

Q2) Explain working of a Generative Adversarial Network (GAN) with neat diagram. Your answer should include:

- Generator & Discriminator roles
- Training process
- Types of GAN (any two)
- Application: Real vs Fake Image Detection.

→ A Generative Adversarial Network (GAN) consists of two Neural Networks competing with each other.

1) Generator (G)

- takes random noise  $z$  as input
- Generates fake data (images, etc)
- Goal: Fool the discriminator into thinking fake data is real.

2) Discriminator (D)

- takes real + Fake data as input
- Outputs probability (real or fake)
- Goal: Correctly distinguish real vs fake

\* Training Process.

1) Generator creates fake data

2) Discriminator evaluates

- Real Data  $\rightarrow$  label = 1

- Fake data  $\rightarrow$  label = 0

3) Discriminator updates weights to improve classification

4) Generator updates weights based on discriminator feedback

5) Process repeats until generator produces real data.

\* Objective Function.

$$\min_G \max_D V(D, G) = \mathbb{E}[\log D(x)] + \mathbb{E}[\log (1 - D(G(z)))]$$



# SPPU-BE-COMP-CONTENT - KSKA Git

## \* Types of GAN

### ① DCGAN (Deep Convolutional GAN)

- uses convolutional layers
- looks well for image generation

### ② Conditional GAN (cGAN)

- Adds condition to generator

## \* Application.

Real vs fake image detection

- Discriminator acts as binary classifier.

used in

DeepFake detection

Fake profile image detection

Fraud detection systems

Q2) Explain Reinforcement Learning Framework with Markov Decision Process (MDP), Also answer.

Q-Learning algo with formula

Difference between Q-Learning & Deep Q-Networks.

→ Reinforcement Learning involves an agent interacting with an environment

## \* Components.

Agent → learner / Decision maker

Environment → where agent acts

State (S) → Current situation

Action (A) → Decision taken

Reward (R) → Feedback signal.



## \* Markov Decision Process (MDP)

MDP is defined as  $(S, A, P, R, \gamma)$ 

- $S$  = States
- $A$  = Actions
- $P$  = transition probability
- $R$  = Reward
- $\gamma$  (gamma) = Discount factor.

→ Next state depends only on current state. (Markov property)

## \* Q-Learning Algorithm.

- Q-Learning learns a Q-value. (state action value)

$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

$\alpha$  = Learning rate.

$\gamma$  = Discount factor.

$r$  = Reward

$S'$  = Next state.

## \* Difference.

Q-Learning vs ~~QDN~~ DQN

Feature.	Q-Learning	DQN
Representation	Q-table.	Neural Network
State space.	Small	Large / complex.
Memory	table based.	Experience replay
Scalability	Poor	High
example.	Tic-Tac-Toe.	Atari games.



\* Reward system for tic-tac-toe Agent

Designed reward badly = useless agent.  
So keep it simple & logical.

Suggested Reward System

- Win  $\rightarrow +10$
- Lose  $\rightarrow -10$
- Draw  $\rightarrow +5$
- valid move  $\rightarrow 0$
- invalid move  $\rightarrow -5$