

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/361495121>

# Artificial Intelligence & Machine Learning Unit 5: Reinforced and Deep Learning Question bank and its solution

Presentation · June 2022

DOI: 10.13140/RG.2.2.35149.92643

CITATIONS

0

READS

12,869

1 author:



[Abhishek D. Patange](#)

ABB

115 PUBLICATIONS 753 CITATIONS

[SEE PROFILE](#)



# Artificial Intelligence & Machine Learning

Course Code: 302049

## Unit 5: Reinforced and Deep Learning

*Third Year Bachelor of Engineering (Choice Based Credit System)*

*Mechanical Engineering (2019 Course)*

*Board of Studies – Mechanical and Automobile Engineering, SPPU, Pune*

*(With Effect from Academic Year 2021-22)*

### Question bank and its solution

by

**Abhishek D. Patange, Ph.D.**

**Department of Mechanical Engineering**

**College of Engineering Pune (COEP)**



## Unit 5: REINFORCED AND DEEP LEARNING

### Syllabus:

Content	Theory	Mathematics	Numerical
<b>• Reinforced learning</b>			
Characteristics of reinforced learning	✓	✗	✗
Algorithms: Value Based, Policy Based, Model Based; Positive Vs. Negative Reinforced Learning	✓	✗	✗
Models: Markov Decision Process, Q Learning	✓	✓	✓
<b>• Deep Learning</b>			
Characteristics of Deep Learning	✓	✗	✗
Artificial Neural Network	✓	✗	✗
Convolution Neural Network	✓	✗	✗
<b>• Application of Reinforced and Deep Learning in Mechanical Engineering</b>			

### Type of question and marks:

Type	Theory	Mathematics	Numerical
<b>Marks</b>	2 or 4 or 6 marks	4 marks	2 or 4 marks

Topic: Characteristics of reinforced learning

Theory	Mathematics	Numerical
✓	✗	✗

Theory questions

1. What is reinforcement learning? State one practical example.

- Reinforcement Learning is a **feedback-based Machine learning** technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the **agent learns automatically using feedbacks** without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to **learn by its experience only**.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that **"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.

• **Example:**

Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.

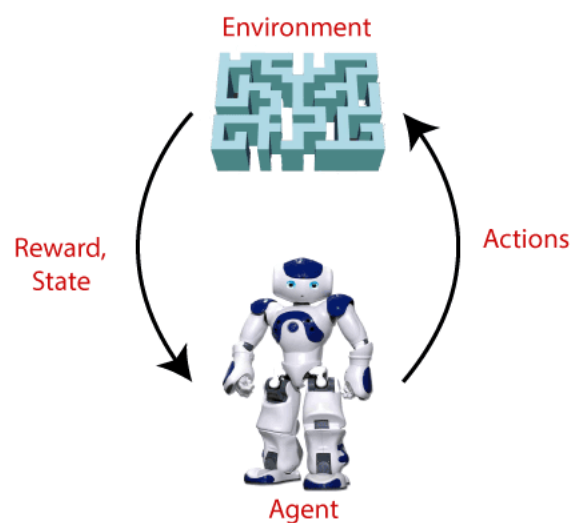
The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.

The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

### 2. State key constituents of reinforcement learning. (Explain key terms in reinforcement learning.)

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term return with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).



### 3. State key features of reinforcement learning.

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

### 4. Explain approaches to implement reinforcement learning.

OR

### Explain value-based, policy-based, and model-based reinforcement learning.

There are mainly three ways to implement reinforcement-learning in ML, which are:

- **Value-based:** The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy  $\pi$ .
- **Policy-based:** Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy:
  - ❖ **Deterministic:** The same action is produced by the policy ( $\pi$ ) at any state.
  - ❖ **Stochastic:** In this policy, probability determines the produced action.
- **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

### 5. Explain elements of reinforcement learning.

There are four main elements of Reinforcement Learning, which are given below:

Policy, Reward Signal, Value Function, Model of the environment

- **Policy:** A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

**For deterministic policy:**  $a = \pi(s)$

**For stochastic policy:**  $\pi(a | s) = P[A_t = a | S_t = s]$

- **Reward Signal:** The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

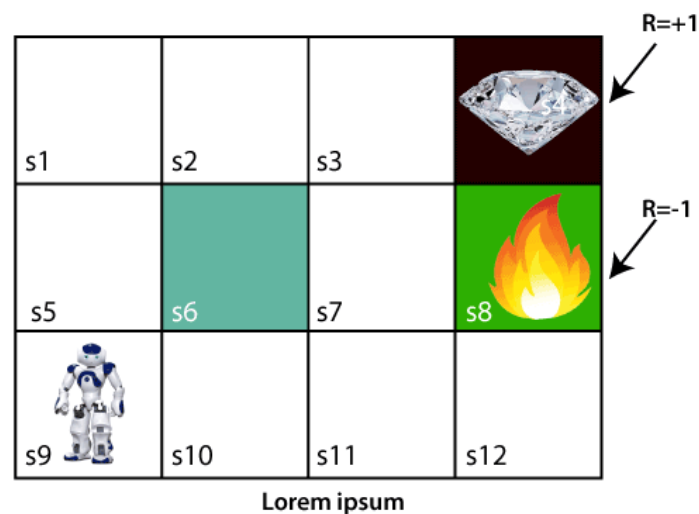
- **Value Function:** The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.
- **Model:** The last element of reinforcement learning is the model, which mimics the behaviour of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward. The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

### 6. How does Reinforcement Learning Work?

To understand the working process of the RL, we need to consider two main things:

- **Environment:** It can be anything such as a room, maze, football ground, etc.
- **Agent:** An intelligent agent such as AI robot.

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:






In the above image, the agent is at the very first block of the maze. The maze is consisting of an  $S_6$  block, which is a **wall**,  $S_8$  a **fire pit**, and  $S_4$  a **diamond block**.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING






The agent cannot cross the  $S_6$  block, as it is a solid wall. If the agent reaches the  $S_4$  block, then get the **+1 reward**; if it reaches the fire pit, then gets **-1 reward point**. It can take four actions: **move up, move down, move left, and move right**.

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path **S9-S5-S1-S2-S3**, so he will get the +1-reward point.

The agent will try to remember the preceding steps that it has taken to reach the final step. To memorize the steps, it assigns 1 value to each previous step. Consider the below step:

$V=1$ s1	$V=1$ s2	$V=1$ s3	 s4
$V=1$ s5	s6	s7	 s8
 $V=1$ s9	s10	s11	s12

Now, the agent has successfully stored the previous steps assigning the 1 value to each previous block. But what will the agent do if he starts moving from the block, which has 1 value block on both sides? Consider the below diagram:

 s1	 $V=1$ s2	$V=1$ s3	 s4
 $V=1$ s5	s6	s7	 s8
$V=1$ s9	s10	s11	s12

It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.



7. What is the Bellman Equation? How is it helpful in RL?

The Bellman equation was introduced by the Mathematician **Richard Ernest Bellman in the year 1953**, and hence it is called as a Bellman equation. It is associated with dynamic programming and used to calculate the values of a decision problem at a certain point by including the values of previous states.

It is a way of calculating the value functions in dynamic programming or environment that leads to modern reinforcement learning.

The key-elements used in Bellman equations are:

- Action performed by the agent is referred to as "a"
- State occurred by performing the action is "s."
- The reward/feedback obtained for each good and bad action is "R."
- A discount factor is Gamma "γ."

The Bellman equation can be written as:

$$V(s) = \max [R(s,a) + \gamma V(s')]$$

Where,

**V(s)= value calculated at a particular point.**

**R(s,a) = Reward at a particular state s by performing an action.**

**γ = Discount factor**

**V(s') = The value at the previous state.**

In the above equation, we are taking the max of the complete values because the agent tries to find the optimal solution always.

So now, using the Bellman equation, we will find value at each state of the given environment. We will start from the block, which is next to the target block.

**For 1st block:**

$$V(s_3) = \max [R(s,a) + \gamma V(s')], \text{ here } V(s') = 0 \text{ because there is no further state to move.}$$

$$V(s_3) = \max[R(s,a)] \Rightarrow V(s_3) = \max[1] \Rightarrow \mathbf{V(s_3) = 1.}$$

**For 2nd block:**

$$V(s_2) = \max [R(s,a) + \gamma V(s')], \text{ here } \gamma = 0.9(\text{lets}), V(s') = 1, \text{ and } R(s, a) = 0, \text{ because there is no reward at this state.}$$

$$V(s_2) = \max[0.9(1)] \Rightarrow V(s_2) = \max[0.9] \Rightarrow \mathbf{V(s_2) = 0.9}$$

**For 3rd block:**

$$V(s_1) = \max [R(s,a) + \gamma V(s')], \text{ here } \gamma = 0.9(\text{lets}), V(s') = 0.9, \text{ and } R(s, a) = 0, \text{ because there is no reward at this state also.}$$

$$V(s_1) = \max[0.9(0.9)] \Rightarrow V(s_1) = \max[0.81] \Rightarrow \mathbf{V(s_1) = 0.81}$$

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

### For 4th block:

$V(s_5) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 0.81$ , and  $R(s, a) = 0$ , because there is no reward at this state also.




$$V(s_5) = \max[0.9(0.81)] \Rightarrow V(s_5) = \max[0.73] \Rightarrow \mathbf{V(s_5) = 0.73}$$

### For 5th block:




$V(s_9) = \max [R(s,a) + \gamma V(s')]$ , here  $\gamma = 0.9$  (lets),  $V(s') = 0.73$ , and  $R(s, a) = 0$ , because there is no reward at this state also.

$$V(s_9) = \max[0.9(0.73)] \Rightarrow V(s_9) = \max[0.66] \Rightarrow \mathbf{V(s_9) = 0.66}$$




Consider the below image:

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5		s7	 s8
 V=0.66 s9	s10	s11	s12

Now, we will move further to the 6<sup>th</sup> block, and here agent may change the route because it always tries to find the optimal path. Let's consider from the block next to the fire pit.

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5		 s7	 s8
V=0.66 s9	s10	s11	s12

Now, the agent has three options to move; if he moves to the blue box, then he will feel a bump if he moves to the fire pit, then he will get the -1 reward. But here we are taking only positive rewards, so for this, he will move to upwards only. The complete block values will be calculated using this formula. Consider the below image:

V=0.81 s1	V=0.9 s2	V=1 s3	 s4
V=0.73 s5	 s6	V=0.9 s7	 s8
V=0.66 s9	V=0.73 s10	V=0.81 s11	V=0.73 s12

**8. Explain types of reinforcement learning: (Positive & Negative reinforcement)**

**Positive Reinforcement:** The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior. This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences. Advantages are:

- Maximizes Performance
- Sustain Change for a long period of time
- Too much Reinforcement can lead to an overload of states which can diminish the results

**Negative Reinforcement:** The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition. It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior. Advantages are:

- Increases Behavior
- Provide defiance to a minimum standard of performance
- It Only provides enough to meet up the minimum behavior

**9. How to represent the agent state?**

We can represent the agent state using the **Markov State** that contains all the required information from the history. The State  $S_t$  is Markov state if it follows the given condition:

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \dots, S_t]$$

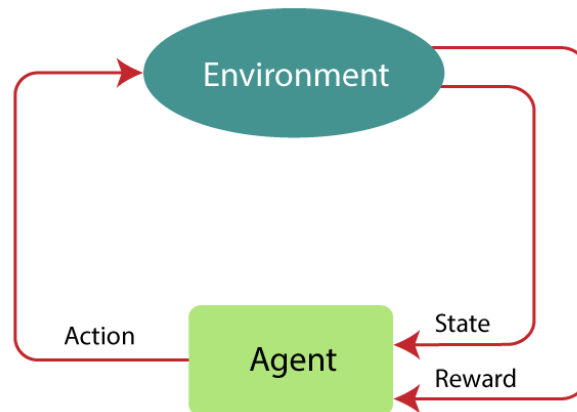
The Markov state follows the **Markov property**, which says that the future is independent of the past and can only be defined with the present. The RL works on fully observable

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

environments, where the agent can observe the environment and act for the new state. The complete process is known as Markov Decision process, which is explained below:

### 10. Explain Markov Decision Process

Markov Decision Process or MDP, is used to **formalize the reinforcement learning problems**. If the environment is completely observable, then its dynamic can be modeled as a **Markov Process**. In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.



MDP is used to describe the environment for the RL, and almost all the RL problem can be formalized using MDP.

MDP contains a tuple of four elements ( $S, A, P_a, R_a$ ):

- A set of finite States  $S$
- A set of finite Actions  $A$
- Rewards received after transitioning from state  $S$  to state  $S'$ , due to action  $a$ .
- Probability  $P_a$ .

MDP uses **Markov property**, and to better understand the MDP, we need to learn about it.

**Markov Property:** It says that *"If the agent is present in the current state  $S_1$ , performs an action  $a_1$  and move to the state  $s_2$ , then the state transition from  $s_1$  to  $s_2$  only depends on the current state and future action and states do not depend on past actions, rewards, or states."* Or, in other words, as per Markov Property, the current state transition does not depend on any past action or state. Hence, MDP is an RL problem that satisfies the Markov property. Such as in a Chess game, the players only focus on the current state and do not need to remember past actions or states.

**Finite MDP:** A finite MDP is when there are finite states, finite rewards, and finite actions. In RL, we consider only the finite MDP.

#### Markov Process:

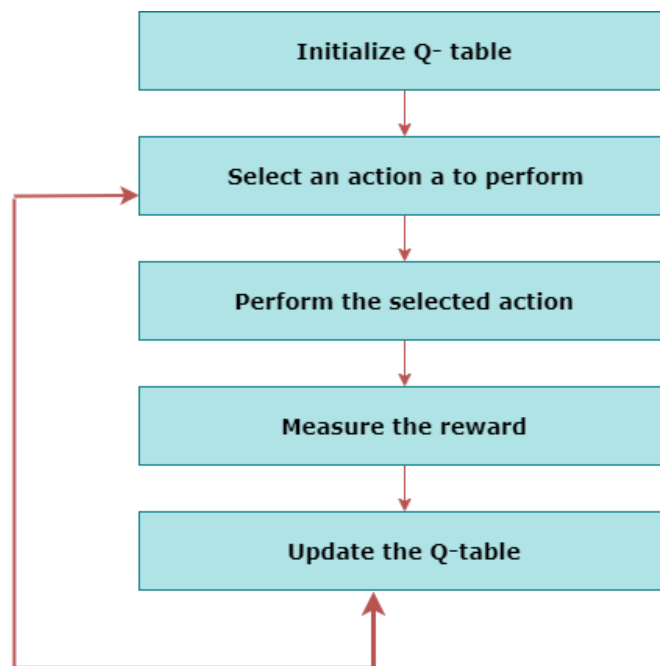
Markov Process is a memoryless process with a sequence of random states  $S_1, S_2, \dots, S_t$  that uses the Markov Property. Markov process is also known as Markov chain, which is a tuple ( $S,$

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

P) on state  $S$  and transition function  $P$ . These two components ( $S$  and  $P$ ) can define the dynamics of the system.

### 11. Explain Q-Learning.

- Q-learning is an **off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
- It learns the value function  $Q(S, a)$ , which means how good to take action "**a**" at a particular state "**s**."
- The below flowchart explains the working of Q- learning:



### State Action Reward State action (SARSA):

- SARSA stands for **State Action Reward State action**, which is an **on-policy** temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.
- The goal of SARSA is to calculate the  **$Q \pi(s, a)$  for the selected current policy  $\pi$  and all pairs of  $(s-a)$** .
- The main difference between Q-learning and SARSA algorithms is that **unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table**.
- In SARSA, new action and reward are selected using the same policy, which has determined the original action. The SARSA is named because it uses the quintuple  **$Q(s, a, r, s', a')$** .

Where,

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

**s:** original state

**a:** Original action

**r:** reward observed while following the states

**s' and a':** New state, action pair

### Deep Q Neural Network (DQN):

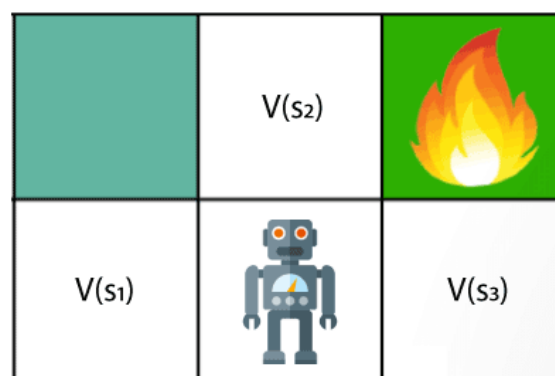
- As the name suggests, DQN is a **Q-learning using Neural networks**.
- For a big state space environment, it will be a challenging and complex task to define and update a Q-table.
- To solve such an issue, we can use a DQN algorithm. Where, instead of defining a Q-table, neural network approximates the Q-values for each action and state.
- Now, we will expand the Q-learning.

### Q-Learning Explanation:

- Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation.
- **The main objective of Q-learning is to learn the policy which can inform the agent that what actions should be taken for maximizing the reward under what circumstances.**
- It is an **off-policy RL** that attempts to find the best action to take at a current state.
- The goal of the agent in Q-learning is to maximize the value of Q.
- The value of Q-learning can be derived from the Bellman equation. Consider the Bellman equation given below:

$$V(s) = \max [R(s,a) + \gamma \sum_{s'} P(s, a, s') V(s')]$$

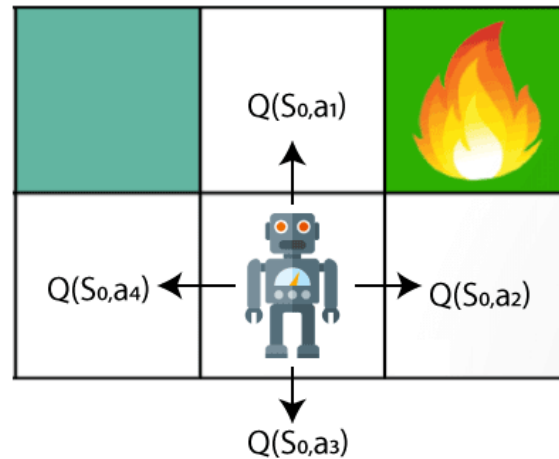
- In the equation, we have various components, including reward, discount factor ( $\gamma$ ), probability, and end states  $s'$ . But there is no any Q-value is given so first consider the below image:



- In the above image, we can see there is an agent who has three values options,  $V(s_1)$ ,  $V(s_2)$ ,  $V(s_3)$ . As this is MDP, so agent only cares for the current state and the future state.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

The agent can go to any direction (Up, Left, or Right), so he needs to decide where to go for the optimal path. Here agent will take a move as per probability bases and changes the state. But if we want some exact moves, so for this, we need to make some changes in terms of Q-value. Consider the below image:



- Q represents the quality of the actions at each state. So instead of using a value at each state, we will use a pair of state and action, i.e.,  $Q(s, a)$ . Q-value specifies that which action is more lubricative than others, and according to the best Q-value, the agent takes his next move. The Bellman equation can be used for deriving the Q-value.
- To perform any action, the agent will get a reward  $R(s, a)$ , and also he will end up on a certain state, so the Q -value equation will be:

$$Q(S, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V(s')$$

- Hence, we can say that,  $V(s) = \max [Q(s, a)]$

$$Q(S, a) = R(s, a) + \gamma \sum_{s'} (P(s, a, s') \max_{a'} Q(s', a'))$$

- **The above formula is used to estimate the Q-values in Q-Learning.**

**What is 'Q' in Q-learning?**

- The Q stands for **quality** in **Q-learning**, which means it specifies the quality of an action taken by the agent.

**Q-table:** A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, i.e.,  $[s, a]$ , and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table. The RL agent uses this Q-table as a reference table to select the best action based on the q-values.

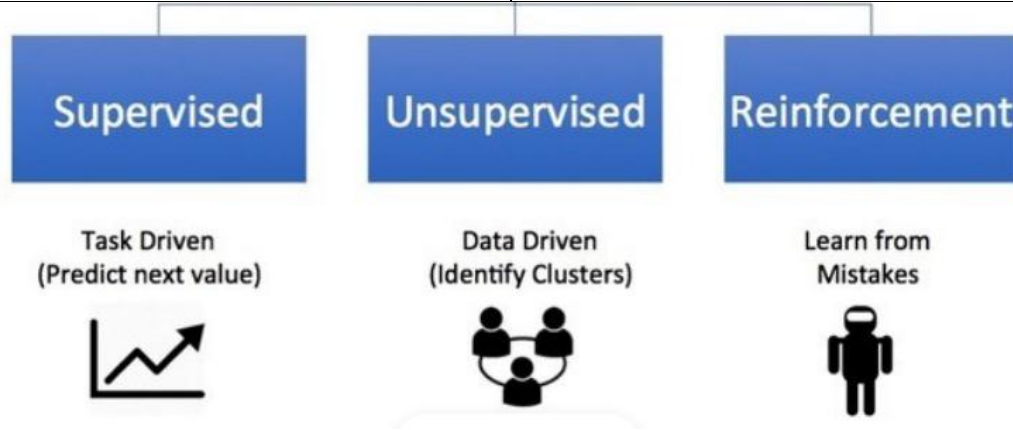
### 12. Difference between Reinforcement Learning and Supervised Learning.

The Reinforcement Learning and Supervised Learning both are the part of machine learning, but both types of learnings are far opposite to each other. The RL agents interact with the

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

environment, explore it, take action, and get rewarded. Whereas supervised learning algorithms learn from the labeled dataset and, on the basis of the training, predict the output. The difference table between RL and Supervised learning is given below:

Reinforcement Learning	Supervised Learning
RL works by interacting with the environment.	Supervised learning works on the existing dataset.
The RL algorithm works like the human brain works when making some decisions.	Supervised Learning works as when a human learns things in the supervision of a guide.
There is no labeled dataset is present	The labeled dataset is present.
No previous training is provided to the learning agent.	Training is provided to the algorithm so that it can predict the output.
RL helps to take decisions sequentially.	In Supervised learning, decisions are made when input is given.



### 13. Explain various practical applications of reinforcement learning.

#### Applications in self-driving cars

Various papers have proposed Deep Reinforcement Learning for **autonomous driving**. In self-driving cars, there are various aspects to consider, such as speed limits at various places, drivable zones, avoiding collisions—just to mention a few.

Some of the autonomous driving tasks where reinforcement learning could be applied include trajectory optimization, motion planning, dynamic pathing, controller optimization, and scenario-based learning policies for highways.

For example, parking can be achieved by learning automatic parking policies. Lane changing can be achieved using Q-Learning while overtaking can be implemented by learning an overtaking policy while avoiding collision and maintaining a steady speed thereafter.

AWS DeepRacer is an autonomous racing car that has been designed to test out RL in a physical track. It uses cameras to visualize the runway and a reinforcement learning model to control the throttle and direction.



## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

Wayve.ai has successfully applied reinforcement learning to training a car on how to **drive in a day**. They used a deep reinforcement learning algorithm to tackle the lane following task. Their network architecture was a deep network with 4 convolutional layers and 3 fully connected layers. The example below shows the lane following task. The image in the middle represents the driver's perspective.

### Industry automation with Reinforcement Learning

In industry reinforcement, learning-based **robots** are used to perform various tasks. Apart from the fact that these robots are more efficient than human beings, they can also perform tasks that would be dangerous for people.

A great example is the use of AI agents by Deepmind to cool Google Data Centers. This led to a 40% reduction in **energy spending**. The centers are now fully controlled with the AI system without the need for human intervention. There is obviously still supervision from data center experts. The system works in the following way:

- Taking snapshots of data from the data centers every five minutes and feeding this to deep neural networks
- It then predicts how different combinations will affect future energy consumptions
- Identifying actions that will lead to minimal power consumption while maintaining a set standard of safety criteria
- Sending and implement these actions at the data center
- The actions are verified by the local control system.

### Reinforcement learning applications in engineering

In the engineering frontier, Facebook has developed an **open-source reinforcement learning platform**—Horizon. The platform uses reinforcement learning to optimize large-scale production systems. Facebook has used Horizon internally:

- to personalize suggestions
- deliver more meaningful notifications to users
- optimize video streaming quality.
- Horizon also contains workflows for:
  - simulated environments
  - a distributed platform for data preprocessing
  - training and exporting models in production.

A classic example of reinforcement learning in video display is serving a user a low or high bit rate video based on the state of the video buffers and estimates from other machine learning systems. Horizon is capable of handling production-like concerns such as: deploying at scale, feature normalization, distributed learning, serving and handling datasets with high-dimensional data and thousands of feature types.

### Reinforcement Learning in robotics manipulation

The use of deep learning and reinforcement learning can train robots that have the ability to grasp various objects—even those unseen during training. This can, for example, be used in building products in an assembly line. This is achieved by combining large-scale distributed optimization and a variant of deep Q-Learning called QT-Opt. QT-Opt support for continuous action spaces makes it suitable for robotics problems. A model is first trained offline and then deployed and fine-tuned on the real robot. Google AI applied this approach to **robotics grasping** where 7 real-world robots ran for 800 robot hours in a 4-month period. In this experiment, the QT-Opt approach succeeds in 96% of the grasp attempts across 700 trials grasps on objects that were previously unseen. Google AI's previous method had a 78% success rate. **Refer following links for demonstrations**

<https://www.youtube.com/watch?v=W4joe3zzgLU>

<https://aws.amazon.com/fr/deepracer/>

**RL can be used in large environments in the following situations:**

1. A model of the environment is known, but an analytic solution is not available;
2. Only a simulation model of the environment is given (the subject of simulation-based optimization)
3. The only way to collect information about the environment is to interact with it.

### 14. What is deep learning?

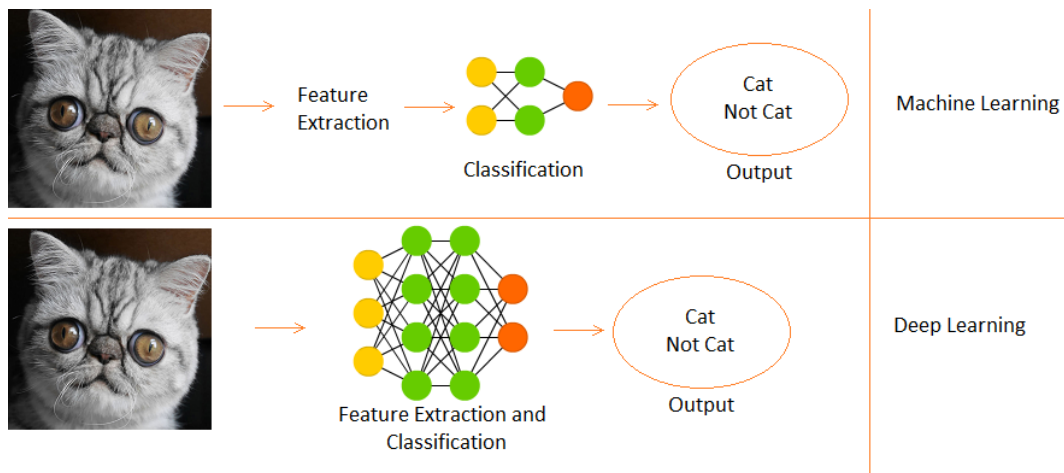
Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is - neurons. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- DL is a subfield of Machine Learning, inspired by the biological neurons of a brain, and translating that to artificial neural networks with representation learning.
- When the volume of data increases, Machine learning techniques, no matter how optimized, starts to become inefficient in terms of performance and accuracy, whereas Deep learning performs soo much better in such cases.
- Well one cannot quantify a threshold for data to be called big, but intuitively let's say a Million sample might be enough to say "It's Big"(This is where Michael Scott would've uttered his famous words "That's what she said")

### 15. State difference between Machine Learning and Deep Learning.

Machine Learning	Deep Learning
Works on small amount of Dataset for accuracy.	Works on Large amount of Dataset.
Dependent on Low-end Machine.	Heavily dependent on High-end Machine.
Divides the tasks into sub-tasks, solves them individually and finally combine the results.	Solves problem end to end.
Takes less time to train.	Takes longer time to train.
Testing time may increase.	Less time to test the data.



### 16. State different architectures of DL network.

- **Deep Neural Network** – It is a neural network with a certain level of complexity (having multiple hidden layers in between input and output layers). They are capable of modeling and processing non-linear relationships.
- **Deep Belief Network(DBN)** – It is a class of Deep Neural Network. It is multi-layer belief networks. **Steps for performing DBN:**
  - a. Learn a layer of features from visible units using Contrastive Divergence algorithm.
  - b. Treat activations of previously trained features as visible units and then learn features of features.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

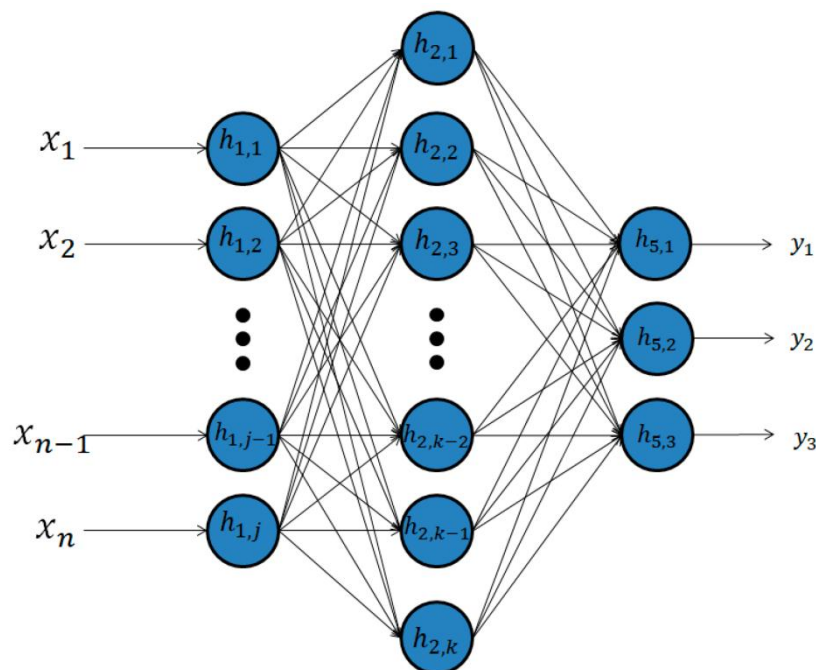
- c. Finally, the whole DBN is trained when the learning for the final hidden layer is achieved.
- **Recurrent** (perform same task for every element of a sequence) **Neural Network** – Allows for parallel and sequential computation. Similar to the human brain (large feedback network of connected neurons). They are able to remember important things about the input they received and hence enables them to be more precise.

### 17. Explain Artificial Neural Network (ANN).

This is another name for Deep Neural network or Deep Learning.

#### **What does a Neural Network mean?**

- What neural network essentially means is we take logistic regression and repeat it multiple times.
- In a normal logistic regression, we have an input layer and an output layer.
- But in the case of a Neural Network, there is at least one hidden layer of regression between these input and output layers.



#### **How many layers are needed to call it a "Deep" neural network?**

- Well of course there is no specific amount of layers to classify a neural network as deep.
- The term "Deep" is quite frankly relative to every problem.
- The correct question we can ask is "How much deep?".
- For example, the answer to "How deep is your swimming pool?" can be answered in multiple ways.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- It could be 2 meters deep or 10 meters deep, but it has "depth". Same with our neural network, it can have 2 hidden layers or "thousands" hidden layers(yes you heard that correctly).
- So I'd like to just stick with the question of "How much deep?" for the time being.

### ***Why are the hidden layers?***

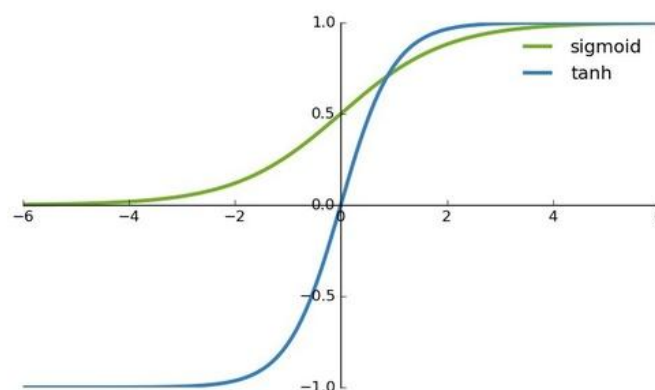
- They are called hidden because they do not see the original inputs( the training set ).
- For example, let's say you have a NN with an input layer, one hidden layer, and an output layer.
- When asked how many layers your NN has, your answer should be "It has 2 layers", because while computation the initial, or the input layer, is ignored.
- Let me help visualize how a 2 layer Neural network looks like.

### **Step by step we shall understand this image.**

1) As you can see here we have a 2 Layered Artificial Neural Network. A Neural network was created to mimic the biological neuron of the human brain. In our ANN we have a "k" number of nodes. The number of nodes is a hyperparameter, which essentially means that the amount is configured by the practitioner making the model.

2) The inputs and outputs layers do not change. We have "n" input features and 3 possible outcomes.

3) Unlike Logistic regression, neural networks use the *tanh* function as their activation function instead of the *sigmoid* function which you are quite familiar with. The reason is that the mean of its output is closer to 0 which makes the more centered for input to the next layer. *tanh* function can cause an increase in non-linearity which makes our model learn better.



4) In normal logistic regression: Input => Output.

Whereas in a Neural network: Input => Hidden Layer => Output. The hidden layer can be imagined as the output of part 1 and input of part 2 of our ANN.

Now let us have a more practical approach to a 2 Layered Neural Network.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

(Important Note: We shall continue where we left off in the previous article. I'm not going to waste your time and mine by loading the dataset again and preparing it. The link to the Part 1 of this series is given above.)

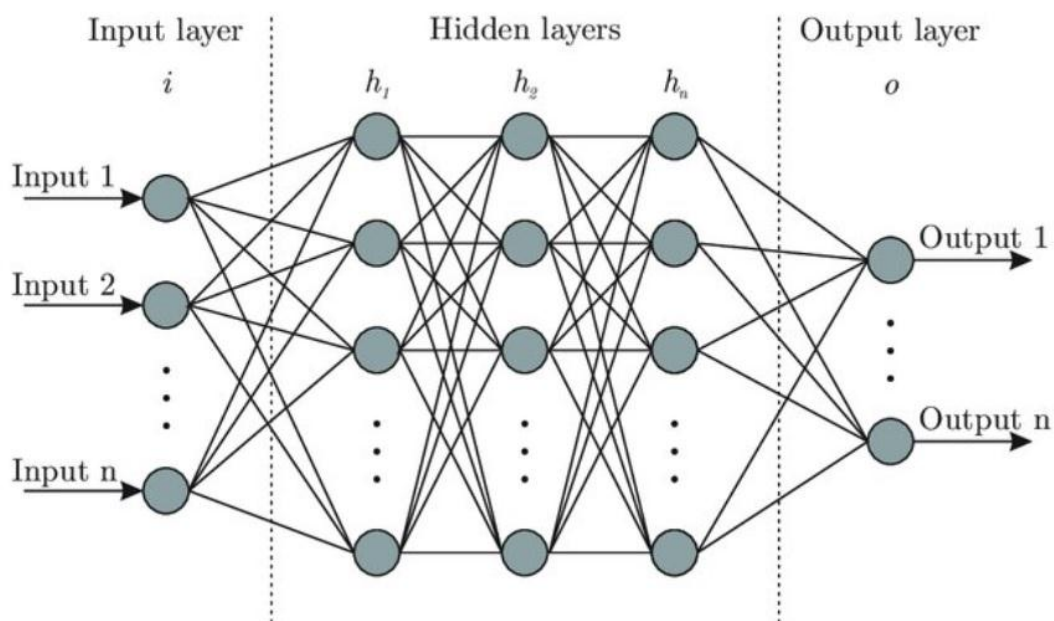
### 18. Explain elements of Deep Learning?

Researchers tried to mimic the working of the human brain and replicated it into the machine making machines capable of thinking and solving complex problems. Deep Learning (DL) is a subset of Machine Learning (ML) that allows us to train a model using a set of inputs and then predict output based. Like the human brain, the model consists of a set of neurons that can be grouped into 3 layers:

**a) Input Layer:** It receives input and passes it to hidden layers.

**b) Hidden Layers:** There can be 1 or more hidden layers in Deep Neural Network (DNN). "Deep" in DL refers to having more than 1 layer. All computations are done by hidden layers.

**c) Output Layer:** This layer receives input from the last hidden layer and gives the output.

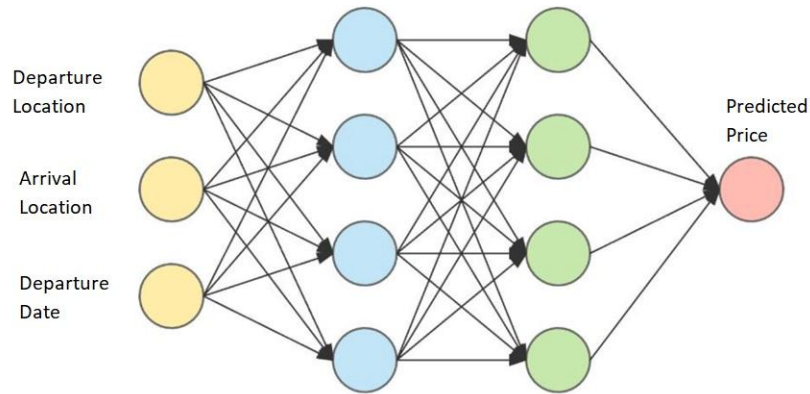


### 19. Explain working of deep Learning with an example.

We will see how DNN works with the help of the train price prediction problem. For simplicity, we have taken 3 inputs, namely, Departure Station, Arrival Station, Departure Date. In this case, the **input layer** will have 3 neurons, one for each input. The first **hidden layer** will receive input from the input layer and will start performing mathematical computations followed by other hidden layers. The number of hidden layers and number of neurons in each hidden layer is hyperparameters that are challenging task to decide. **The output layer** will give the predicted price value. There can be more than 1 neuron in the output layer. In our case, we have only 1 neuron as output is a single value.



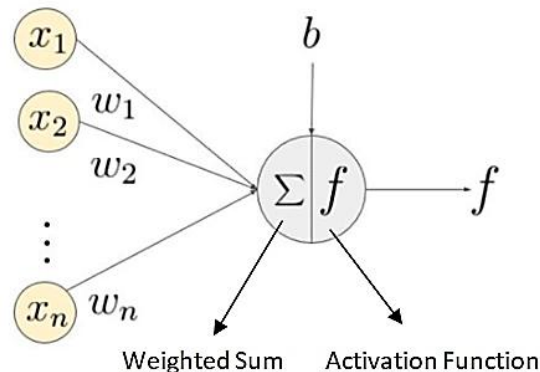
## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING



Now, how the price prediction is made by hidden layers? How computation is done inside hidden layers? This will be explained with help of activation function, loss function, and optimizers.

### 20. What is activation Functions?

Each neuron has an **activation function** that performs computation. Different layers can have different activation functions but neurons belonging to one layer have the same activation function. In DNN, a weighted sum of input is calculated based on weights and inputs provided. Then, the activation function comes into the picture that works on weighted sum and converts it into output.



### 21. Why activation functions are required?

Activation functions help model learn complex relationship that exists within the dataset. If we do not use the activation function in neurons and give weighted sum as output, in that case, computations will be difficult as there is no specific range for weighted sum. So, the activation function helps to keep output in a particular range. Secondly, the non-linear activation function is always preferred as it adds non-linearity to the dataset which otherwise would form a simple linear regression model incapable of taking the benefit of hidden layers. The relu function or its variants is mostly used for hidden layers and sigmoid/ softmax function is mostly used for final layer for binary/ multi-class classification problems.

## 22. What is Loss/ Cost Function?

To train the model, we give input (departure location, arrival location and, departure date in case of train price prediction) to the network and let it predict the output making use of activation function. Then, we compare predicted output with the actual output and compute the error between two values. This error between two values is computed using **loss/ cost function**. The same process is repeated for entire training dataset and we get the average loss/error. Now, the objective is to minimize this loss to make the model accurate. There exist weights between each connection of 2 neurons. Initially, weights are randomly initialized and the motive is to update these weights with every iteration to get the minimum value of the loss/ cost function. We can change the weights randomly but that is not efficient method. Here comes the role of **optimizers** which updates weights automatically.

## 23. What are different loss functions and their use case?

Loss function is chosen based on the problem.

a. Regression Problem

Mean squared error (MSE) is used where real value quantity is to be predicted.

MSE in case of train price prediction as price predicted is real value quantity.

b. Binary/ Multi-class Classification Problem

Cross-entropy is used.

c. Maximum- Margin Classification

Hinge loss is used.

## 24. Explain optimizers. Why optimizers are required?

Once loss for one iteration is computed, optimizer is used to update weights. Instead of changing weights manually, optimizers can update weights automatically in small increments and helps to find the minimum value of the loss/ cost function. **Magic of DL!!** Finding minimum value of cost function requires iterating through dataset many times and thus requires large computational power. Common technique used to update these weights is **gradient descent**.

## 25. What is Gradient Descent (GD) and its variants?

It is used to find minimum value of loss function by updating weights. There are 3 variants:

### a) Batch/ Vanila Gradient

- In this, gradient for entire dataset is computed to perform one weight update.
- It gives good results but can be slow and requires large memory.

### b) Stochastic Gradient Descent (SGD)

- Weights are updated for each training data point.



## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- Therefore, frequent updates are performed and thus can cause objective function to fluctuate.

### c) Mini-batch Gradient Descent

- It takes best of batch gradient and SGD.
- It is algorithm of choice.
- Reduces frequency of updates and thus can lead to more stable convergence.
- Choosing proper learning rate is difficult.

### 26. What are GD optimization methods and which optimizer to use?

To overcome challenges in GD, some optimization methods are used by AI community. Further, less efforts are required in hyperparameter tuning.

#### a) Adagrad

- Algorithm of choice in case of sparse data.
- Eliminate need of manually tuning learning rate unlike GD.
- Default value of 0.01 is preferred.

#### b) Adadelta

- Reduces adagrad's monotonically decreasing learning rate.
- Do not require default learning rate.

#### c) RMSprop

- RMSprop and adadelta were developed for same purpose at same time.
- Learning rate = 0.001 is preferred.

#### d) Adam

- It works well with most of problems and is algorithm of choice.
- Seen as combination of RMSprop and momentum.
- AdaMax and Nadam are variants of Adam.

To sum up, DNN takes the input which is worked upon by activation function to make computation and learn complex relationship within dataset. Then, loss is computed for entire dataset based on actual and predicted values. Finally, to minimize the loss and making the predicted values close to actual, weights are updated using optimizers. This process continues till model converges with the motive of getting minimum loss value.

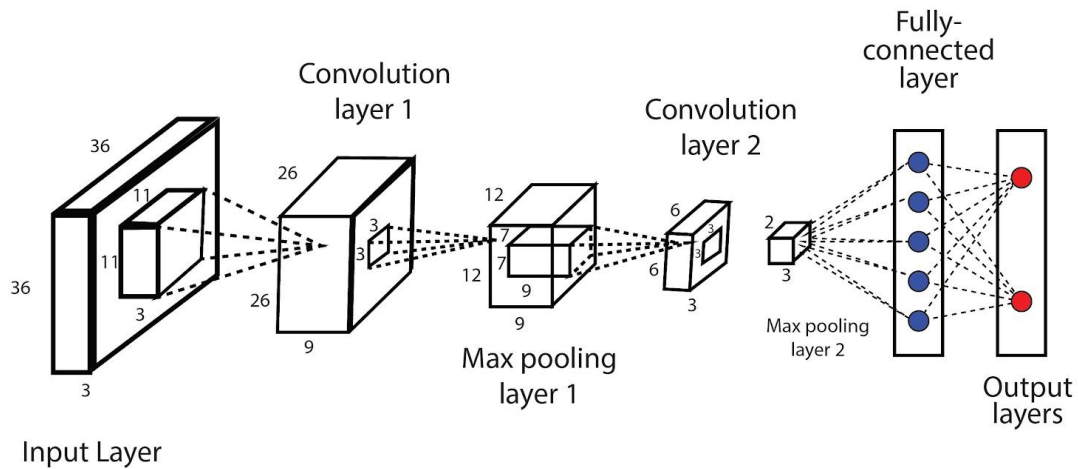
### 27. What is Convolutional Neural Network (CNN)?

"Convolution neural networks" indicates that these are simply neural networks with some mathematical operation (generally matrix multiplication) in between their layers called convolution. It was proposed by **Yann LeCun** in 1998. It's one of the most popular uses in Image Classification. Convolution neural network can broadly be classified into these steps:

#### 1. Input layer

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

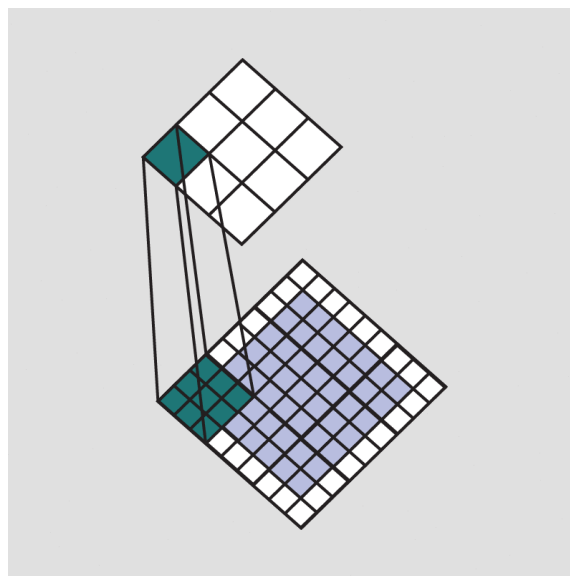
2. **Convolutional layer**
3. **Output layers**



### 28. Explain the architecture of Convolutional Neural Networks (CNN)?

Input layers are connected with convolutional layers that perform many tasks such as padding, striding, the functioning of kernels, and so many performances of this layer, this layer is considered as a building block of convolutional neural networks. We will be discussing its functioning in detail and how the fully connected networks work.

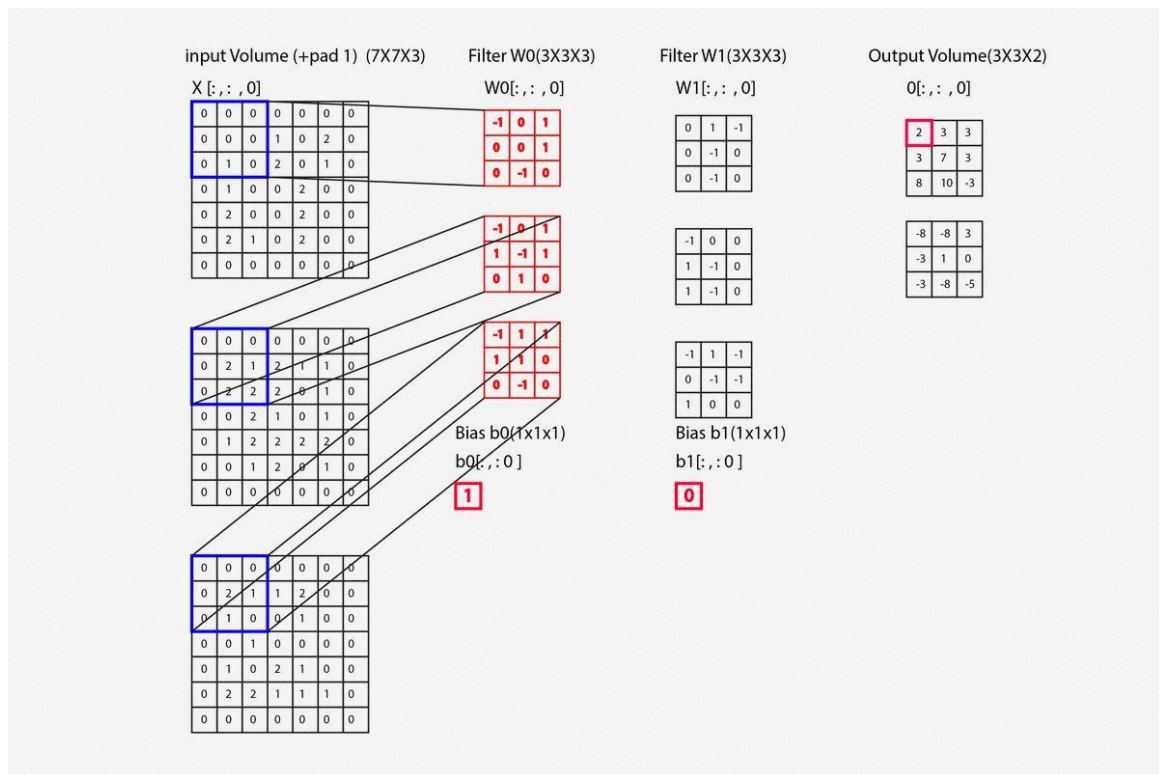
**Convolutional Layer:** The convolutional layer's main objective is to extract features from images and learn all the features of the image which would help in object detection techniques. As we know, the input layer will contain some pixel values with some weight and height, our kernels or filters will convolve around the input layer and give results which will retrieve all the features with fewer dimensions. Let's see how kernels work;



Formation and arrangement of Convolutional Kernels

With the help of this very informative visualization about kernels, we can see how the kernels work and how padding is done.

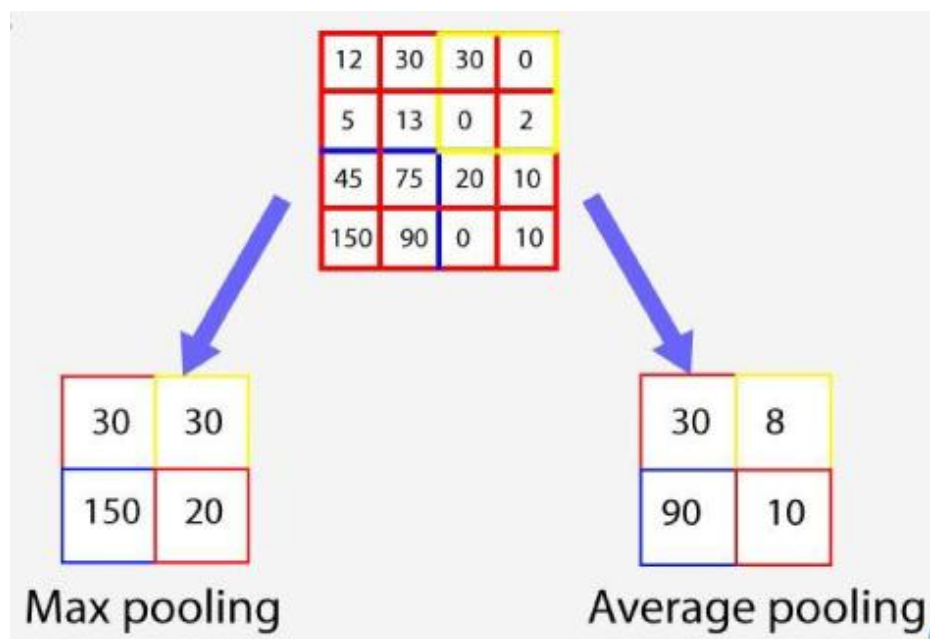
## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING



Matrix visualization in CNN

**Need for Padding:** We can see padding in our input volume, we need to do padding in order to make our kernels fit the input matrices. Sometimes we do zero paddings, i.e. adding one row or column to each side of zero matrices or we can cut out the part, which is not fitting in the input image, also known as valid padding. Let's see how we reduce parameters with negligible loss, we use techniques like Max-pooling and average pooling.

**Max pooling or Average pooling:**



Matrix formation using Max-pooling and average pooling

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

Max pooling or average pooling reduces the parameters to increase the computation of our convolutional architecture. Here, 2\*2 filters and 2 strides are taken (which we usually use). By name, we can easily assume that max-pooling extracts the maximum value from the filter and average pooling takes out the average from the filter. We perform pooling to reduce dimensionality. We have to add padding only if necessary. The more convolutional layer can be added to our model until conditions are satisfied.

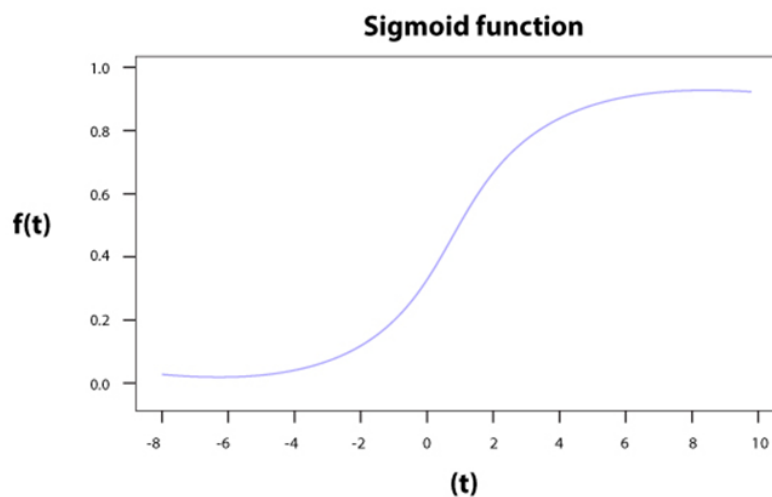
### 29. Explain activation functions in CNN?

An activation function is added to our network anywhere in between two convolutional layers or at the end of the network. So you must be wondering what exactly an activation function does, let me clear it in simple words for you. It helps in making the decision about which information should fire forward and which not by making decisions at the end of any network. In broadly, there are both linear as well as non-linear activation functions, both performing linear and non-linear transformations but non-linear activation functions are a lot helpful and therefore widely used in neural networks as well as deep learning networks. The four most famous activation functions to add non-linearity to the network are described below.

#### 1. Sigmoid Activation Function

The equation for the sigmoid function is

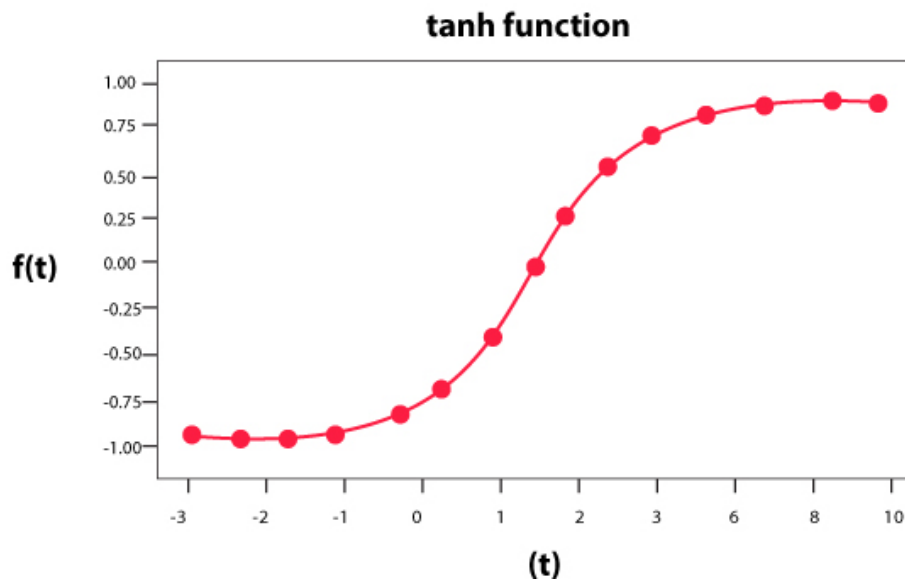
$$f(x) = \frac{1}{1+e^{-x}}$$



Sigmoid Activation function

The sigmoid activation function is used mostly as it does its task with great efficiency, it basically is a probabilistic approach towards decision making and ranges in between 0 to 1, so when we have to make a decision or to predict an output we use this activation function because of the range is the minimum, therefore, the prediction would be more accurate.

## 2. Hyperbolic Tangent Activation Function(Tanh)

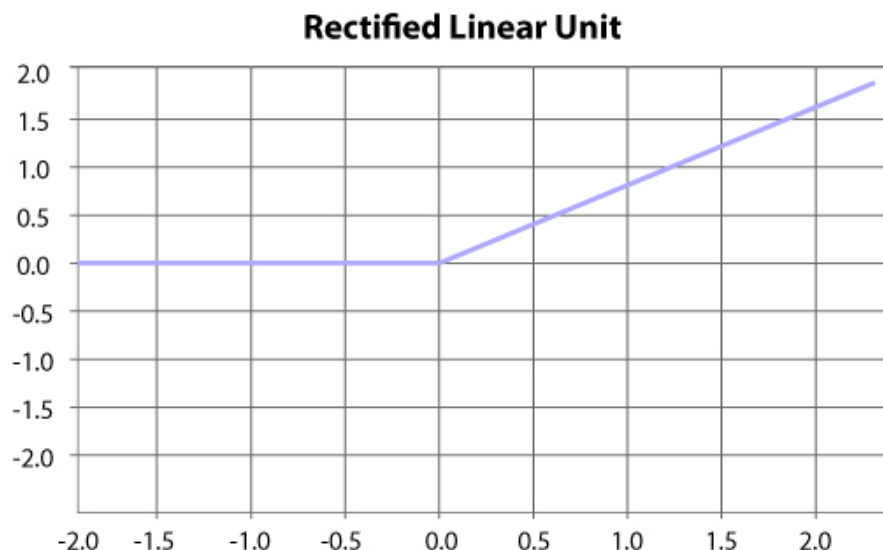


Tanh Activation function

This activation function is slightly better than the sigmoid function, like the sigmoid function it is also used to predict or to differentiate between two classes but it maps the negative input into negative quantity only and ranges in between -1 to 1.

## 3. ReLU (Rectified Linear unit) Activation function

Rectified linear unit or ReLU is the most widely used activation function right now which ranges from **0 to infinity**, all the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem, but where there is a problem there is a solution.

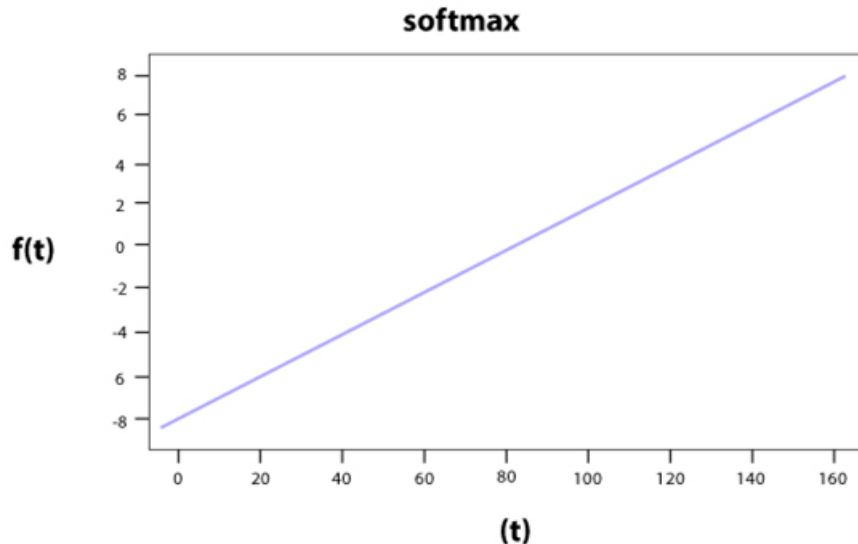


Rectified Linear Unit activation function

We use Leaky ReLU function instead of ReLU to avoid this unfitting, in Leaky ReLU range is expanded which enhances the performance.

### 4. Softmax Activation Function

Softmax is used mainly at the last layer i.e output layer for decision making the same as sigmoid activation works, the softmax basically gives value to the input variable according to their weight, and the sum of these weights is eventually one.



Softmax activation function

For Binary classification, both sigmoid, as well as softmax, are equally approachable but in the case of multi-class classification problems we generally use softmax and cross-entropy along with it.

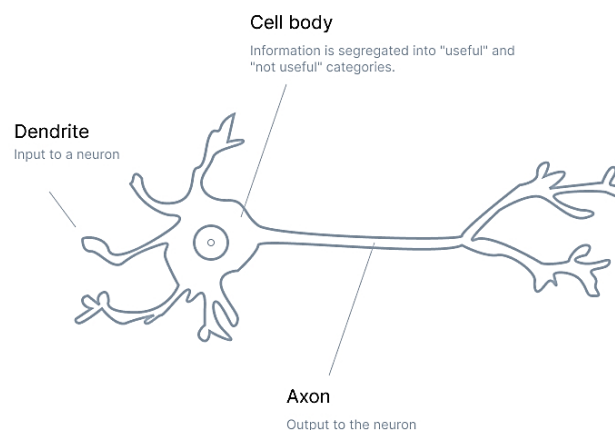
### 30. What's need of activation functions?

*"The world is one big data problem."*

As it turns out—

This saying holds true both for our brains as well as machine learning.

Every single moment our brain is trying to segregate the incoming information into the "useful" and "not-so-useful" categories.

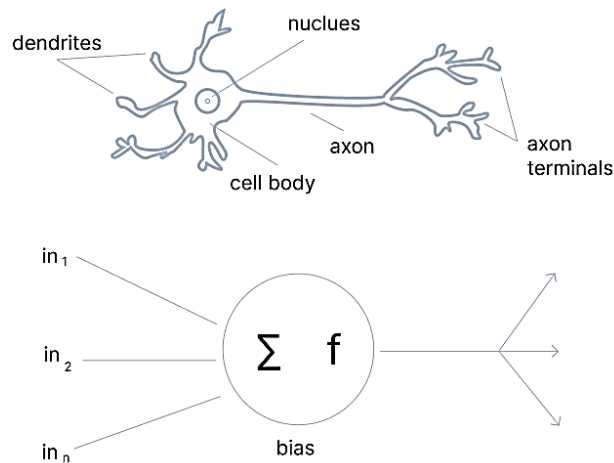


A similar process occurs in artificial neural network architectures in deep learning. The segregation plays a key role in helping a neural network properly function, ensuring that it

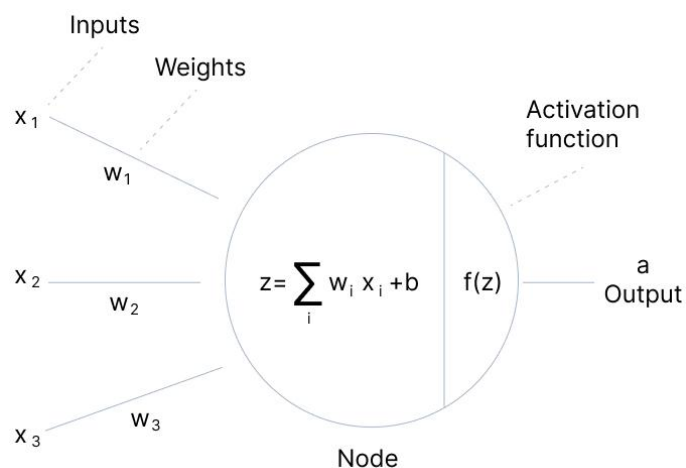
learns from the useful information rather than get stuck analyzing the not-useful part. And this is also where activation functions come into the picture. Activation Function helps the neural network to use important information while suppressing irrelevant data points.

### 31. What is a Neural Network Activation Function?

**An Activation Function** decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. The role of the Activation Function is to derive output from a set of input values fed to a node (or a layer). But—Let's take a step back and clarify: What exactly is a *node*? Well, if we compare the neural network to our brain, a node is a replica of a neuron that receives a set of input signals—external stimuli.



Depending on the nature and intensity of these input signals, the brain processes them and decides whether the neuron should be activated ("fired") or not. In deep learning, this is also the role of the Activation Function—that's why it's often referred to as a *Transfer Function* in Artificial Neural Network. The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.



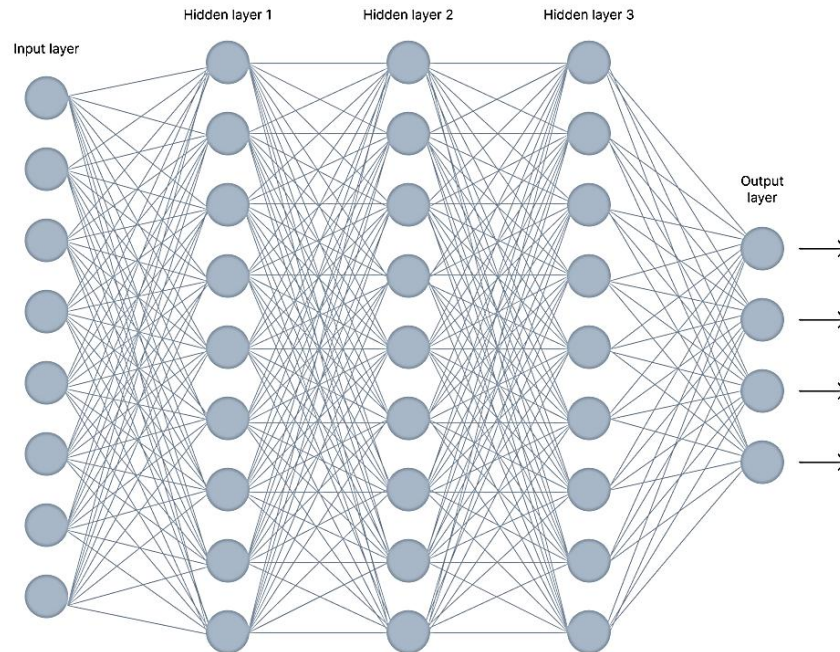


Now, let's have a look at the Neural Networks Architecture.

### Elements of a Neural Networks Architecture

Here's the thing— If you don't understand the concept of neural networks and how they work, diving deeper into the topic of activation functions might be challenging.

That's why it's a good idea to refresh your knowledge and take a quick look at the structure of the Neural Networks Architecture and its components. Here it is.



In the image above, you can see a neural network made of interconnected neurons. Each of them is characterized by its **weight**, **bias**, and **activation function**.

#### **Input Layer**

The input layer takes raw input from the domain. No computation is performed at this layer. Nodes here just pass on the information (features) to the hidden layer.

#### **Hidden Layer**

As the name suggests, the nodes of this layer are not exposed. They provide an abstraction to the neural network.

The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer.

#### **Output Layer**

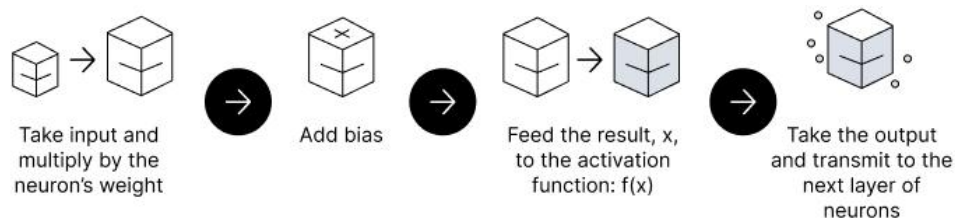
It's the final layer of the network that brings the information learned through the hidden layer and delivers the final value as a result.

**Note:** All hidden layers usually use the same activation function. However, the output layer will typically use a different activation function from the hidden layers. The choice depends on the goal or type of prediction made by the model.



### **Feedforward vs. Backpropagation**

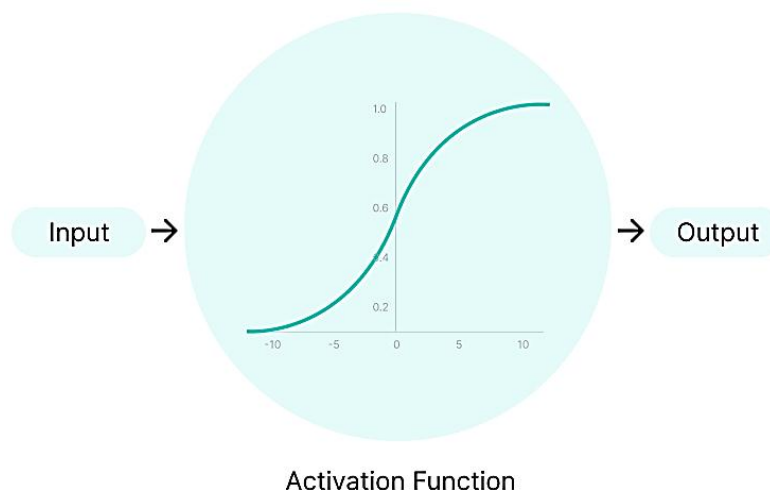
When learning about neural networks, you will come across two essential terms describing the movement of information—feedforward and backpropagation. Let's explore them. Feedforward Propagation - the flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate an output. In the feedforward propagation, the Activation Function is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer.



Backpropagation - the weights of the network connections are repeatedly adjusted to minimize the difference between the actual output vector of the net and the desired output vector. To put it simply—backpropagation aims to minimize the cost function by adjusting the network's weights and biases. The cost function gradients determine the level of adjustment with respect to parameters like activation function, weights, bias, etc.

### **Why do Neural Networks Need an Activation Function?**

So we know what Activation Function is and what it does, but— *Why* do Neural Networks need it? Well, the purpose of an activation function is to add non-linearity to the neural network.



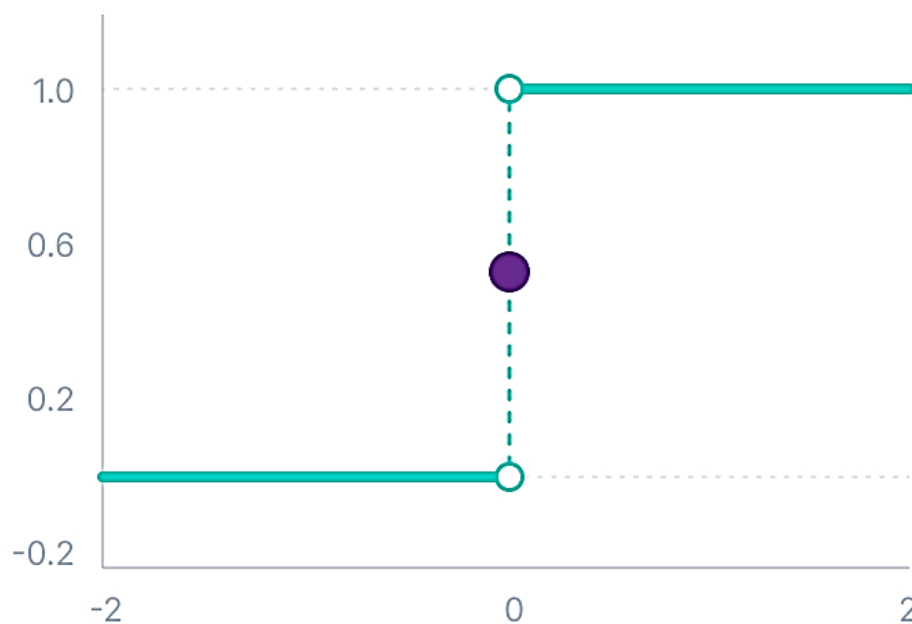
Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it. Here is why— Let's suppose we have a neural network working *without* the activation functions. In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it

doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself. Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model.

### 3 Types of Neural Networks Activation Functions

Now, as we've covered the essential concepts, let's go over the most popular neural networks activation functions.

**Binary Step Function:** Binary step function depends on a threshold value that decides whether a neuron should be activated or not. The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



Binary Step Function

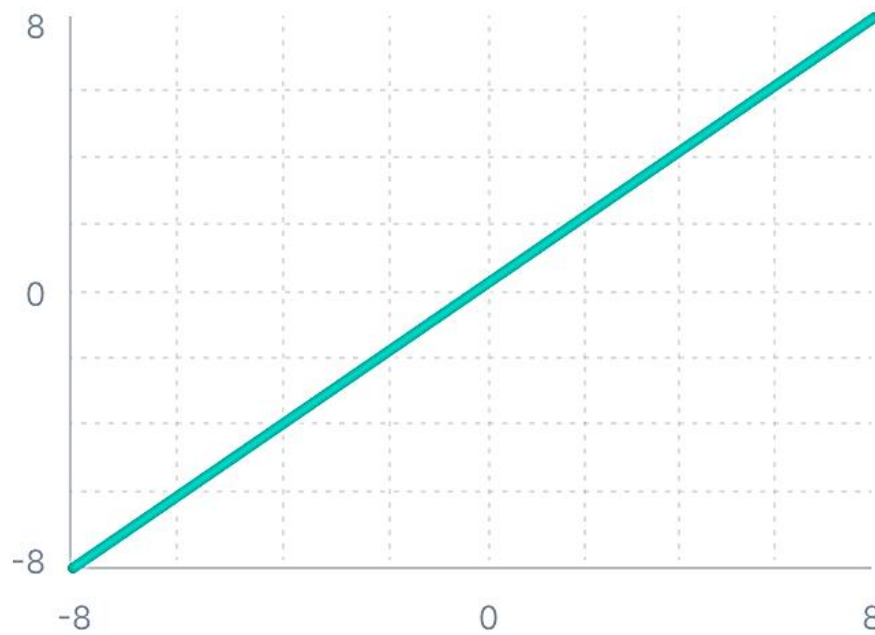
Mathematically it can be represented as:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Here are some of the limitations of binary step function:

- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

**Linear Activation Function:** The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input. The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.



Linear Activation Function

Mathematically it can be represented as:

$$f(x) = x$$

However, a linear activation function has two major problems :

- It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input  $x$ .
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

### Non-Linear Activation Functions

The linear activation function shown above is simply a linear regression model.

Because of its limited power, this does not allow the model to create complex mappings between the network's inputs and outputs.

Non-linear activation functions solve the following limitations of linear activation functions:

- They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

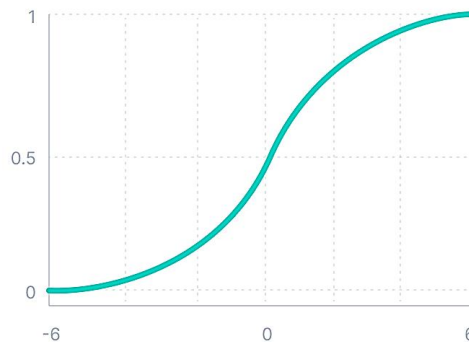
Now, let's have a look at ten different non-linear neural networks activation functions and their characteristics.

**32. Explain non-linear Neural Networks activation functions.**

**Sigmoid / Logistic Activation Function**

This function takes any real value as input and outputs values in the range of 0 to 1.

The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.



Sigmoid/Logistic Activation Function

Mathematically it can be represented as:

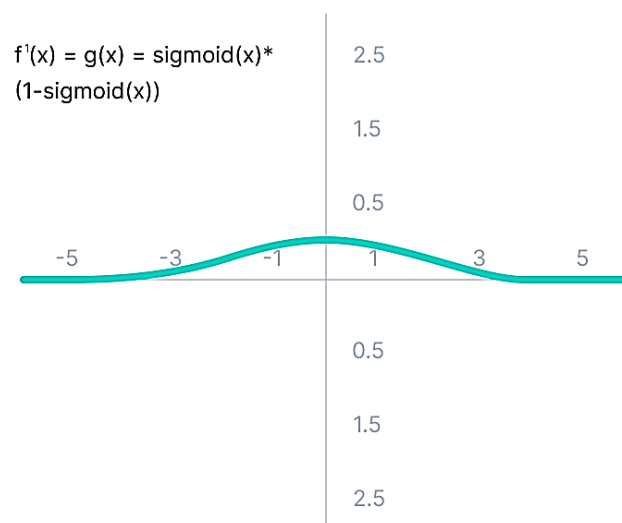
$$f(x) = \frac{1}{1 + e^{-x}}$$

Here's why sigmoid/logistic activation function is one of the most widely used functions:

- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

The limitations of sigmoid function are discussed below:

- The derivative of the function is  $f'(x) = \text{sigmoid}(x) * (1 - \text{sigmoid}(x))$ .

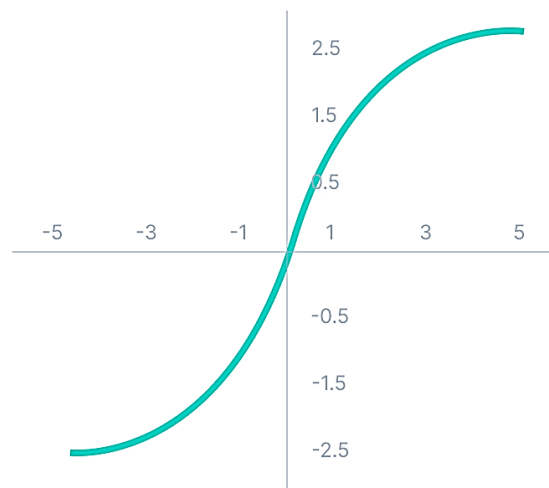


### The derivative of the Sigmoid Activation Function

As we can see from the above Figure, the gradient values are only significant for range -3 to 3, and the graph gets much flatter in other regions. It implies that for values greater than 3 or less than -3, the function will have very small gradients. As the gradient value approaches zero, the network ceases to learn and suffers from the *Vanishing gradient* problem. The output of the logistic function is not symmetric around zero. So the output of all the neurons will be of the same sign. This makes the training of the neural network more difficult and unstable.

### **Tanh Function (Hyperbolic Tangent)**

Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.



Tanh Function (Hyperbolic Tangent)

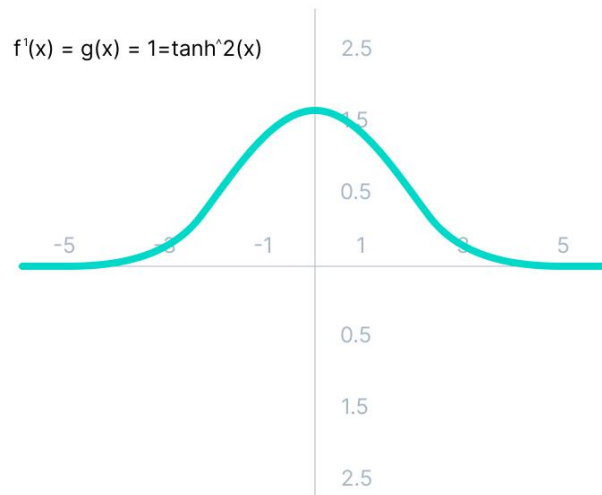
Mathematically it can be represented as:

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

Advantages of using this activation function are:

- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between -1 to 1; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data and makes learning for the next layer much easier.

Have a look at the gradient of the tanh activation function to understand its limitations.

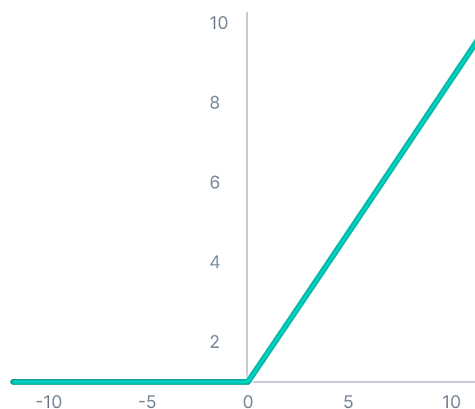


Gradient of the Tanh Activation Function

As you can see— it also faces the problem of vanishing gradients similar to the sigmoid activation function. Plus the gradient of the tanh function is much steeper as compared to the sigmoid function. Note: Although both sigmoid and tanh face vanishing gradient issue, tanh is zero centered, and the gradients are not restricted to move in a certain direction. Therefore, in practice, tanh nonlinearity is always preferred to sigmoid nonlinearity.

### **ReLU Function**

ReLU stands for Rectified Linear Unit. Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0.



ReLU Activation Function

Mathematically it can be represented as:

$$f(x) = \max(0, x)$$

The advantages of using ReLU as an activation function are as follows:

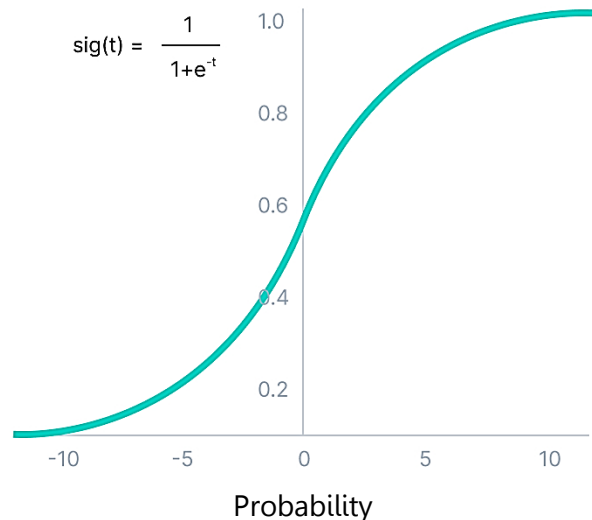
- Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

### **Softmax Function**

Before exploring the ins and outs of the Softmax activation function, we should focus on its building block—the sigmoid/logistic activation function that works on calculating probability values.



The output of the sigmoid function was in the range of 0 to 1, which can be thought of as probability. But — This function faces certain problems. Let's suppose we have five output values of 0.8, 0.9, 0.7, 0.8, and 0.6, respectively. How can we move forward with it? The answer is: We can't. The above values don't make sense as the sum of all the classes/output probabilities should be equal to 1. You see, the Softmax function is described as a combination of multiple sigmoids. It calculates the relative probabilities. Similar to the sigmoid/logistic activation function, the SoftMax function returns the probability of each class. It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification. Mathematically it can be represented as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Softmax Function

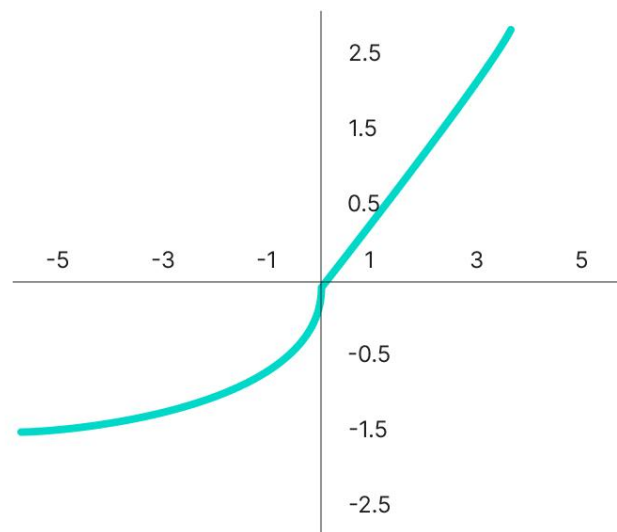
Let's go over a simple example together.

Assume that you have three classes, meaning that there would be three neurons in the output layer. Now, suppose that your output from the neurons is [1.8, 0.9, 0.68]. Applying the softmax function over these values to give a probabilistic view will result in the following outcome: [0.58, 0.23, 0.19]. The function returns 1 for the largest probability index while it returns 0 for the other two array indexes. Here, giving full weight to index 0 and no weight to index 1 and index 2. So the output would be the class corresponding to the 1st

neuron(index 0) out of three. You can see now how softmax activation function make things easy for multi-class classification problems.

### **Scaled Exponential Linear Unit (SELU)**

SELU was defined in self-normalizing networks and takes care of internal normalization which means each layer preserves the mean and variance from the previous layers. SELU enables this normalization by adjusting the mean and variance. SELU has both positive and negative values to shift the mean, which was impossible for ReLU activation function as it cannot output negative values. Gradients can be used to adjust the variance. The activation function needs a region with a gradient larger than one to increase it.



SELU Activation Function

Mathematically it can be represented as:

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

SELU has values of alpha  $\alpha$  and lambda  $\lambda$  predefined.

Here's the main advantage of SELU over ReLU: Internal normalization is faster than external normalization, which means the network converges faster. SELU is a relatively newer activation function and needs more papers on architectures such as CNNs and RNNs, where it is comparatively explored.

### **33. Why are deep neural networks hard to train?**

There are two challenges you might encounter when training your deep neural networks.

Let's discuss them in more detail.

**Vanishing Gradients:** Like the sigmoid function, certain activation functions squish an ample input space into a small output space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small. For shallow networks with only a few layers that use these



activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.

**Exploding Gradients:** Exploding gradients are problems where significant error gradients accumulate and result in very large updates to neural network model weights during training. An unstable network can result when there are exploding gradients, and the learning cannot be completed. The values of the weights can also become so large as to overflow and result in something called NaN values.

### 34. How to choose the right Activation Function?

You need to match your activation function for your output layer based on the type of prediction problem that you are solving—specifically, the type of predicted variable.

Here's what you should keep in mind.

As a rule of thumb, you can begin with using the ReLU activation function and then move over to other activation functions if ReLU doesn't provide optimum results.

And here are a few other guidelines to help you out.

1. ReLU activation function should only be used in the hidden layers.
2. Sigmoid/Logistic and Tanh functions should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).
3. Swish function is used in neural networks having a depth greater than 40 layers.

Finally, a few rules for choosing the activation function for your output layer based on the type of prediction problem that you are solving:

1. **Regression** - Linear Activation Function
2. **Binary Classification**—Sigmoid/Logistic Activation Function
3. **Multiclass Classification**—Softmax
4. **Multilabel Classification**—Sigmoid

The activation function used in hidden layers is typically chosen based on the type of neural network architecture.

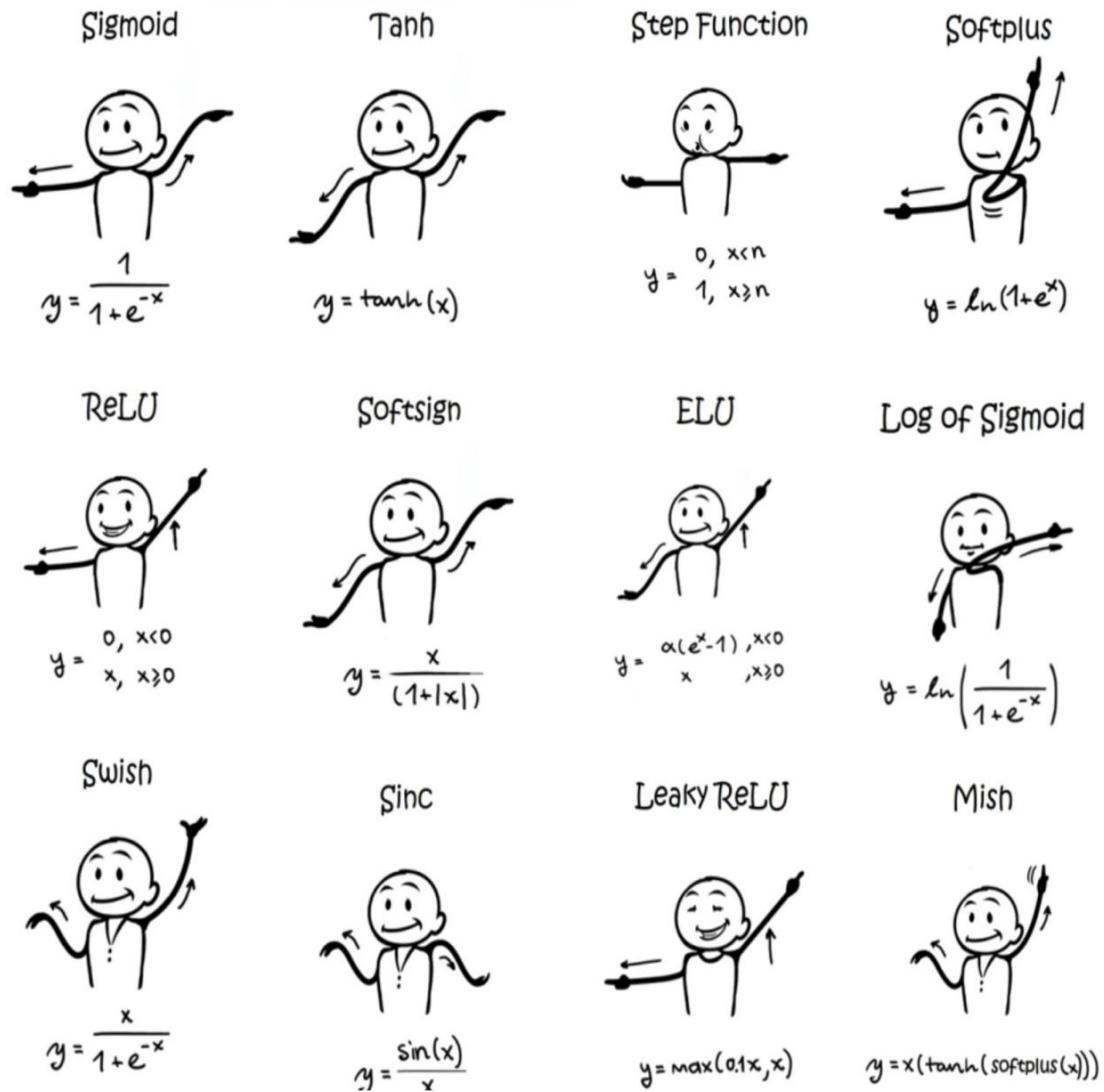
5. **Convolutional Neural Network (CNN):** ReLU activation function.
6. **Recurrent Neural Network:** Tanh and/or Sigmoid activation function.

Summary:

- Activation Functions are used to introduce non-linearity in the network.
- A neural network will almost always have the same activation function in all hidden layers. This activation function should be differentiable so that the parameters of the network are learned in backpropagation.
- ReLU is the most commonly used activation function for hidden layers.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- While selecting an activation function, you must consider the problems it might face: vanishing and exploding gradients.
- Regarding the output layer, we must always consider the expected value range of the predictions. If it can be any numeric value (as in case of the regression problem) you can use the linear activation function or ReLU.
- Use Softmax or Sigmoid function for the classification problems.



### 35. How does deep learning attain such impressive results?

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images. While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

1. Deep learning requires large amounts of **labeled data**. For example, driverless car development requires millions of images and thousands of hours of video.
2. Deep learning requires substantial **computing power**. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

### 36. State examples of Deep Learning.

Deep learning applications are used in industries from automated driving to medical devices.

**Automated Driving:** Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

**Aerospace and Defense:** Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

**Medical Research:** Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

**Industrial Automation:** Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

**Electronics:** Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

### 37. What's the Difference Between Machine Learning and Deep Learning?

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs “end-to-end learning” – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically. Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases. In machine learning, you manually choose features and a classifier to sort images. With deep learning, feature extraction and modeling steps are automatic.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

<https://in.mathworks.com/videos/ai-for-engineers-building-an-ai-system-1603356830725.html>

<https://in.mathworks.com/videos/object-recognition-deep-learning-and-machine-learning-for-computer-vision-121144.html>

<https://in.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>

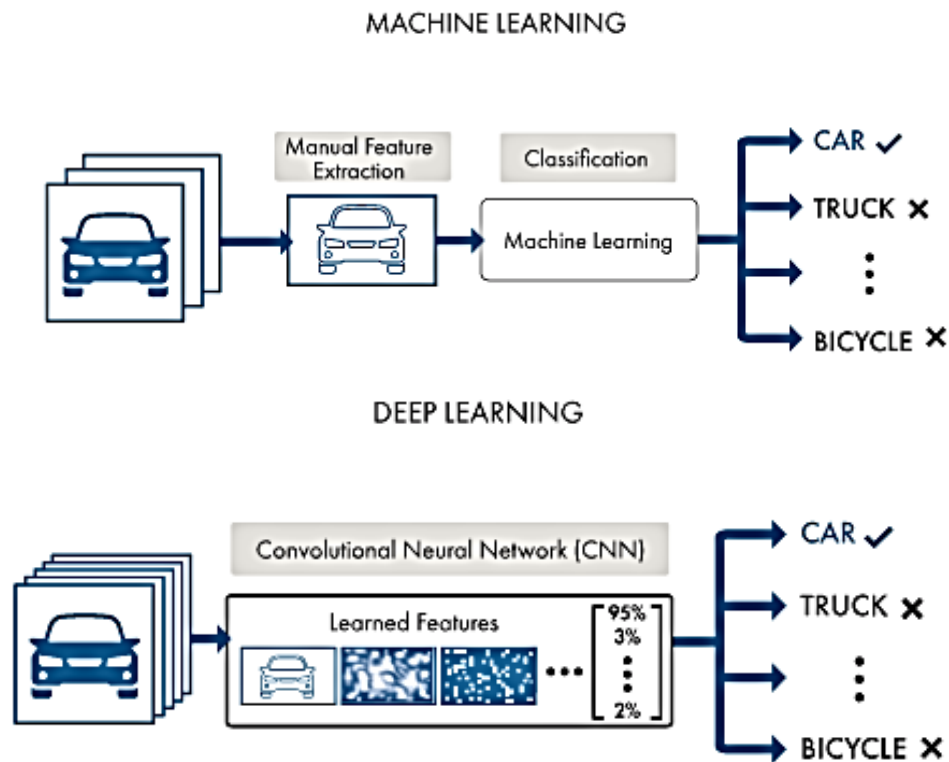


Figure: Comparing a machine learning approach to categorizing vehicles (left) with deep learning (right).

### 38. Choosing Between Machine Learning and Deep Learning

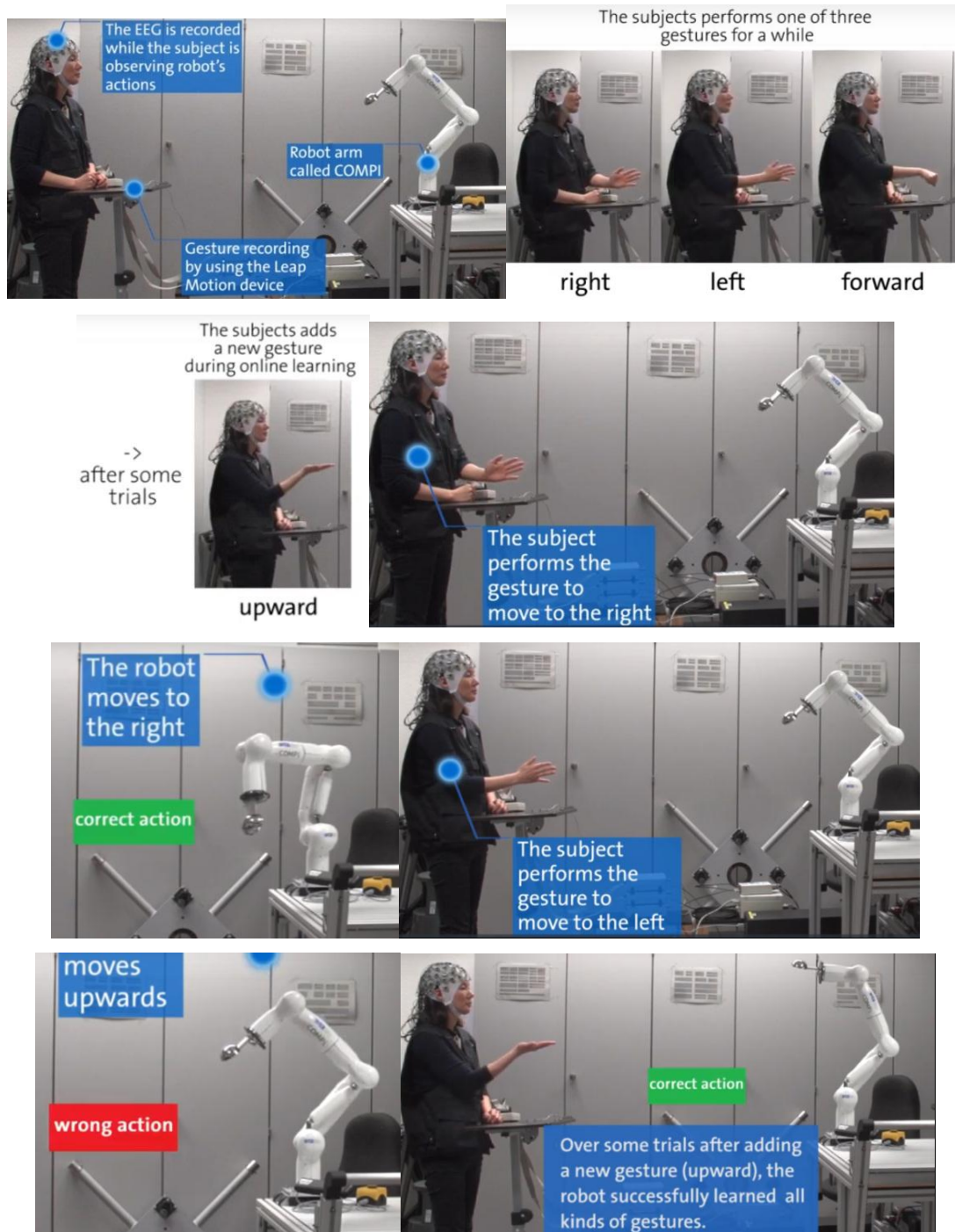
- Machine learning offers a variety of techniques and models you can choose based on your application, the size of data you're processing, and the type of problem you want to solve.
- A successful deep learning application requires a very large amount of data (thousands of images) to train the model, as well as GPUs, or graphics processing units to rapidly process your data.
- When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labeled data.
- If you don't have either of those things, it may make more sense to use machine learning instead of deep learning.
- Deep learning is generally more complex, so you'll need at least a few thousand images to get reliable results.

## QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING

- Having a high-performance GPU means the model will take less time to analyze all those images.

### 39. Application oriented questions

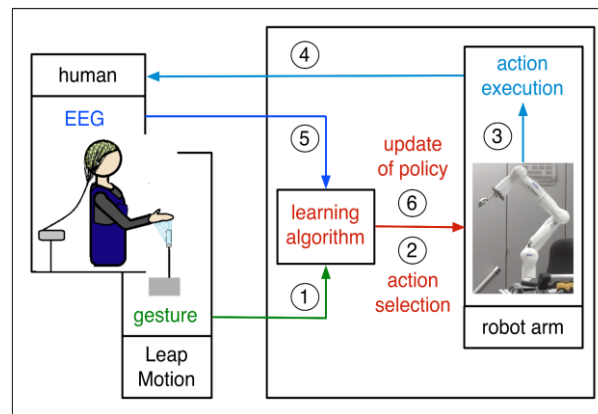
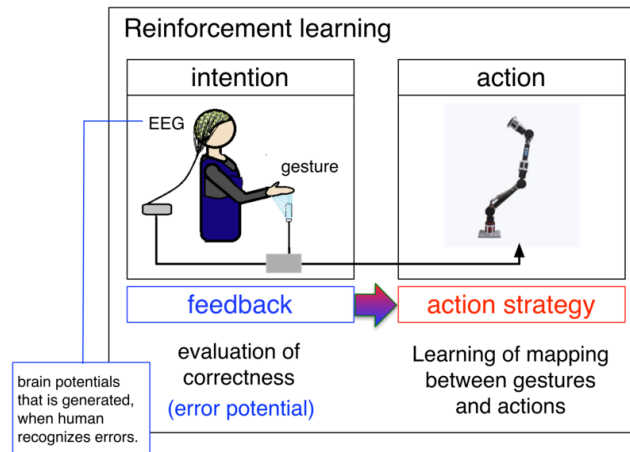
Explain role of reinforcement learning in following example. Identify environment, agent, different actions, reward, punishment etc. Draw its block diagram.



The reinforcement learning provides the means for robots to learn complex behavior from interaction on the basis of generalizable behavioural primitives. From the human negative feedback, the robot learns from its own misconduct.



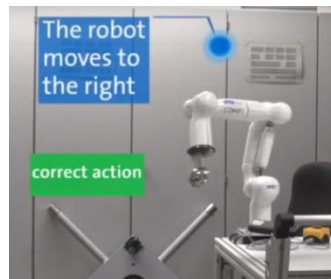
# QUESTION BANK FOR UNIT 5: REINFORCED AND DEEP LEARNING



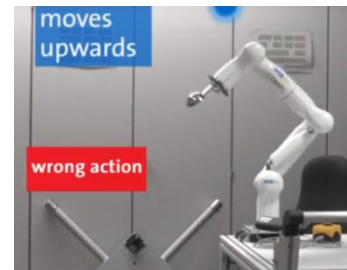
Agent



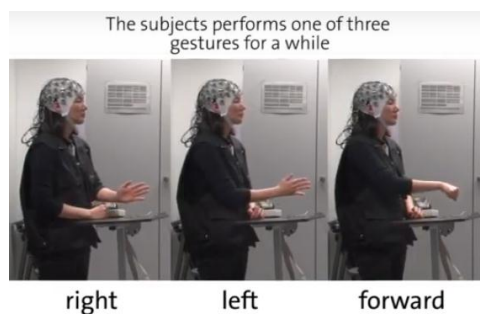
Reward



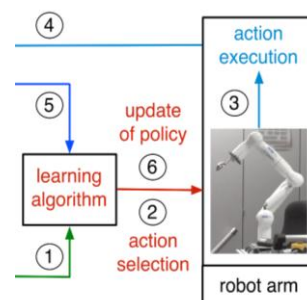
Punishment



Different actions



Environment



Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp,  
email ID: [adp.mech@coep.ac.in](mailto:adp.mech@coep.ac.in) and [abhipatange93@gmail.com](mailto:abhipatange93@gmail.com)

\*\*\*\*\*

Dr. Abhishek D. Patange, Mechanical Engineering, College of Engineering Pune (COEP)