DEEP LEARNING

## 

## **What is DEEP LEARNING ?**

Deep Learning is a subset of **machine learning (ML)** that focuses on using **artificial neural networks (ANNs)** to model and solve complex problems. It mimics the way the human brain processes information and recognizes patterns.

## **Use of DEEP LEARNING :**

### **1. Computer Vision 🖼️**

* **Facial Recognition** – Used in security systems and social media (e.g., Face ID, Facebook tagging).
* **Object Detection** – Self-driving cars detect pedestrians, traffic signs, and obstacles.
* **Medical Imaging** – Identifies diseases like cancer from X-rays, MRIs, and CT scans.

### **2. Natural Language Processing (NLP) 🗣️**

* **Chatbots & Virtual Assistants** – Siri, Alexa, and ChatGPT use DL for better human-like responses.
* **Language Translation** – Google Translate uses deep learning for real-time translations.
* **Sentiment Analysis** – Detects emotions in text, useful in customer service and social media monitoring.

### **3. Autonomous Systems 🚗**

* **Self-Driving Cars** – Tesla, Waymo use DL to make real-time driving decisions.
* **Drones & Robotics** – Used in agriculture, delivery services, and industrial automation.

### **4. Healthcare & Drug Discovery 🏥**

* **Disease Prediction** – AI models predict diseases like Alzheimer’s, cancer, and heart conditions.
* **Personalized Medicine** – AI suggests the best treatment plans based on patient data.
* **Drug Discovery** – Speeds up the process of finding new medicines.

### **5. Finance & Fraud Detection 💰**

* **Credit Scoring & Loan Approval** – AI predicts creditworthiness.
* **Stock Market Prediction** – Analyzes trends for better investment decisions.
* **Fraud Detection** – Identifies fraudulent transactions in banking.

### **6. Gaming & Entertainment 🎮**

* **Game AI** – DeepMind’s AlphaGo defeated world champions in Go.
* **Content Recommendation** – Netflix, YouTube, and Spotify suggest personalized content.
* **Deepfake Technology** – Creates realistic AI-generated images and videos

## **Key Features of DEEP LEARNING:**

1. **Uses Neural Networks** – Especially deep neural networks (DNNs) with multiple layers.
2. **Learns Hierarchical Features** – Extracts simple features (like edges) in early layers and complex patterns (like faces) in deeper layers.
3. **Requires Large Data** – Performs well with large datasets, improving accuracy as more data is available.
4. **Computationally Intensive** – Needs powerful GPUs and TPUs for training

## **ARCHITECTURE :**

We use different **Deep Learning architectures** because each one is optimized for specific types of problems. Choosing the right architecture depends on the type of data (images, text, time-series, etc.) and the task (classification, prediction, detection, etc.).

1. Feedforward Neural Networks (FNNs)
2. Convolutional Neural Networks (CNNs)
3. Recurrent Neural Networks (RNNs)
4. Long Short-Term Memory (LSTM) & Gated Recurrent Unit (GRU)
5. Transformers (e.g., BERT, GPT)
6. Generative Adversarial Networks (GANs)
7. Autoencoders
8. Deep Reinforcement Learning (DRL)

## **1. Feedforward Neural Networks (FNN)**

* The most basic type of neural network.
* Consists of **input, hidden, and output layers** with neurons connected in one direction (no loops).
* Used in **simple classification and regression tasks**.

🔸 **Example Applications:** ✅ Predicting house prices  
 ✅ Spam email classification

## **2. Convolutional Neural Networks (CNN)**

* Designed for **image processing & computer vision** tasks.
* Uses **convolutional layers** to detect patterns like edges, textures, and objects.
* Has **pooling layers** to reduce spatial size and **fully connected layers** for classification.

🔸 **Example Applications:** ✅ Face recognition (e.g., Face ID)  
 ✅ Object detection (self-driving cars)  
 ✅ Medical image analysis

## **3. Recurrent Neural Networks (RNN)**

* Used for **sequential data** (time series, speech, text).
* Has loops that allow **memory of previous inputs**.
* Problem: **Vanishing gradient** (difficulty in learning long-term dependencies).

🔸 **Example Applications:** ✅ Speech recognition  
 ✅ Language modeling (e.g., autocomplete)

## **4. Long Short-Term Memory (LSTM) & Gated Recurrent Units (GRU)**

* Improvements over RNNs to handle **long-term dependencies**.
* **Why?** Solves the **vanishing gradient problem** in RNNs.
* Uses **gates (forget, input, output)** to decide which information to keep/forget.
* GRU is a simpler, more efficient alternative to LSTM.

🔸 **Example Applications:** ✅ Machine translation (Google Translate  
 ✅ Chatbots (e.g., Siri, Alexa)  
 ✅ Stock market prediction

## **5. Transformers (e.g., BERT, GPT)**

* Replaced RNNs for NLP tasks.
* Uses **self-attention mechanisms** to process words in parallel (much faster than RNNs).
* **GPT (ChatGPT)** and **BERT** are based on this architecture.

🔸 **Example Applications:** ✅ Chatbots & virtual assistants  
 ✅ Text summarization  
 ✅ Language translation

## **🔹 6. Generative Adversarial Networks (GANs)**

* Consists of two networks:  
   1️⃣ **Generator** – Creates fake data  
   2️⃣ **Discriminator** – Tries to detect fake data
* Used for **generating new images, videos, and music**.

🔸 **Example Applications:** ✅ Deepfake technology  
 ✅ AI-generated art  
 ✅ Image enhancement (e.g., old photo restoration)

## **🔹 7. Autoencoders**

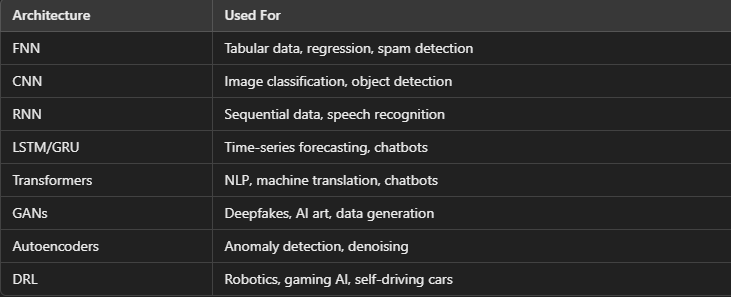
* Used for **dimensionality reduction** and **data compression**.
* Learns an **efficient representation** of data.
* Variational Autoencoders (VAEs) are used in generative tasks.

🔸 **Example Applications:** ✅ Anomaly detection (fraud detection, medical diagnosis)  
 ✅ Image compression

## **🔹 8. Deep Reinforcement Learning (DRL)**

* Combines **Deep Learning + Reinforcement Learning**.
* Uses **CNNs or RNNs** for high-dimensional inputs.

🔸 **Example Applications:** ✅ Self-driving cars (Tesla, Waymo)  
 ✅ Robotics (Boston Dynamics)  
 ✅ AI gaming (AlphaGo, OpenAI Five)



**How to Choose the Right Architecture?**

* If working with images ➝ Use CNN
* If working with sequential data (text, speech, time-series) ➝ Use RNN, LSTM, or Transformers
* If working with real-time decision-making (games, robotics) ➝ Use DRL
* If working with data compression or anomaly detection ➝ Use Autoencoders
* If working with data generation (art, deepfakes) ➝ Use GANs

**Difference Between AI, ML, DL:**

* This means that DL models can learn to identify complex patterns and relationships in data without the need for human intervention, making them more powerful and scalable than traditional ML models.
* AI is the field of study that aims to create machines
* ML is a subset of AI that focuses on developing algorithms that can learn from data and make predictions or decisions based on that learning.[jst like how human learns]
* DL is a subset of ML that focuses on developing artificial neural networks that can learn from large amounts of data.[like our brain taking automatic decision without human involvement]

**HOW DL WORKS:**

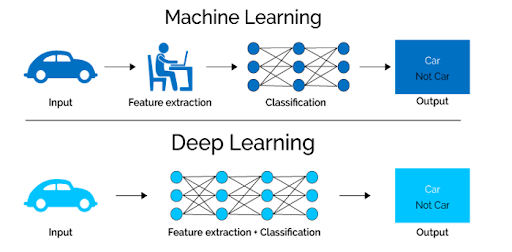
DL uses artificial neural networks, which are modeled after the human brain. These networks are made up of layers of interconnected nodes, called neurons. Each neuron processes information and passes it on to the next layer.

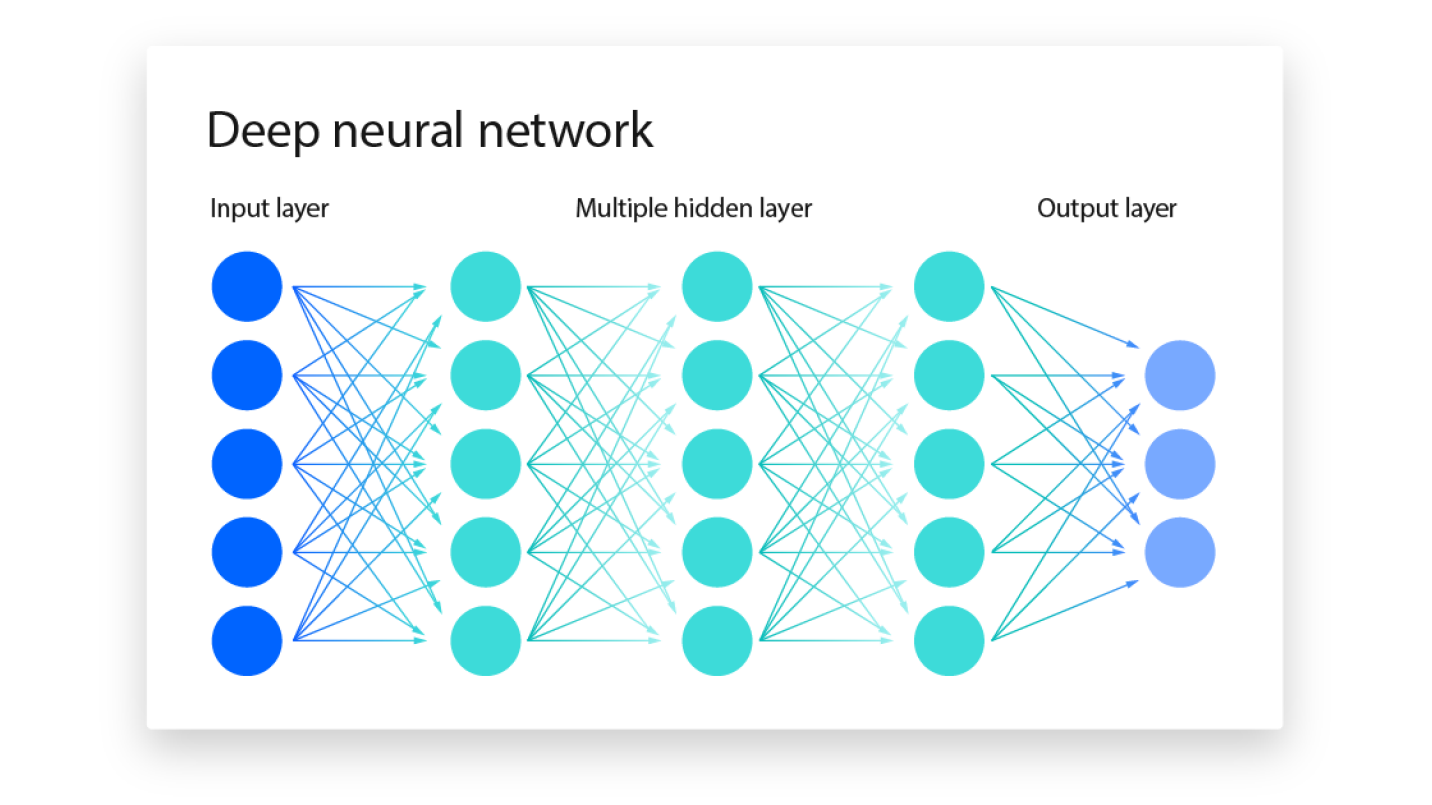
**1. Data Input:** You feed the network a large dataset, like images of cats and dogs.

**2. Feature Extraction:** The network learns to identify patterns and features in the data, like the shape of ears, the texture of fur, etc.

**3. Weight Adjustment:**  The network adjusts the connections between neurons (weights) to improve its ability to recognize these features.

**4. Prediction:**  When you give the network a new image, it uses the learned weights to predict whether it's a cat or a dog.

This process of learning from data and adjusting weights is called **training.** The more data you feed the network, the better it gets at making predictions**.**

****

* Here every circle represensor neuron and every channel contains weights. Every neuron in hidden layer has a specific bias member which is constant. This neurons are the core processing units of the network.
* Each neuron takes the pixel as a input feed and the neurones of first layer connected to neurones are second layer through channels which contains weights
* the input of second urine will depend on the first neuron with the formula

**Xi \* W + Bias**

* Here x i is the input that coming towards the second layer neuron and w is the weight that carried by the Xi and bias is constant that is given to every neuron.
* **Weights:**it says how much neuron should be activated or deactivated.
* **Bias**:constant value added to the weighted sum of inputs before applying an activation function.
* It shifts the activation function, allowing the model to learn complex patterns even when **all input values are zero.**
* **Activation function** It is apply to every neuron at the output of every neuron.

**Uses of Activation Function:**

* Adds non-linearity → Without it, a neural network would behave like a simple linear function (like a single-layer perceptron)
* Helps learn complex relationships → Enables deep networks to model real-world problems like image recognition and NLP.
* Controls neuron activation → Decides which neurons should fire

In output layer the neuron having the highest value is decide as the output.

**FORWARD PROPAGATION:**

**Input Layer:** Receives raw input data (e.g., pixel values of an image or text embeddings). **Hidden Layers:** Applies weights and biases, followed by an activation function to introduce non-linearity**.  
Output Layer**: Produces the final prediction (e.g., class probabilities in classification tasks).



## **What Happens After Forward Propagation?**

**Loss Calculation:** Compare predicted output with actual values (e.g., using cross-entropy for classification).

**Backward Propagation:** Compute gradients and update weights using gradient descent to minimize the loss.

Forward propagation **computes predictions** using weights, biases, and activation functions.

**No learning happens** in forward propagation (that happens in backpropagation).

# BACKWARD PROPAGATION

**Backward Propagation (Backprop)** is the key algorithm used to train deep learning models. It **updates the weights and biases** of a neural network by minimizing the error (loss function) using **gradient descent**.

## **Why is Backpropagation Needed?**

In **forward propagation**, we compute predictions. However, if the prediction is incorrect, we need a way to **adjust the weights** so the model improves over time. **Backpropagation helps achieve this by calculating gradients and updating parameters accordingly.**

## **Steps of Backpropagation**

### **1️⃣ Forward Propagation**

* Compute the weighted sum and apply the activation function.
* Calculate the **loss function** (difference between predicted and actual values).

### **2️⃣ Compute Gradients (Partial Derivatives)**

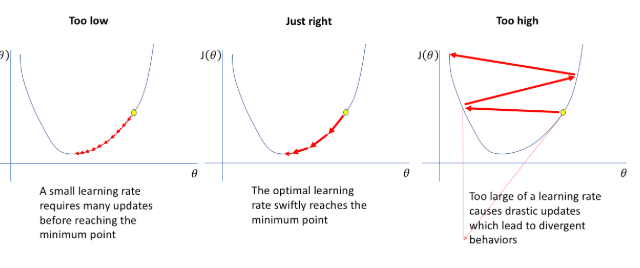
* Use the **chain rule** of calculus to calculate the derivative of the loss function with respect to weights and biases.

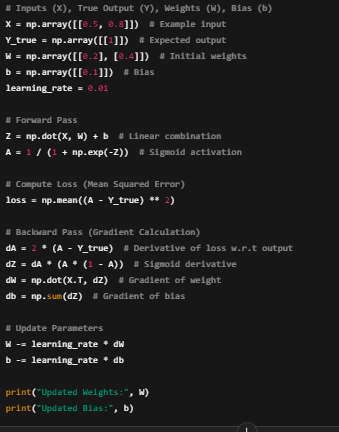
### **3️⃣ Update Weights (Gradient Descent/Optimization)**

* Adjust the weights and biases **opposite to the gradient** direction to minimize loss.
* Learning rate (α\alphaα) controls how much we adjust the weights in each step.
* **Learning rates** always be minimal so that the point goes to global minimum slowly and in a good speed without any jumpings that produces noise.

**Wnew = Wold - (∂L​ / ∂W​old )**

**Where: α is learning rate = 0.01 , ∂L is loss function**



PYTHON BACKWARD:  


## **Optimizers in Backpropagation**

* **Optimizers** are algorithms that adjust the weights and biases of a neural network **to minimize the loss function** and improve model performance. They guide the learning process by updating parameters efficiently during backpropagation.

**TYPES OF OPTIMIZERS:**

**Gradient Descent-Based Optimizers:**

* Updates weights after processing the entire dataset.
* **Disadvantages** are if there are 1 million record be required huge Ram and GPU to run this Increases time complexity
* **W=W−α∂W∂L​**

**Stochastic Gradient Descent (SGD):**

* Updates weights after each individual training sample
* Faster and works well for large datasets.
* Disadvantages are Noisy and last table

**Adam (Adaptive Moment Estimation) [MOST USED]:**

* Combines Momentum and RMSprop for fast and stable learning.
* Works well in most deep learning models.
* **Used in:** CNNs, NLP, GANs, Transformers

**CHAIN RULE : how much each weight affects the loss**

Enables weight updates by finding how changes in each parameter affect the loss**.**

The chain rule itself does not decide how much to increase or decrease the weights. Instead, it helps calculate the gradient (i.e., how much each weight affects the loss). The actual weight updates depend on the optimizer, which uses the computed gradients to adjust the weights**.**

## **How Does the Chain Rule Help?**

The chain rule computes the partial derivative of the loss function with respect to each weight:

**∂W /∂L ​=(∂L / ∂Outputneuron )×(∂Output∂A​×∂W)**

This tells us:  
 **Direction:** Whether the weight should increase or decrease.  
 **Magnitude:** How much the weight contributes to the error.

## **Summary**

✅ **The chain rule calculates gradients**, showing how much each weight affects the loss.  
 ✅ **Optimizers (like SGD, Adam) decide the actual weight update** based on these gradients.  
 ✅ The **learning rate** controls how much the weights change per step.

**Activation Function**

An activation function in a neural network introduces **non-linearity** to the model, allowing it to **learn complex patterns**. Without activation functions, the network would behave like a simple linear regression model, unable to capture intricate relationships in data.

**Types of non linear Function:**

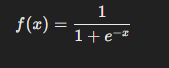
1. **Sigmoid**
2. **Tanh**
3. **ReLU**
4. **Leaky ReLU**
5. **Softmax**

**1 SIGMOID ACTIVATION FUNCTION:**

**Output range:** (0,1) **Good for:** Binary classification

**Pros:** Smooth and differentiable  
 Converts values into probabilities

**Cons:** Vanishing gradient problem – Small gradients slow down learning in deep networks.  
 Not centered around zero (causes inefficient updates).

****

**2 Tanh (Hyperbolic Tangent)**

**Output range**: (-1,1)  
**Good for:** Hidden layers in neural networks

**Pros:**  
 Zero-centered (better than Sigmoid)  
 Works well for data with negative values

**Cons:**  
 Vanishing gradient problem (like Sigmoid).

****

**3 ReLU (Rectified Linear Unit) [MOST COMMON]**

**Output range:** (0, ∞) **Good for:** Deep networks, CNNs

**Pros:** Efficient and simple  
 No vanishing gradient for positive values  
 Speeds up training

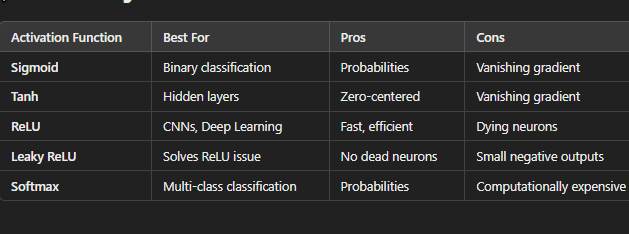
**Cons:**Dying ReLU problem – Neurons with negative inputs become inactive (output = 0).

**4 Leaky ReLU:**

Fixes dying ReLU problem by allowing small negative values

**5 Softmax (for Multi-Class Classification)**

**Good for:** Multi-class classificationConverts outputs into probabilities that sum to 1



**Choosing the Right Activation Function**

1. **Binary classification problem:**

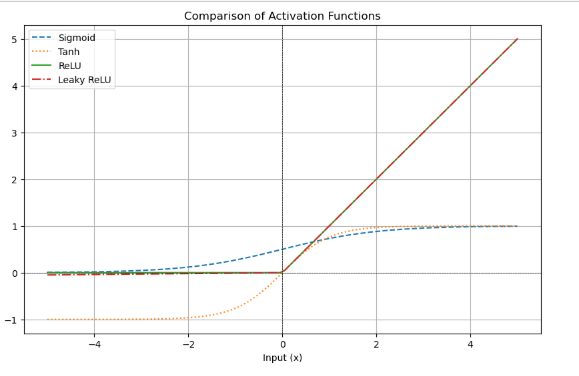
* Use **ReLu** activation function in hidden layer
* **Sigmoid** activation function in output layer

**2 MultiClass classification problem:(Having multipleoutputparameter):**

* We use **ReLu** activation function hidden layer
* In output layer we use **soft Max** activation function

**3 Regreesion problems:**

* Linear activation function output layer



**LOSS FUNCTION:**

A loss function **measures how well a neural network is performing** by comparing its predictions with the actual target values. The goal of training a model is to minimize the loss by updating weights using backpropagation and an optimizer

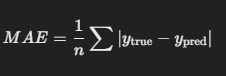
* It helps to measure model accuracy

**Types of Loss Functions:**

**1 Mean Squared Error (MSE)**

* Adv: Smooth and differentiable, good for gradient descent.  
  Con:Sensitive to outliers (since errors are squared).
* 

**2 Mean Absolute Error (MAE)**

* Less sensitive to outliers than MSE.
* Gives equal weight to all errors.
* Con:Not differentiable at zero
* 

**3 Huber Loss (Mix of MSE and MAE)**

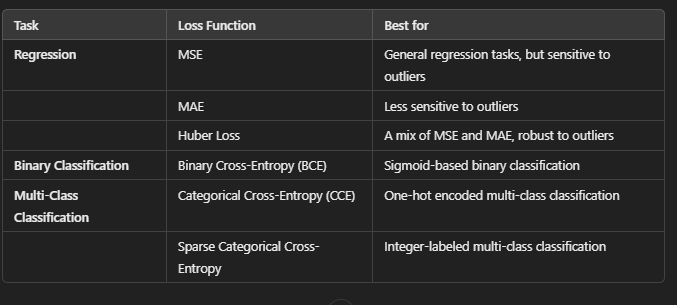
* Combines MSE (for small errors) and MAE (for large errors).
* Less sensitive to outliers than MSE

**4 Binary Cross-Entropy (For Binary Classification).**

* Used when predicting probabilities between 0 and 1.
* Works well with the Sigmoid activation function.

**5 Categorical Cross-Entropy (For Multi-Class Classification)**

* Used when output classes are **one-hot** encoded.
* Works well with the Softmax activation function.
* Encourages correct class probabilities to be higher.

**OONE HOT ENCODE:**

Consider example of student marks

| Maths | Phy | O/P | Good | Bad | Neutral |
| --- | --- | --- | --- | --- | --- |
| 80 | 89 | Good | 1 | 0 | 0 |
| 66 | 78 | Neutral | 0 | 0 | 1 |
| 43 | 38 | Bad | 0 | 1 | 0 |

This has multiple outputs. hands this is multiclass classification problem

**convert categorical variables into numerical form** so that machine learning and deep learning models can process them.

## **Pros & Cons of One-Hot Encoding**

✅ **Pros:** ✔ Works well for categorical variables.  
 ✔ Prevents the model from assuming an order between categories.

❌ **Cons:** ⛔ Creates **too many columns** if there are many categories (**high dimensionality**).  
 ⛔ Not efficient for large datasets

**Flash Note:**

**1 Binary Classification problem:**

* **ReLu** at hidden layers,
* **Sigmoid** at output layer,
* **BinaryCross entropy**(BCE loss function)
* **SGD or Adam** optimiser

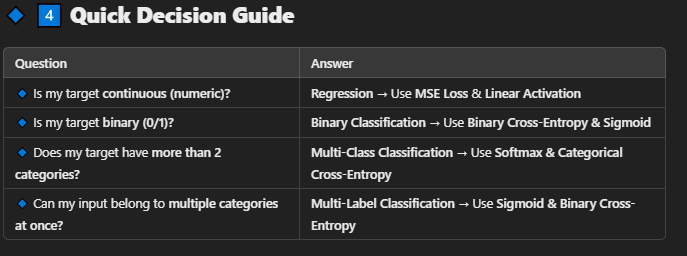
**2 Multi Class classification problem:**

* **ReLu** at hidden layer
* **SoftMax** at output layer
* **Categorical Entropy** (loss function)
* **SGD or Adam** optimiser

**3 Regression classification problem**

* **ReLu** at hidden layer
* **Linear Regression** at output layer
* **MSE , MAE ,** Huber loss functions.
* **SGD or Adam** optimiser

OTHER ML PROBLEM TYPES:

****

**Multi classification problem like: Sentimental analysis**

**Reinforcement problem :auto cars,**