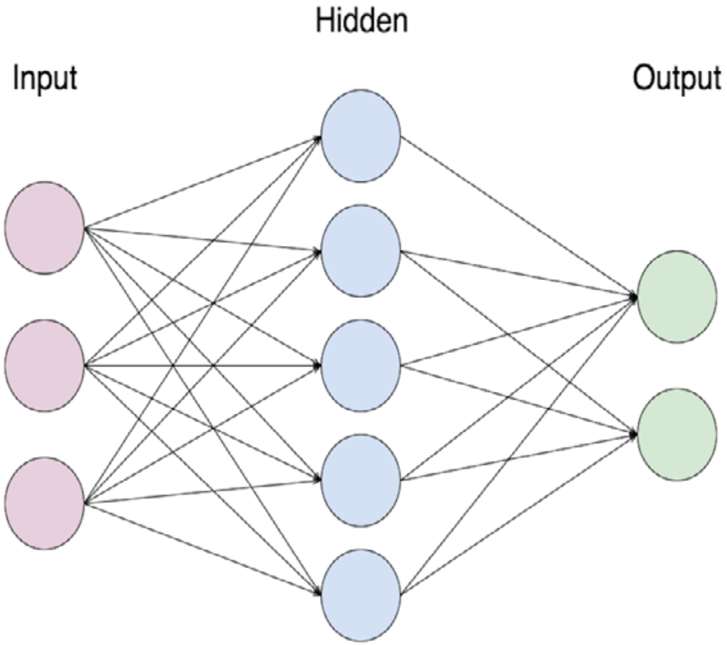
Agenda

1. **Neural Network Building Blocks**
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron
2. Why we need Neural Networks
3. Working of a Neural Network
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks
    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

1. Artificial Neural Network (ANNs)



- Composed of a large number of highly interconnected processing elements called neurons

- Like people, learn by example
- Configured for specific applications
- Like pattern recognition or data classification

Agenda

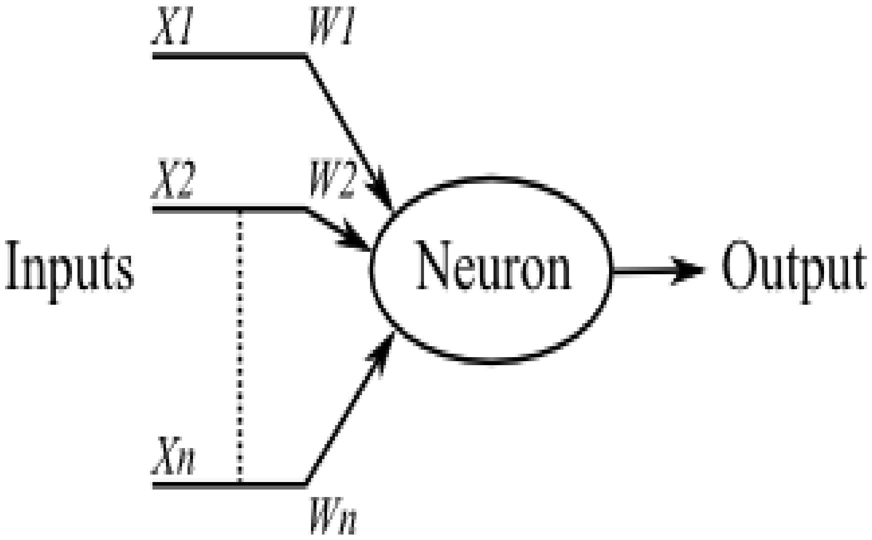
1. Neural Network Building Blocks
    - **What is a Neuron?**
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron
2. Why we need Neural Networks
3. Working of a Neural Network
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

- Hidden Layers
- Why Deep Neural Networks?
- Types of DNNs
    - Multilayer Perceptron (ANN)
    - Convolutional Neural Network
    - Recurrent Neural Network
    - Generative Adversarial Networks (GANs)

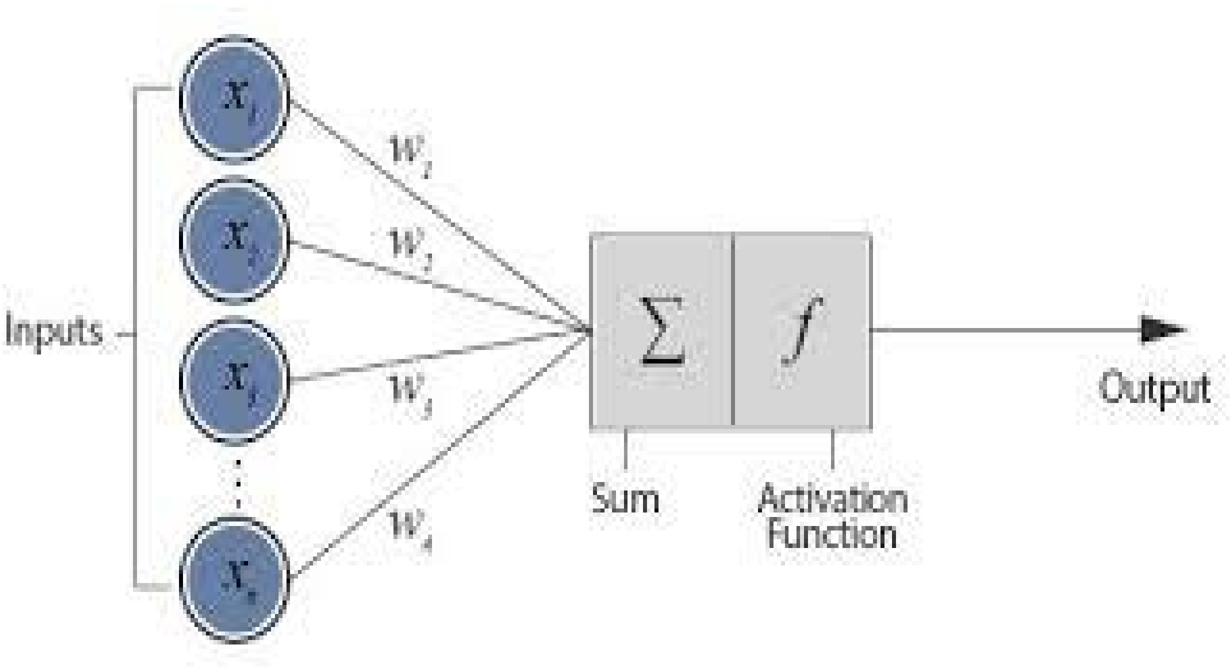
5. Applications of Deep Neural Networks

What is a Neuron?



- A device with many inputs and one output
- Also called McCulloch and Pitts model
- Effect that each input has at decision making depends on the weight of the particular input

Can You Suggest the Final Outcome?



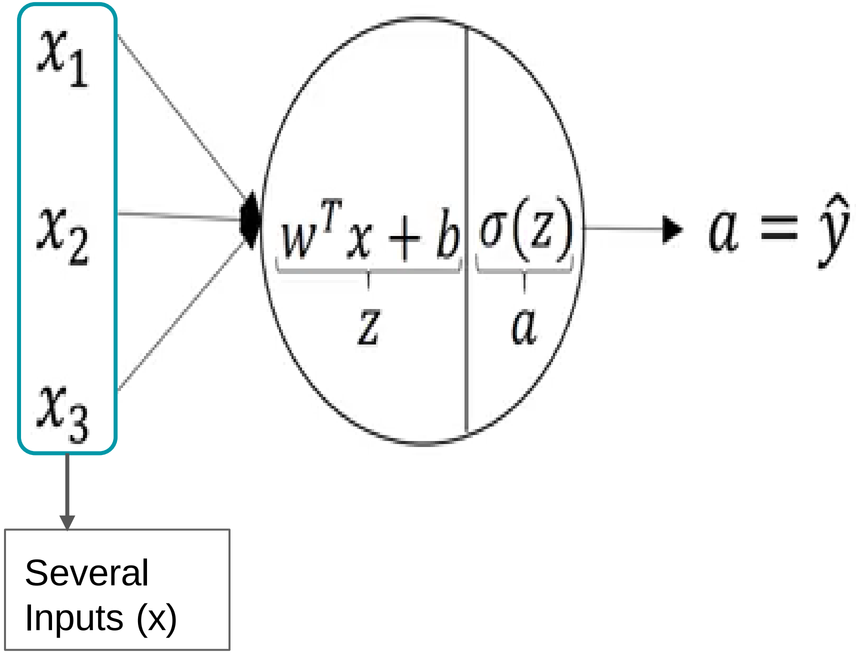
Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - **Working of a Neuron**
    - Analogy with Human Brain
    - Perceptron
2. Why we need Neural Networks
3. Working of a Neural Network
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks
    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

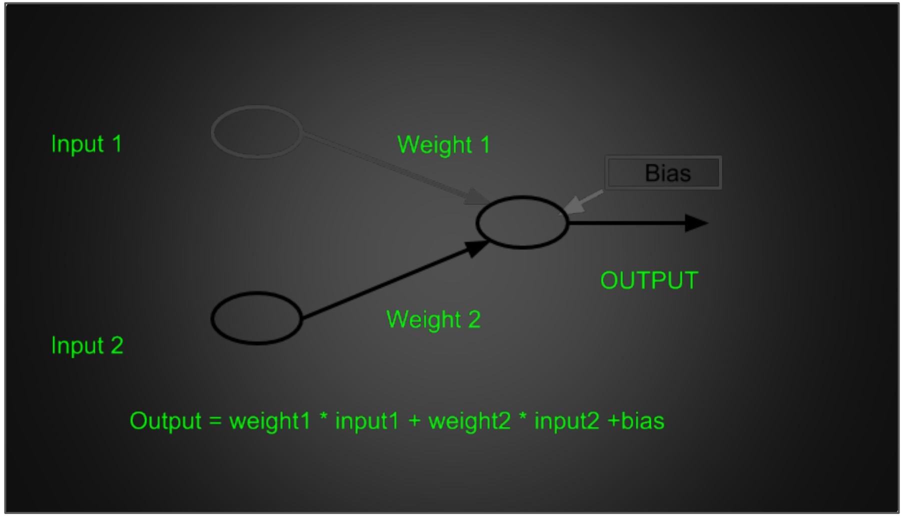
5. Applications of Deep Neural Networks

Working of a Neuron



- First, a neuron computes the matrix product of the weights and inputs
- It also adds a bias to the above term
- Second, it uses an activation function to get the output

Working of a Neuron - Summation

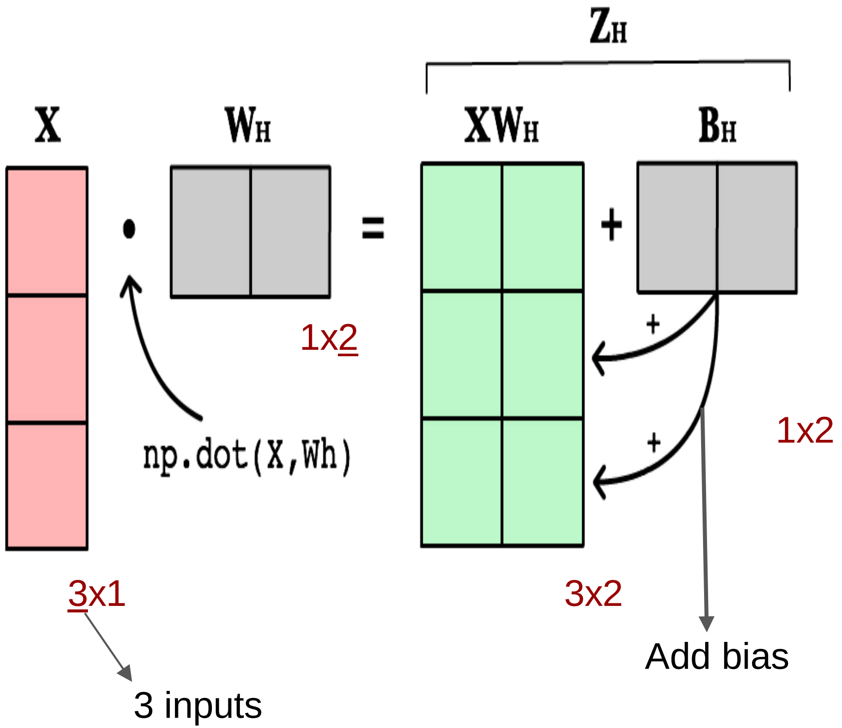


- Firstly, the inputs are matrix multiplied to the weight vector

$$\begin{bmatrix} Input1 & Input2 \end{bmatrix} * \begin{bmatrix} Weight1 \\ Weight2 \end{bmatrix}$$

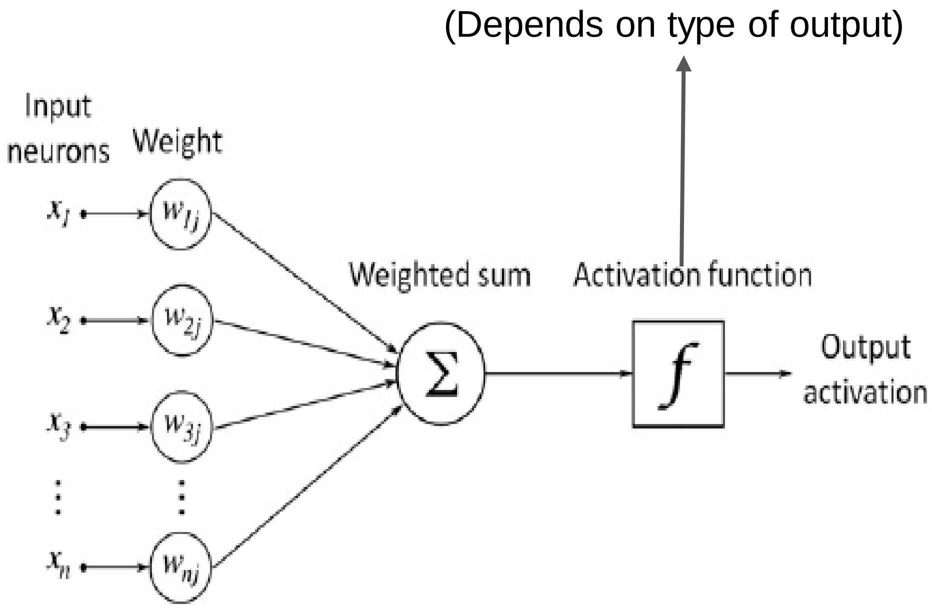
- Size of the weight vector = Number of inputs
- Then, bias is added to get the output ‘z’

Working of a Neuron - Matrix Operations



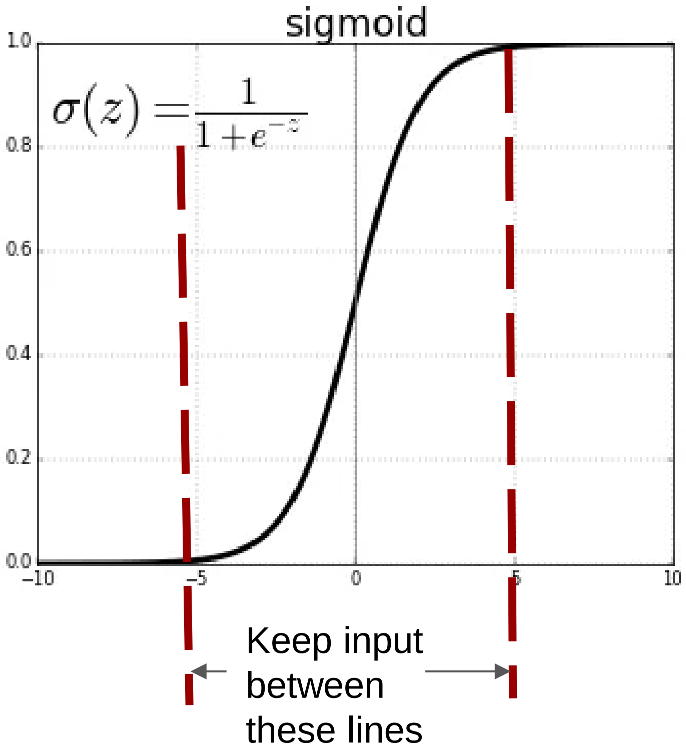
- Ensure that dimensions of input and weights are set up for matrix multiplication
- Bias is broadcasted to be added to each row

Working of a Neuron - Activation



- An activation is applied to output ‘z’ to compute final output ‘a’ of a neuron
- Different activations modify the output in different ways

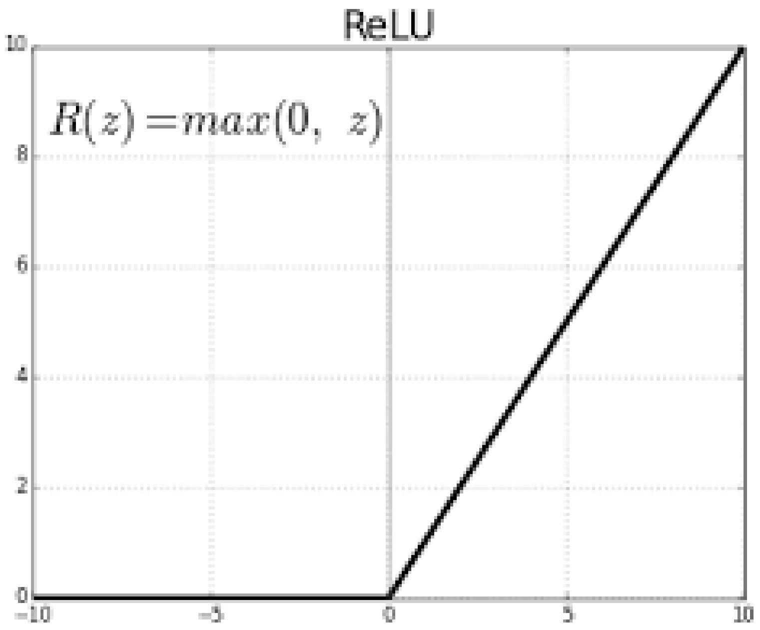
Working of a Neuron - Sigmoid Activation Function



- Sigmoid activation is a nonlinear function
- It exponentially transforms the values the further away they are from 0
- We expect to get a non-zero gradient

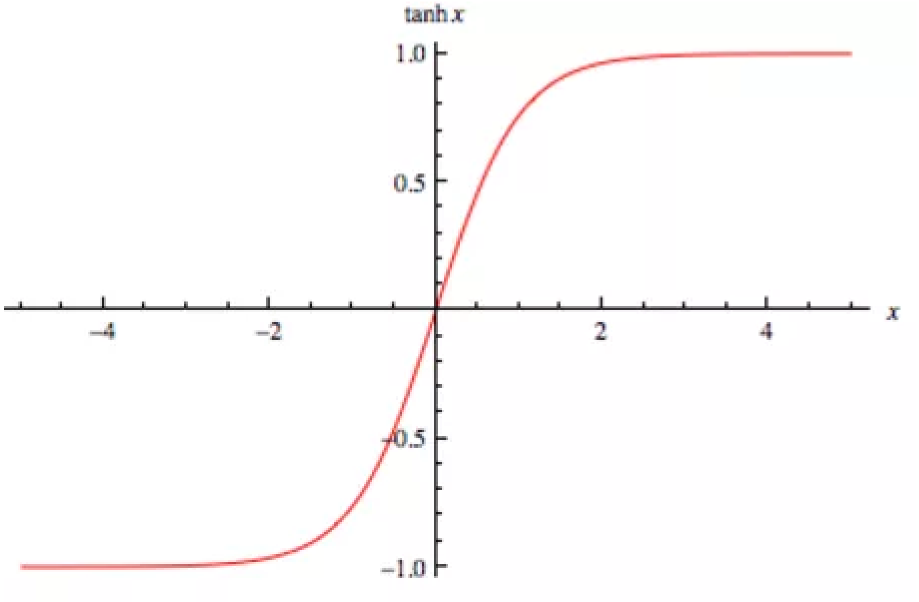
  between the red lines

Working of a Neuron - ReLU Activation Function



- Rectified Linear Unit or ReLU is also a nonlinear function
- It nullifies inputs less than zero to zero
- The inputs greater than zero are kept same

Working of a Neuron - tanh Activation Function



- Hyperbolic tangent or tanh is another nonlinear activation
- Similar to sigmoid except that for negative inputs, it gives negative output

*Details of the activation functions will be covered in later sessions

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - **Analogy with Human Brain**
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

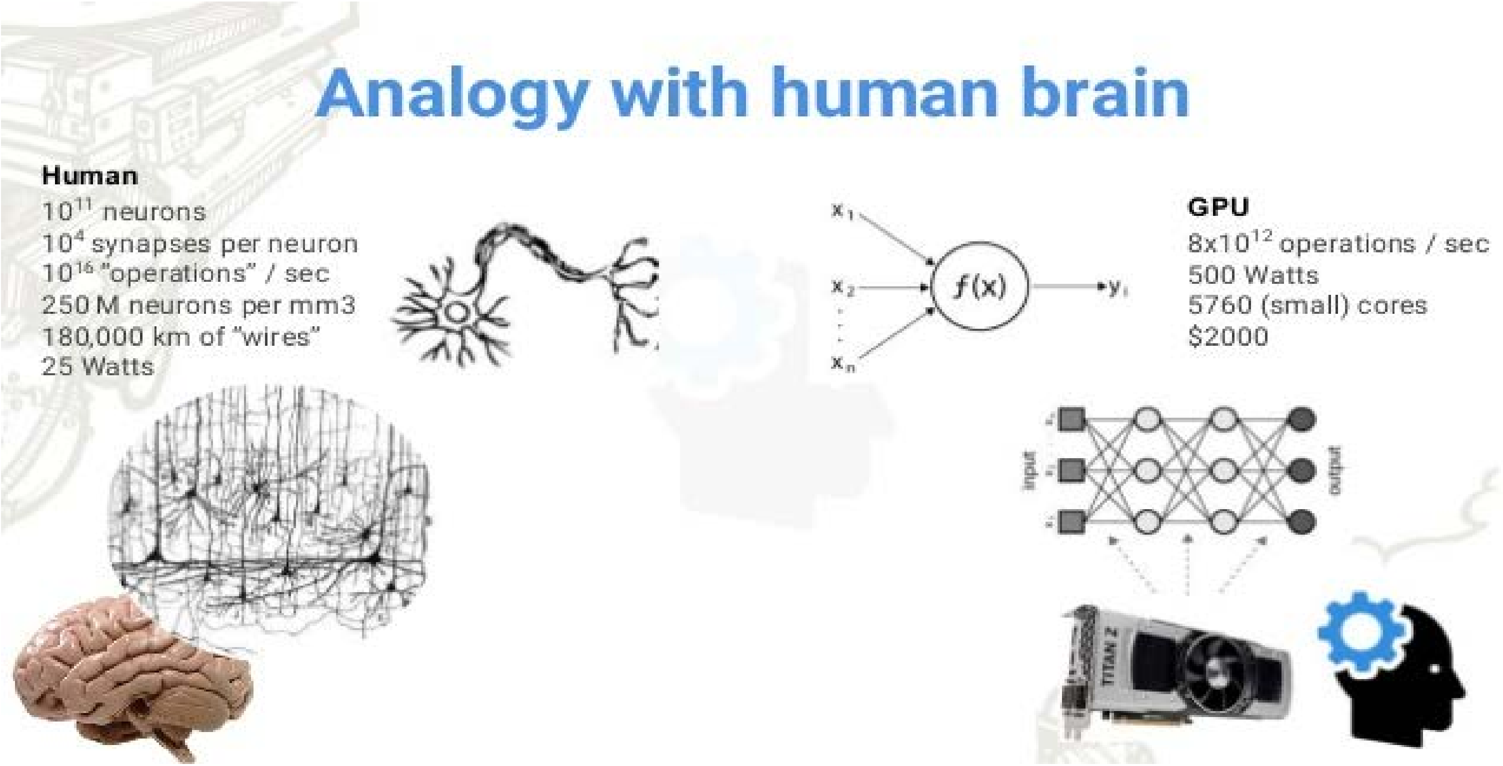
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

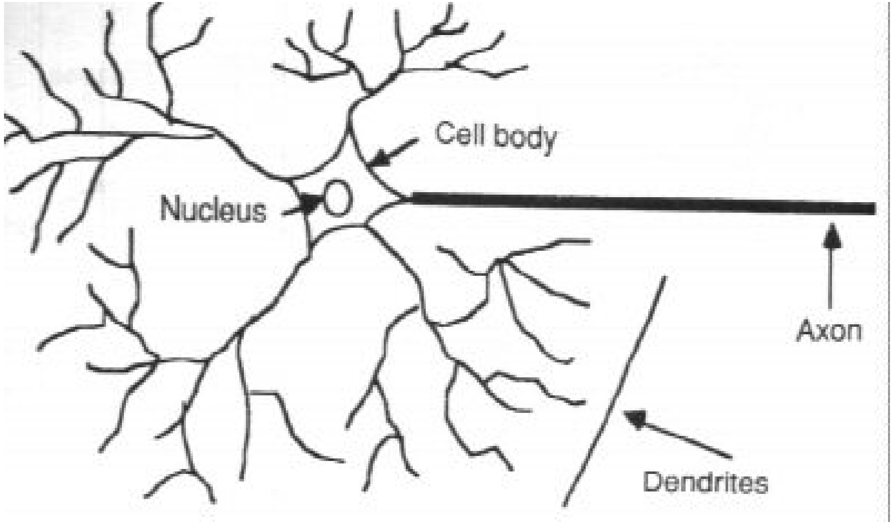
    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Analogy with Human Brain

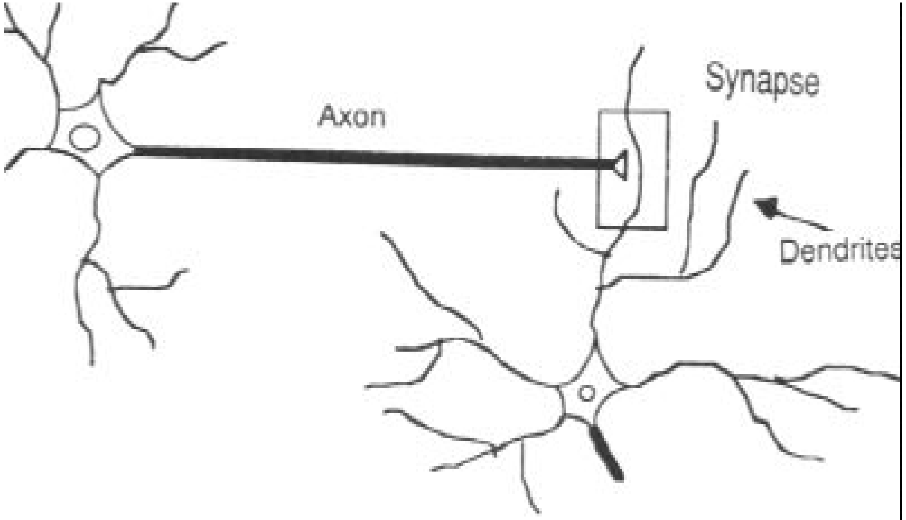


How Human Brain Learns



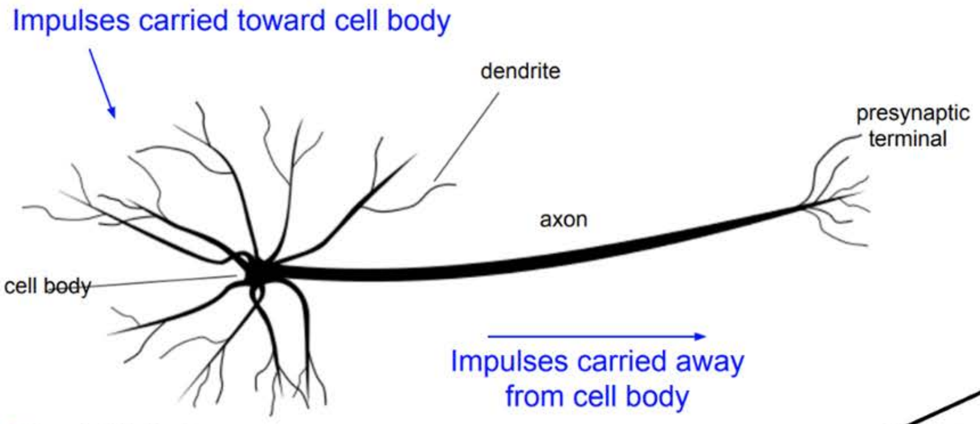
- A typical neuron collects signals through a host of fine structures called *dendrites*
- Neuron sends out spikes of electrical activity through a long thin strand known as an *axon*
- Axons split into thousands of branches

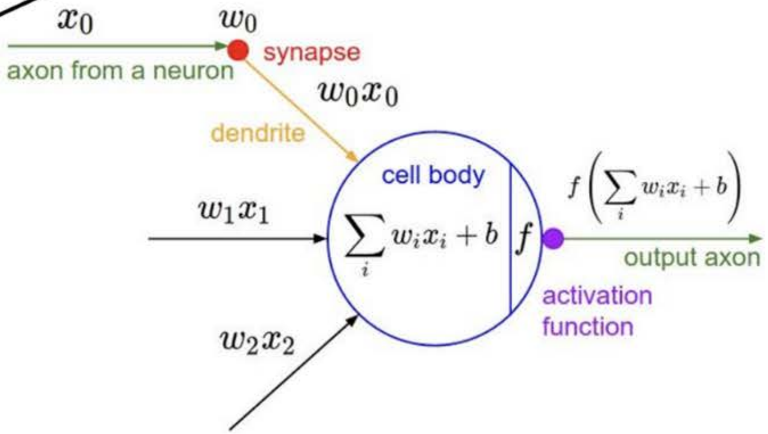
How Human Brain Learns



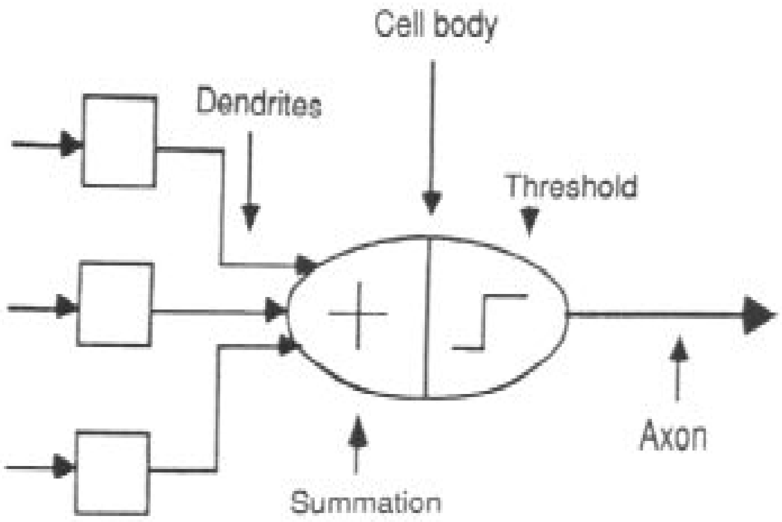
- At the end of each branch, a *synapse* converts activity from axon into electrical effects
- These inhibit or excite activity in the connected neurons
- Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

Analogy of Brain to Neural Network





Neural Network terminology in a Neuron



- We first try to deduce essential features of neurons and their interconnections
- Since our knowledge of neurons is incomplete and computing power is limited
- Our models are actually gross idealisations of real networks of neurons

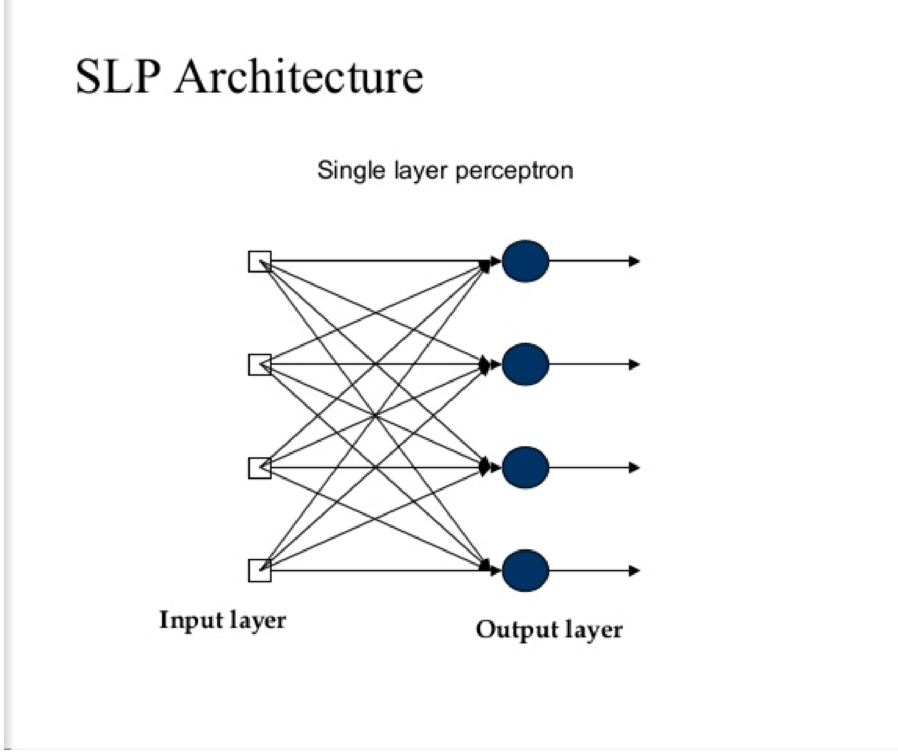
Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - **Perceptron**
2. Why we need Neural Networks
3. Working of a Neural Network
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks
    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

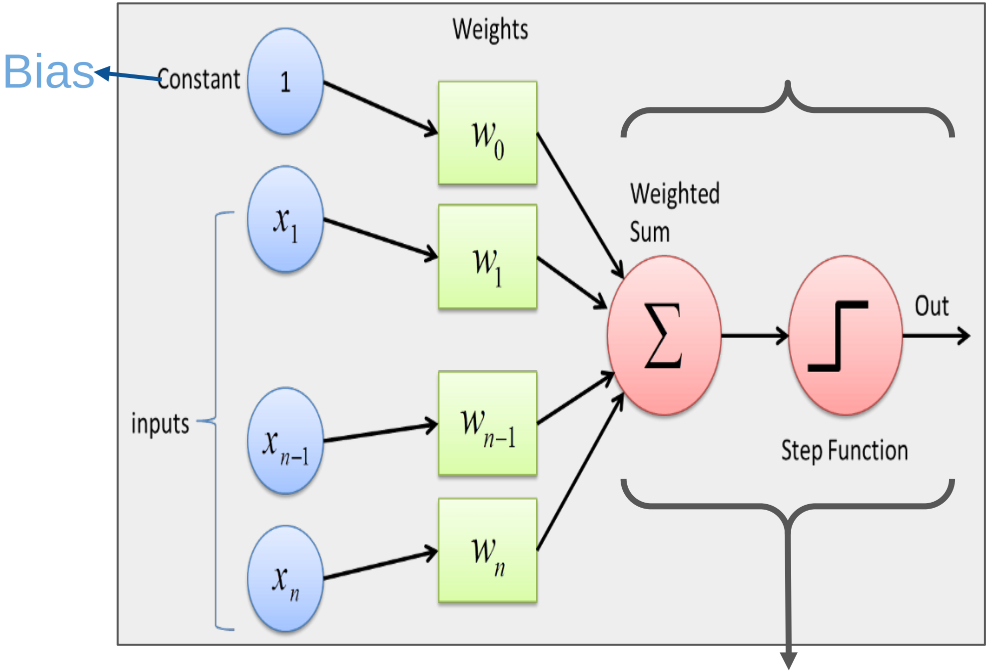
5. Applications of Deep Neural Networks

Perceptron - Introduction



- A perceptron is commonly taken as a Single Layer Perceptron (SLP)
- This means that there is only a single layer of neurons
- The number of neurons in the SLP may vary according to the input

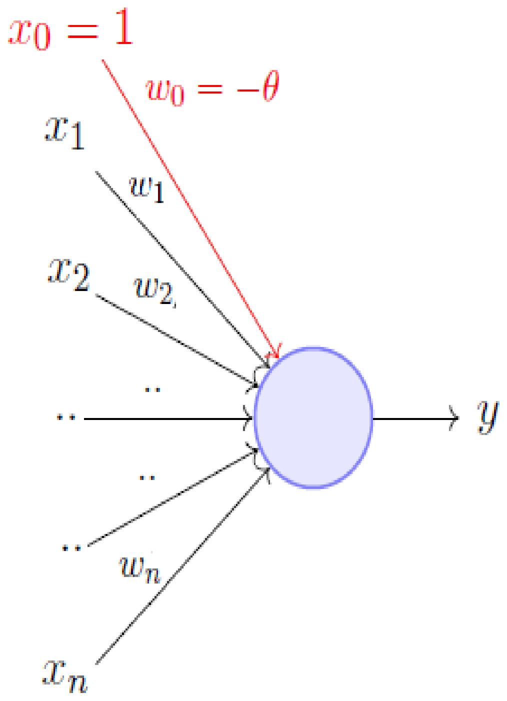
Perceptron



Single neuron of the layer.
Multiple such neurons can exist in an SLP

- Perceptron is a more general computational model than McCulloch-Pitts neuron
- It takes an input and aggregates it (weighted sum)
- Returns 1 only if the aggregated sum is more than some threshold, else returns 0

Perceptron - Working



A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

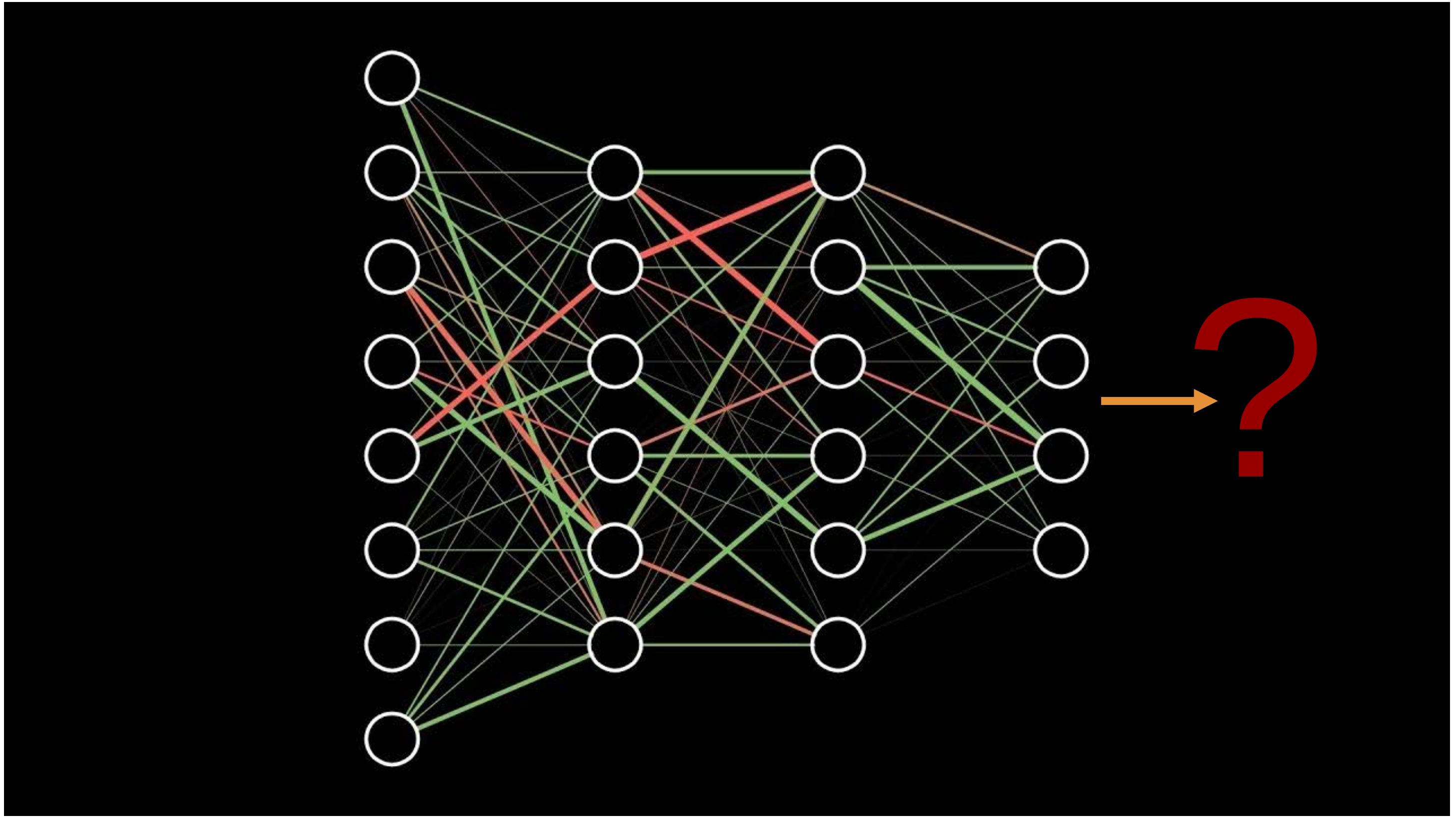
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

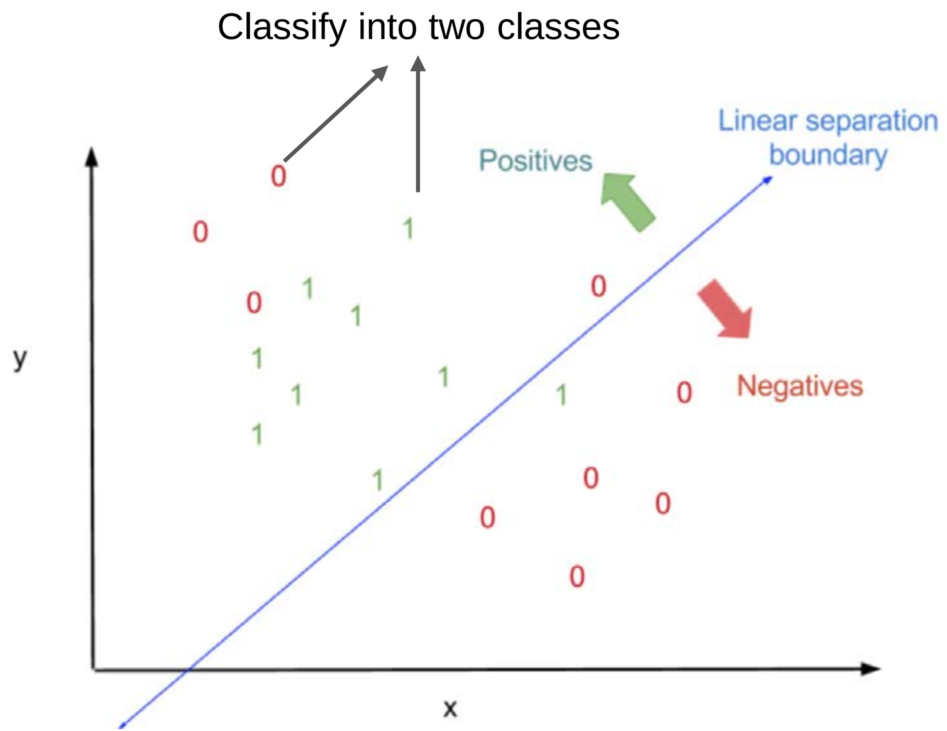
    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Why do we need Neural Networks - Example



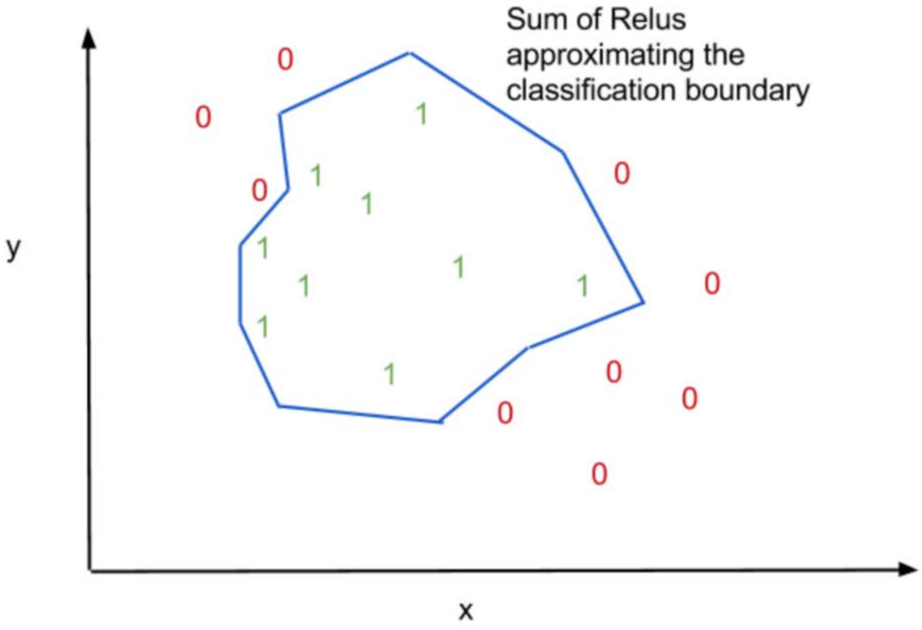
Classification Example - Linear Classifier



- Linear classifier - fails miserably
- Boundary of separation - meant to be nonlinear

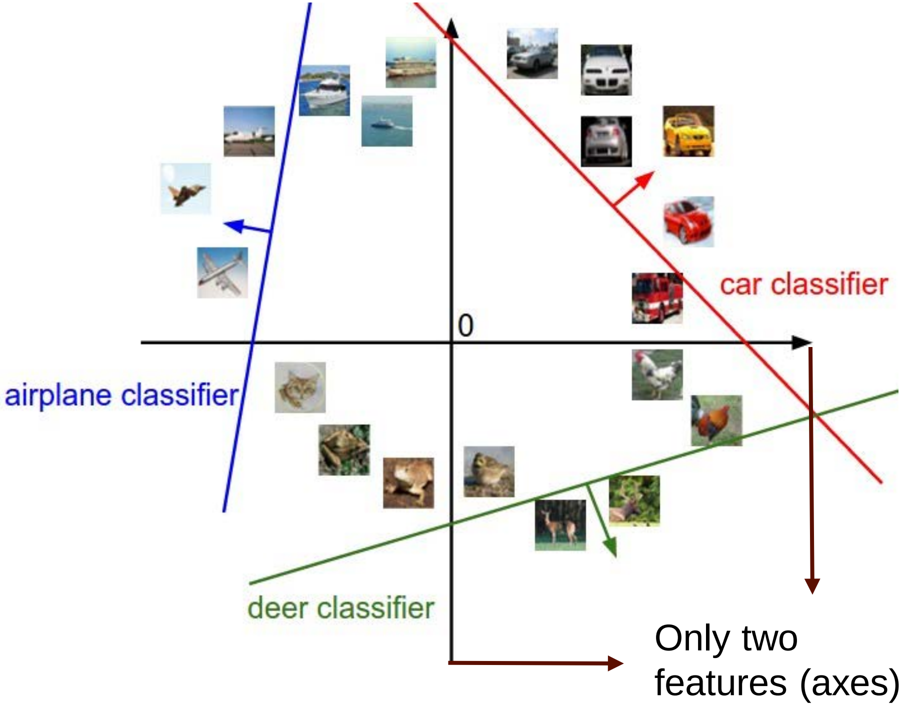
- Regular machine learning algorithms won't work

Classification Problem - Neural Network Classifier



- Neural networks have the ability to build *arbitrary shaped classification boundaries*
- This difference becomes even more evident when there are more than two classes

Multiclass Classification Problem



- Linear classifiers face even more problems for multiclass classification

- 256x256 sized image - 2^{16} features
- Neural Networks classify multi-outputs on large no. of features and classes

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. **Working of a Neural Network**

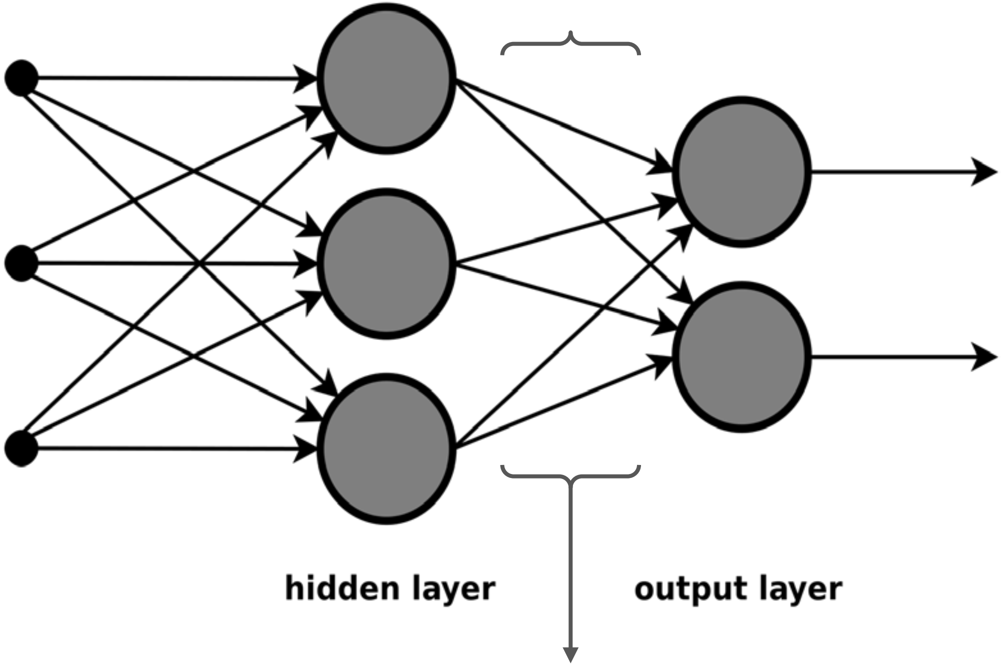
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Neural Networks - Working



Output of one layer used as input to next
Hidden layers > 1 => Multi-layer Perceptron

- One perceptron = one decision
- What about multiple decisions? ◦ E.g. Multi-image classification
- Stack as many output nodes as the number of possible outcomes into the final layer
  - Neural network

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

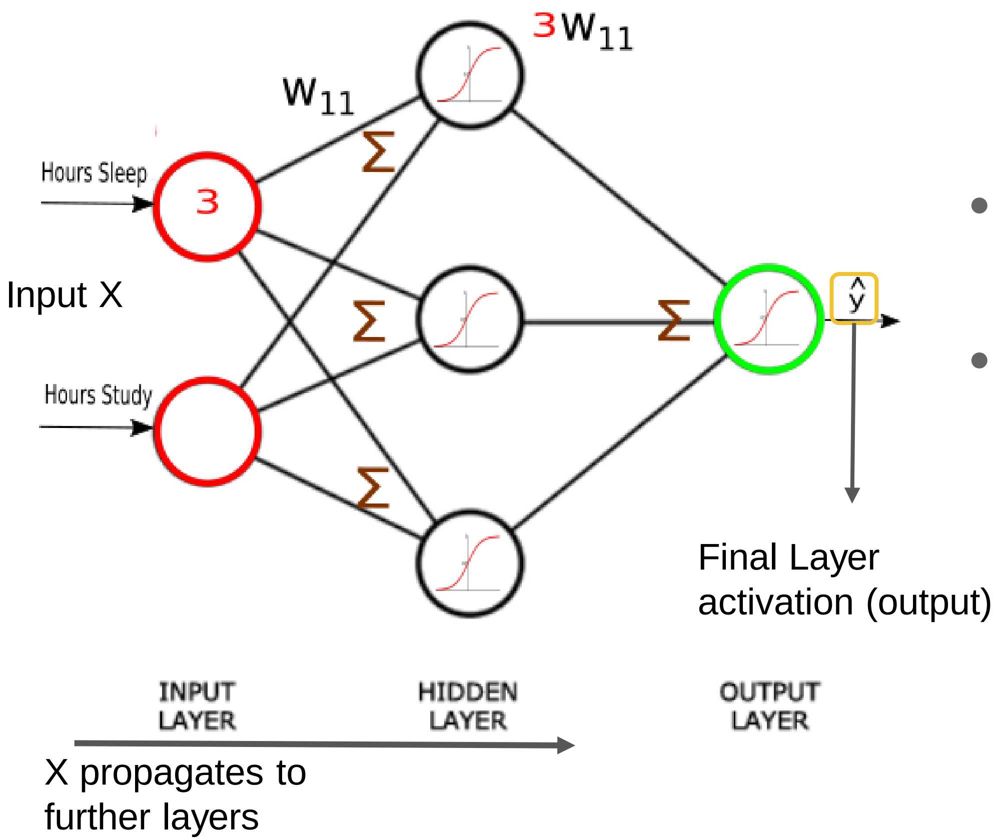
    - **Forward Propagation**
    - Backward Propagation

4. Deep Neural Networks

    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

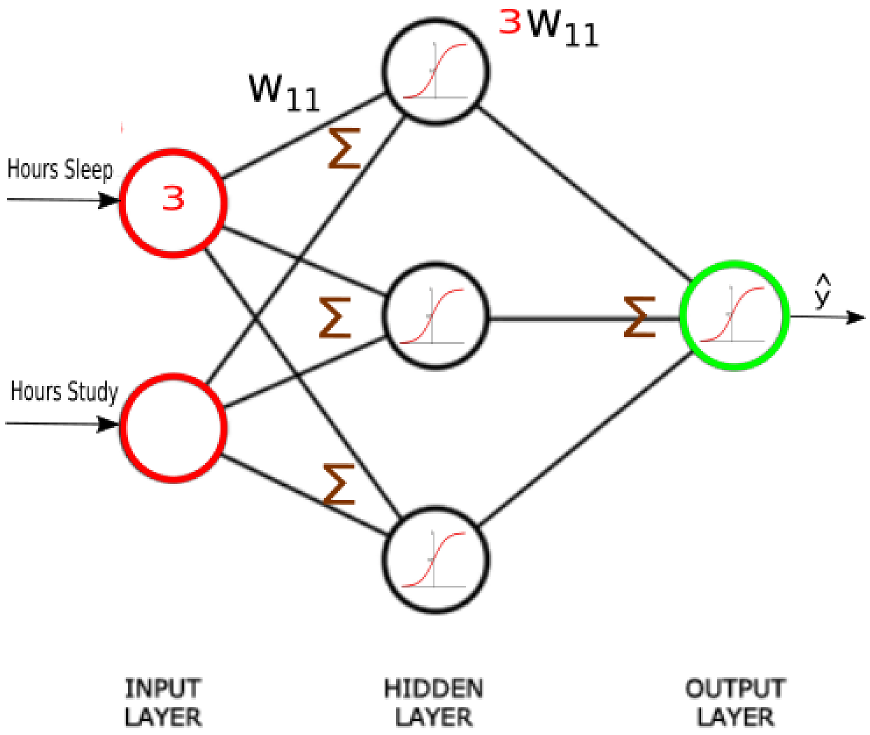
5. Applications of Deep Neural Networks

Forward Propagation



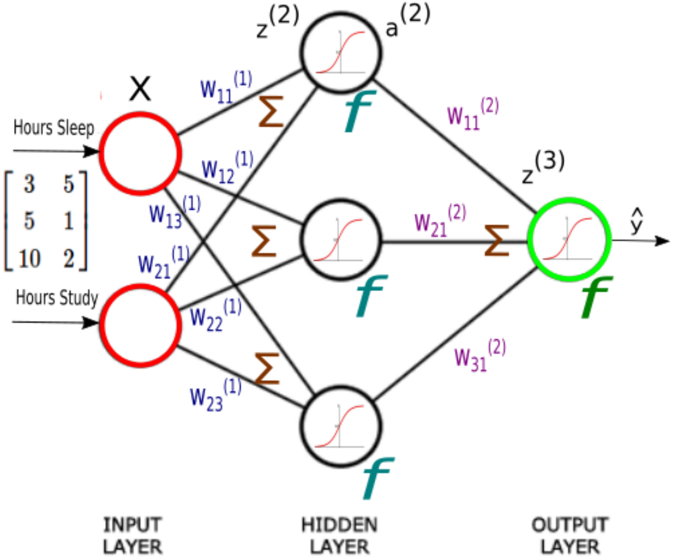
- Let’s take student study-sleep dataset as an example
- At each layer, compute matrix product and its activation, pass to the next layer as input

Forward Propagation



- Two features => two input nodes
- Regular neural network => single hidden layer
- Nodes in hidden layers - our choice
- Output - test score
- Need only one node

Forward Propagation

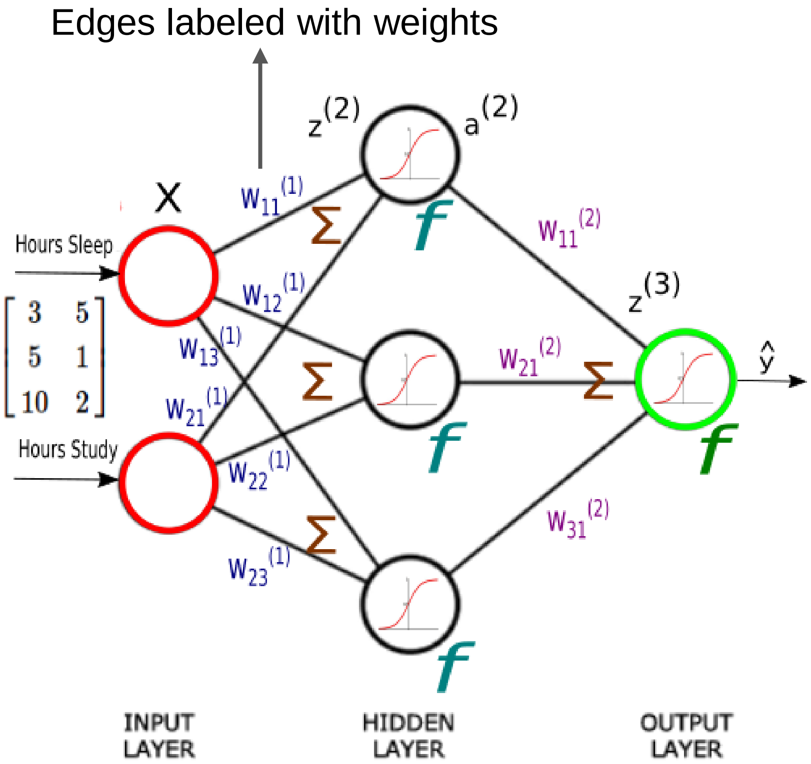


Variables

Math Symbol	Definition	Dimensions	Example
X	Input Data, each row in an example	(numExamples, inputLayerSize)	(N, 2)
y	target data	(numExamples, outputLayerSize)	(N,1)
$W^{(1)}$	Layer 1 weights	(inputLayerSize, hiddenLayerSize)	(2,3)
$W^{(2)}$	Layer 2 weights	(hiddenLayerSize, outputLayerSize)	(3,1)
$z^{(2)}$	Layer 2 activation	(numExamples, hiddenLayerSize)	(N,3)
$a^{(2)}$	Layer 2 activity	(numExamples, hiddenLayerSize)	(N,3)
$z^{(3)}$	Layer 3 activation	(numExamples, outputLayerSize)	(N,1)

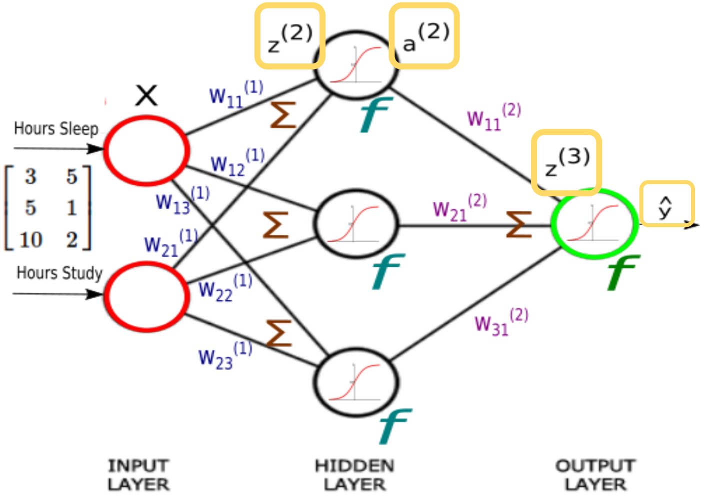
N - No. of training examples

Forward Propagation



- Example matrix of 3 instances, each of which has an hours slept and hours studied value
- Eg. instance 1 - 3 hours of sleep and 5 hours of study
- Predict the test scores for the three instances

Forward Propagation



Matrix product at layer 2 - $z^{(2)} = XW^{(1)}$

Activation at layer 2 - $a^{(2)} = f\left(z^{(2)}\right)$

Matrix product at layer 3 - $z^{(3)} = a^{(2)}W^{(2)}$

Activation at layer 3 (output) - $\hat{y} = f\left(z^{(3)}\right)$

- These are the main equations which will be used to calculate the output

- We ignore the bias here for simplicity
- We do the same steps progressively for each layer

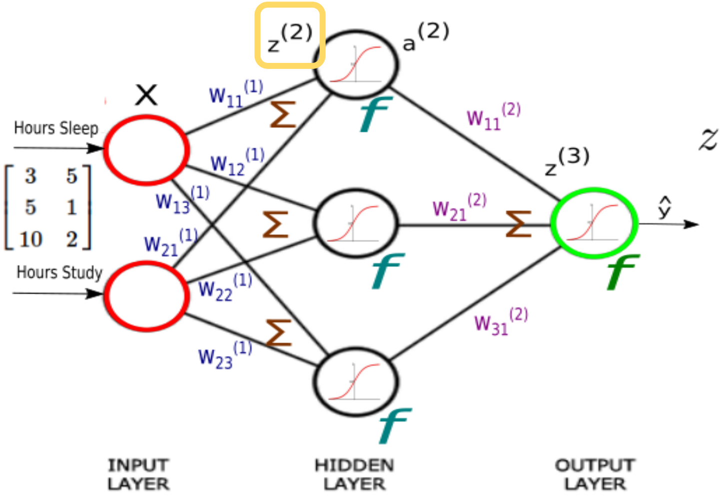
Forward Propagation - Matrix Multiplication (1st layer)

For first layer - $$z^{(2)} = XW^{(1)}$$

$$= \begin{bmatrix} 3 & 5 \\ 5 & 1 \\ 10 & 2 \end{bmatrix}_{\underline{3}\text{x}2} \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix}_{2\text{x}\underline{3}}$$

- Multiply 3 x 2 input matrix with 2 x 3 weight matrix
- For eg. $W^{(1)}_{23}$ means the weight of the edge from the second (2) input feature to the third (3) node of the first (1) hidden layer

Forward Propagation



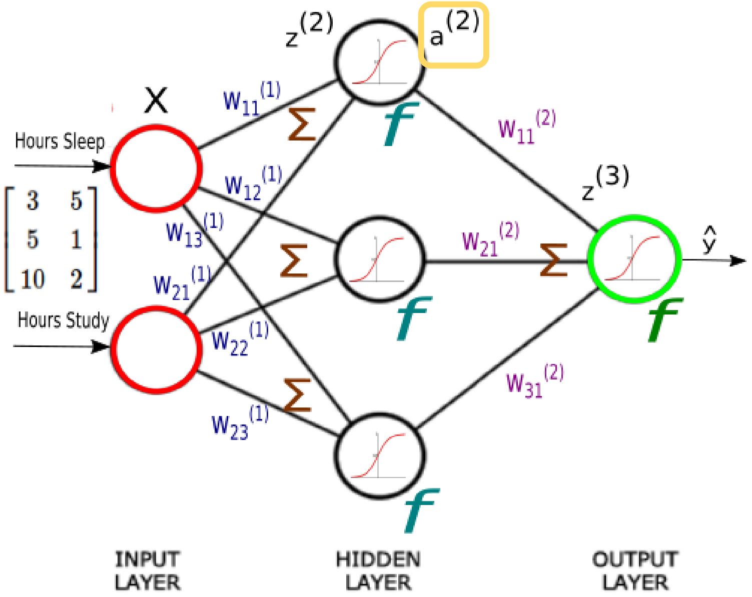
$$z^{(2)} = XW^{(1)} = \begin{bmatrix} 3 & 5 \\ 5 & 1 \\ 10 & 2 \end{bmatrix} \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} 3W_{11}^{(1)} + 5W_{21}^{(1)} & 3W_{12}^{(1)} + 5W_{22}^{(1)} & 3W_{13}^{(1)} + 5W_{23}^{(1)} \\ 5W_{11}^{(1)} + W_{21}^{(1)} & 5W_{12}^{(1)} + W_{22}^{(1)} & 5W_{13}^{(1)} + W_{23}^{(1)} \\ 10W_{11}^{(1)} + 2W_{21}^{(1)} & 10W_{12}^{(1)} + 2W_{22}^{(1)} & 10W_{13}^{(1)} + 2W_{23}^{(1)} \end{bmatrix}$$

- Each entry in z - sum of weighted inputs to each hidden neuron.
- z is 3×3 matrix, one row for each sample, and one column for each hidden unit (N=3, -> 3x3 matrix)

Forward Propagation - Activation

- Matrix activated using an activation function
- Sigmoid activation function as example
- Calculate the sigmoid of each value in the matrix



$$a^{(2)} = sigmoid\left(\begin{bmatrix} 3W_{11}^{(1)} + 5W_{21}^{(1)} & 3W_{12}^{(1)} + 5W_{22}^{(1)} & 3W_{13}^{(1)} + 5W_{23}^{(1)} \\ 5W_{11}^{(1)} + W_{21}^{(1)} & 5W_{12}^{(1)} + W_{22}^{(1)} & 5W_{13}^{(1)} + W_{23}^{(1)} \\ 10W_{11}^{(1)} + 2W_{21}^{(1)} & 10W_{12}^{(1)} + 2W_{22}^{(1)} & 10W_{13}^{(1)} + 2W_{23}^{(1)} \end{bmatrix}\right)$$

$$= \begin{bmatrix} smd(3W_{11}^{(1)} + 5W_{21}^{(1)}) & smd(3W_{12}^{(1)} + 5W_{22}^{(1)}) & smd(3W_{13}^{(1)} + 5W_{23}^{(1)}) \\ smd(5W_{11}^{(1)} + W_{21}^{(1)}) & smd(5W_{12}^{(1)} + W_{22}^{(1)}) & smd(5W_{13}^{(1)} + W_{23}^{(1)}) \\ smd(10W_{11}^{(1)} + 2W_{21}^{(1)}) & smd(10W_{12}^{(1)} + 2W_{22}^{(1)}) & smd(10W_{13}^{(1)} + 2W_{23}^{(1)}) \end{bmatrix}$$

* Sigmoid abbreviated as smd

Forward Propagation - Initialization

Random Initialization

$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \longrightarrow \begin{bmatrix} 0.8 & 0.4 & 0.3 \\ 0.2 & 0.9 & 0.5 \end{bmatrix}$$

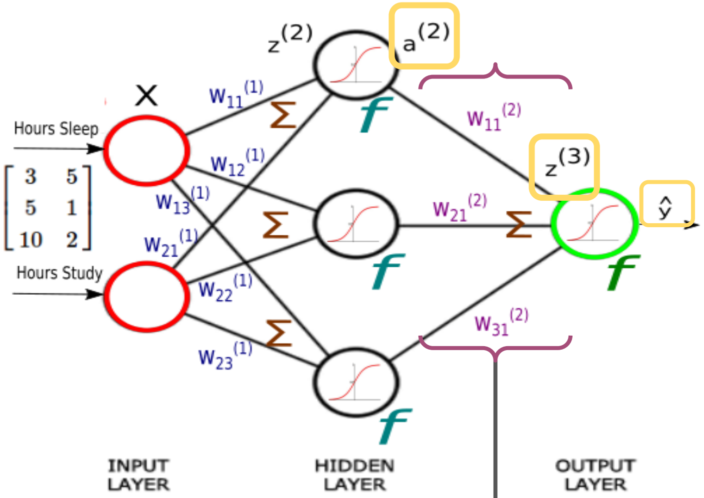
Normalized (0,1) values for easy calculation

Zero Initialization

$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

All computed gradients evaluate to 0

Forward Propagation



$$a^{(2)} = f\left(z^{(2)}\right)$$

$$z^{(3)} = a^{(2)} W^{(2)}$$

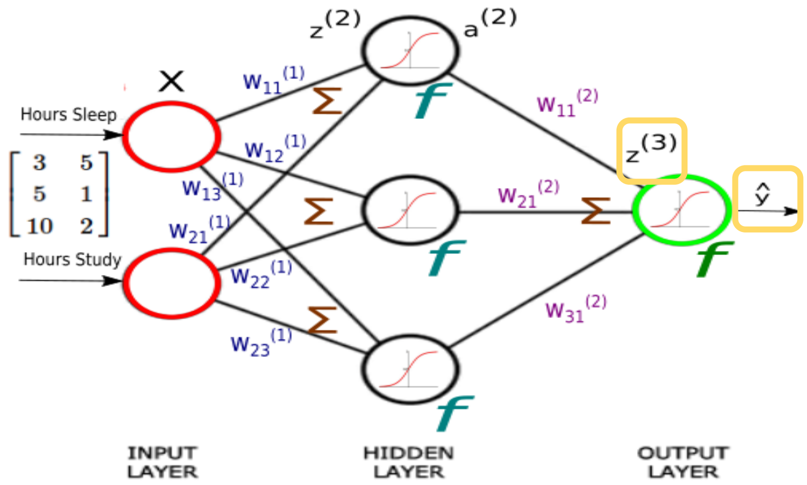
$$\hat{y} = f\left(z^{(3)}\right)$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix}$$

- $a^{(2)}$ passed as input to the third (final) layer
- Matrix multiplication - $a^{(2)}$ (3 x 3) and $W^{(2)}$ (3 x 1)

- Final output - activation matrix of the final layer

Forward Propagation



- Final output - sigmoid of the matrix $z^{(3)}$
- No training => Poor output predictions of test scores
- This is one iteration of forward propagation
- To get good results, we train our network i.e. use a feedback mechanism to realign weights

$$z^{(3)} = a^{(2)} W^{(2)}$$

$$= \begin{bmatrix} smd(3W_{11}^{(1)} + 5W_{21}^{(1)}) & smd(3W_{12}^{(1)} + 5W_{22}^{(1)}) & smd(3W_{13}^{(1)} + 5W_{23}^{(1)}) \\ smd(5W_{11}^{(1)} + W_{21}^{(1)}) & smd(5W_{12}^{(1)} + W_{22}^{(1)}) & smd(5W_{13}^{(1)} + W_{23}^{(1)}) \\ smd(10W_{11}^{(1)} + 2W_{21}^{(1)}) & smd(10W_{12}^{(1)} + 2W_{22}^{(1)}) & smd(10W_{13}^{(1)} + 2W_{23}^{(1)}) \end{bmatrix} \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix} \xrightarrow{f()} \hat{y} = f\left(z^{(3)}\right)$$

Agenda

1. Neural Network Building Blocks
   - What is a Neuron?
   - Working of a Neuron
   - Analogy with Human Brain
   - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

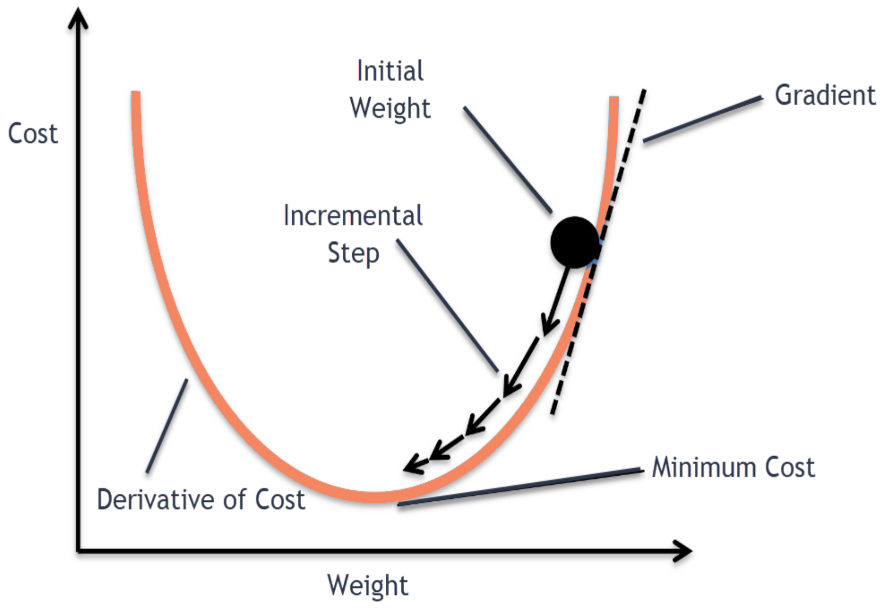
   - Forward Propagation
   - **Backward Propagation**

4. Deep Neural Networks

   - Hidden Layers
   - Why Deep Neural Networks?
   - Types of DNNs
     - Multilayer Perceptron (ANN)
     - Convolutional Neural Network
     - Recurrent Neural Network
     - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Backward Propagation (Backprop)



- Need to calculate the cost function gradient
- Requires known, target data for each input
- Supervised training

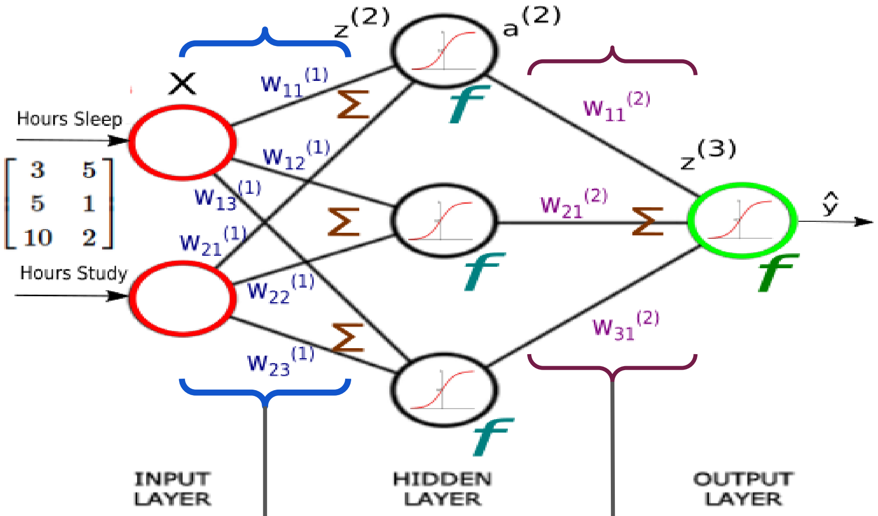
Backward Propagation - Cost

Minimize cost function (sum of squared loss) -

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- Here, N is the no. of samples, y_i is the actual output, $\hat{y}_i$ is our prediction
- Complete equations for gradient descent - too long and complex
- Since we have already dealt with the concept, we use it with basic hints

Backward Propagation - Weights



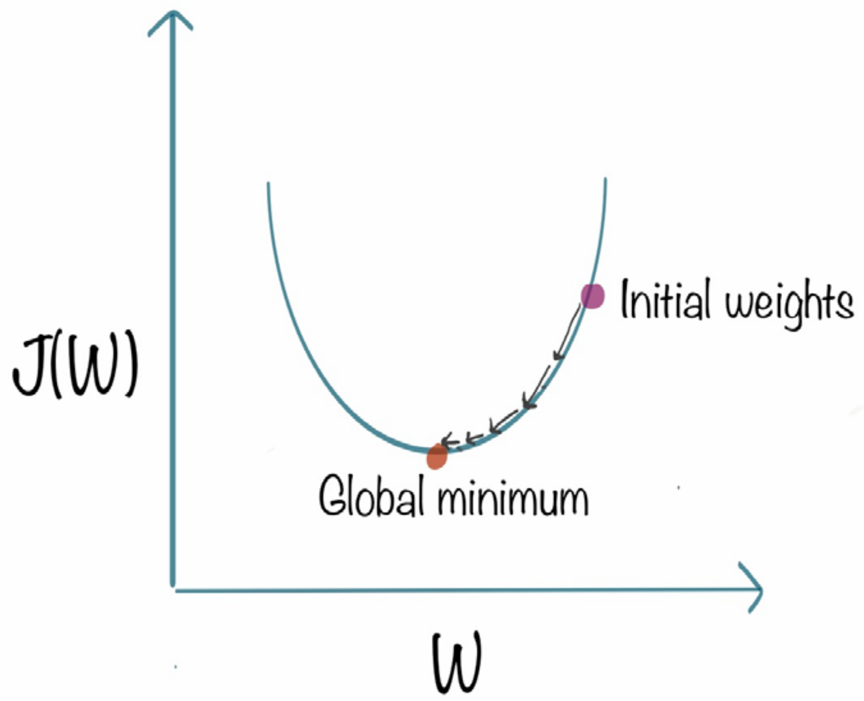
$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix}$$

- We have a collection of 9 weights
- We're going to make our cost (J) as small as possible
- This can be done using an optimal combination of the weights

Backward Propagation - Brute Force vs GD

Brute Force	Gradient Descent
• Check cost by only tweaking the first weight W_{11} for 1000 different values = 0.11 secs	• Calculate the cost function using all samples
• Check 1000 values of W_{11} and W_{12} = ~100 secs	• Calculate gradient to update the weights = ~10 seconds
• Check 1000 values for all 9 weights = 1 trillion millenium	• Update the weights using the gradient = ~30 seconds
• That's a billion times more than the **age of the universe**	

Backward Propagation - Training (Gradient Descent)



- Smarter way - Gradient Descent
- Capable of incredible speedups in higher dimensions
- Gradient = derivative of the cost wrt the weights
- Three choices of **gradient descent**: batch(standard), stochastic or mini-batch

Backward Propagation - Batch, Stochastic and Mini-batch GD

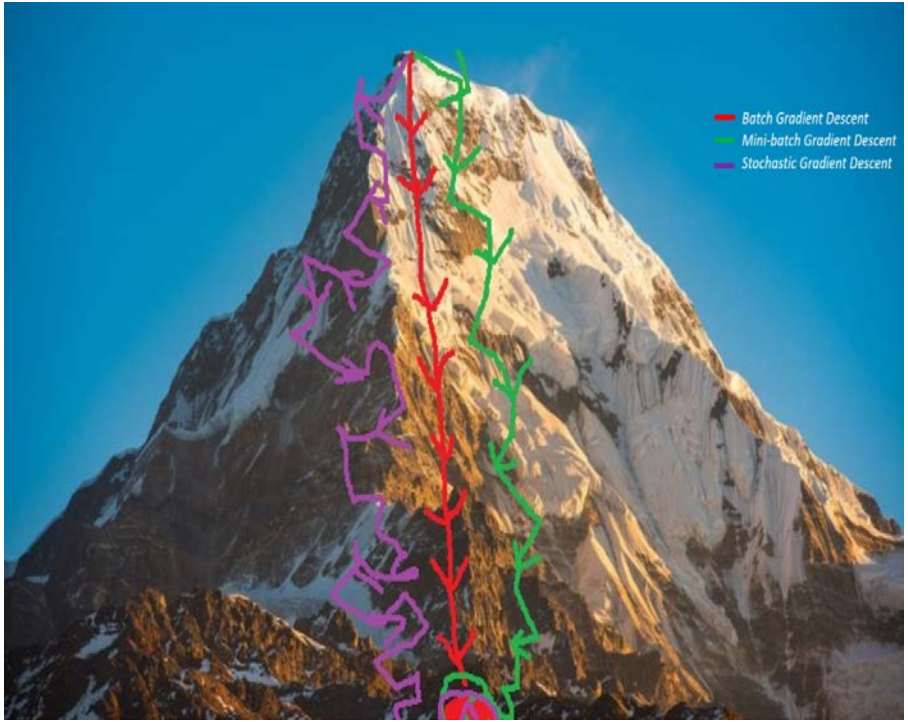
Batch Gradient Descent - $\sum_{i=1}^{N} \frac{\partial J}{\partial W}$

Stochastic Gradient Descent - $\frac{\partial J}{\partial W}$

Mini-batch Gradient Descent - $\sum_{i=1}^{s} \frac{\partial J}{\partial W}$

- **Batch** gradient descent method sums up all the derivatives of J for all samples (N)
- **Stochastic** gradient descent (SGD) method uses one derivative at one sample and move to another sample point
- **Mini-batch** gradient descent uses fixed batch size of samples (here, s), usually much smaller than N

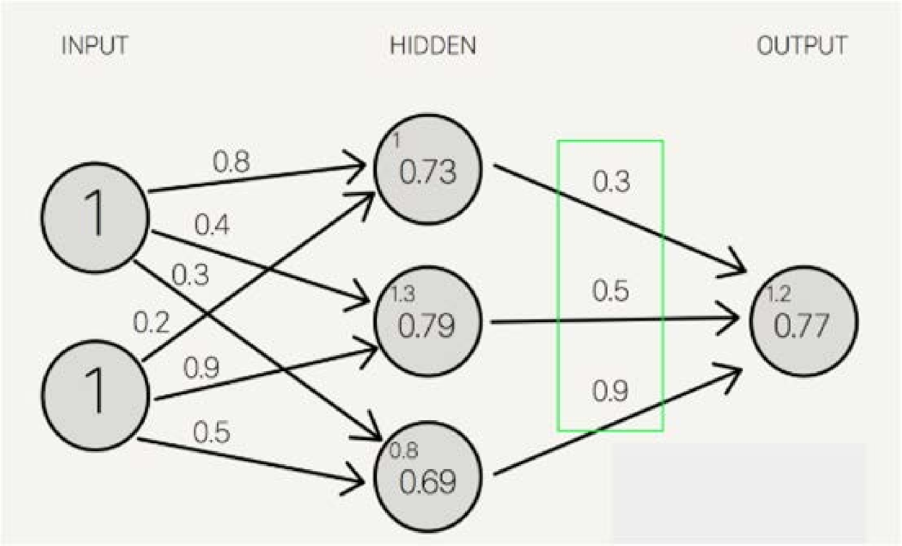
Batch vs Stochastic vs Mini-Batch Gradient Descent



- **Batch** : takes fewer steps, high no. of calculations per step
- **Stochastic** : takes high no. of steps, only one calculation per step
- **Mini-batch** : Middle ground, takes intermediate no. of steps, few calculations per step
- Performance :

  Mini-batch > Stochastic > Batch

Backward Propagation - Output



- Backprop uses gradient descent to train our neural networks efficiently
- Gives an accurate measure very close to the actual output

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

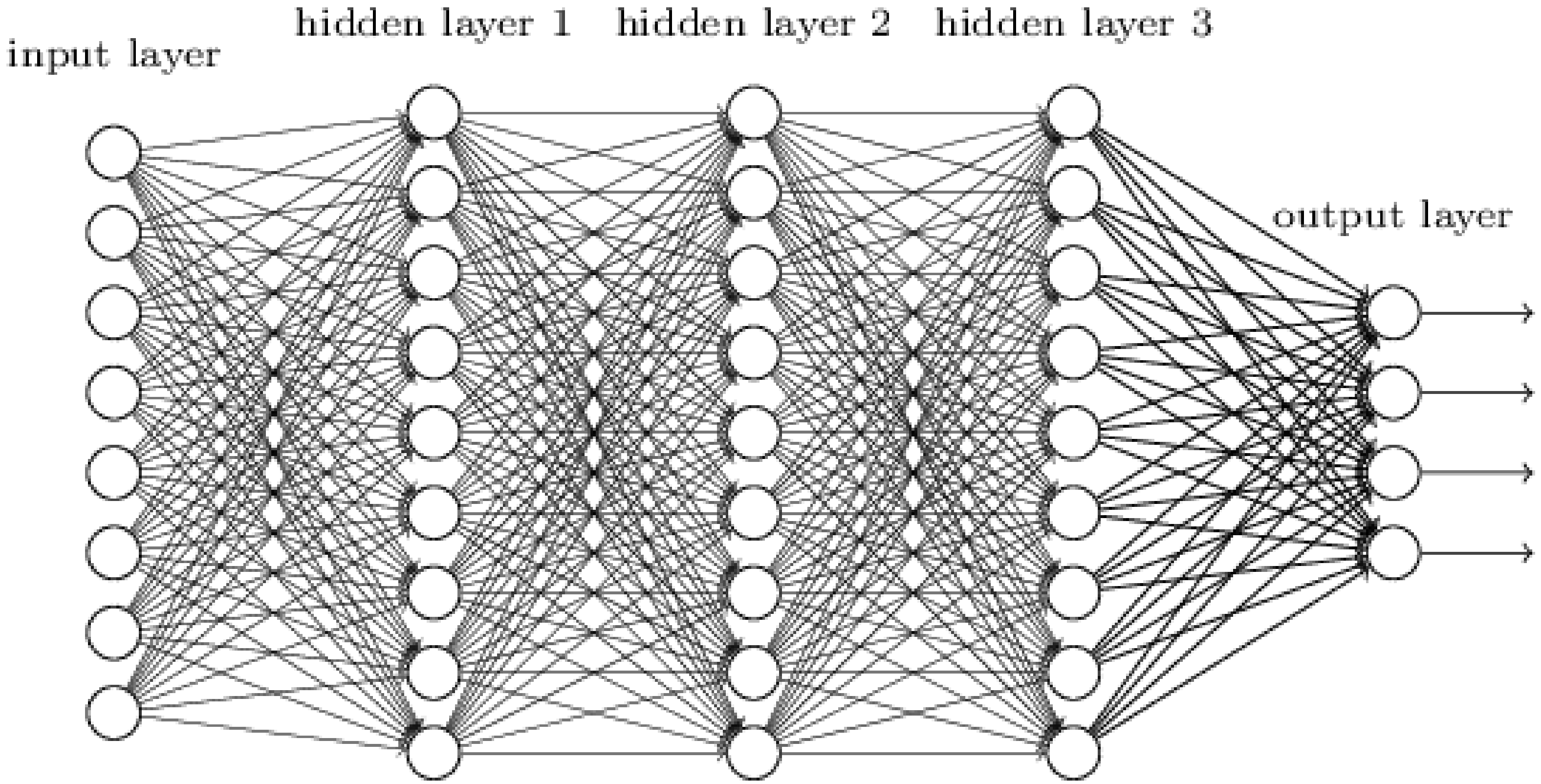
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

- Hidden Layers
- Why Deep Neural Networks?
- Types of DNNs
    - Multilayer Perceptron (ANN)
    - Convolutional Neural Network
    - Recurrent Neural Network
    - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Deep Neural Networks



Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

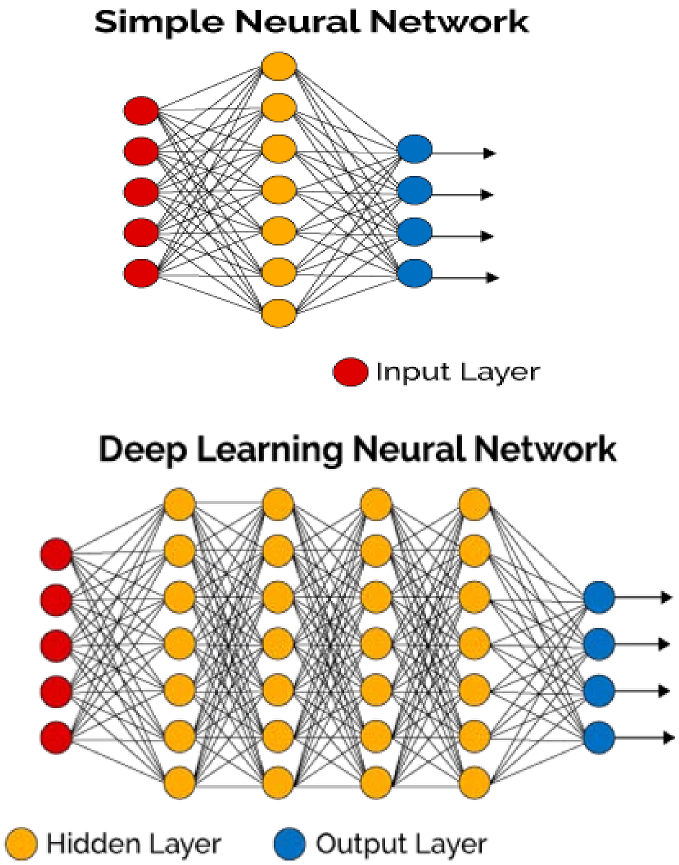
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

    - **Hidden Layers**
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

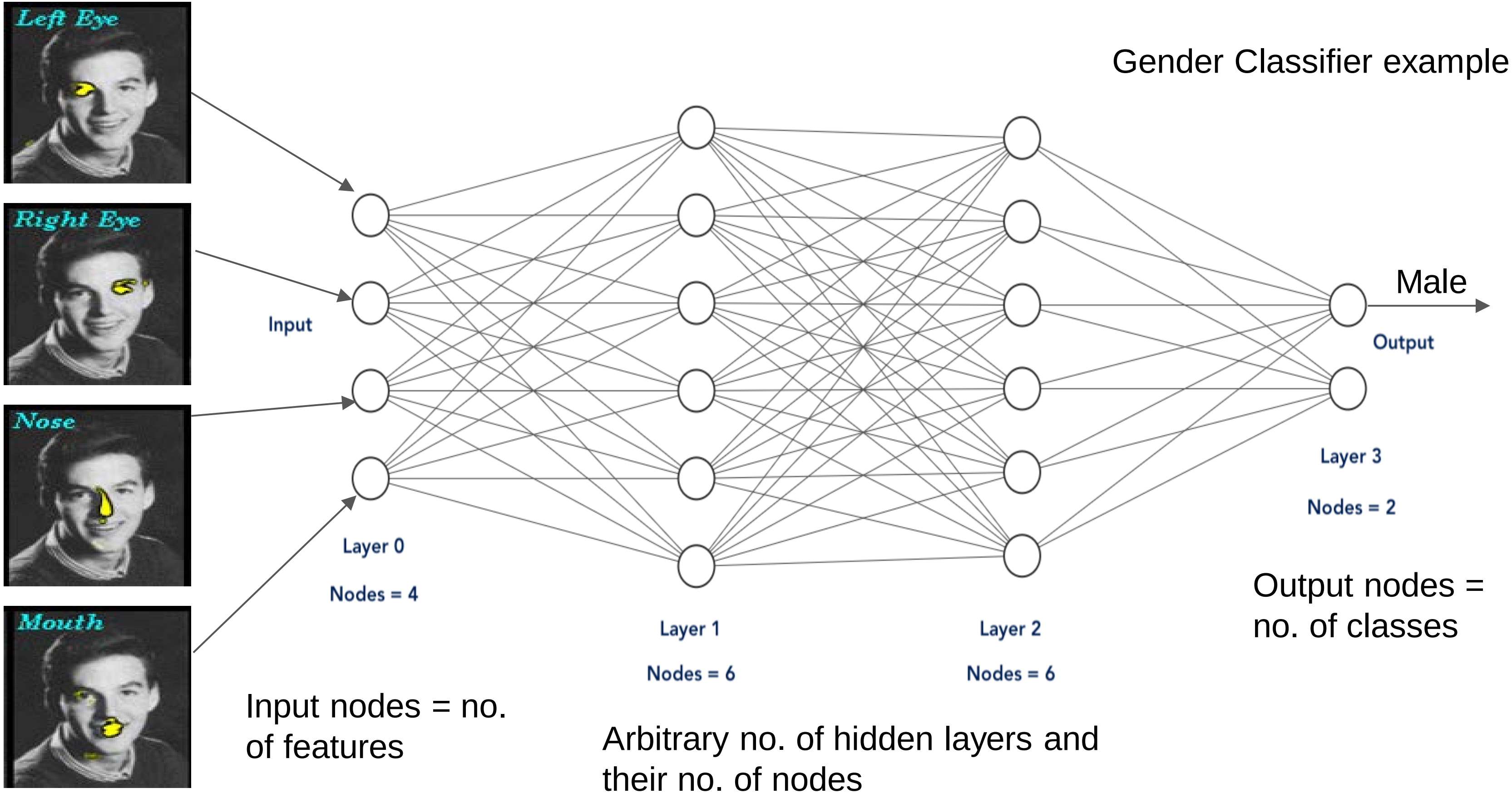
Deep Neural Networks (DNN) - Hidden Layers



- Conventional NN - one hidden layer
- DNN - two or more hidden layers
- *Deep* layers => *Deep* Learning
- Previous algorithms like forward propagation, gradient descent, backward propagation etc. remain same

Deep Neural Network - Hidden Layers

Gender Classifier example

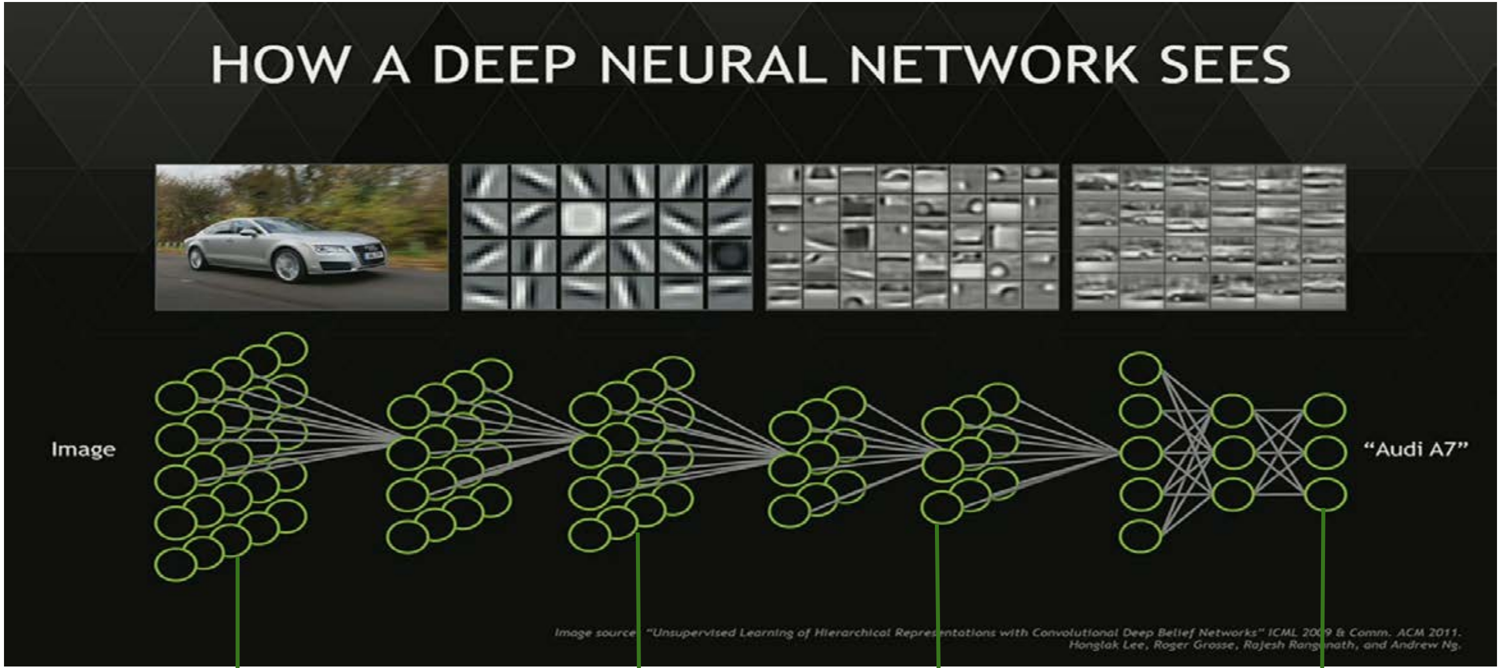


Input nodes = no. of features

Arbitrary no. of hidden layers and their no. of nodes

Output nodes = no. of classes

Deep Neural Network - Hidden Layers

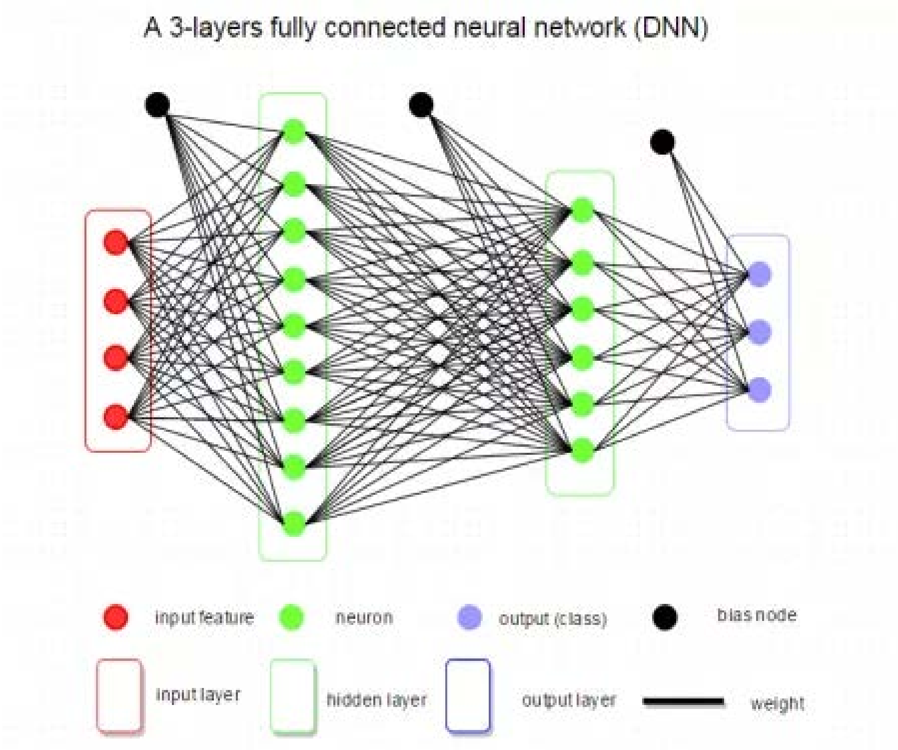


Input layer -> nodes = no. of features

Hidden layers capture features like edges, shapes, wheels, etc.

Output layer predicts object

Choosing No. of layers and nodes



- Generic way - Experimentation
- Develop intuition with experience
- Depth - the more the better, but with diminishing returns
- Take ideas from papers, models

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

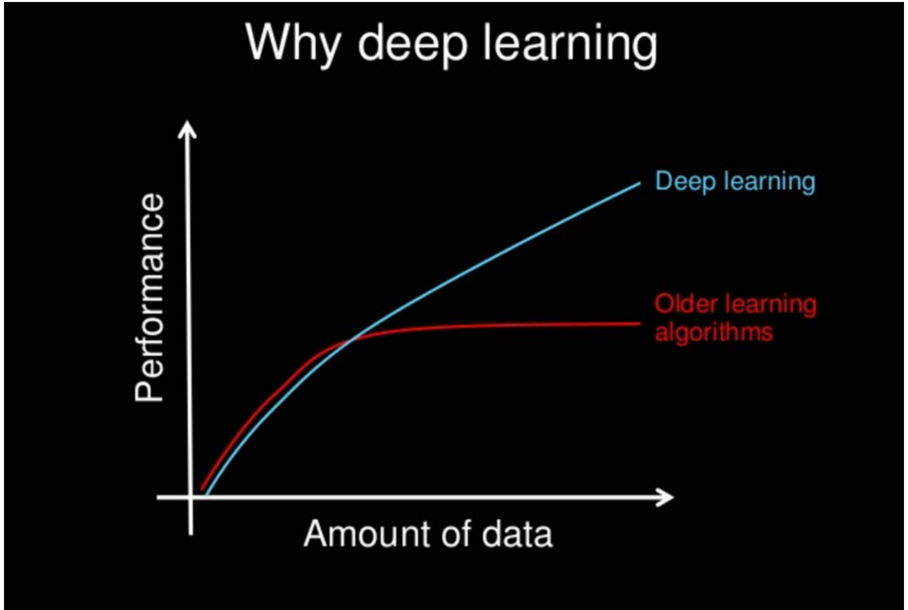
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

    - Hidden Layers
    - **Why Deep Neural Networks?**
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

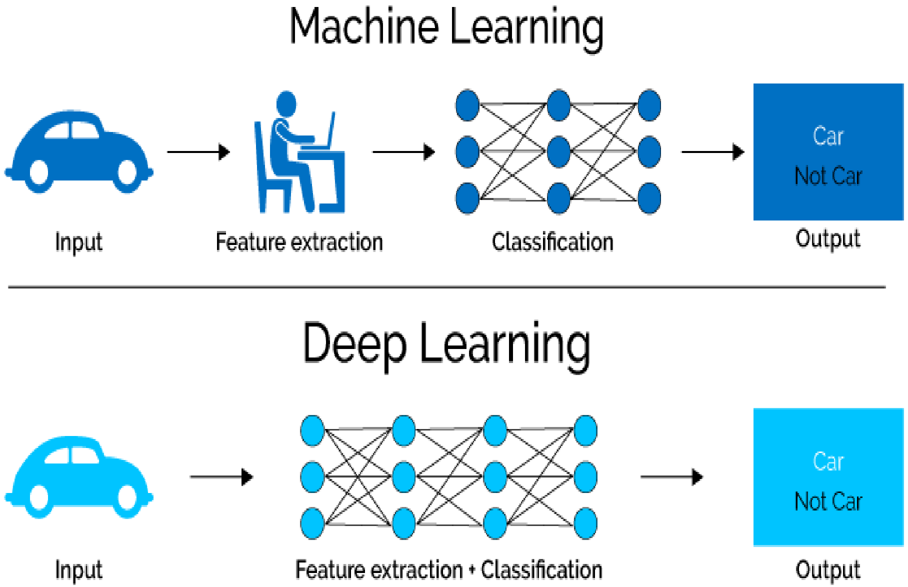
Why *Deep* Neural Networks?



How do data science techniques scale with amount of data?

- Data - Today, more than 2 trillion terabytes of data is generated each year
- Hardware - We now have devices and hardware capable of dealing with such volume of data, eg. GPUs, TPUs, NPUs, etc.
- Performance - Deep Learning > Machine Learning almost universally now

Why *Deep* Neural Networks - Performance



- Selecting features - human limitation
- Deep learning eliminates this requirement by doing feature selection within the network internally
- Often still need to do manual preprocessing to make DNN work

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

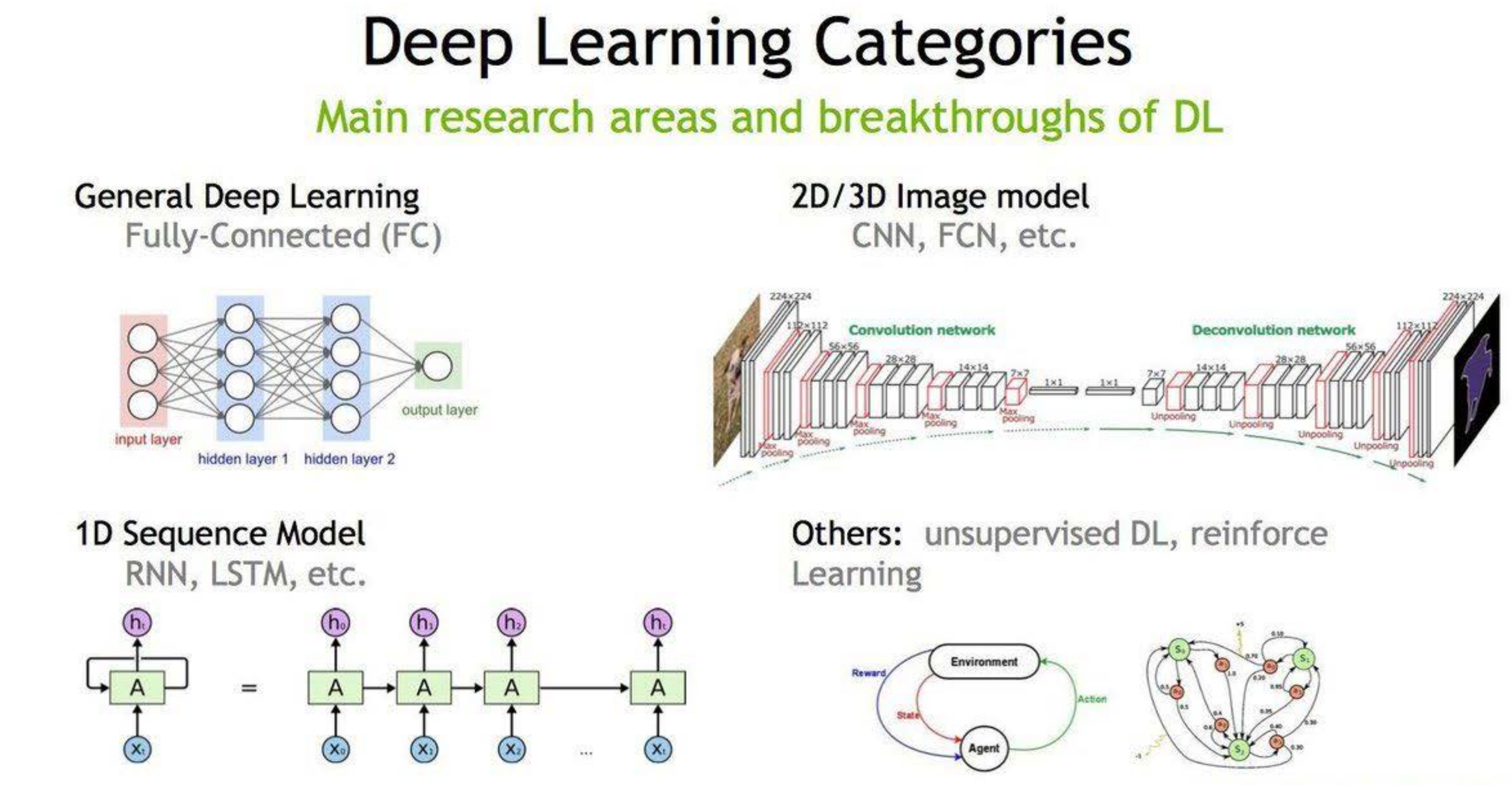
    - Forward Propagation
    - Backward Propagation

4. Deep Neural Networks

    - Hidden Layers
    - Why Deep Neural Networks?
    - **Types of DNNs**
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Types of *Deep* Neural Networks - Breakthroughs



Agenda

1. Neural Network Building Blocks
   - What is a Neuron?
   - Working of a Neuron
   - Analogy with Human Brain
   - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

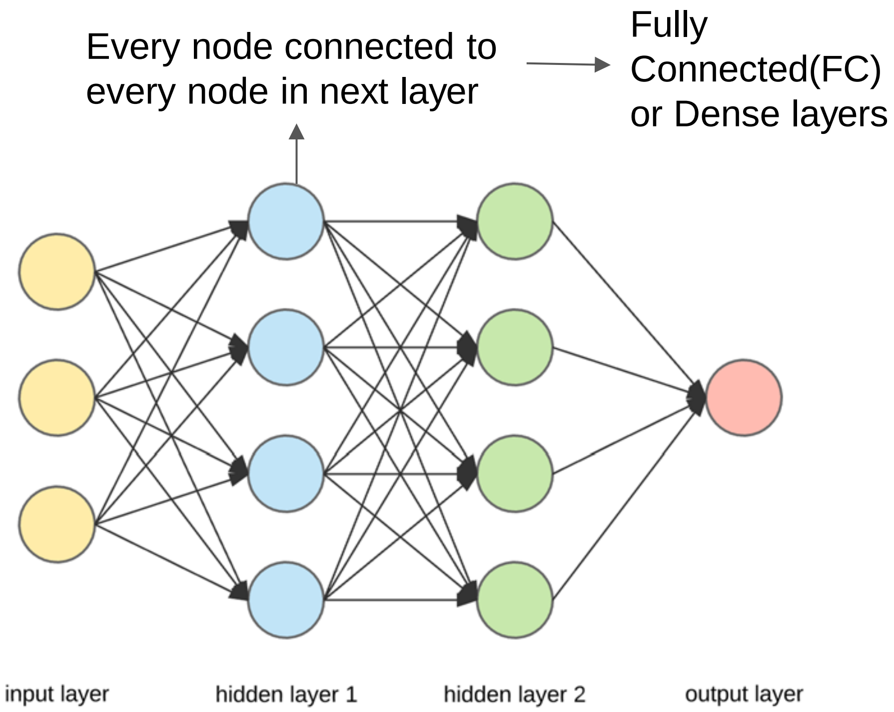
   - Forward Propagation
   - Backward Propagation

4. Deep Neural Networks

   - Hidden Layers
   - Why Deep Neural Networks?
   - Types of DNNs
     - **Multilayer Perceptron (ANN)**
     - Convolutional Neural Network
     - Recurrent Neural Network
     - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

Types of Neural Networks - Multilayer Perceptron



- Also known as conventional Artificial Neural Network (ANN) or Vanilla Neural Network

- Hyperparameters -
  - No. of hidden layers
  - No. of nodes in a layer
  - Learning rate for gradient descent, etc

- Optimal combination of hyperparameters - experimentation

Agenda

1. Neural Network Building Blocks
   - What is a Neuron?
   - Working of a Neuron
   - Analogy with Human Brain
   - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

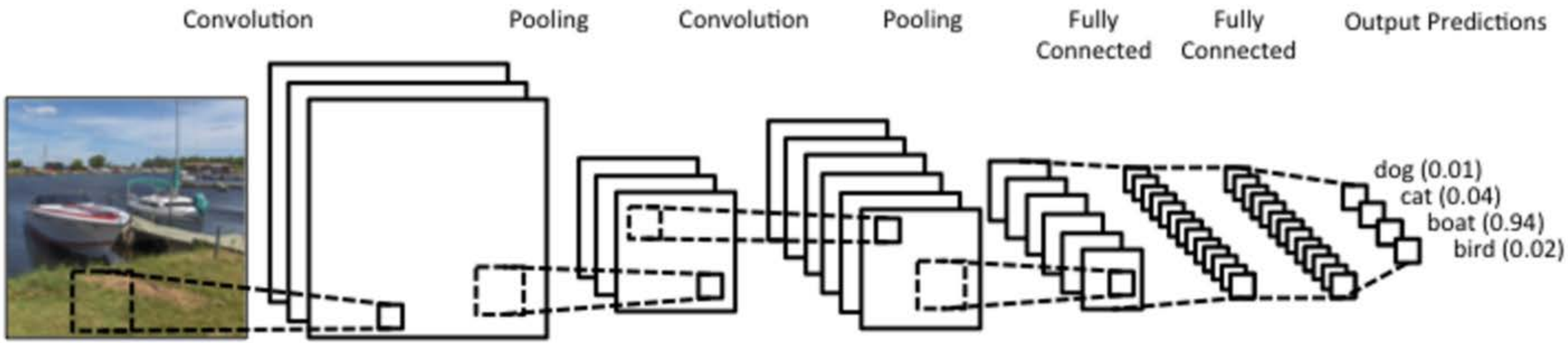
   - Forward Propagation
   - Backward Propagation

4. Deep Neural Networks

   - Hidden Layers
   - Why Deep Neural Networks?
   - Types of DNNs
     - Multilayer Perceptron (ANN)
     - **<u>Convolutional Neural Network</u>**
     - Recurrent Neural Network
     - Generative Adversarial Networks (GANs)

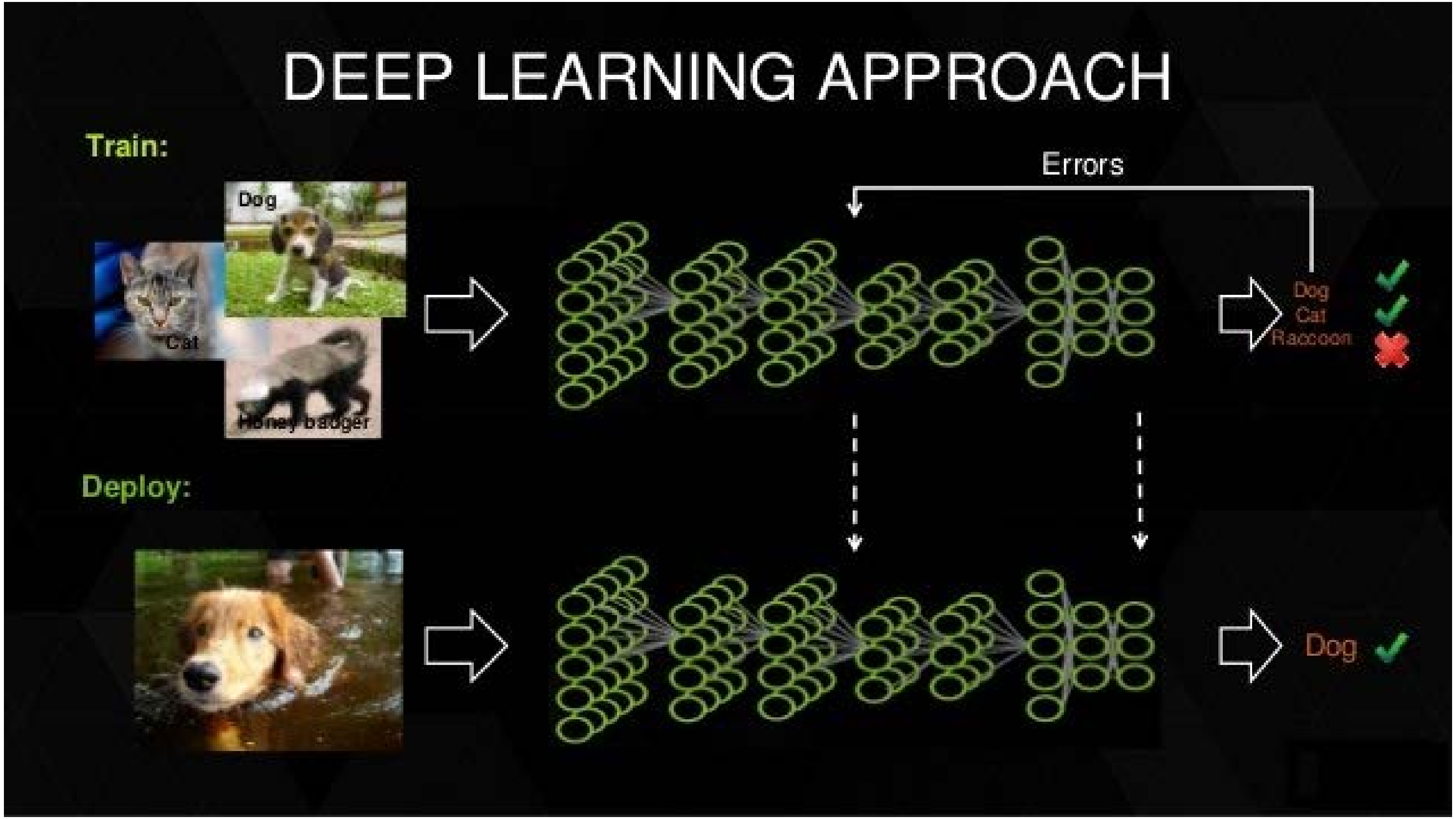
5. Applications of Deep Neural Networks

Types of NN - Convolutional Neural Network (CNN)



- CNNs (by Geoffrey Hinton) are a type of NN used often for image/video data
- Use a window to scan over an image to look for prominent features in images
- Introduced new layers for better data processing like pooling, dropout, etc.

Deep Neural Network - CNN as Eyes



Agenda

1. Neural Network Building Blocks
   - What is a Neuron?
   - Working of a Neuron
   - Analogy with Human Brain
   - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

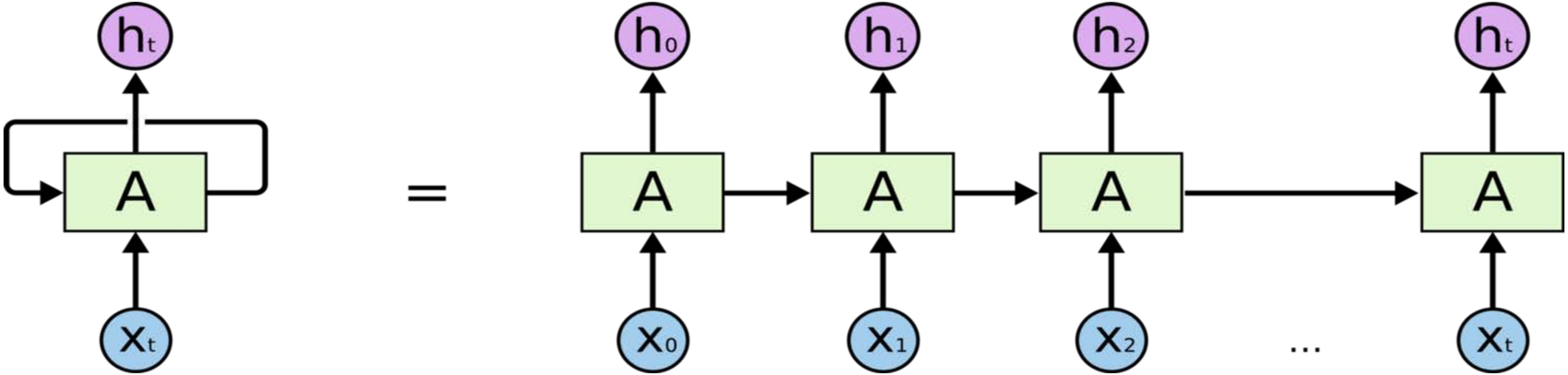
   - Forward Propagation
   - Backward Propagation

4. Deep Neural Networks

   - Hidden Layers
   - Why Deep Neural Networks?
   - Types of DNNs
     - Multilayer Perceptron (ANN)
     - Convolutional Neural Network
     - **Recurrent Neural Network**
     - Generative Adversarial Networks (GANs)

5. Applications of Deep Neural Networks

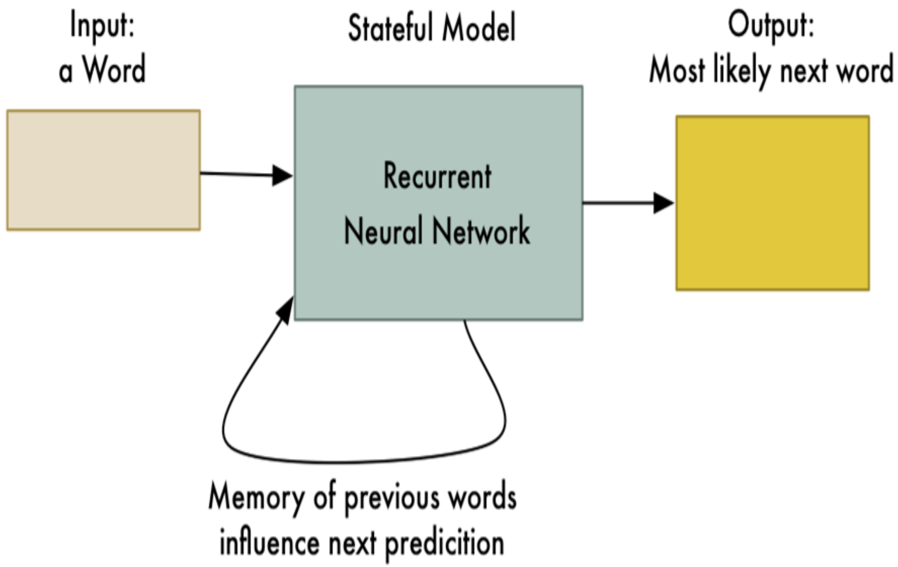
Types of NN - Recurrent Neural Network (RNN)



- Use internal memory to predict next state making it ideal for audio, text and other sequential data
- Each layer has two inputs, the present and the recent past i.e. the current layer output and the previous layer output

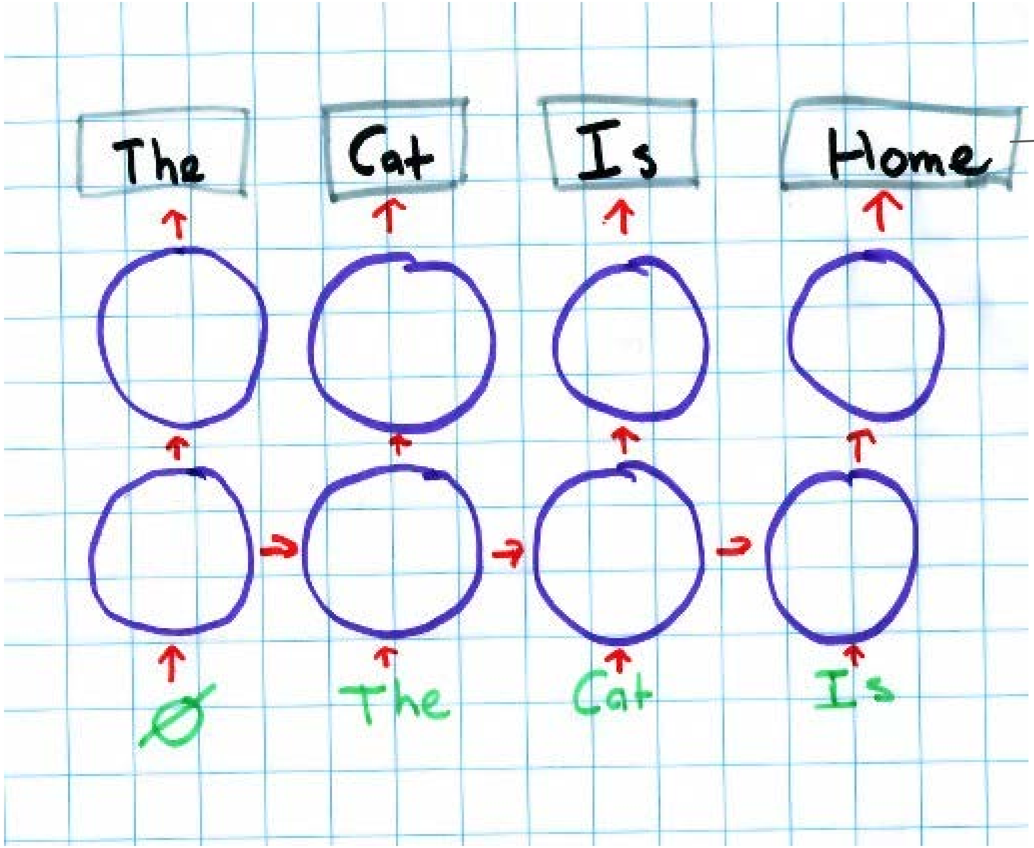
- Remember the previous input states to give output for further states

Deep Neural Network - RNN as Ears and Mouth



- RNNs are used to read and predict text
- They can also be used for identifying sounds such as song lyrics
- We also use them to generate text, as well as speech

RNN - Working Example - Predict next word in sentence



Prediction - Home
Probability - 0.7

- Predict the next word based on previous
- Output at each time-step - list of next probable words
- Pick most probable

Agenda

1. Neural Network Building Blocks
   - What is a Neuron?
   - Working of a Neuron
   - Analogy with Human Brain
   - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

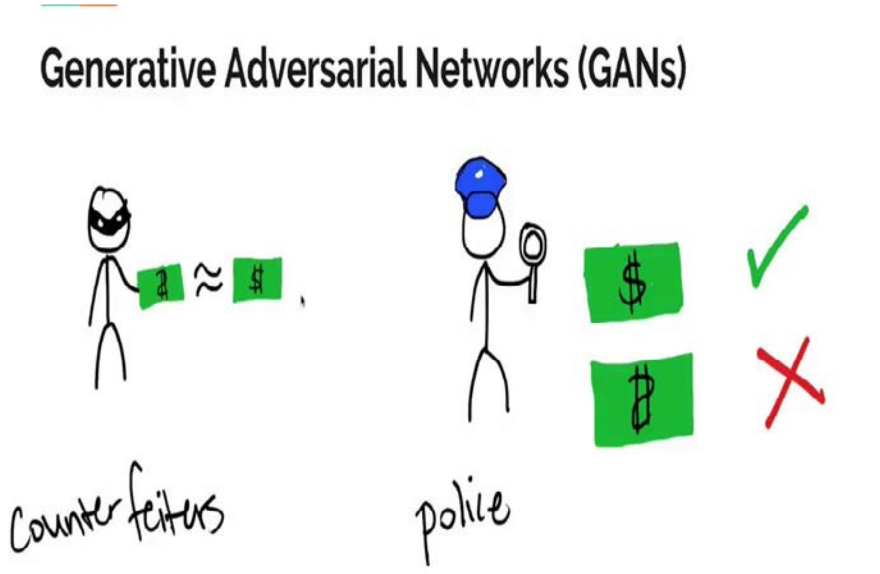
   - Forward Propagation
   - Backward Propagation

4. Deep Neural Networks

   - Hidden Layers
   - Why Deep Neural Networks?
   - Types of DNNs
     - Multilayer Perceptron (ANN)
     - Convolutional Neural Network
     - Recurrent Neural Network
     - **Generative Adversarial Networks (GANs)**

5. Applications of Deep Neural Networks

Types of NN - Generative Adversarial Networks (GANs)

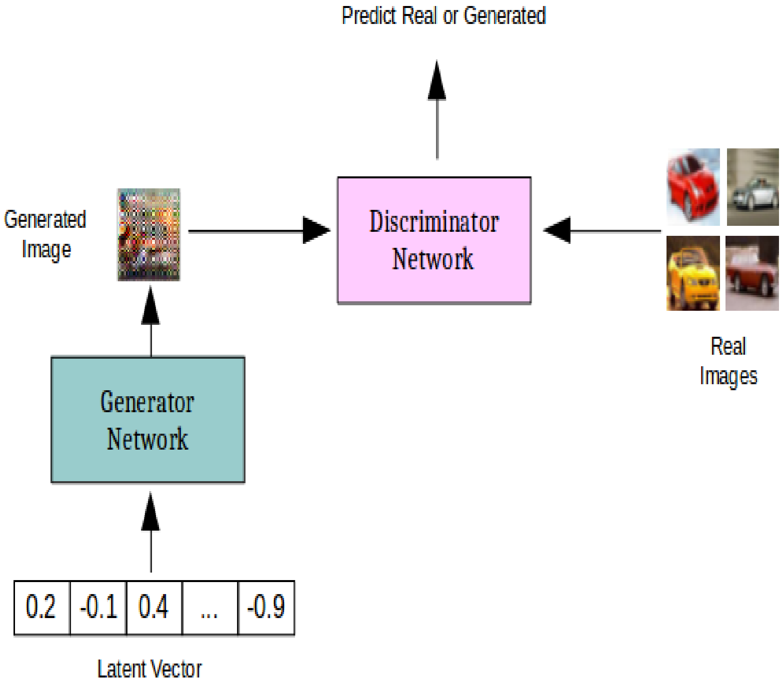


Try to develop fake currency to fool police

Use methods to identify the fake currency

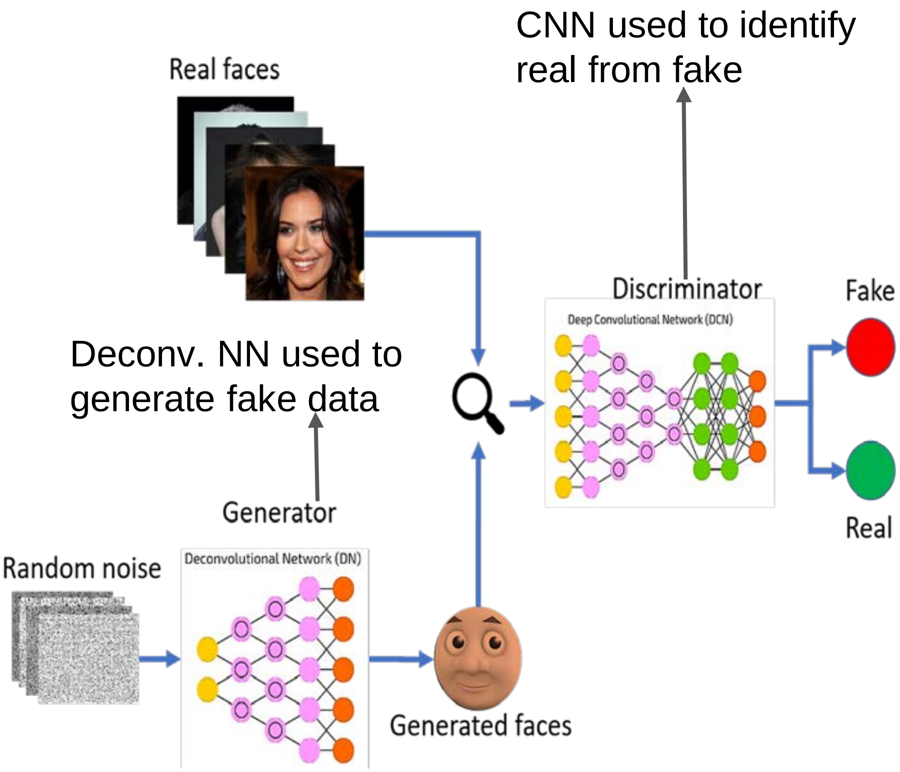
- Deal with data generation and imitation
- Two networks -
  - **Generator** - generates new data
  - **Discriminator** - evaluates authenticity
- Generator : Counterfeiter as Discriminator : Police

Generative Adversarial Networks (GANs)



- **Generator** - takes random data vector (noise), transforms it to target data (eg. cars)
- **Discriminator** - Tries to differentiate the generated data from real, outputs a probability

Generative Adversarial Networks (GANs) - Working



- If discriminator fooled - discriminator improved to identify better
- If discriminator catches forgery - generator makes more authentic data

Agenda

1. Neural Network Building Blocks
    - What is a Neuron?
    - Working of a Neuron
    - Analogy with Human Brain
    - Perceptron

2. Why we need Neural Networks

3. Working of a Neural Network

    - Forward Propagation
    - Backward Propagation

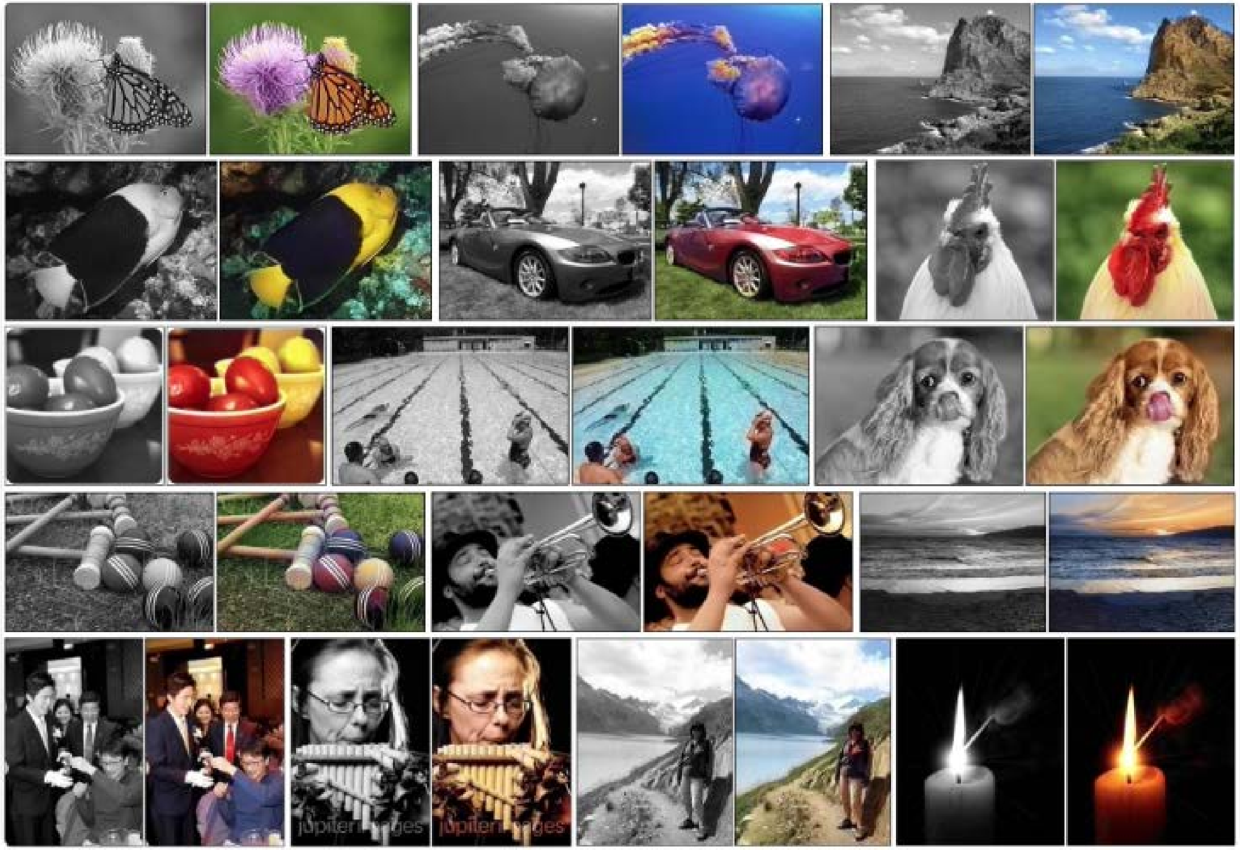
4. Deep Neural Networks

    - Hidden Layers
    - Why Deep Neural Networks?
    - Types of DNNs
        - Multilayer Perceptron (ANN)
        - Convolutional Neural Network
        - Recurrent Neural Network
        - Generative Adversarial Networks (GANs)

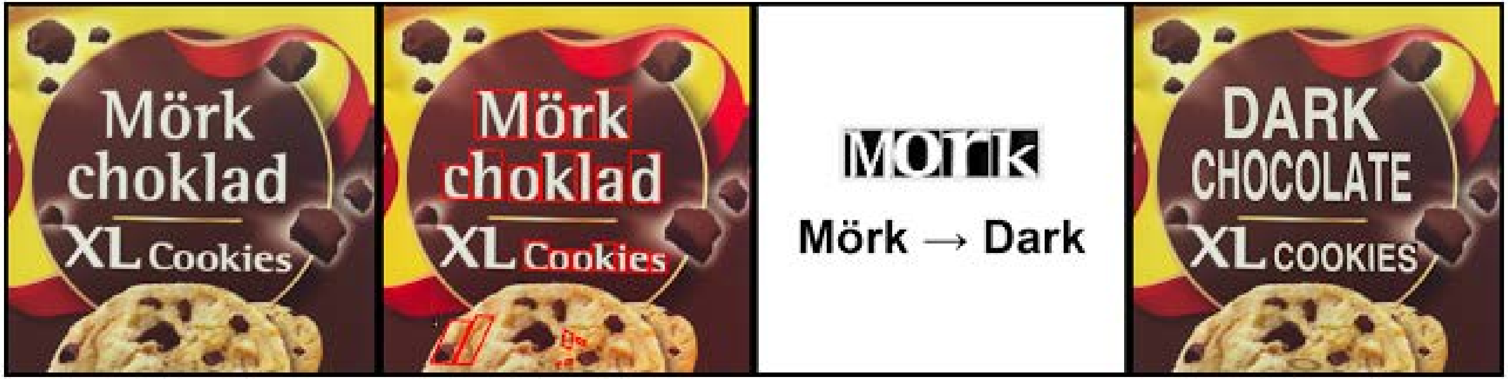
5. Applications of Deep Neural Networks

Applications of Deep Neural Networks

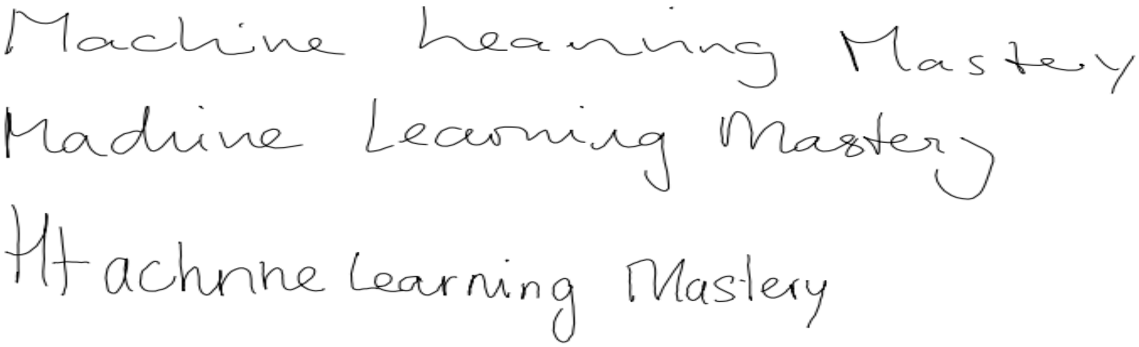
1. Image Colourization



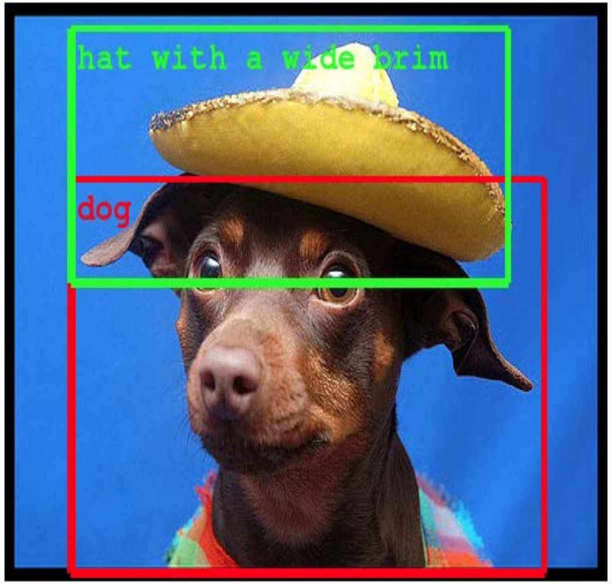
2. Machine Translation



3. Auto Handwriting Generation



4. Object Classification and Detection



5. Image Captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."

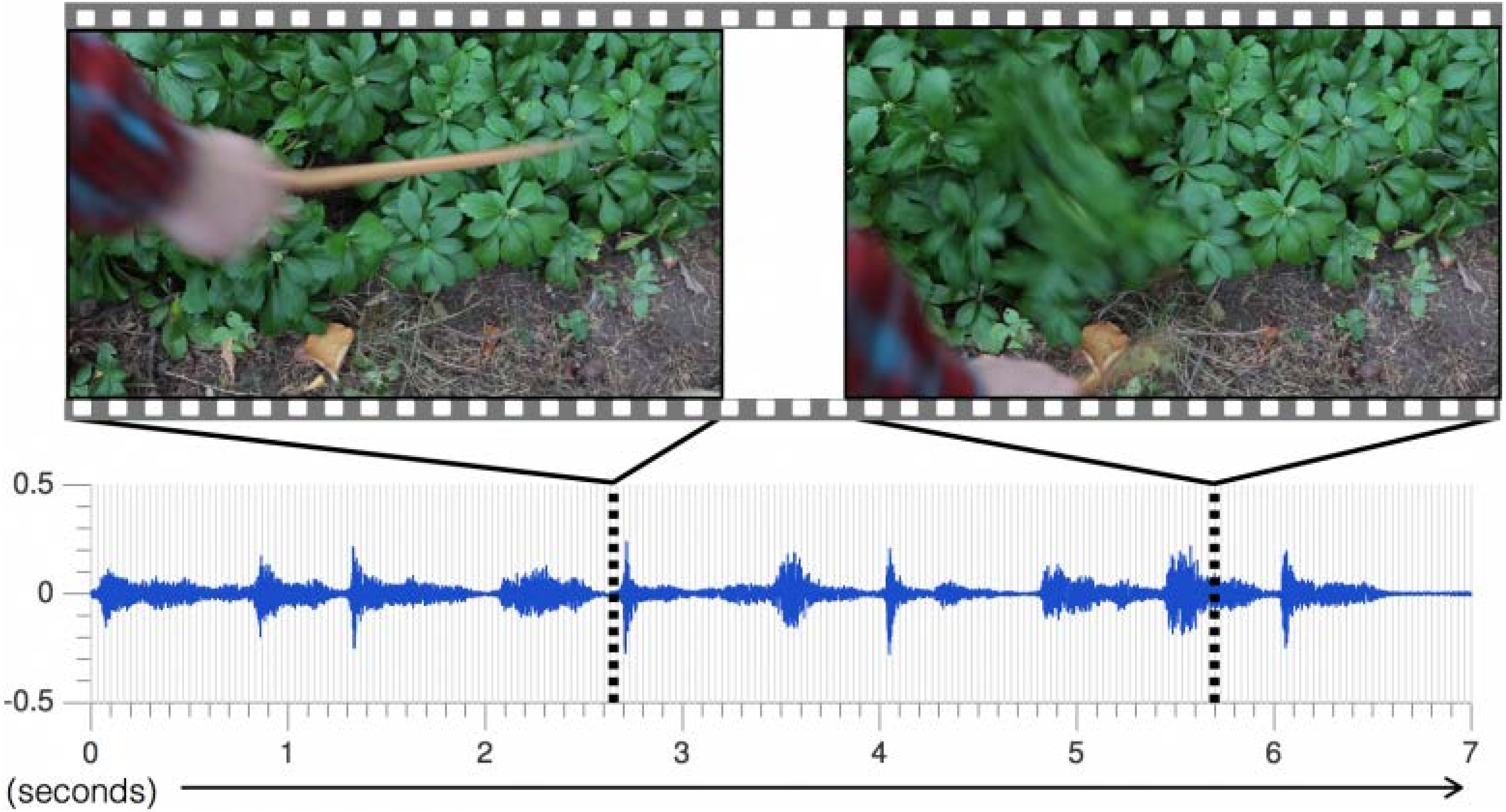
"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

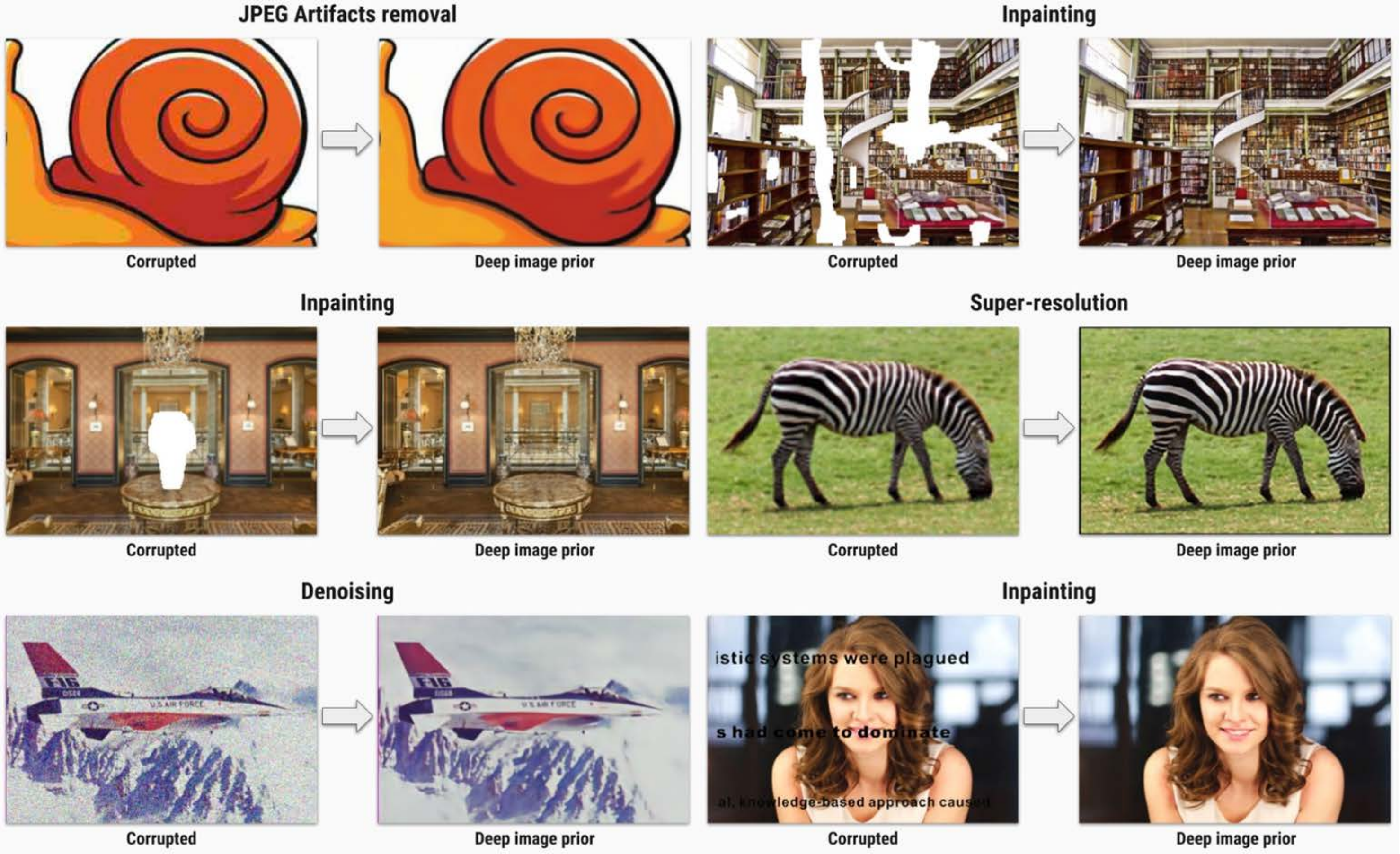
6. Self-Driving Cars

7. Style Transfer

8. Adding Sounds to Silent Movies

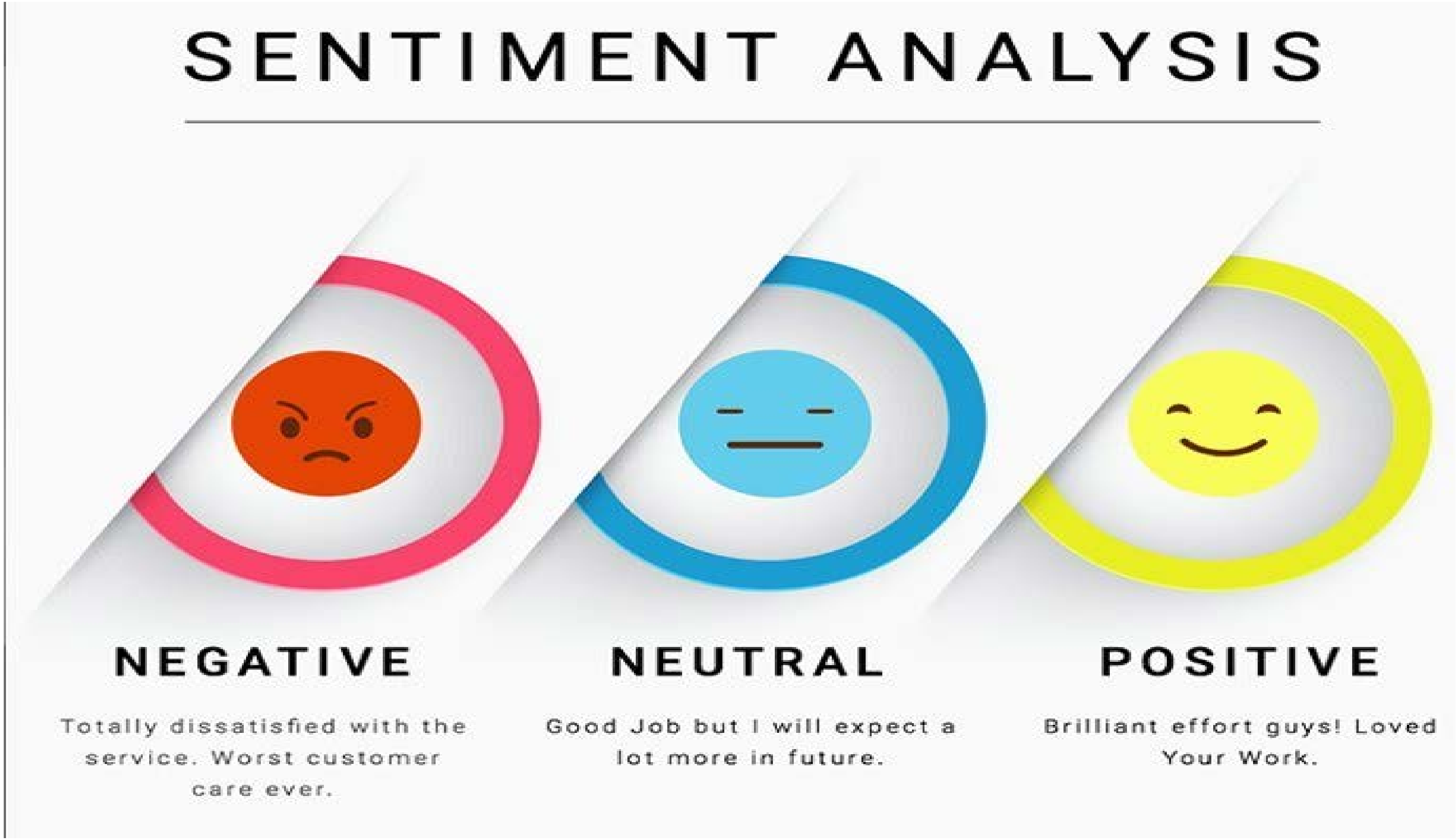


9. Image Handling

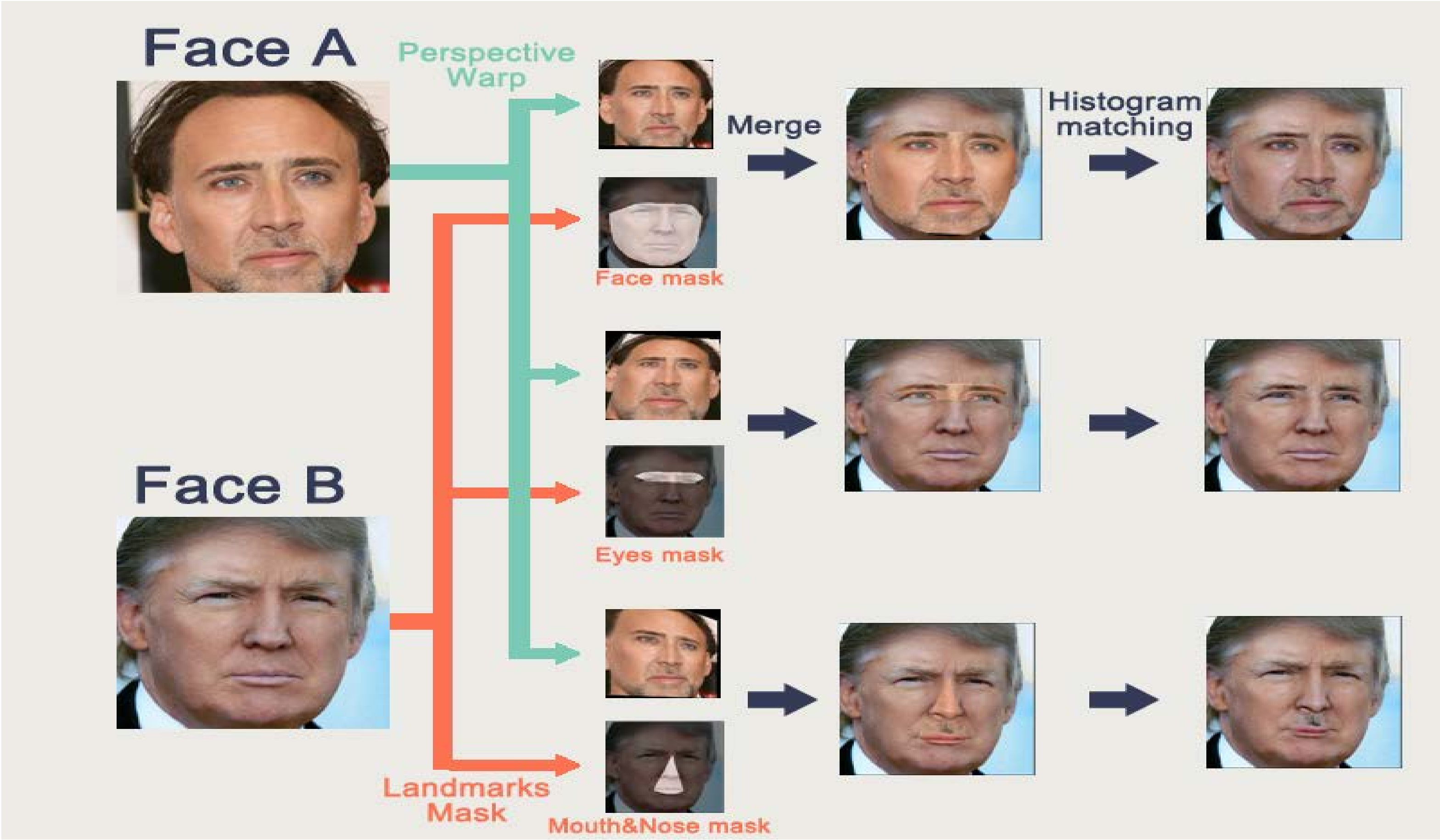


Deep Image Prior - a GAN

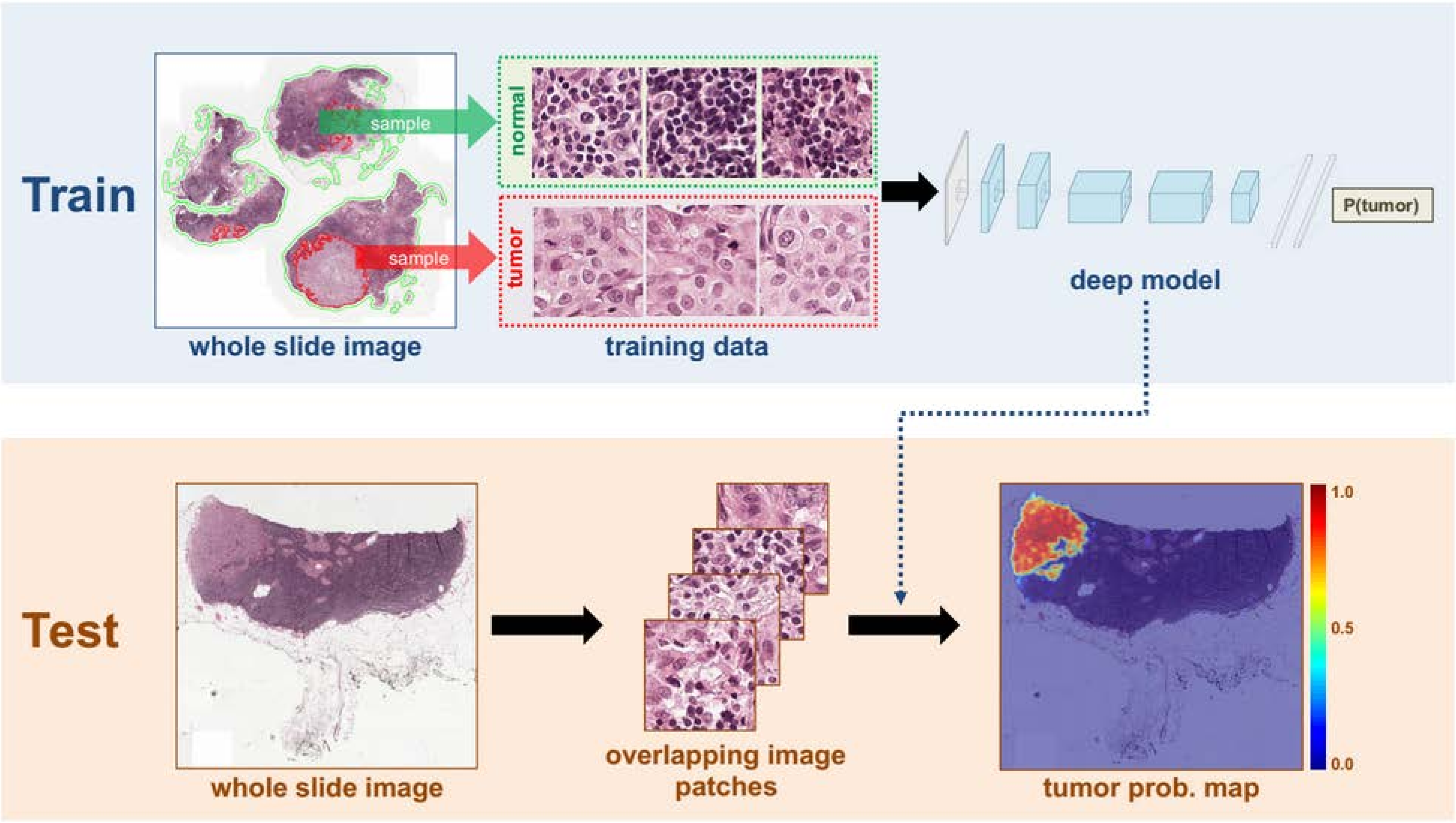
10. Sentiment Analysis



11. Face Swapping



12. Cancer Detection and Healthcare



And many more...