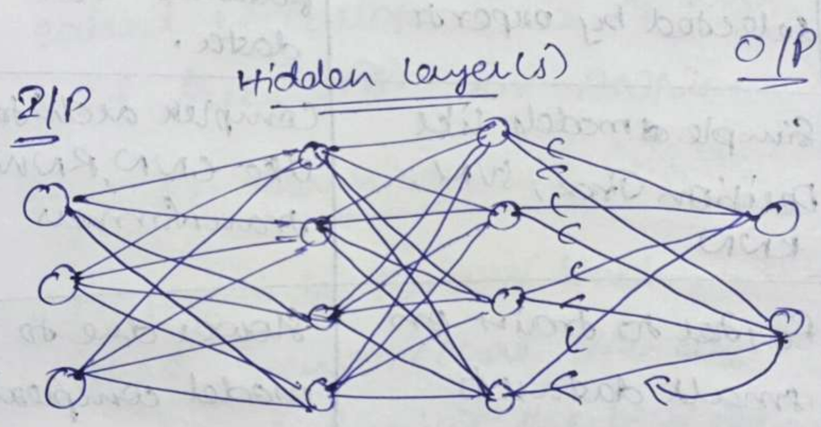
Unit-3

② Deep Learning:-

- It is a subfield of ML that uses artificial neural networks with multiple layers to learn patterns & make predictions.
- It mimics how the human brain works by processing large amounts of data and identifying complex patterns.
- As they use ANNs, they are also called Deep Neural Networks (DNNs).
- It has become increasingly popular in recent years because of the advances in processing power and the availability of large datasets.
- The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes.
- These networks can learn complex representations of data by discovering heirarchical patterns & features in the data.
- These algorithms can automatically learn & improve from data without the need for manual feature engineering.
- Some popular deep learning architectures include CNNs, RNNs, & Deep Belief Networks (DBNs)

→ Applications
- Computer Vision – understand visual data
- NLP – text generation, lang. translation
- RL – Game playing, Robotics



→ Working :-

① I/P Layer

② Hidden Layers

③ O/P Layer

④ Backpropogation

→ Adv :-
- High accuracy in complex tasks.
- Automatically extracts imp features from data
- Effective for large-scale data problems.

→ Disadv :-
- Requires large amounts of labeled data
- High computational cost (works without revealing its process)
- Difficult to interpret ("black-box" nature)

④ Difference b/w Deeplearning & Machine learning:-

Aspect	ML	DL
Definition	Subset of AI focused on creating models to learn from data	Subset of ML using neural n/ws for advanced learning tasks
Data Dependency	works well with small to medium datasets.	Requires large datasets for better performance.
Feature Extraction	Features are manually selected by experts	Automatically extracts features from raw data.
Algorithms	Simple models like Decision Tree, SVM, KNN	Complex architectures like CNN, RNN & transformers
Training time	faster to train on small datasets	Slower due to high model complexity
Computational Power	Requires moderate computational resources (CPU)	Requires high computation power (GPUs, TPUs)
Data Type	Works best with structure data like tables	Best suited for unstructured data like video, audio, text
Performance	Less accurate for complex tasks	High accuracy for complex tasks.
Interpretability	Easier to interpret & debug.	Difficult to interpret, acts as a black box.
Apps	fraud detection, spam filtering, stock price prediction	Image recognition, speech processing, autonomous vehicles.

Historical Trends in Deep Learning :-

→ Deep Learning (DL) has evolved significantly over the years, marked by key milestones & achievements. Below is a simple & detailed timeline of its historical trends:

1940s – 1960s : Early foundations

1. McCulloh & Pitts Model (1943):
   - Proposed first artificial neuron model called McCulloh-Pitts neuron which laid foundation for ANNs.
2. Hebbian Learning Rule (1949):
   - Proposed by Donald Hebb, which suggested that "neurons that fire together wire together", forming the basis for learning NNs.
3. Perceptron Model (1958):-
   - Developed by Frank Rosenblatt, which is a single layer NN capable of solving simple problems.

1970s – 1980s : Decline & Revival:-

1. XOR Problem (1969):-
   - Marvin Minsky & Seymour Papert showed that perceptrons couldn't solve non-linear problems, leading to a decline in interest.

2) Backpropagation Alg. (1986):

· Introduced by Rumelhart, Hinton & Williams which allowed multi-layer n/ws to learn by adjusting weights through error gradients.

3) Hopfield N/ws (1982):

· Proposed by John Hopfield, which is used for associative memory models & optimization problems.

1990s: Emergence of CNNs:-

1) LeNet (1998):-

· Developed by Yann LeCun, which successfully recognized handwritten digits in the MNIST dataset.

· First practical application of CNN in character recognition.

2000s: Growth of DL:-

1) Restricted Boltzmann Machines (RBMs):-

· Popularized by Geoffrey Hinton which is used for pretraining NNs.

2) Support Vector Machines (SVMs):-

· Although not directly a deeplearning model, SVMs became popular for classification tasks.

2010s: DL Revolution:-

1) AlexNet (2012):-

- Developed by Krizhevsky, Sutskever, & Hinton which won the ImageNet competition & demonstrated the power of deep CNNs.

2) RNNs & LSTMs (2014):-

- Long-Short Term Memory (LSTM) n/ws became popular for sequence based tasks like speech-recognition.

3) GANs (2014):-

- Introduced by Ian Goodfellow which used for image generation & unsupervised learning.

4) Transformer Architecture (2017):-

- Proposed by Vaswani et al which revolutionized NLP.

2020s : Advancements & Applications:-

1) Large Lang. Models (LLMs):-

- Ex include GPT & BERT, widely used in conversational NLP tasks.

2) Explainable AI (XAI):-

- Focus on making DL models more interpretable.

3) Edge & Federated Learning:-

- Deployment of DL models on edge devices for privacy & efficiency.

④ Deep Feed-forward Networks (DFFNs):-

→ DFFNs also called Feeds Forward Neural Networks that are the simplest type of ANNs.

→ These n/ws are called "feed-forward" because info flows in one direction, from i/p layer to the o/p layer through one or more hidden layers.

→ Structure:-

3 layers are all connected to one another:

1) I/P layer :- Accepts i/p features.

2) Hidden Layers:- Process the i/p by applying weights, biases & activation function.

3) O/P layer: Produces the final result or prediction.

→ Here, info moves forward in a layer-by-layer fashion, without going back or looping, with no feedback loops.

→ Working:-

1) I/P :- I/P layer recieves raw data (pixel values of an image)

2) Weighted sum:-

Each neuron computes a weighted sum of its i/ps.

$$z = \Sigma (w_i \cdot x_i) + b$$

$w_i \rightarrow$ weights, $x_i \rightarrow$ i/ps, $b \rightarrow$ bias

3) Activation Function:

The weighted sum (z) is passed through an activation function (sigmoid, tanh, ReLU) to decide o/p of neuron.

4) O/P layer:

The final layer o/ps the result, such as a classification label or regression value.

→ Adva:-

1) Easy to design & implement
2) Capable of approximating any continuous function.
3) Suitable for various tasks like classification, regression, & pattern recognition.

→ Disadv:-

1) Can't handle sequential data
2) Requires large amount of data & computational power
3) Overfitting problems

→ App:

1) Image classification
2) Speech Recognition
3) NLP
4) Medical Diagnosis

(10) Gradient Based Learning:-

→ It is a technique used to optimize neural networks by adjusting the weights & biases during the training process.

→ The key objective is to min the error b/w predicted o/p & actual target o/p by finding the best values for these weights.

→ The gradient is a vector of partial derivatives that shows the direction & rate of the steepest ascent of a function.

→ In neural networks, it represents how much the loss function changes w.r.t weights & biases.

→ The loss function measures the error b/w predicted & actual o/ps.

→ Common loss functions are Mean Squared Error for regression tasks & Cross-Entropy loss for classification tasks.

→ Steps involved in Gradient-Based Learning:

i) Forward Propogation:

- I/p data is passed through the n/w layer by layer.
- The n/w generates an o/p.
- The error is computed using the loss function.

2) Back Propogation:

- The error is propogated backward to calculate gradients of the loss function w.r.t to each weight & bias.
- This step uses the chain rule of calculus.

3) Weight Update:

- Weights & biases are updated using the gradient values to reduce the error.

$$W_{new} = W_{old} - \alpha \cdot \frac{\partial L}{\partial w}$$

where, α - learning rate

$\frac{\partial L}{\partial w}$ - gradient of loss w.r.t weight.

→ Learning rate controls how much weights are updated in each iteration. A small 'α' makes the process slow while a large value may lead to overshooting.

→ When the weights reach optimal or near optimal values, the learning process is said to converge.

→ Adv:-

- Efficient for large-scale data
- Suitable for complex models

→ Disadv:

- May get stuck in local minima
- Sensitive to learning rate.

④ Hidden Units :-

→ Hidden units are the individual neurons in the hidden layers of a neural network.

→ These units recieve input from the previous layer, process it by applying weights, biases and an activation function, and pass the o/p to the next layer.

→ Roles :-

1) Feature Extraction :-

Hidden units learn imp features or patterns from i/p data.

2) Data Transformation :-

They transform the data by applying nonlinear functions, making complex tasks solvable.

3) Information Flow :-

Pass intermediate results b/w i/p & o/p layers to build complex models.

→ Components :-

1) Weights (w)

2) Bias (b)

3) Activation Function (f)

Formula :

$h_i = f(w \cdot x + b)$

where, h_i → o/p of hidden unit

w → weights

x → i/p from previous layer

f → activation function

→ Importance:-

1) Hidden units allow the n/w to solve non-linear problems by applying activation function. [Non-Linearity]

2) Multiple hidden units across several layers help the n/w understand complex heirarchical features. [Heirarchical Learning]

3) Increasing the no. of hidden units can improve learning but also increase the risk of overfitting. [Model Complexity]

→ Choosing no. of Hidden units:-

1) Too few hidden units → Underfitting
2) Too many hidden units → Overfitting
3) The no. of hidden units depends on:
   - complexity of the problem
   - Size of i/p data
   - Availability of computational resources.

④ Architecture Design:-

→ It refers to deciding the structure of n/w, including no. of layers, types of layers, no. of neurons (units) per layer & activation functions used.

→ Key Components of Architecture Design:

1) I/P Layer
2) Hidden Layer

3) O/P Layer

4) Activation functions
(to introduce non linearity & to decide when to fire a neuron)

5) Weights & Biases

6) Loss function (MSE for regression, Cross Entropy for classification)

7) Optimization Algorithm
- Adjusts weights & biases to min loss function.
- Gradient Descent, Adam & RMSprop.

→ Steps for Designing a Neural N/W Architecture:

1) Define the problem: Identify whether it's a classification, regression or clustering task.

2) Preprocess the Data: Normalise data & handle missing values.

3) Select the N/w Type:
CNN for image processing tasks
RNN for time-series or sequence data.

4) Choose no. of Layers:
Simple problems → Fewer layers
complex problems → More layers

5) Select Neurons per Layer:
Start with a moderate number and adjust based on performance.

6) Activation Functions:-

- Use ReLU for hidden layers &
  Sigmoid/Tanh for o/p layers in
  classification

7) Loss function & Optimizer:
   Match the loss function to the task type.

8) Training & Evaluation:

- Train the n/w using forward propogation
  & backpropogation
- Evaluate using metrics like accuracy
  or RMSE

→ Types of NN Architectures:-

1) FNNs
2) CNNs
3) RNNs
4) GANs