IDC410
A course on Image Processing and Machine Learning
(Lecture 08)

Shashikant Dugad,

IISER Mohali

Reading Material

Suggested Books:

1. **Neural Networks and Deep Learning by Michael Nielsen**
2. **Fundamentals of Deep Learning by Nikhil Buduma**

Source for this presentation:

https://www.scaler.com/topics/deep-learning/introduction-to-feed-forward-neural-network/

https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network

http://machine-learning-for-physicists.org. by Florian Marquardt

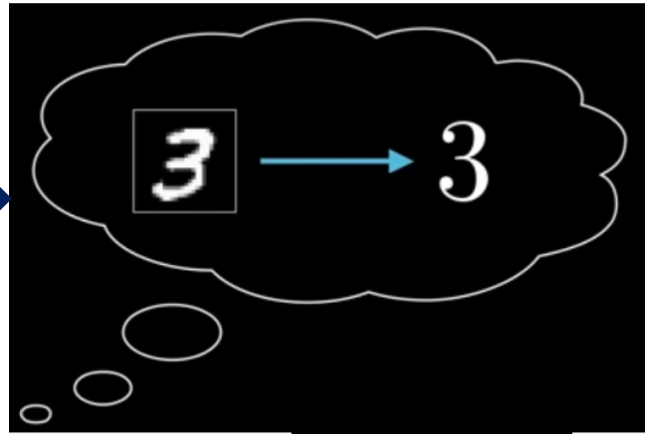
3Blue1Brown (Youtube Videos)

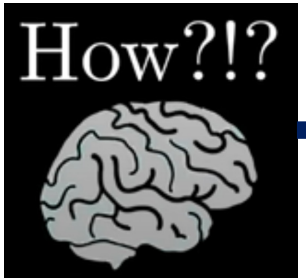
https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks

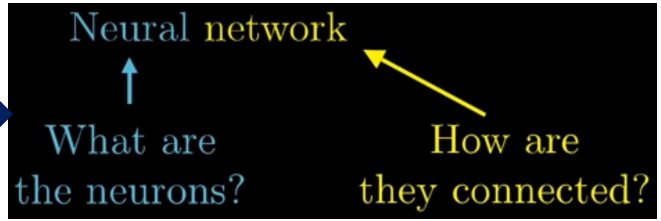
Feed Forward Neural Network (FFN)

Pattern Recognition in our brain: How does it happen?

MNSIT Dataset of handwritten numbers: 28 x28 pixels

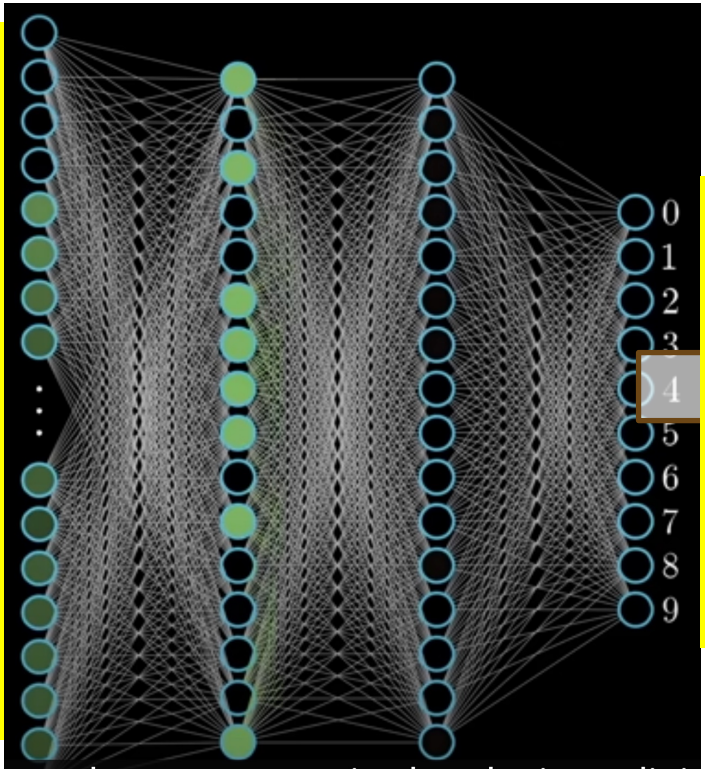






Possible neurological process in brain

Input image: 28x28 pixels
784 inputs of gray scale intensity



0.01
0.02
0.01
0.01
0.88
0.02
0.01
0.01
0.02
0.01

10 binary outputs representing numbers between 0 to 9

Each Neuron contains a number

Feedforward Neural Network (FFN)

- In Feedforward Neural Network (FNN) the information moves in ONLY one direction, i.e., forward from the input nodes, through the hidden nodes (if any), and to the output nodes
- There are no cycles or loops in the network
- Feedforward neural networks are simpler than their counterparts like recurrent neural networks (RNN) and convolutional neural networks (CNN)
- The architecture of a feedforward neural network consists of three types of layers: the input layer, hidden layers, and the output layer. Each layer is made up of units known as neurons, and the layers are interconnected by weights.
- **Input Layer:** This layer consists of neurons that receive inputs and pass them on to the next layer. The number of neurons in the input layer is determined by the dimensions of the input data.

Feedforward Neural Network (FFN)

- **Hidden Layers:** These layers are not exposed to the input or output and can be considered as the computational engine of the neural network. Each hidden layer's neurons take the weighted sum of the outputs from the previous layer, apply an activation function, and pass the result to the neurons of the next layer. The network can have zero or more hidden layers.

- **Output Layer:** The final layer that produces the output for the given inputs. The number of neurons in the output layer depends on the number of possible outputs the network is designed to produce

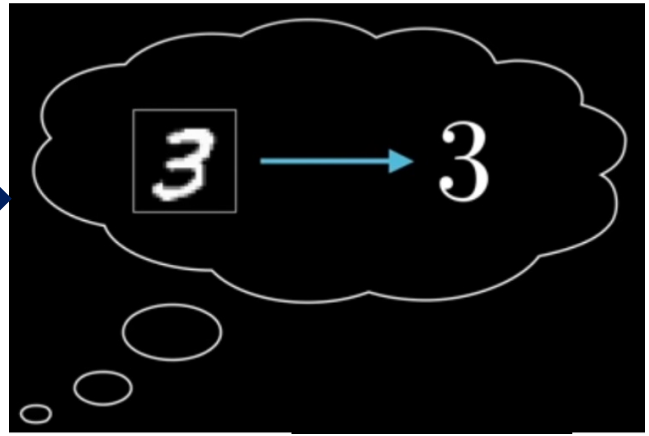
- **Fully connected FFN:** When each neuron in one layer is connected to every neuron in the next layer, then it makes *fully connected (feedforward) neural network* (FCN)

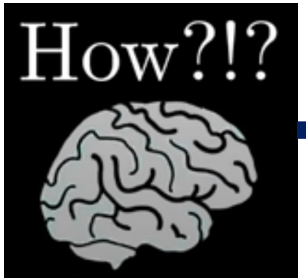
Feedforward Neural Network

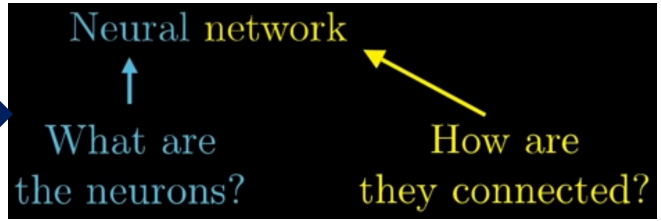
- **Feedforward Phase:** In this phase, the input data is fed into the network, and it propagates forward through the network. At each hidden layer, the weighted sum of the inputs is calculated and passed through an activation function, which introduces non-linearity into the model. This process continues until the output layer is reached, and a prediction is made
- **Backpropagation Phase:** Once a prediction is made, the error (difference between the predicted output and the actual output) is calculated. This error is then propagated back through the network, and the weights are adjusted to minimize this error. The process of adjusting weights is typically done using a gradient descent optimization algorithm
- **Training:** A known dataset is used to adjust the weights through an iterative process by passing the known dataset through the network multiple times, and each time, the weights are updated to reduce the error in prediction. This process is known as gradient descent, and it continues until the network performs satisfactorily on the training data.

Pattern Recognition in our brain: How does it happen?

MNSIT Dataset of handwritten numbers: 28 x28 pixels

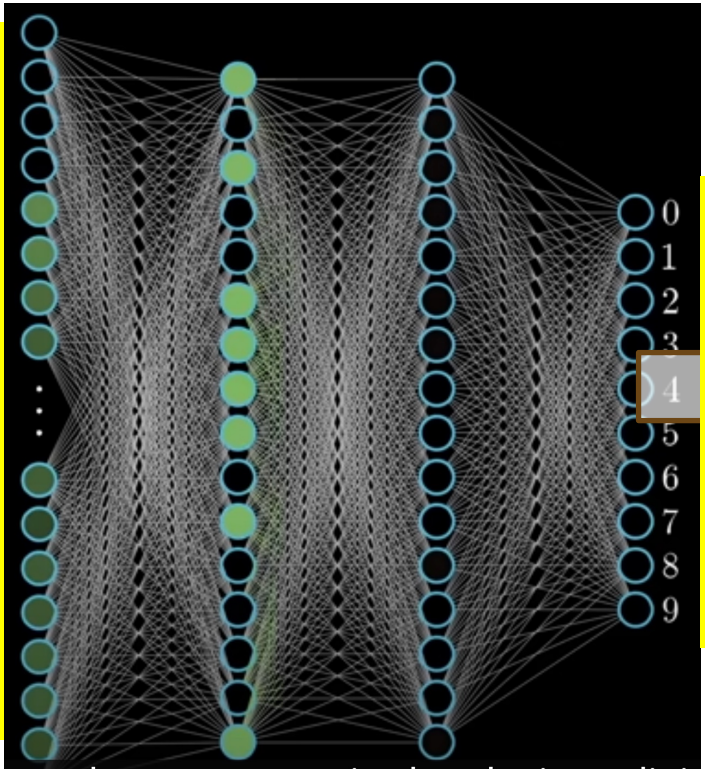






Possible neurological process in brain

Input image: 28x28 pixels
784 inputs of gray scale intensity

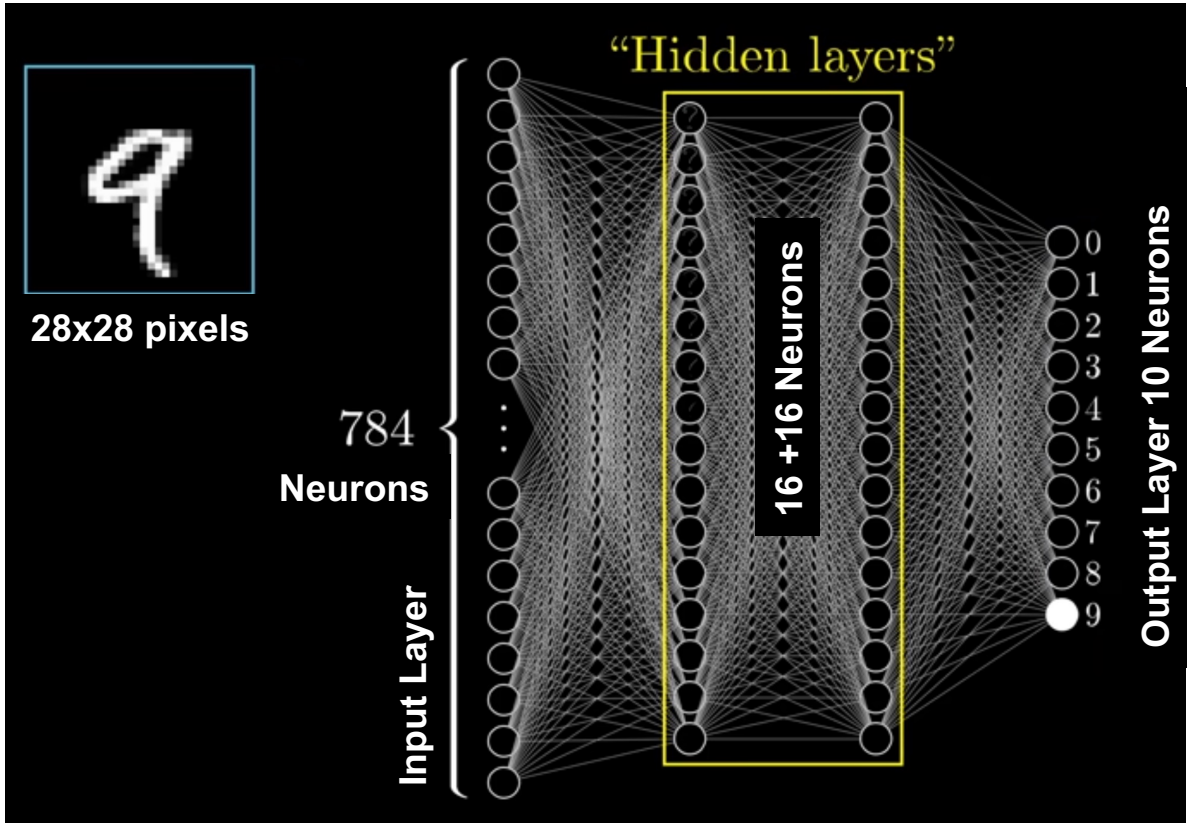


0.01
0.02
0.01
0.01
0.88
0.02
0.01
0.01
0.02
0.01

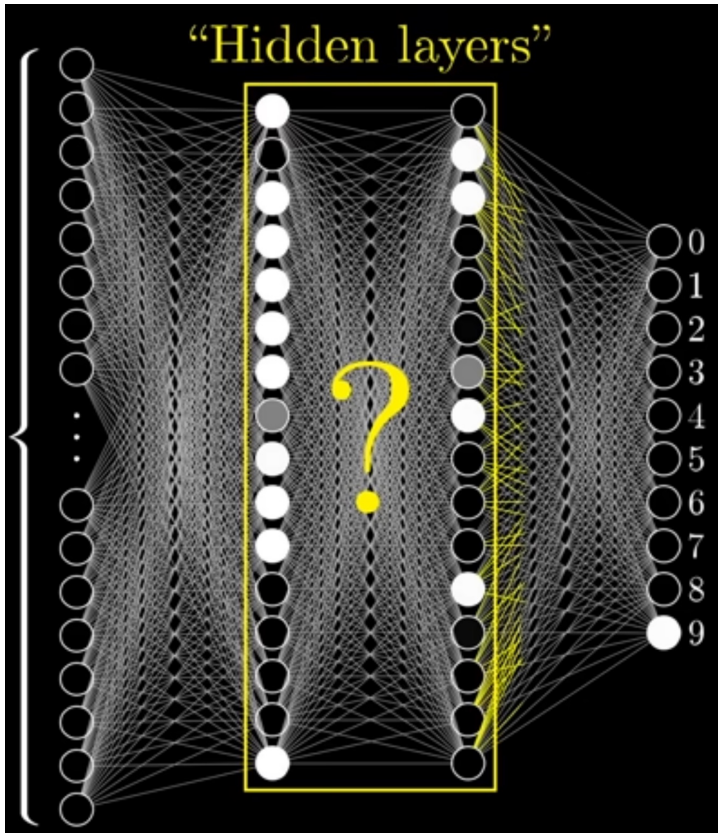
10 binary outputs representing numbers between 0 to 9

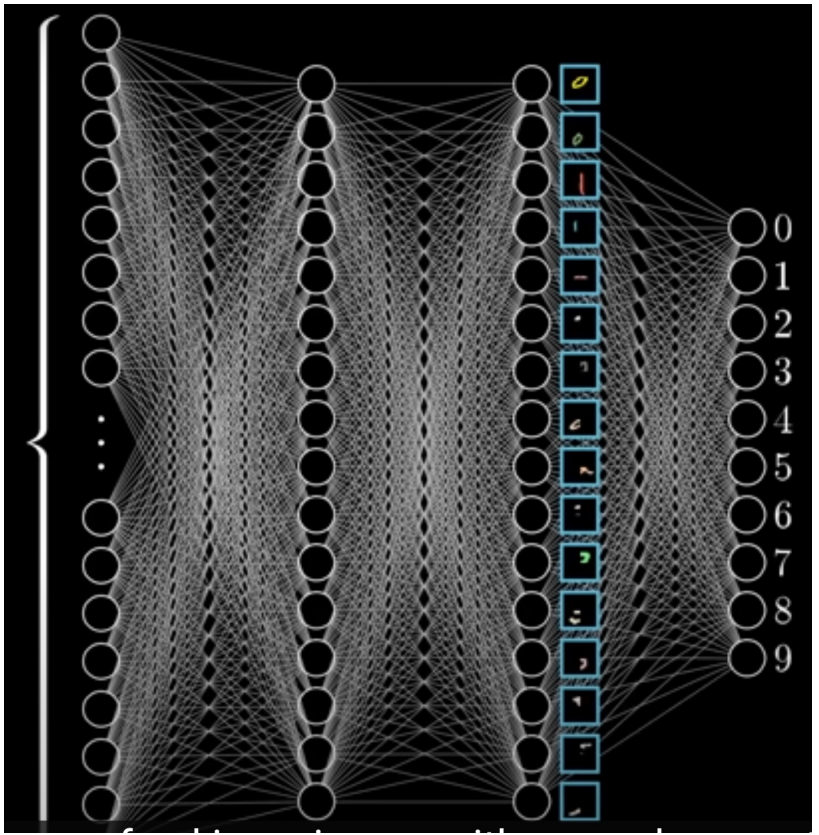
Each Neuron contains a number

Formulation of thought process!

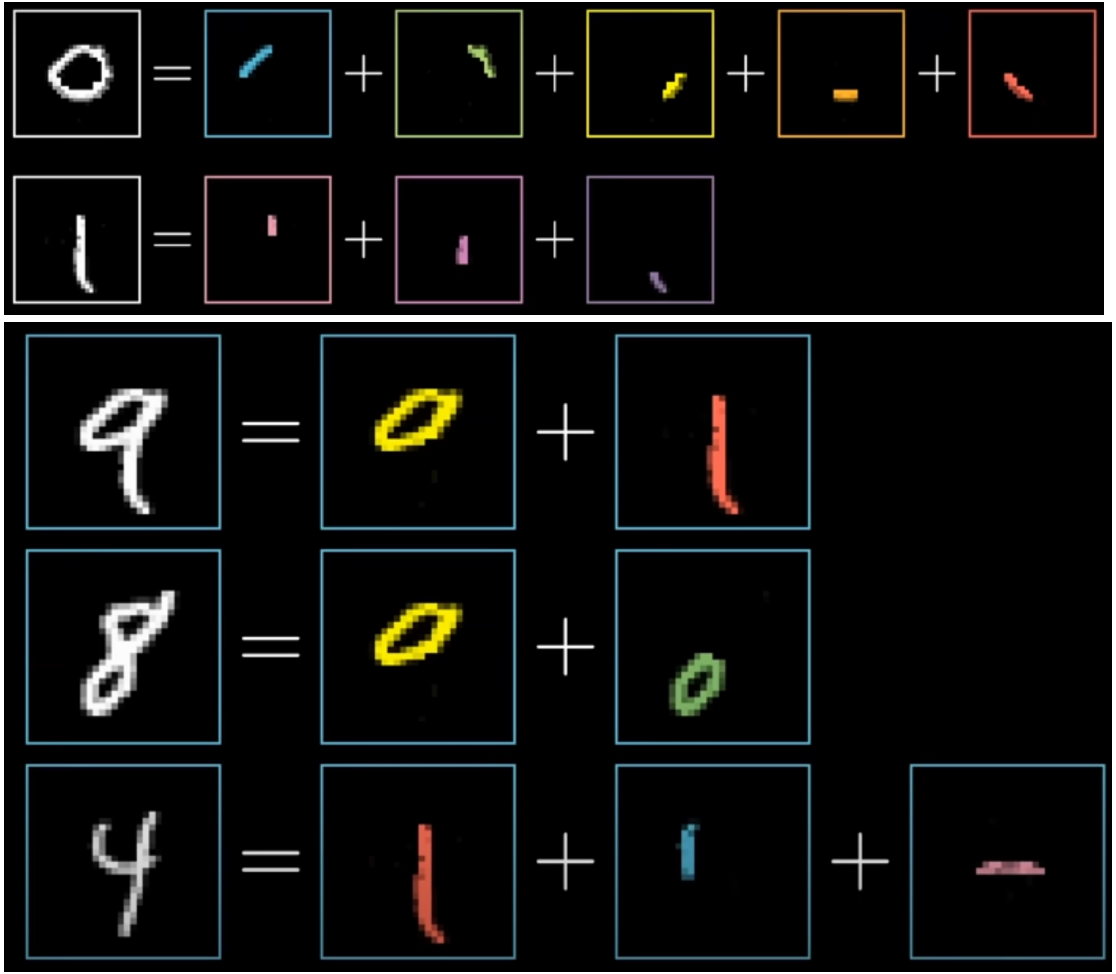


Quantification of thought process!

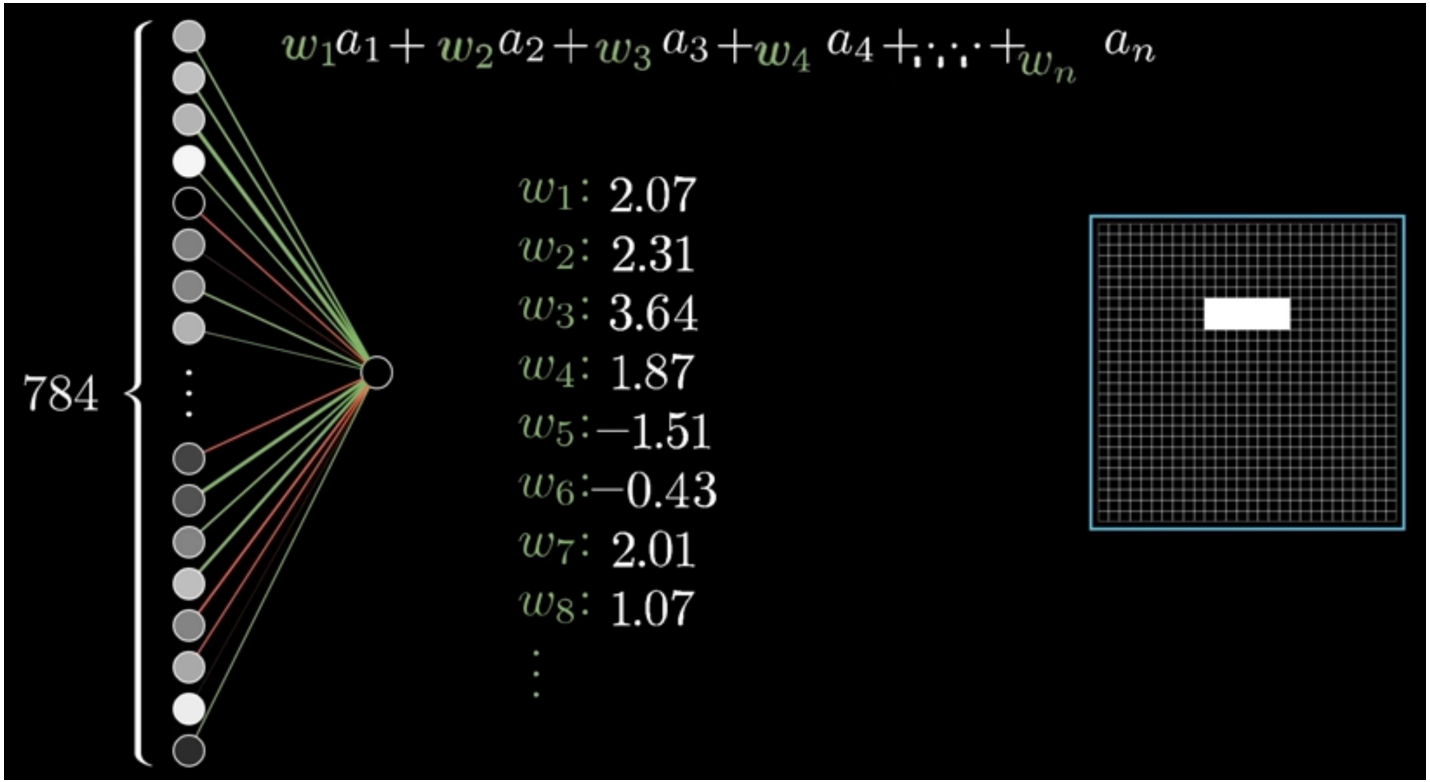




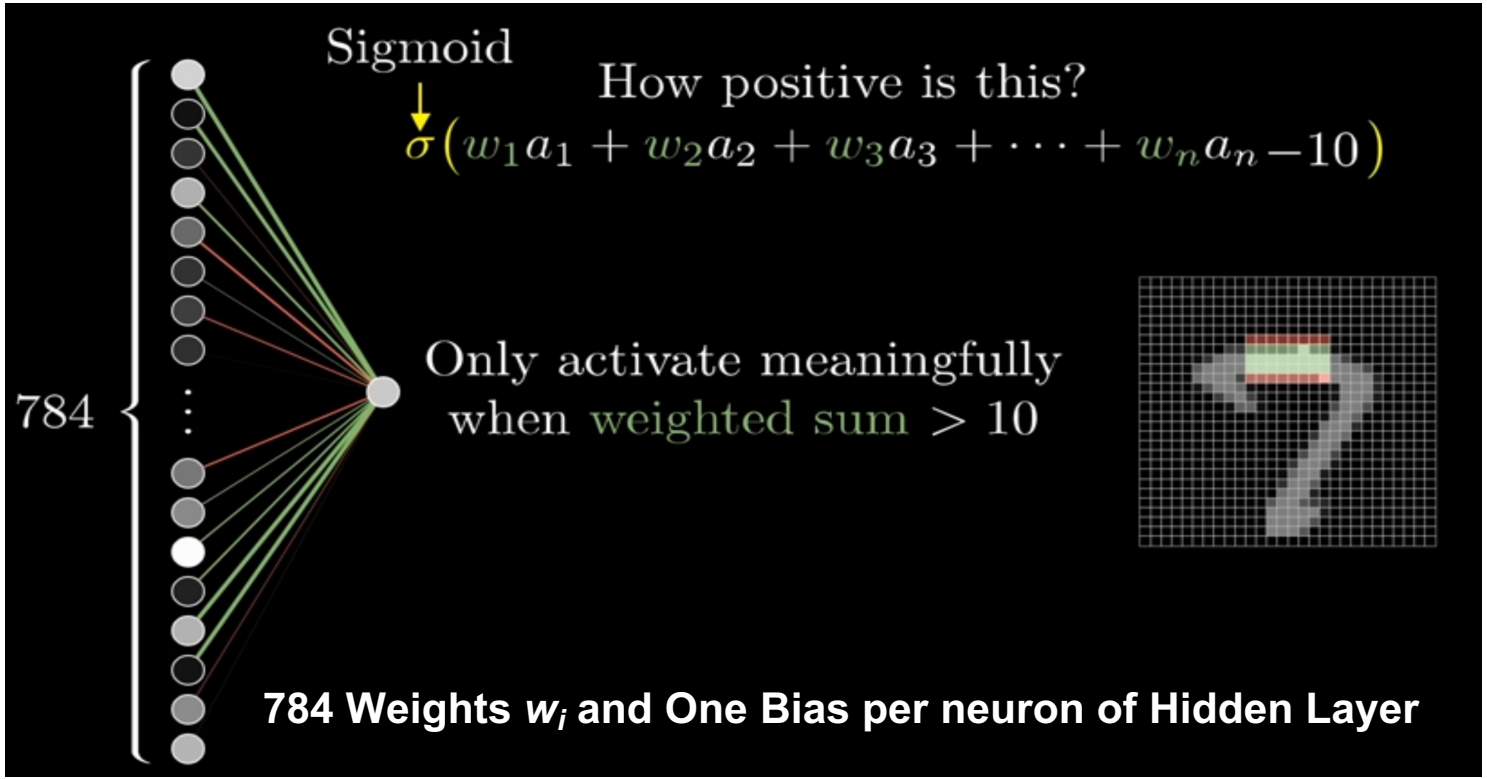
Features at kernel level



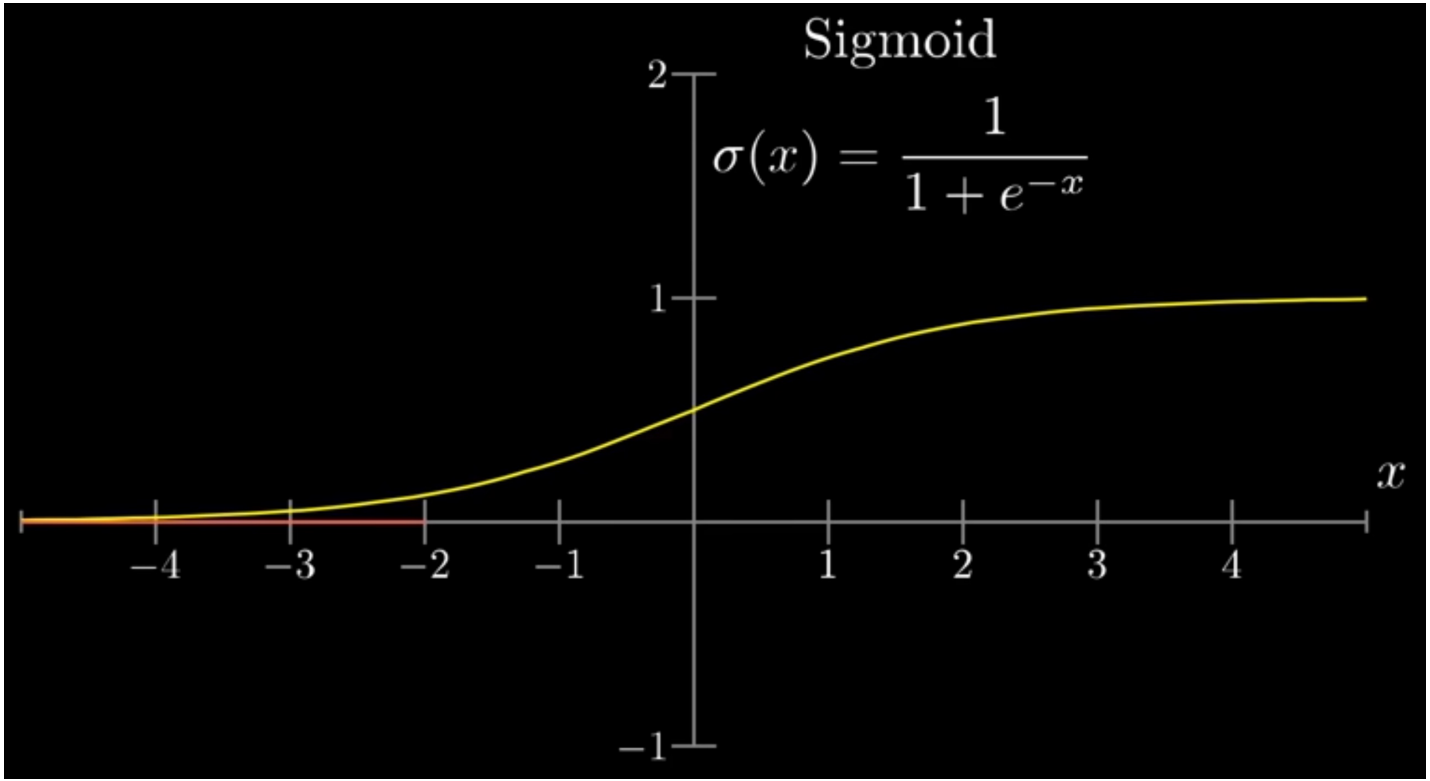
Concept of weights and Biases



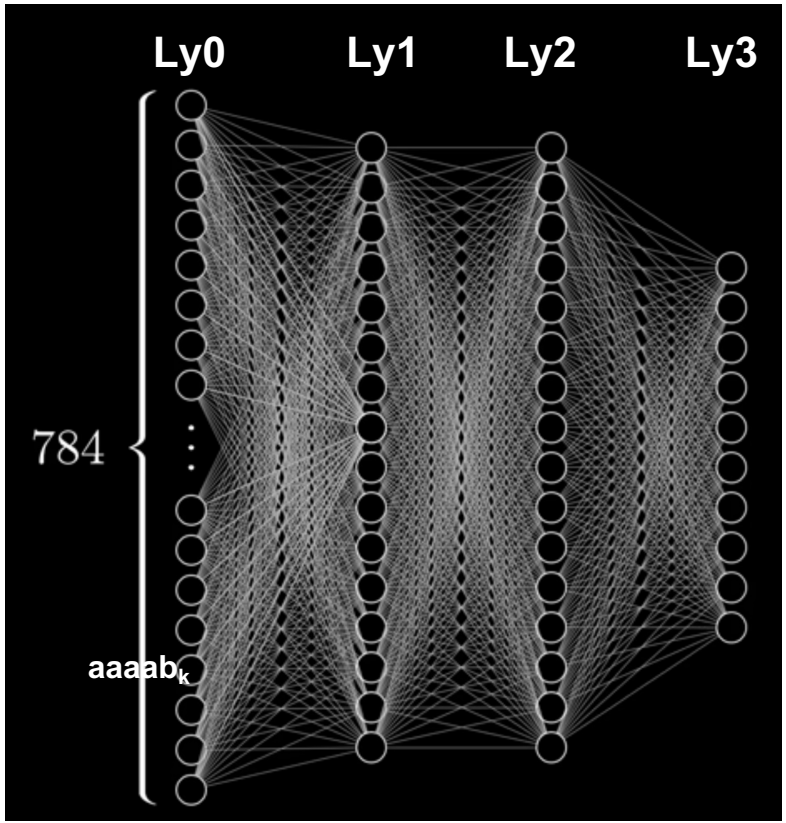
Concept of Weights and Biases



Activation Function



Evaluation of Total # of Weights and Biases



- Total No. Layers: 4
- Input Layer: Layer 0
- Hidden Layers: Layer 1 & Layer 2
- Output Layer: Layer 4

- # of Neurons in Ly0: 28 x 28 = 784
- # of Neurons in Ly1: 16
- # of Neurons in Ly2: 16
- # of Neurons in Ly3: 10

- # of weights for Ly0: 784 x 16 = 12544
- # of weights for Ly1: 16 x 16 = 256
- # of weights for Ly2: 16 x 10 =. 160
- Total # of weights: 12960

- Total # of biases: 16 + 16 + 10 = 42

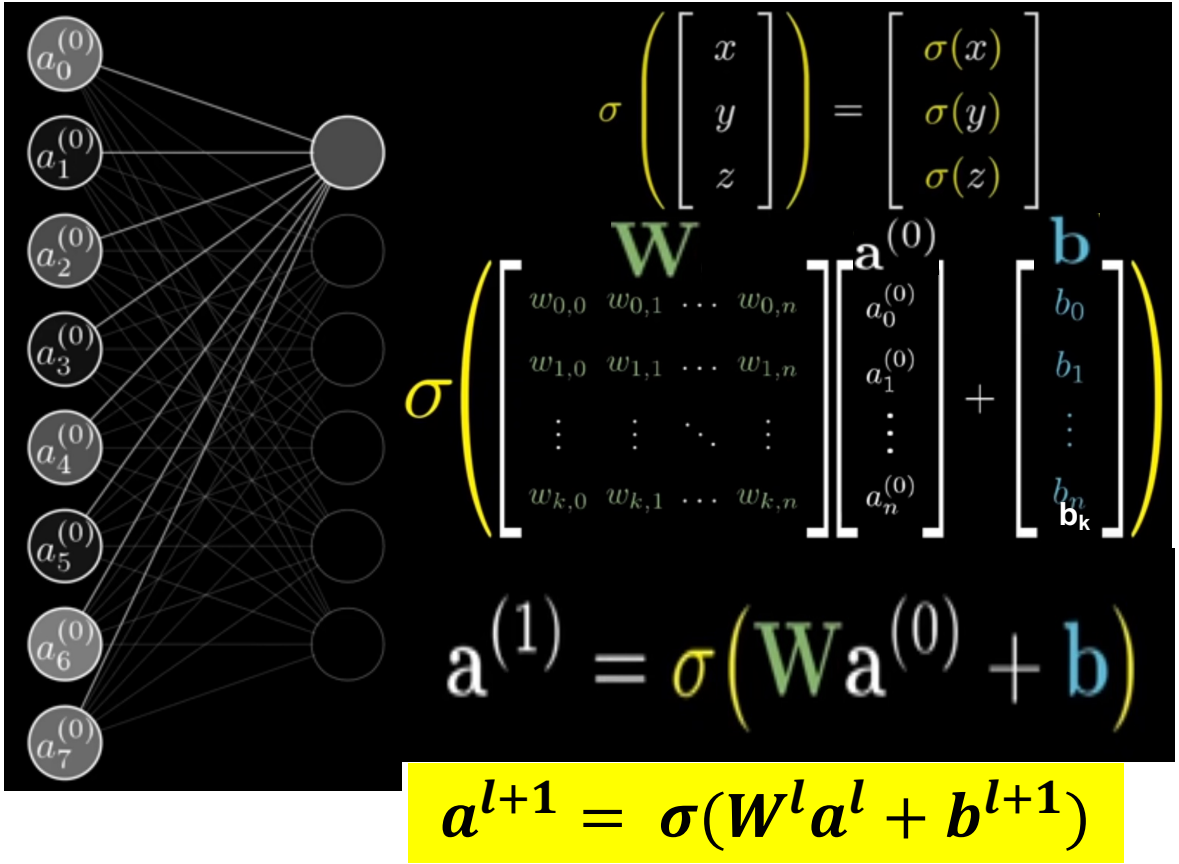
Compact form for Weights and Biases

- Consider two layers, with Ly0 having ***n*** neurons and Ly1 has ***k*** neurons
- a_i^0 is the normalized value of intensity of neuron *i* in Ly0
- w_{ji}^0 is the weight associated with neuron *i* in Ly0 connected to neuron j in Ly1
- Weighted sum at neuron j in Ly1

$$s_j = \sum_{i=0}^{n} w_{ji}^0 a_i^0$$

- s_j is subjected to activation function to get the value a_j^1 of neuron j in Ly1

$$a_j^1 = \sigma(s_j)$$



$$a^{l+1} = \sigma(W^l a^l + b^{l+1})$$

General Formula for Weights and Biases

- Total number layers in a network: **N**
- Number of Neurons per layer: n_i where i=1, 2, 3 …N is layer number. n_i is number of neurons in layer i.
- Calculate the total number of weights and biases for such network

Activation Functions

Activation Function for a Neuron and Perceptrons

- **Without activation functions, neural networks would just consist of linear operations like matrix multiplication. All layers would perform linear transformations of the input, and no non-linearities would be introduced.**
- **A neural network without an activation function would essentially behave like a simple linear regression model, meaning it can only learn linear relationships between inputs and outputs, unable to capture complex patterns in real-world data due to the lack of handling non-linearity limiting its ability to solve intricate problems like image recognition or natural language processing etc.**
- **Activation functions are the building block of neural networks, enabling to learn complex patterns in data.**
- **Introducing activation functions can mitigate this problem and enable neural networks to learn non-linear behaviours of the input. This greatly increases the flexibility and power of neural networks to model complex data**
- **There are several type of activation functions, each having its own unique properties and is suitable for certain use cases.**

Activation Function for a Neuron and Perceptrons

1. A biological neuron only fires when a certain conditions satisfied
2. Similarly, the artificial neuron will also fire, only when the sum of the inputs (weighted sum) exceeds a certain threshold value, say 0
3. This was the reason for choosing a ***Unit Step (Threshold) function*** as an ***activation function***, originally used by Rosenblatt
4. A smoother version of the above function such as the ***sigmoid*** function, the ***Hyperbolic tangent*** (tan***h***) function are preferred
5. Both ***sigmoid*** and ***tanh*** functions suffer from vanishing gradients problems
6. ***ReLU*** and ***Leaky ReLU*** are the most popularly used activation functions. They are comparatively stable over deep networks.