

R. N. G. Patel Institute of Technology
Department of Computer Science & Engineering

Study Material

Subject: Special topics in Artificial Intelligence

Unit – 2

➤ **Self-Play Networks**

Que-1: What is a self-play network, and how does it work?(3)

- A **self-play network** is a type of machine learning system in which an AI model improves itself by playing against its own past versions or copies. This approach is widely used in reinforcement learning, particularly in training game-playing agents and decision-making systems.
- Self-play is a method in reinforcement learning where an agent plays against itself to improve its decision-making skills. This technique is particularly useful in environments like games, where the agent continuously competes with itself to refine its strategies. By playing against itself, the agent generates a wide variety of experiences, which enhances its learning without requiring external opponents. Notable examples include AlphaZero in chess and Go, where the agent plays millions of games against itself to improve its performance.

How It Works

1. **Initial Setup:** The agent begins with a random policy, meaning it makes decisions randomly at first.
2. **Playing Against Itself:** The agent plays games against copies of itself. This ensures that the opponent is always at a similar skill level, providing a balanced challenge.
3. **Policy Update:** After each game, the agent updates its policy based on the game's outcome. This can be done using various methods, such as temporal difference learning or policy gradient.
4. **Iterative Improvement:** The process is repeated many times, with the agent's policy gradually improving. As the agent gets better, the difficulty of the task it faces also increases, providing a natural curriculum of increasingly challenging tasks

Que-2: Explain various self-play network techniques in detail. (7)

Self-play networks are a powerful approach in reinforcement learning (RL), where an AI agent learns by competing against itself, leading to continuous improvement. There are several techniques used in self-play networks, each with different strategies for balancing exploration, exploitation, and policy optimization. Below are the key self-play techniques explained in detail:

1. Direct Self-Play

Concept:

- The AI plays against its exact copy or past versions.
- It improves by identifying weaknesses in its own strategy and refining them.

How It Works:

1. The AI starts with an initial policy (random or heuristic-based).
2. It plays against itself, collecting data on successful and unsuccessful moves.
3. The reinforcement learning model updates based on rewards from these games.
4. This process repeats iteratively, refining the model with each self-play cycle.

Advantages:

- Simple and effective for structured environments like board games.
- No need for human-labeled data.

Disadvantages:

- Can lead to overfitting if the AI keeps reinforcing its own weaknesses.
- May get stuck in local optima (suboptimal strategies).

2. Population-Based Self-Play

Concept:

- Instead of playing against itself, the AI plays against a population of different versions of itself.
- Helps avoid stagnation by introducing diverse strategies.

How It Works:

1. Maintain a pool of AI agents at different stages of learning.
2. New models are periodically introduced into the population.
3. The AI competes against multiple opponents, ensuring diverse learning experiences.
4. The best-performing models are selected and evolved.

Advantages:

- Encourages diversity in strategies, avoiding local optima.
- More robust than direct self-play.

Disadvantages:

- Requires more computational resources due to maintaining multiple agents.
- More complex training setup.

3. Fictitious Self-Play (FSP)

Concept:

- Combines reinforcement learning with game theory.
- The AI learns a mixed strategy by considering past strategies instead of only the current best one.

How It Works:

1. The AI maintains a history of past strategies.
2. It updates its policy by playing against all previous versions, giving weight to different strategies.

3. The goal is to learn an optimal mixed strategy over time.

Advantages:

- Prevents the AI from overfitting to a single strategy.
- Useful in multi-agent systems where adversaries constantly adapt.

Disadvantages:

- Computationally expensive due to storing and evaluating multiple strategies.
- Harder to implement for large-scale environments.

4. AlphaZero-Style Self-Play (Monte Carlo Tree Search + Deep Learning)

Concept:

- Used in **AlphaGo**, **AlphaZero**, and **MuZero**.
- Integrates deep neural networks with **Monte Carlo Tree Search (MCTS)** for decision-making.

How It Works:

1. **Neural Network Policy:** A deep neural network predicts move probabilities and value estimates.
2. **Monte Carlo Tree Search (MCTS):** Explores future moves, balancing exploration vs. exploitation.
3. **Self-Play Games:** The model plays games against itself, generating training data.
4. **Reinforcement Learning Update:** The network is trained using the new data, replacing older policies.

Advantages:

- Achieves superhuman performance in games like Chess, Go, and Shogi.
- Efficient and scalable with deep learning.

Disadvantages:

- Requires significant computational power.
- Harder to generalize beyond structured environments like board games.

5. Curriculum Learning in Self-Play

Concept:

- Instead of learning from scratch, the AI gradually learns from easier to harder versions of self-play.

How It Works:

1. The AI starts with basic opponents (e.g., random or weak strategies).
2. As it improves, the difficulty of opponents increases (e.g., previous stronger versions of itself).
3. This controlled progression ensures stable learning.

Advantages:

- Prevents the AI from being overwhelmed by complex strategies too early.
- Helps in training agents in environments with increasing difficulty (e.g., robotics, autonomous driving).

Disadvantages:

- Requires careful tuning of difficulty levels.
- Can slow down the learning process if not well-balanced.

6. Exploitability Reduction via Nash Averaging

Concept:

- Uses **game-theoretic** techniques to reduce exploitability by maintaining a **Nash equilibrium**.
- The AI tries to minimize the advantage an opponent can take over it.

How It Works:

1. The AI keeps track of how exploitable its strategy is.
2. Adjustments are made to ensure it is harder to predict or counter.

3. This is particularly useful in zero-sum competitive environments.

Advantages:

- Leads to highly robust strategies that are difficult to exploit.
- Works well in adversarial environments like poker, military simulations, and economic models.

Disadvantages:

- Requires advanced game-theoretic modeling.
- Can be computationally expensive.

Comparison of Self-Play Techniques

Technique	Best For	Pros	Cons
Direct Self-Play	Structured games like chess, Go	Simple and effective	Risk of overfitting, local optima
Population-Based Self-Play	Dynamic environments (multi-agent RL)	Promotes diverse strategies	High computational cost
Fictitious Self-Play (FSP)	Multi-agent systems, game theory	Prevents overfitting, good for mixed strategies	Computationally expensive
AlphaZero-Style Self-Play	Games with clear rules (chess, Go)	Achieves superhuman performance	Requires heavy computation
Curriculum Learning	Tasks with increasing complexity	Ensures smooth learning progression	Needs careful difficulty tuning
Nash Averaging	Adversarial settings (poker, security)	Highly robust against exploitation	Requires advanced game-theoretic analysis

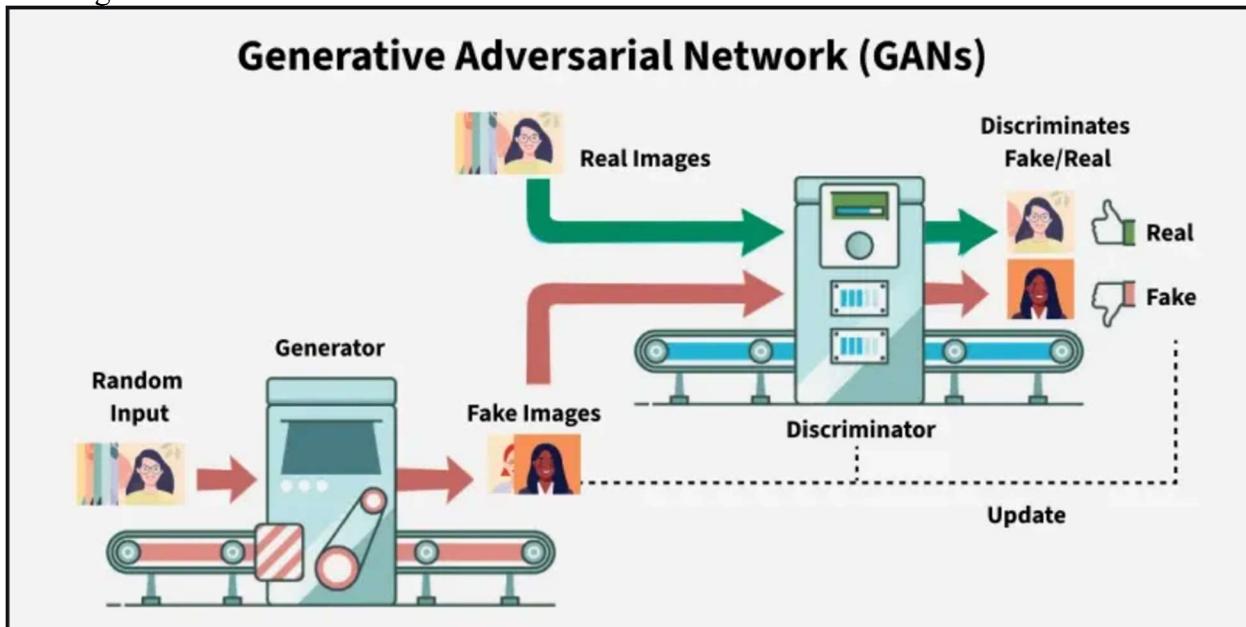
➤ Generative Adversarial Networks (GANs)

Que-3: What are some common applications of GANs in real-world scenarios? (4)

- **Image Generation:** GANs can generate realistic images from noise or based on certain conditions. For example, generating high-resolution images of faces or creating artwork.
- **Data Augmentation:** GANs can create synthetic data to augment datasets, improving the performance of machine learning models in tasks like image classification and object detection.
- **Super-Resolution:** GANs can enhance the resolution of images, converting low-resolution images into high-resolution ones, which is useful in medical imaging and satellite imagery.
- **Style Transfer:** GANs can transfer the style of one image to another, such as converting a photo into a painting in the style of a famous artist.
- **Deepfake Technology** – GANs are used to create realistic fake videos, replacing faces in videos or synthesizing speech for entertainment and security applications.

Que-4: Describe the architecture of Generative Adversarial Networks (GANs) and explain how the generator and discriminator interact during training.(7)

Generative Adversarial Networks (GANs) consist of two neural networks: a Generator and a Discriminator. These networks are trained simultaneously through a process of adversarial learning.



Architecture of GANs

1. Generator (G)

- **Objective:** To generate realistic data samples (e.g., images) from random noise.
- **Structure:** Typically consists of a series of upsampling layers (e.g., deconvolutional layers) that transform a low-dimensional noise vector into a high-dimensional data sample. The generator takes a random noise vector z as input and produces a data sample $G(z)$.

2. Discriminator (D)

- **Objective:** To distinguish between real data samples and fake samples generated by the generator.
- **Structure:** Typically consists of a series of downsampling layers (e.g., convolutional layers) that transform a high-dimensional data sample into a single scalar output. The discriminator takes a data sample x as input and outputs a probability $D(x)$ indicating whether the sample is real or fake.

Interaction During Training

1. **Initialization:** The generator and discriminator are initialized with random weights.
2. **Adversarial Training:**
 - **Step 1: Training the Discriminator:**
 - The discriminator is trained on a batch of real samples from the dataset, updating its weights to correctly classify these samples as real.
 - The discriminator is then trained on a batch of fake samples generated by the generator, updating its weights to correctly classify these samples as fake.
 - **Step 2: Training the Generator:**
 - The generator is trained by using the discriminator's feedback. The generator's weights are updated to maximize the probability that the discriminator classifies its generated samples as real.
 - This is done by minimizing the generator's loss, which measures how well the generator can fool the discriminator.
3. **Iterative Process:** These steps are repeated iteratively, with the generator and discriminator improving their abilities over time. The generator becomes better at producing realistic samples, while the discriminator becomes better at distinguishing real from fake samples.

Adversarial Loss Functions

Discriminator Loss

The discriminator's loss function measures its ability to correctly classify real and fake samples. It is typically defined as:

$$L_D = -\frac{1}{m} \sum_{i=1}^m [\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))]$$

Where:

- $D(x^{(i)})$ is the discriminator's output for real sample $x^{(i)}$.
- $D(G(z^{(i)}))$ is the discriminator's output for generated sample $G(z^{(i)})$.
- m is the batch size.

Generator Loss

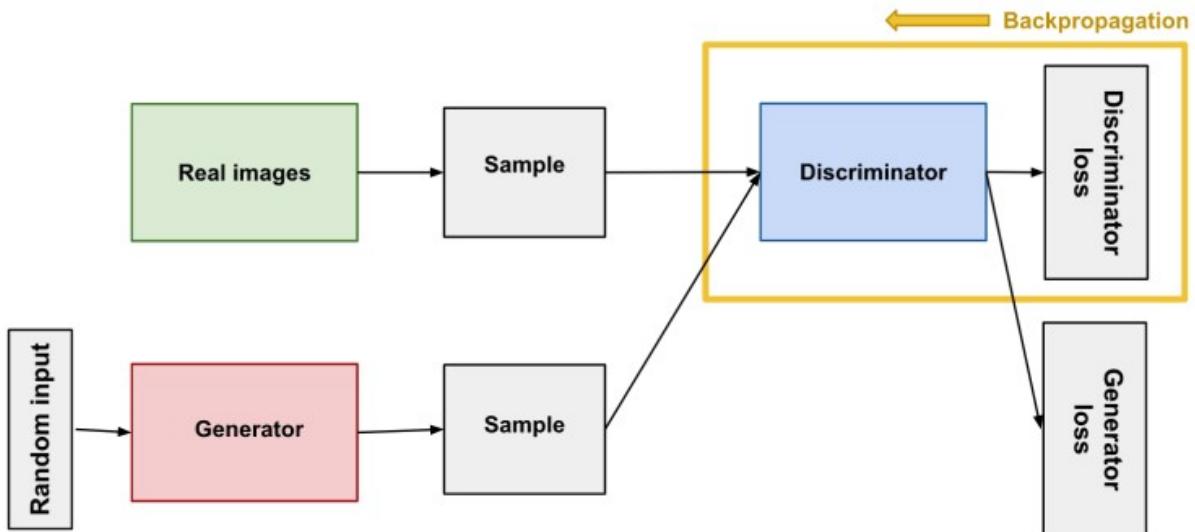
The generator's loss function measures its ability to fool the discriminator into classifying its generated samples as real. It is typically defined as:

$$L_G = -\frac{1}{m} \sum_{i=1}^m \log(D(G(z^{(i)})))$$

Where:

- $D(G(z^{(i)}))$ is the discriminator's output for generated sample $G(z^{(i)})$
- m is the batch size.

These loss functions ensure that the generator and discriminator are trained adversarially , with the generator improving its ability to generate realistic samples and the discriminator improving its ability to distinguish real from fake.



Que-5: Write and explain the loss function for a GAN model (4)

GANs follow a minimax optimization where the generator and discriminator are adversaries:

$$\min_G \max_D(G, D) = [\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(g(z)))]]$$

Where,

- G is generator network and D is the discriminator network
- Actual data samples obtained from the true data distribution $p_{\text{data}}(x)$ are represented by x.
- Random noise sampled from a previous distribution $p_z(z)$ (usually a normal or uniform distribution) is represented by z.
- $D(x)$ represents the discriminator's likelihood of correctly identifying actual data as real.
- $D(G(z))$ is the likelihood that the discriminator will identify generated data coming from the generator as authentic.

The generator aims to **minimize** the loss, while the discriminator tries to **maximize** its classification accuracy.

➤ Concept-Drifting Data Streams

Que-6: Discuss different types of concept drift. (7)

Concept drift occurs when the statistical properties of the target variable change over time, leading to a decrease in the performance of machine learning models. It is common in dynamic environments such as finance, healthcare, and fraud detection. There are different types of concept drift, categorized based on how the data distribution changes.

1. Sudden (Abrupt) Drift

Definition: The concept changes abruptly at a specific point in time.

Example: A sudden economic crisis drastically alters consumer spending behavior overnight.

Impact: Model performance drops instantly, requiring rapid adaptation.

Solution:

- Use adaptive models (e.g., online learning).
- Implement **change detection algorithms** to trigger model retraining.

2. Incremental Drift

Definition: The concept changes gradually over time.

Example: A slow shift in customer preferences, such as a gradual decline in landline phone usage.

Impact: Models trained on old data become less effective over time.

Solution:

- Regularly retrain models with recent data.
- Implement **sliding window approaches** to include newer patterns.

3. Gradual (Blended) Drift

Definition: The old and new concepts coexist for some time before the shift is complete.

Example: Seasonal changes in shopping trends where summer products fade out as winter approaches.

Impact: Performance may fluctuate as the model encounters mixed distributions.

Solution:

- Use **ensemble learning** to balance both concepts.
- Maintain multiple models for different phases of the transition.

4. Recurring (Seasonal) Drift

Definition: A previously seen concept reappears periodically.

Example: Online traffic patterns change during holidays and return to normal afterward.

Impact: The model might struggle if it assumes a consistent distribution.

Solution:

- Store historical patterns and train models to detect seasonal trends.
- Implement **context-aware learning models** to anticipate recurrences.

5. Feature (Covariate) Drift

Definition: The distribution of input features changes, even if the relationship between features and labels remains stable.

Example: A change in customer demographics alters feature distributions in a recommendation system.

Impact: Models trained on past feature distributions may make incorrect predictions.

Solution:

- Monitor feature distributions with statistical tests.
- Apply **domain adaptation techniques** to adjust model parameters.

6. Real Concept Drift

Definition: The actual relationship between input features and target labels changes.

Example: A fraud detection model trained on past fraud patterns becomes ineffective as fraudsters adopt new strategies.

Impact: The model needs a fundamental update since the relationship between features and outcomes has changed.

Solution:

- Continuously update training data.
- Use adaptive learning techniques like **online learning** or **reinforcement learning**.

Comparison of Concept Drift Types

Type	Speed of Change	Example	Solution
Sudden Drift	Instant	Policy changes in financial markets	Change detection & retraining
Incremental Drift	Slow & continuous	Evolving consumer preferences	Sliding window approaches
Gradual Drift	Temporary overlap	Seasonal product transitions	Ensemble learning
Recurring Drift	Periodic reappearance	Holiday shopping trends	Context-aware learning
Feature Drift	Input feature shift	Changing customer demographics	Feature monitoring & adaptation
Real Concept Drift	Feature-label relationship change	New fraud techniques	Online learning & retraining

Que-7: Explain the general framework for concept drift detection (7)

General Framework for Concept Drift Detection

Concept drift detection involves monitoring data streams and identifying changes in data distributions that may affect a machine learning model's performance. A general framework for concept drift detection consists of the following key steps:

1. Data Collection & Preprocessing

Objective: Gather and prepare incoming data for analysis.

Process:

- Continuously collect new data from real-world environments.
- Preprocess data (handling missing values, normalization, etc.).
- Track distributions of key features and labels over time.

2. Drift Detection Methods

Objective: Identify whether a drift has occurred using statistical, model-based, or data-driven techniques.

A. Statistical Techniques

- **Kolmogorov-Smirnov (KS) Test** – Measures changes in data distributions.
- **Kullback-Leibler (KL) Divergence** – Detects shifts in probability distributions.
- **Chi-Square Test** – Compares categorical feature distributions.

B. Error Rate Monitoring

- Track the **model's accuracy or error rate** over time.
- If accuracy suddenly drops, it indicates possible drift.

C. Data Distribution Comparison

- Compare old vs. new feature distributions using distance metrics (e.g., Wasserstein distance).

D. Drift Detection Algorithms

- **Page-Hinkley Test** – Monitors mean changes in a data stream.
- **DDM (Drift Detection Method)** – Triggers drift alerts when the error rate increases significantly.
- **ADWIN (Adaptive Windowing)** – Dynamically adjusts window size to detect drifts.

3. Drift Classification

Objective: Determine the **type of drift** (e.g., sudden, incremental, recurring).

Process:

- Analyze the rate and magnitude of detected changes.
- Classify the drift based on predefined thresholds.

4. Adaptation & Model Update

Objective: Take action based on detected drift.

Strategies:

- **Retrain the model** with recent data.
- **Use ensemble learning** to balance old and new patterns.
- **Apply online learning** to dynamically adjust model weights.

5. Continuous Monitoring & Feedback Loop

Objective: Ensure ongoing model stability and prevent performance degradation.

Process:

- Set up automated drift alerts.
- Periodically analyze detection results.
- Fine-tune detection thresholds to improve accuracy.

Que-8: What is the difference between real concept drift and virtual concept drift

Aspect	Real Concept Drift	Virtual Concept Drift
Definition	Changes in the relationship between input features and the target variable.	Changes in the distribution of input features without affecting the target variable.
Impact on Model	Requires retraining because the decision boundary shifts.	May not always require retraining, but monitoring is needed.
Example	A fraud detection model becomes ineffective as fraudsters adopt new tactics.	A shift in customer demographics without affecting overall purchasing behavior.
Causes	Changes in business rules, market conditions, fraud techniques, or user behavior.	Data preprocessing issues, sensor calibration changes, seasonal fluctuations.
Detection	Requires tracking both feature and label changes over time.	Can be detected by monitoring feature distributions without label changes.
Solution	Adaptive learning, online model updating, retraining with new data.	Feature engineering, data normalization, and domain adaptation.