**What is Deep Learning:**

Deep learning is a subbranch of machine learning , concerned with solving high level problems by emulating the working of human brain.

**Neural Networks:**

Neural networks are a type of computational model inspired by the structure and function of the human brain. They consist of interconnected nodes, called neurons, organized into layers. Neural networks are used in machine learning and deep learning to process and analyze data.

Here's a basic overview of neural networks:

1. \*\*Input Layer:\*\* This layer receives the initial data or features that the network will process.

2. \*\*Hidden Layers:\*\* Between the input and output layers, one or more hidden layers can exist. These layers contain neurons that perform mathematical operations on the input data, introducing non-linearity and complexity to the network.

3. \*\*Output Layer:\*\* The final layer produces the network's output, which could be a classification label, a numerical prediction, or any other relevant result, depending on the task.

Each connection between neurons is associated with a weight, which determines the strength of the connection. During training, neural networks adjust these weights based on the error between their predictions and the actual target values, aiming to minimize this error. This process, known as backpropagation, involves gradient descent optimization algorithms.

**Weight:**

In a neural network, a weight is a parameter associated with each connection between neurons. These weights determine the strength of the connection and are used to multiply the input data or activations when calculating the output of a neuron. During training, these weights are adjusted through optimization algorithms like gradient descent to make the network's predictions more accurate.

**Gradient Descent:**

Gradient descent is an optimization algorithm used in machine learning and deep learning to minimize the loss or error of a model by iteratively adjusting its parameters, such as weights and biases

**What is Keras:**

TensorFlow is a software library created by Google to implement large scale machine learning models and to solve complex numerical problems.

**Why use keras:**

* Keras prioritizes developer experience
* Used in Research
* Keras makes it easy to turn models into products

Keras is a high-level api and this high-level api can run on multiple low-level api such as tensorflow etc. Keras works on the frontend and supports multiple backend engine.

We can train the data and divide the data on multiple gpu.

Records --→ 100

divide --→ 5 mini batches

Train --- > On multiple seperate GPUs.

**What is Keras?**

Keras is a high level api and it is written in python.And it can run on the top of tensorFlow , CNTK etc. It is designed to be modular, fast and easy to use.

Models Available in Keras:

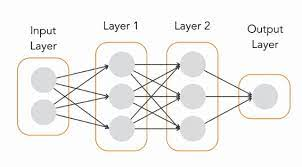
Two types of models:

* Sequential Model
* Functional Model

Sequential Models:

Linear Stack of Layers. (Sequence of Layers)

Input Layer(top of input layer)--→ add --→(Hidden layer 1)-→(Hidden layer 1 top)-→add—→ (Hidden Layer 2)--→(Hidden layer 2 top)--> add --→ (Output layer)



**First Step:**

Create an instance:

model = Sequential()

**Adding Layer:**

**Dense: Layername with 32 units(neurons)**

**784 features**

**The output of each of those 32 neurons will have the ReLU function applied to it individually.**

Rectified Linear Unit (ReLU): **ReLU is a simple but effective activation function.**

model.add(Dense(32 , input\_shape = 784 ,)))

model.add(Activation(‘relu’))

**Problem:**

It follows the Sequence. When we add input layer between them this is not possible.

**Functional Models:**

Used for defining complex models, such as multi-output models, directed acyclic graphs or model with shared layers.

Any layer can be connected with another layer like layer1 connected with layer 4.

**Steps:**

* Defining the Input
* Build layers
* Connecting Layers
* Creating the Model

from keras.layers import Input

from keras.layers import Dense

visible = Input(shape = (2,)) ---→Input Define

Build layers:

hidden = Dense(2)(visible) --→ connect with visible layer

Create Model:

model = Model(inputs = visible , outputs = hidden)

**Predefined Neural Network Layers:**

Core layers

Convolutional Layers

Pooling Layers

Locally-connected Layers

Recurrent Layers

Normalization Layers

Noise Layers

Embedding Layers

Merge Layers

Advanced Activation Layers

**Performing Regularization Using Keras:**

* Underfitting
* Appropriate fitting
* Overfitting

**Reduce Overfitting:**

By using Regularization

Regularization --→ penalizes the weight matrices of nodes

(Nodes ki values ko km kr daina)

To much alter the weights --→ underfitting problem rises

**First Technique:**

**Dropout:**

At every iteration, it randomly selects some nodes and remove them , alog with all of their incoming and outgoing connections.

Iteration --→ different set of nodes --→ different set of outputs

Just like ensemble technique in Machine learing.

**Data Augmentation:**

Reduce Overfitting is to increase the size of training data

Dealing with images(digits dataset) --→ training data --→ rotating image , flipping image , shifting etc etc

* Improving the accuracy of model
* Mandatory trick to improve our predictions