UTV
RNN

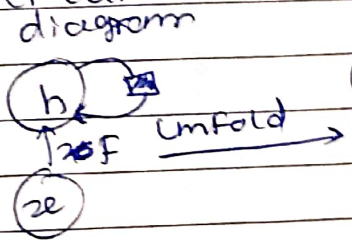
* Unfolding Computational graph -

1) Computational graph - A Computational graph is a graphical representaⁿ of a series of maths operaⁿ or Computaⁿ. It's commonly used in deep learning Framework like tensorflow & pytorch to represent and optimize complex math expression.

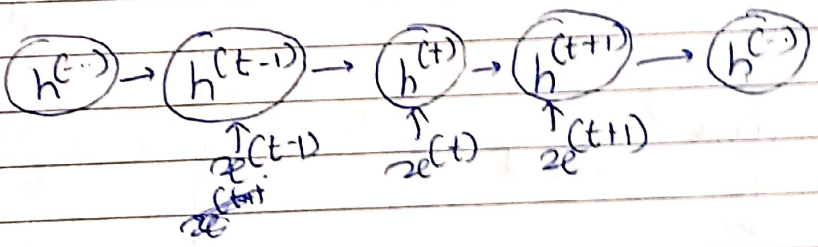
2) Unfolding Computational graph refers to expand it over time or across iteraⁿ. The graph is unrolled to visualize the sequence of operaⁿ across iteration.

) Example - Unfolding RNN -

Circuit diagram



Unfolded computaⁿ graph.



Equation $h^{(t)} = F(h^{(t-1)}, xe^{(t)}; O)$

We can represent unfolded reccurrence after t steps with a function $g^{(t)}$:

$$h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, \ldots, xe^{(2)}, xe^{(1)})$$
$$= F(h^{(t)}, xe^{(t)}; O)$$

Funcⁿ $g^{(t)}$ takes sequence of past xe inputs $(xe^{(t)}, xe^{(t-1)} \ldots xe^{(2)} xe^{(1)})$ as input & produces the current state, but the unfolded recurrent structure allows us to factorize $g^{(t)}$ into repeated applicaⁿ of a function F.

3) Advantages - 1) Unfolding enables the use of same function at every step of the sequence.

2) Unfolding ensures that regardless of sequence length learned model has consistent input size.

3) Single model 'F' can operate on all time steps and sequence lengths.

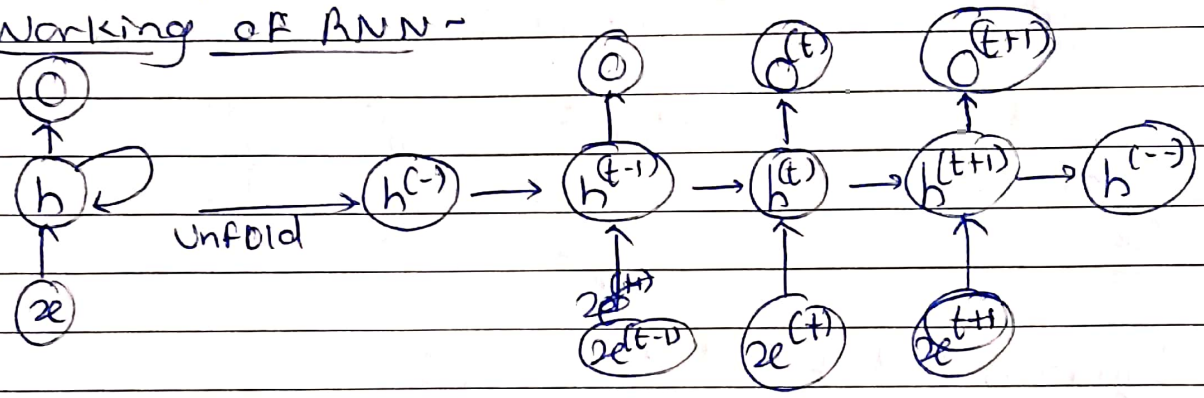
Teacher's Sign.:

✱ RNN (Recurrent Neural Network) -

RNN is a type of Artificial Neural Network that processes sequential data input and converts it into sequential data output. RNNs have internal memory in hidden layers which remembers previous state and make accurate predictions.

> Need for RNN - Traditional Feedforward network processes each input independently lacking the ability to capture dependencies or patterns in longer sequences RNNs address these limitation by introducing connection to the previous states to retain information from previous inputs & context.

> Working of RNN -



1) Input - at each time step t, the RNN recieves an input $x^{(t)}$, which could be single element of sequence (eg. a word, ~~a pixel etc~~).

2) Hidden state - RNN maintains a hidden state $h^{(t)}$ that catures informa" from previous time steps. This hidden state acts as memory of network, allowing to retain info from previous time step.

3) Output - Based on the current input $x^{(t)}$ and the previous hidden state ~~$h^{(t)}$~~ $h^{(t-1)}$ the RNN computes the current hidden state $h^{(t)}$ and produces output ~~$o^{(t)}$~~ $O^{(t)}$ at each time step.

eqn - $h^{(t)} = F(W_{hx} \cdot x^{(t)} + W_{hh} \cdot h^{(t-1)} + b_h)$

$O^{(t)} = g(W_{yh} \cdot h^{(t)} + b_y)$.

Where, W_{hx}, W_{yh}, W_{hh} are weights, b_h & b_y are biases.

Teacher's Sign.: _____

LSTM (Long Short Term Memory)

LSTM networks are a type of RNNs designed to address vanishing gradient problem and capture long term dependencies in sequential data

1) LSTM is the memory cell which is capable of storing info for long duration. This helps to capture information over long sequences.
2) LSTM incorporates 3 Types of gates -
   1) Forget gate - Determines which info from previous cell state should be discarded.
   2) Input gate - which new info should be stored
   3) Output gate - Chooses which parts of the cell should be used to compute output.

4) Working of LSTM -
   1) At each step, the LSTM cell recieves input x_t & previous hidden state h_{t-1}
   2) Forget gate determines which info from previous cell state C_{t-1} should be discarded.
   3) Input gate decides which new info should be stored.
   4) The output gate determines the new hidden state (h_t) based on current cell state (C_t) and previous hidden state h_{t-1}
   5) The output layer produces an output based on h_t, reflecting sequence's context up to current time step.
5) Iteration - Steps 1 to 5 are repeated for each element in the sequence

BiLSTM (Bidirectional LSTM).

Extension to LSTM, enhancing capability of LSTM in both direction of input sequence. This allows the model to capture info from past & future.

~~Working~~ - Architecture -

1) Dual LSTM -
   1) Forward - Left to right
   2) backward - Right to Left.
2) Independent processing - Each LSTM layer operates independently using internal gates to manage info flow & capture long-dependencies.
3) Merging the knowledge -

Working -

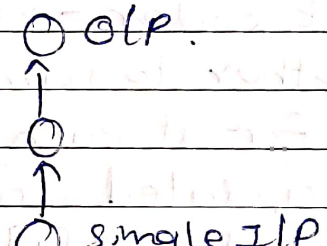
1) BiLSTM receives an input sequence $X = \{x_1, x_2, \ldots, x_T\}$ where T is the length of sequence.
2) Forward pass - input sequence processed from beginning to end. computes hidden state based on current i/p
3) Backward pass - input sequence is processed from the end to the beginning.
4) Combining hidden states - Once forward & backward pass is completed, the hidden states from both directions are combined to capture bidirectional context.
   $h_t = [h_t^{(F)}; h_t^{(b)}]$. $[;]$ denotes concatenation.
5) Combined hidden state h_t represent output at the BiLSTM at each time step. These hidden state can be used for various task such as sequence generation.

A How to Train RNN -

1) Prepare data
2) Design model - choose RNN architecture
3) Initialize parameters. - initialize wieghts & biases
4) Forward pass - Input data sequentially, compute hidden state & output
5) ~~Compute loss -~~
6) Backpropagaⁿ -
5) Train - use Backpropagaⁿ Through Time (BPTT) with gradient clipping
6) Compute loss
7) Backpropagaⁿ to update weights.
8) Evaluate.

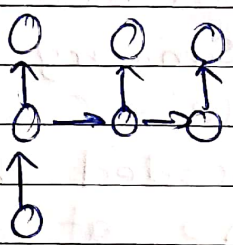
Types of RNN -

1) One to one RNN -



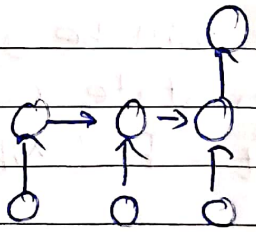
This type of neural net is understood bec oz the vanilla Neural Net used for general machine learning problem

2) One to Many RNN -



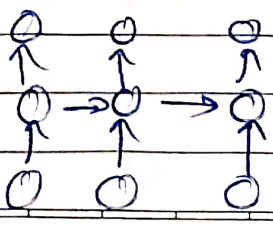
Used in ~~caption~~ image caption generaⁿ.

3) Many to one RNN -



example - Sentiment analysis

4) Many to Many



Ex - Machine Translation.

Teacher's Sign.: _____

Variants of RNN -

1) Basic RNN (Vanilla RNN)

2) LSTM

3) GRU - has gating mechanism, combining the forget gate & input gates into single update gate.

4) Bidirectional RNN -

5) Deep RNN.

Encoder - Decoder Seq 2 Seq Architecture -

1) Encoder -> Multiple RNN cells can be stacked together to form encoder. RNN reads sequentially.
   > For at each time step (input) t, hidden state h is updated according to the input at that timestep X[i].
   > After all time step inputs are ready by encoder model, final hidden state of model represents context / summary of whole input sequence.
   > Example - Consider input sentence "Cat sat on mat" to be encoded There will be 5 timestamps - 4 timestamp, at each time stamp step, the hidden state 'h' will be updated using previous hidden state.
   > hidden state computed using eqn.

$$h_t = f(W_{hh} h_{t-1} + W_{hx} x_t).$$

2) Encoder vector -
> Final hidden state produced by encoder RNN it is a compressed representation that captures the meaning & context of entire input sequence.
> Provides essential info for the decoder to generate the output sequence accurately. Like a starting point for the decoder.

3) Decoder -
> Another RNN that takes context vector from the encoder as its initial state.
> Generates the output sequence one element at a time.
> Predicts the next element in the output sequence based on context vector and previously generated outputs.
> We use softmax activation function at the output layer used to produce probability distribution from a vector of values.

Re RNN VS Recursive NN.

RNN	~~RNN~~ RNN
1) Networks having hierarchical Structure	chain like structure
2) Processes hierarchical data	known as sequential structure Sequential data
3) Limited context handeling	Captures context through Sequential memory
4) Network requires specific Tree traversal algorithm for training	Involves training backpropagation through time
5) ~~Vanishing~~ gradient problem	~~Computational complex~~.
5) ~~Internal~~ Computationally complex.	~~Int~~ vanishing gradient.
6) Internal memory	Limited memory.

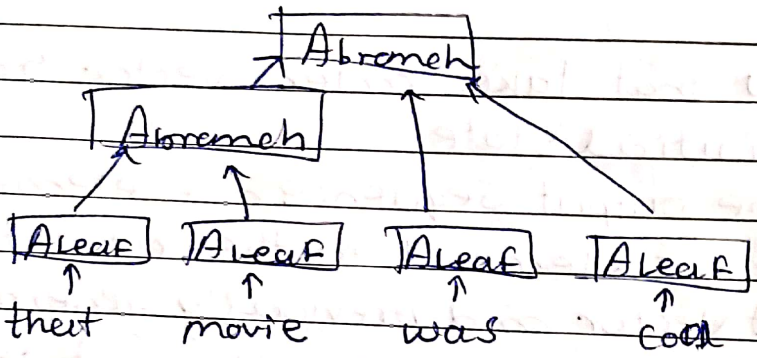
Teacher's Sign.: _____

✱ Recursive Neural Network (RvNN) -

RvNN are specialized type of deep learning architecture designed to handle data with hierarchical structures. RvNN focuses on understanding nested relationships within data.

Characteristics -

1) Hierarchical data - RvNN are built to process data organized in a tree-like fashion



2) Recursive processing - They operate by recursively breaking down the data structure. They process smaller parts (leaf nodes) and then combine result as they ascend the hierarchy.
They recursively apply some neural network operation at each level of the tree.

3) Examples - Include parsing, sentiment analysis hierarchical document classification.

Optimizer for long term dependencies -

> Long term dependencies pose a significant challenge in training RNN. The vanishing gradient problem can hinder the network's ability to learn relationships between elements.

Here are some optimizing techniques

> Gradient clipping - During backpropagation, gradients can become very large or very small, leading to unstable training. Gradient Clipping prevents these extremes by setting max threshold for of gradient.

> Truncated Backpropagation Through Time (BPTT) - Limits the number of time steps used for propagation reducing the impact of vanishing gradients but potentially sacrificing accuracy.

Explicit Memory - Introducing seperate memory component to store info for extended periods

> External Memory Network - Introduce an external memory ~~usage~~ storage unit alongside the RNN. The network can access this memory using read & write operation

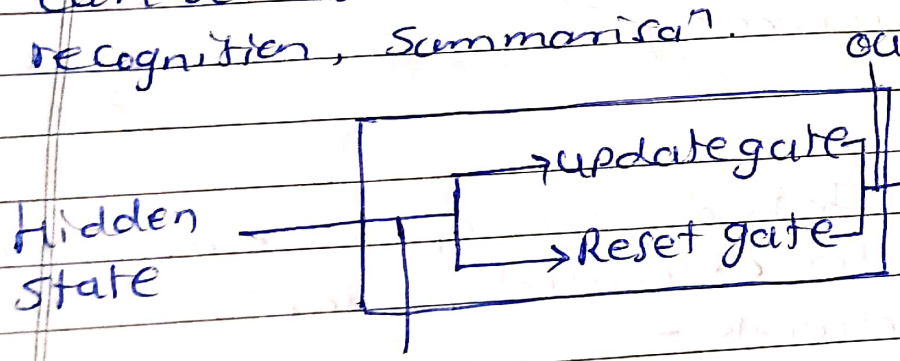
> Neural Turing Machines - A specialized type of external memory network with multiple read/write heads for accessing different parts of the memory

> Differentiable Neural Dictionaries (DNDs) - Newer approach using dictionary as memory component. Stores key-value pairs allowing network to learn association betn inputs & corresponding o/p.

✗ Gated Recurrent UNIT (GRU) -

A type of RNN designed to address the vanishing gradient problem.

> Similar to LSTM but simpler with fewer gates.
  1) Reset gate - Decides how much of the previous hidden state to forget.
  2) Update gate - controls how much new information to incorporate from current input.
> Like LSTMs GRU are adept at processing sequential data such as text, speech, time series.
> They can learn long term dependencies within sequence.
> It has simpler structure faster to train can be used for tasks like machine translation, speech recognition, summarisation.

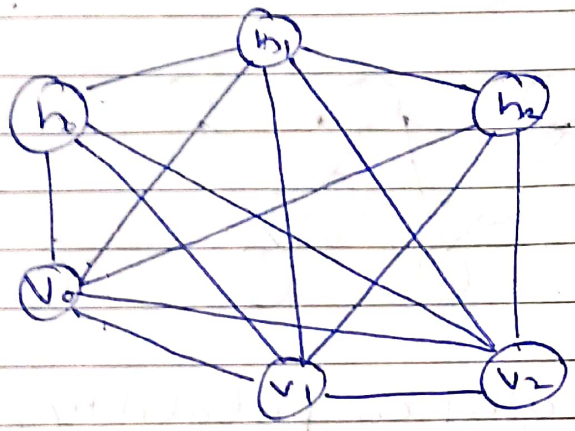


Unit V

Deep generative models

(BM)

Boltzman Machine — It is an unsupervised DL model in which every node is connected to every other node. Boltzman machines are undirected It is not a deterministic DL model but a stochastic or generative DL model. There are 2 Node Types of nodes visible nodes & Hidden nodes connected to each other.



V - Visible
H - Hidden.

> BM is an energy based model where each Configura" is associated with energy value. The probability distribu" over configura" is determined by using Boltzman distribu", which assigns higher probability value with lower energy.

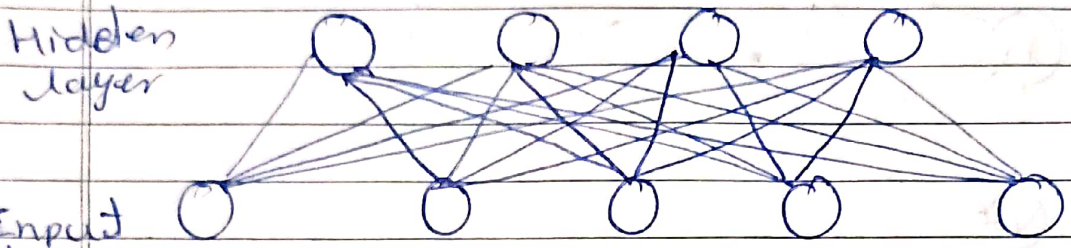
> Training Boltzman Machine involves adjusting weights between units to minimize energy of observed data configura" & increase the likelihood of generating similar data configura".

> Every node is conected to all other nodes irrespective of whether they are input or hidden nodes. This enables them to share info among themselves and self generate subsequent data. After input is provided BM is able to capture pattern and understand parameters and correlation among the data.

Teacher's Sign.:

* Types of Boltzman Machine -

1) Restricted Boltzman Machine. - Type of Stochastic generative model used for unsupervised learning.
2) RBMs consist of 2 layers input & Hidden layer Connection exist between every pair of visible & hidden units, but there are no connections within same layer. This restric^n simplifies model makes learning more tractable.



3) RBMs operate based on probability distribu^n. Each unit in hidden layer has a certain activa^n probability based on the state of visible layer units its connected to.
4) The probability distribu^n over configura^n is determined by the Boltzmann distribu^n which assigns higher probability to low or energy.
5) Training RBMs involves adjusting weights between visible and hidden units to minimize the energy of observed data configura^n and increase likelihood of generating similar data configura^n.
6) Applica^n -
   1) Feature extraction.
   2) Recommenda^n system.
   3) Pretraining of Deep networks

Teacher's Sign.:

★ Deep Belief Network (DBN)

> DBNs ~~const~~ consists of multiple layers of RBMs stacked on top of each other. ~~Each layer serves as hidden layer for the RBM above it.~~ Such that the outputs of the first RBM are the inputs of the subsequent RBM. Connection within individual layers are undirected, while connecⁿ between layers are directed.

> DBNs work in 2 phases - pretraining & Finetuning.

> Pretraining phase, the network learns to represent input data layer by layer, Each layer trained independently as an RBM which allows network to learn ~~data~~ probability distribuⁿ of the inputs.

> In Finetuning phase, the DBN adjusts its parameters for specific tasks. This is typically done using ~~a~~ backpropagaⁿ, errors are used to update network's parameters.

★ Generative Adverserial Network (GANs) -

GANs are a class of ML models that excel at generating new data. Unlike traditional model GAN consists of 2 Neural Net a generator & descriminator.

> During training, the generator & descriminator are trained simultaneously. The generator tries to generate realistic samples to fool descriminator, while descriminator tries to distinguish betⁿ read and fake samples.

> The generator creates new data samples. The discriminator recieves both ~~read~~ real data samples and generated one. It tries to classify them as real or fake

> Based on descriminator's feedback, the generator refines its approach to produce more realistic data in next ~~g~~ iteraⁿ.

Teacher's Sign.: _____

	Generative Model	Descriminator
1)	Generate fresh data & to understand the probability distribution of input.	Data classification into distinct categories ie real or fake.
2)	Large amount of data required to learn from data	Labeled input data are necessary in order to classify data.
3)	Generated data and and original data are similar	An output label from descriminative models identifies the kind of input.
4)	Generative models are complex to construct than discriminative models	Descriminative models are easier to create since they simply need to understand category.
5)	Usage of generative model is common in voice & picture recognition	In classification tasks like voice and picture recognition.
6)	Unlabeled data can be analyzed	Labeled data are necessary

* Deep Generative model -

> Deep generative models are powerful class of algos in DL that is Specialize in creating entirely new data.

> All types of generative models aim at learning the true data distribⁿ to generate new data with some variations.

> Two most commonly used efficient approaches are variational Autoencoder (VAE) and GANs. VAE aims at maximizing the lower bound of the data log. and GAN aims at achieving an equilibrium betⁿ Generator & descriminator.

> Autoregressive models works on conditional probability distribⁿ of each element in data sequence. They generated data sequentially one at a time based on probability distribⁿ.

> Applicaⁿ -

1) Image generaⁿ
2) Text to text image generaⁿ
3) Audio generaⁿ.

(VII)
Reinforcement Learning

* Deep Reinforcement learning (DRL) –
DRL is a powerful subfield of ML that combines the strengths of Deep learning & reinforcement learning. It allows agents to learn complex behaviours and make optimal decisions through trial and error, interacting with an environment & recieving rewards.

> Reinforcement Learning focuses on training agents to make optimal choices in an environment by trial & error. Agents recieve rewards for actions.

> DRL leverages power of deep learning enables to learn intricate pattern & relationships from data.

> key components.
Agent - Entity interacts with environment.
Environment - External system with which agent interacts
Action -
Reward.

> DRL Algorithms -
There are various DRL Algos -
Deep Q Network (DQN), Proximal Policy optimizaⁿ.
Soft Actor-Critic.

> Application -
1> Robotics
2> Game playing
3> Recommendaⁿ sys.

* challenges in Reinforcement Learning -

1) explora^n & VS Exploita^n -
Bt Balancing explora^n - trying new actions to discover better strategies, & exploration - focusing on Actions known to yield high rewards is crucial.

2) Sample efficiency - Training RL agent can be automatica computationally expensive, especially for complex tasks requiring many interactions.

3) High variance - RL Algos experience high variance due to their stochastic nature. This can make it difficult to evaluate the true performance of agent.

4) Credit Assignment problem - In complex environment with delayed rewards, it can be difficult for agent to determine which past action contributed to a particular reward. Hinders Agent's ability to learn long-term dependencies.

5) Reward Engineering - Defining clear & well structured reward signals that guides agent towards the desired behaviours is critical for successful RL implemen^n

~~* Q learning -~~

* Markov decision Process (MDP) - MDP is a mathematical framework that models decision making in situations where outcomes are both random and under control of decision maker.
> MDPs are built on the Markov chain a stochastic model that describes a sequence of events where probability of each event depends only on the previous ~~one~~ event

> Key components of an MDP –

1) States – Represent all the possible situation the agent can be in within the environment.
2) Action (A) – Set of choices the agent can make in each state.
3) Transition probability (P) – Defines probability of transitioning from one state to another after taking specific action.
4) Reward Fun^n (R) – Represents immediate reward received when transitioning from state.
5) Decision Making Process –
   1) observe current state.
   2) Select an action based on current state.
   3) Transition to a new state based on Action.
   4) Recieve reward based on transition.
   5) repeat Step 1 to 4.
6) $P(S_{n+1} | S_1, S_2, \ldots S_n) = P(S_{n+1} | S_n)$.
eq^n represents probability of transitioning to state S_{n+1} considering the entire history state $S_1, S_2, \ldots S_n$.

Ø Q Learning – Imagine an agent navigating maze learning optimal path through trial & error. This is the essense of Q learning.
> The agent interacts with environment (the maze) taking action (moving directions) and observing the resulting rewards (reaching the goal).
> It maintains Q-value table a map that stores the estimated Q-value (expected future reward) for each state-action pair.
> After taking action, the Q value for that state-pair is updated based on the recieved reward & estimated future rewards from new state.

Teacher's Sign.: _____

> Q-learning doesn't require a pre-built model of environment, making it adaptable to varias Situations.
> Qlearning can learn from past experiences.
> Limitations -
  > Q-table size grows exponentially with no. of states & actions.
  > Explora^n vs Exploita^n - Balancing explora^n (trying new actions) & Exploita^n (focusing on known action) can be challenging.

* Deep Q-learning - (DQN) -
DQN address the limita^n of Q-learning by leveraging dt DNN -
> Instead of Q table, DQN ~~use DNN~~ extends Q-learning by employing DNN. ~~It~~ DQN ~~stores~~ approximates Q-value function using DNN.
> It follows all the steps like Qlearning additionally including buffer to store state, action, reward, & next state, ~~in a m~~
> Random batches of experiences are sampled from memory buffer and used to train Q-network The network adjusts its weights to minimize the difference between the predicted Q-value & actual Q-value.
> The agent continues to interact with environment, update Q-~~value~~ network based on experience for improve its decision making -

Teacher's Sign.: _____

> Challenges in Deep Q Network -
1) Training complexity
2) Explora° vs Exploita°
3) Sample efficiency

* Dynamic programming (DP) for RL -
> DP algos excels at finding policies (action strategies) in scenarios where the agent has a complete model of the environment, it includes -
> Transi° probabilities - The probability of transitioning from one state to another after taking specific action
> Reward function - Reward assigned to each state transi° caused by action.
> Popular DP algos -
1) Policy itera° - Iteratively updates the value fun° computes the value fun° V_π for given policy π by solving Bellman eq° iteratively until convergence.
Greedily updates the policy based on V_π selecting actions maximizing expected rewards.

2) Value itera° - An iterative algo directly computing optimal value function V^*. Updates V^* by selecting actions maximizing expected fun rewards according to current value estimates.
Continues iterating until V^* converges

Simple RL For Tic Tac Toe

> Environment - Tic Tac Toe board of 3X3 size
The agent (player) interacts with environment by selecting moves. (X or O)

> State Repr - State of the game board represe
Current Configura" of X's and O's, corresponding to unique arrangement of 'X' & 'O'

> Action Space - Consists of all possible moves that the agent can make on the board.

> Reward - Agent recieves reward based on
outcome → +1 for winning the game
-1 for losing the game.

> Learning Process.
1) Initializa"
2) Explora"
3) Reward
4) Learning from rewards
5) Improvement.

> challenges -
1) Explora" vs Exploita"
2) Limited Rewards.