* Neural Network:
It is a computer system that tries to work like the human brain to learn from data and make decisions or predictions

* Components of NN

- Input layer:- Takes in data (like no. or img) to be processed
- Hidden layer:- Do the main processing using many small units called neurons
- Neurons (Nodes):- Basic unit that recieves input, process it and pass it on
- weights and Biases:- values that control how imp. each i/p is
- Activation function:- Decides wheather a neuron should be activated or not
- output layer:- Gives the final result or predict

* Advantages:

- learns from example
- solves complex examples
- No need to program rules
- work in many areas
- Remembers patterns
- improve with time

* Disadv:

- Needs lots of data
- Takes time & power
- Hard to explain
- Can make mistakes
- Needs good quality of data
- Not good for small tasks

Types of NN:-

(1) Feedforward NN

- Data moves in one direction - input to output
- use for simple tasks like classifying images or numbers
- ex:- handwriting recognition

(2) Convolution NN:-

- Detects patterns in images using filters
- work best with images and videos
- Ex:- Face detection, object recognition.

(3) Recurrent NN:-

- Remembers previous steps; has memory
- Good for data that comes in sequence
- Ex:- language translat°, speech recognition

(4) Long Short-Term Memory (LSTM):-

- Special type of RNN that remember things for long time
- uses for long sentences, stories or time based data
- Ex:- chatbots, text generat°

(5) Generative Adversarial Network (GAN):-

- Creates new data by learning from real data
- used to generate img, music or fake faces
- Ex:- Creating realistic photos of people who doesn't exists

* ANN

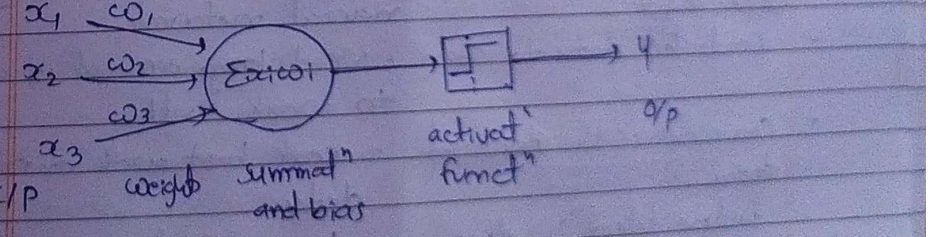
- Made of computer program and math models
- work with no. and data i/p
- learns by adjusting weights in connection
- used for task like img recognition & predict°
- process info in layers
- Needs lots of data and training

BNN

- made of real nerve cells (neurons) in brain
- work with electrical & chemical signals
- learns by making connect° in brains stronger or weaker
- control body, funct°, thoughts and feelings
- process info through neuron network
- learn naturally from experi° and senses

* Perceptron:-

It is simple type of neural network model, used for binary classificat°. It mimics a single neuron that take i/p, process them and produces an output



- Input layer:- Takes i/p values
- weights:- each i/p is multiplied by weight
- Summat°:- Add up all weighted i/p & bias
- Bias:- a constant value added to control the O/p
- Activat° fn°:- apply fn° to the sum to produce final O/p.

cooking:
(1) Take i/p
(2) multiply by weights: each i/p has weight showing its imp
(3) Add bias :- small no. to adjust o/p
(4) Applies activat° fx° :- decides if neuron fires or not
(5) give O/P:

* Mapping ANN elements with BNN

ANN element	BNN element
- Input layer	- Sense organs
- Neuron (Node)	- Brain Neuron
- weight	- synapse strength
- Bias	- threshold
- Activat° fx°	- Neuron firing mechanism
- hidden layer	- Brain processing Area
- O/p layer	- Body response

*

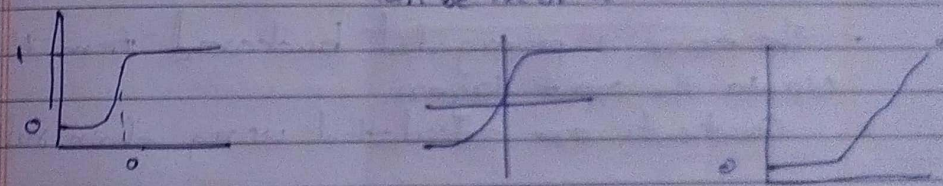
ML	DL
- learns from data using basic algo	- learns from data using neural network
- Needs less data to train	- Need large amount of data to work well
- Can work with structured data	- work best with unstructured data
- feature select° is done manually	- learns features automatically
- Faster to train on small dataset	- Take more time & power to train
- Ex:- Decision Tree	- CNN, RNN, LSTM

* why activat° fx° used:

- It decides whether a neuron should be active or not
- It helps network learn & understand complex pattern
- Add non-linearity, so network can learn more than just straight lines
- without it neural network could behave like simple math funct° and couldn't able to solve real problem like image recognition

	Sigmoid	Tanh	Relu
eqn	$f(x) = \frac{1}{1+e^{-x}}$	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f(x) = \max(0, x)$
o/p range	(0, 1)	(-1, 1)	(0, ∞)
use cases:-	Binary Classificat°	hidden layer	hidden layer in DL
pro adv:-	better for probability	perform when values can be +ve or -ve	fast.

Plot:-



* Backpropagation Algo

- It is a training algo used in artificial neural network to minimize the error between predicted and actual output
- It adjust the weights and biases in network using gradient descent based on calculated

Steps:-

1. forward propagation

- input data is passed through the network
- each neuron computes a weighted sum
  $z = w \cdot x + b$

(2) Compute loss (error)

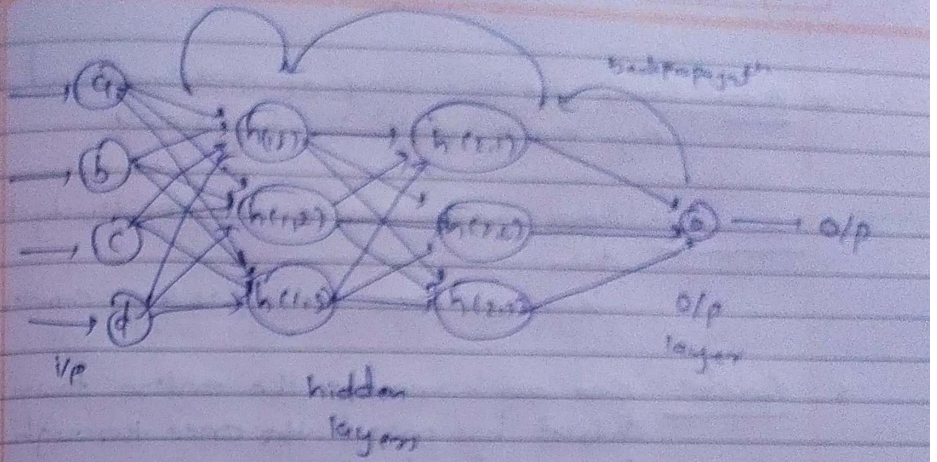
- The loss function measures how far the output is from the actual result
- ex:- using mean square error
  $E = \frac{1}{2}(y_{actual} - y_{predicted})^2$

(3) Backward propagation

- The error is propagated backward from the o/p to the input layer.
- Gradients are calculated using chain rule.

(4) weight & Bias update:

- Adjust weight and biases using gradient descent
- update weights to reduce loss using gradient descent.



* Underfitting:-

High bias low variance

- when model can't learn the pattern from the data
- It perform poorly on both training & testing data
- The model is too basic
- model is not learning enough

Ex: using linear model to predict non linear problem

Can fix by:- Adding more layers or neurons
Training model for more time
using complex model.

* Overfitting:

- Happens when model learns the training data too well, even the noise

low bias high variance

- It does very well on training data but poorly on testing data
- The model becomes too complex and memorizes instead of learning
- fix by:- using more data for training
  - Adding dropout layers

Ex:- using dnn to learn small dataset with noise

* Bias:- Error due to wrong or oversimplified assumption
high bias = model is too simple

* Variance:- Error due to too much sensitivity to training data
high variance = model is too complex

* Epoch:- One epoch means the entire training dataset has passed ~~the~~ once through the neural network.

* Learning rate:- How fast the model learns from it's mistake.
- It controls how much model's weight are updated during training

* Hyperparameter Tuning in NN
- Hyperparameters are the parameters which are set before the training process starts.
- Tuning means changing these hyperparameters to find the best value for better result.
- Try different values to find best combination

- Why important? :- It help the model to learn better & faster
- Good hyperparameter value give high accuracy
- Bad value can cause overfitting or underfitting

Ex:- learning rate
Epoch
Batch size:- no of sample process ate one time
no. of layers and neurons.
Activation fn

* Accuracy:- How many total predictions were correct
Correct prediction / Total prediction

* Precision:- out of all predicted fatigue cases, how many were actually fatigue.

* Recall:- out of all actual fatigue cases, how many did the model detect.

* Loss value:- number that shows how wrong the model is during training

* Relationship b/w no of layers & Complexity
- The more layers means more complex task it can handle
- Each layer helps the network learn deeper features from the data
- Simple task needs fewer layers., Complex need more

Ex:- Simple task - Digit recognition
Complex task - Face recognition

* Regularization:-
means adding rules to a model to keep it simple and stop it from memorizing the training data too much
It help model work well on new data.

Types:-

(1) L1 Regularization:- make some model parts zero to keep only important features
- help to make model simpler and easier to understand

(2) L2 Regularization:- keep model parts small to avoid too much focus on details
- prevent the model from fitting noise in the data

(3) Dropout:- Randomly turns off some neurons while training to avoid over-dependence
- help the model to not rely on specific neurons

(4) Early stopping:- stops training early to prevent the model from learning noise
- saves time by avoiding unnecessary training steps.

Padding:-
means adding extra pixels around the edges of an image before applying filter in neural network.

Types:- (1) Valid padding:-
- no extra pixels are added to i/p
- O/p size become smaller than i/p
- also called "valid" bcz only valid parts are used for O/p

(2) Same padding:-
- add zeros around the i/p to keep size of i/p = O/p same after convolution
- Helps when we want to keep image size
- often used in CNN

(3) Zero padding:-
- Add zeros to the border of the image.
- help the filter fit even at the edges
- make sure network doesn't loose imp edge information.

* Pooling:-
It is used to make image smaller by picking out the most useful parts, so model can learn faster and better.

(1) Max pooling:-
- Take largest value from a group of pixels.
- Reduce image size while keeping imp info

(2) Average pooling:-
- Take average value from group of pixels
- used when we want to reduce size and noise

(3) Global Average pooling:-
- Take average of the entire feature map
- Turns each feature map into single number.
- use just before final O/p layer in 'classification' model.

hidden layer in

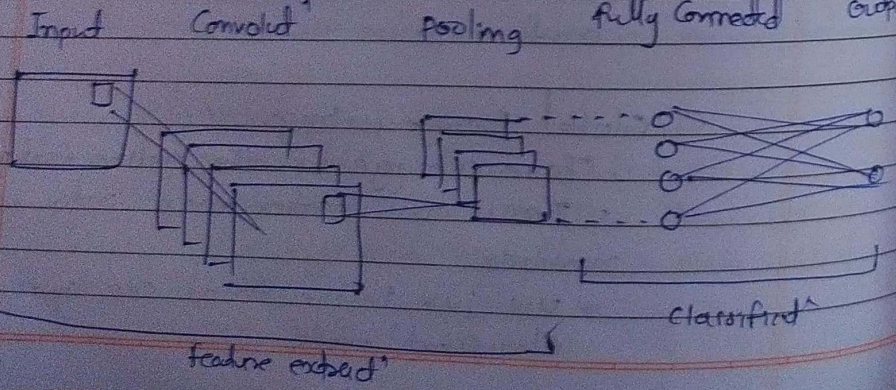
* Role of feature extract:

(1) learn features step-by-step
- Each hidden layer learns to detect different features from i/p
- Early layers find simple features
- Deeper layers combine these to learn complex features

(2) Transform data gradually
- hidden layers transform raw data into meaningful info
- help network understand pattern better for classification

(3) Build hierarchy of features
- create hierarchy from basic to advance feature.
- make accurate predictions

* CNN :-
It is a special type of NN that helps Computer understand images by looking for patterns like edges and shapes.



input layer:
- This is where the raw image or data enter the network
- Usually, images are represented as 3D arrays.

Convolution layer:
- Uses small filters to scan the image and detect features / find imp part in picture
- like edges, shape or color
- Creates feature map showing where these features appear.

Pooling layer:
- make picture smaller but keeps important info
- Reduce data size & Computat' by keeping imp info

fully Connected layer:
- Connects all info to decide what the picture is
- learns pattern to recognize image

Output layer:
- Give final predict'
- Show result like object's name.

* Steps to Train CNN with Tensorflow

(1) Prepare data:
- load and preprocess images
- split data into training & testing sets

(2) Build CNN model:
- Use Tensorflow / Keras to Create layers: Conv, pooling, flatten, Dense.
- Define architecture of your CNN

(3) Compile the model:
- Choose optimizer (like adam)
- Choose loss fund"
- Choose metrics to track (like accuracy)

(4) Train The model
- use .fit() method with training data
- Set batch size and no. of epoch.

(5) Evaluate the model
- Test on validat" or test data using .evaluate() method
- check accuracy and loss to see how well method model learned

(6) make prediction
- Use .predict() on new data to get o/p from trained model.

* Steps to train RNN using Tensorflow

(1) Prepare data:
- load and preprocess your sequence data (like text or time series)
- Convert data into sequence and split into training & testing sets

(2) Build RNN model:
- Use Tensorflow / Keras to Create layers like: simple RNN, LSTM OR GRU
- Add Dense layer after RNN layers for o/p

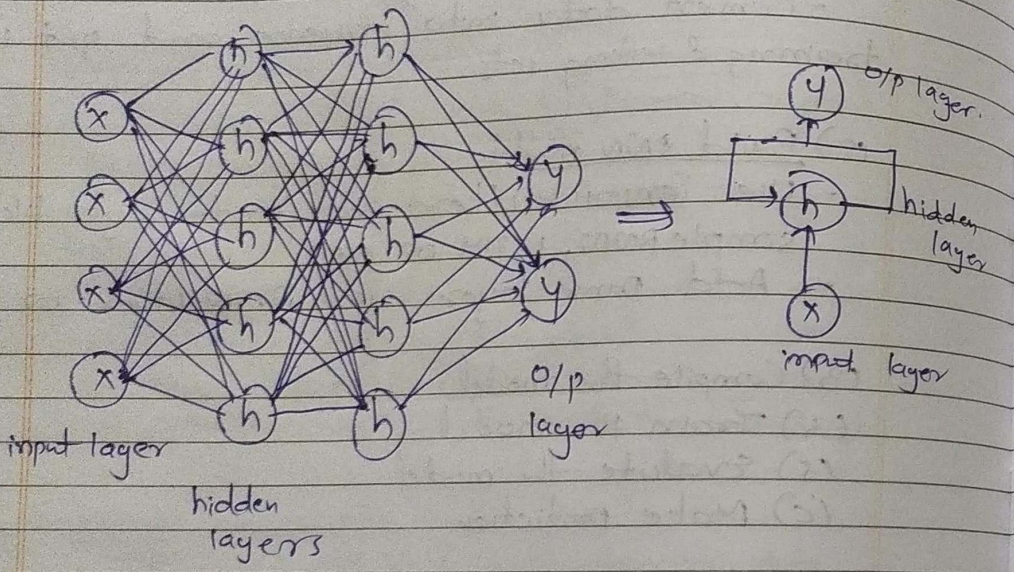
(3) Compile the model
(4) Train the model
(5) Evaluate the model
(6) Make prediction

*

RNN	CNN
- work on sequence data	- work on image or grid data
- Remembers past i/p using loops	- Detects patterns using filters
- Read i/p step by step	- process many parts at once
- Good for text, speech,	- Good for images, videos
- has memory	- no memory
- slower due to sequencial steps	- faster due to parallel processing
- Process data one step at a time	- Process data in parallel

* Architecture RNN

- It is type of NN that good at working with sequence data like text, speech etc.



(1) Input layer
- Take a sequence of data (like words)
- Each i/p is passed one by one

(2) hidden state:
- Remembers past i/p using memory
- passed from one time step to the next

(3) output layer
- Give o/p at each step
- depends on current i/p and hidden state

weights:-
- same weights are used at every step
- help in learning pattern over time.

* Applicat' of ANN:- (one by one) (step by step)

- Text generat'
- language translat' (understand order of words)
- Speech Recognition (it is time based)
- Stock price predict' (based on past values)
- music generat'
- chatbots and conversational AI
- Weather forecasting
- handwriting Recognition

* Types of RNN:-

(1) Vanilla RNN:
- Take input and previous o/p to predict the next o/p
- Basic form of RNN with simple looping from previous time step

(2) LSTM
- Special type of RNN that solve vanishing gradient issue (i/p, forget, o/p)
- use gates to control info. flow
- best for long term dependencies (e.g speech)

(3) GRU (Gated Recurrent Unit)
- Similar to LSTM but simpler and faster
- uses update and reset gate instead of three gates
- performs well like LSTM with fewer parameters

(4) Bidirectional RNN
- process i/p in both forward & backward direction
- useful when context from both part and future is important

(5) Deep RNN:
- multiple RNN layers stacked together
- learns more complex patterns than a single RNN

* Tensorflow:
- It is a dl library made by Google.
- helps build & train ml model using graphs and tensors. (multi dimensional array)

Limitations:-
- Hard to learn :- not beginner friendly : hard for them
- Too much code :- require lot code for small task
- Debugging is Tough :- Error can be confusing & hard to fix
- Not reduced for python users :- writing code doesn't feel smooth/simple in python
- Uses more resources :- Need more memory & power Compare to
- version problem :- new update sometimes break old code causing problem
Training can be slow without GPU

batch is small group of data samples processed together in one step during training

* Eager execut"	Graph execut"
- Runs operations immediately	- Build a computational graph first
- Easy to understand & debug	- harder to debug but faster to run.
- Good for small models or testing	- Best for training big models
- slower	- faster.
- shows o/p instantly	- o/p comes after running the graph
- Easy for beginners & learning	- Better for exp developers.
- memory usage is higher	- optimized memory usage.

* Batch normalization:
- It is a technique used in dl to make training faster and more stable
- It normalizes the i/p of each layer so the value have mean 0, variance 1
- Then add scaling & shifting to keep learning flexible
- It helps network learn faster and better

* Types of Tensors:
(1) Scalar (0-D)
- Just single no.
- Ex:- 5, -3.14

(2) Vector (1-D)
- a list of no. in single line
- Ex:- [2, 4, 6]

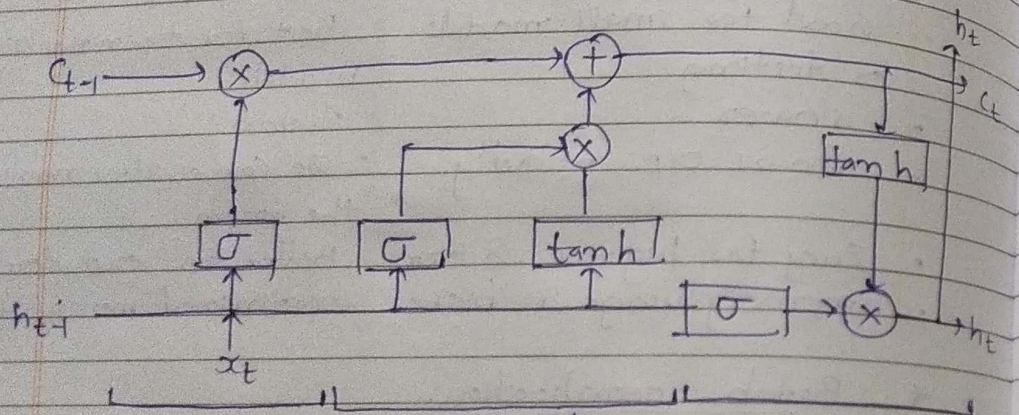
(3) Matrix (2-D)
- a table of no. with rows & columns
- $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

(4) 3-D Tensor
- A stack matrix
- multiple grayscale image stored togeth

higher dim" = (4D and above)
- used for vdo

* LSTM (Long short term memory)

It is special type of RNN that can remember info. for long time.



(1) forget gate:
- Decides whether info to remove from the previous cell state C_{t-1}
- Uses sigmoid fx^n to O/P 0 or 1

(2) Input gate:
- Decide what new info. to store in cell state
- It has two parts: Sigmoid gate
tanh gate

(3) Output Gate:
- Decide what part of the cell state to output as hidden state h_t

pros:- Remembers long sequence
- solves vanishing gradient problem
- handle time series & text well
- learns what to remember or forget

Cons:- Complex architecture
- Slower training
- Hard to tune
- not always the best

* Advantages of Keras:

- Easy to use
- modular & flexible
- Large Community
- Runs on CPU & GPU
- Integrate with Tensorflow
- Supports multiple backends.

* role of DL in CS

(1) Intrusion detect^n system
- help to detect unauthorized access or behaviour
(2) Malware detect^n
- help analyze file, code to detect malicious softwa
(3) phishing detect^n:
- Scan email, URL for signs of fishing attempts