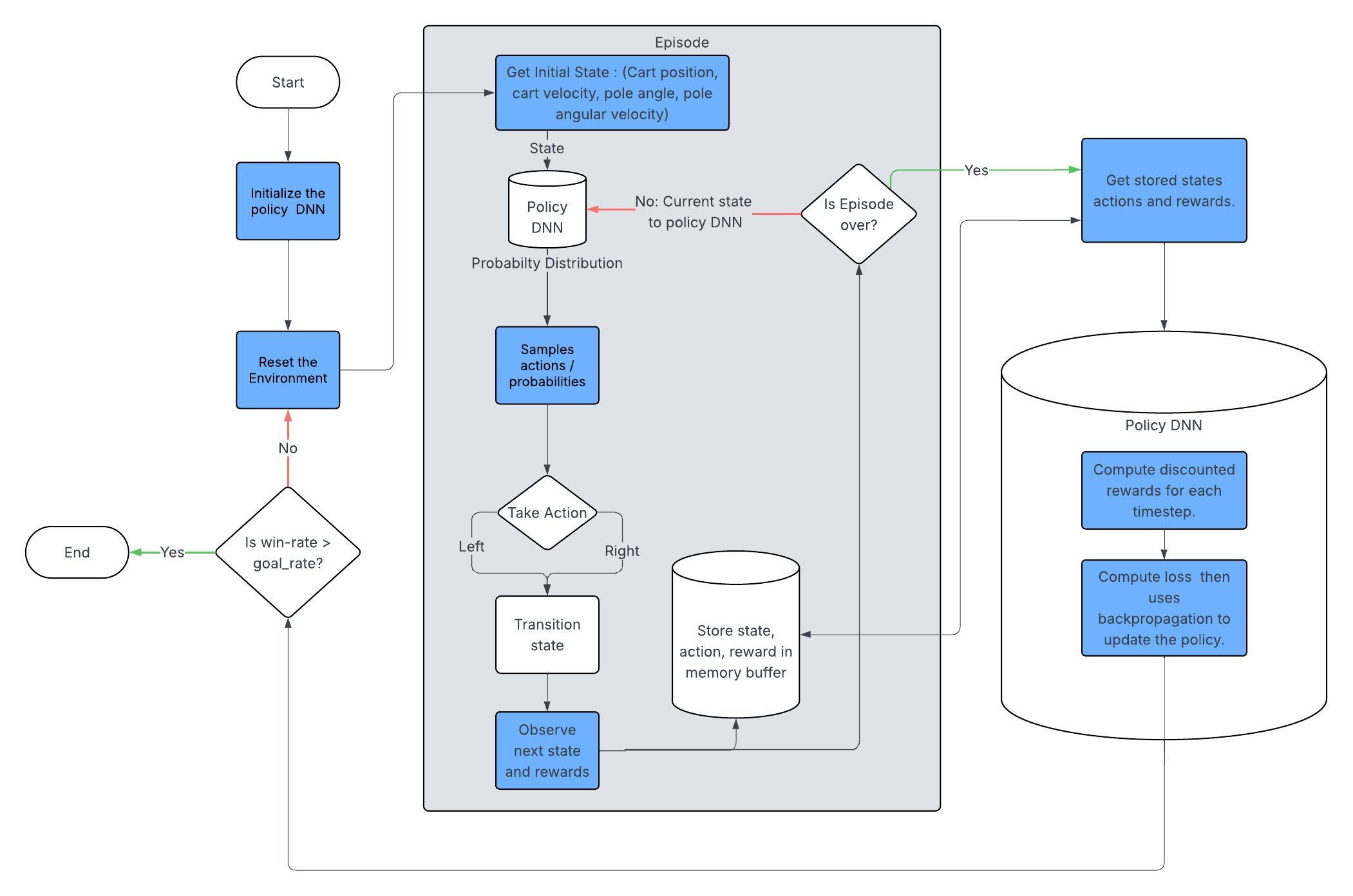
**Solving the cartpole problem with REINFORCE.**

For a quick review, in reinforcement learning, a policy governs how an agent chooses which action to take in different states or situations. In most reinforcement learning algorithms, improving the policy is the primary objective. (*What is a policy,* 2025). The cartpole problem is a game where the agent must balance a pole on a cart by moving the cart left or right. The agent scores a point for each time step the pole remains mostly vertical, and the game ends when the pole falls too far to either side. In a previous module, students solved this problem using deep Q-learning networks; however, it can also be solved using the REINFORCE algorithm.

REINFORCE is a policy-gradient reinforcement learning algorithm that uses a deep neural network to directly update the policy during learning. The cartpole environment, and thus the policy, is stochastic, so updating the policy means updating “the probability distribution of actions.” (*What is a policy,* 2025). In the REINFORCE algorithm, the policy updates “so that actions with higher expected reward have a higher probability value for an observed state.” (Yoon, 2018). In essence, to use REINFORCE to solve the cartpole problem, one would use a DNN to provide a policy, the agent would choose actions based on the policy's probability distribution, then update the policy at the end of each episode. (Miller, 2023). A depiction of the solution to the cartpole problem utilizing the REINFORCE algorithm is displayed on the follow page.



In pseudocode, this REINFORCE solution would appear as:

Initialize environment

Initialize policy network with 4 input nodes and 2 output nodes.

For the maximum number of episodes or until proficiency is obtained (goal score is met consistently):

For the max number of steps:

Get state

Get the probability distribution for available actions.

Choose action based on probability

Log state, action, rewards

If the game is done:

Compute discount returns for each time step

Log probabilities of the actions taken

Update policy with backpropagation

If the average score is above the goal score:

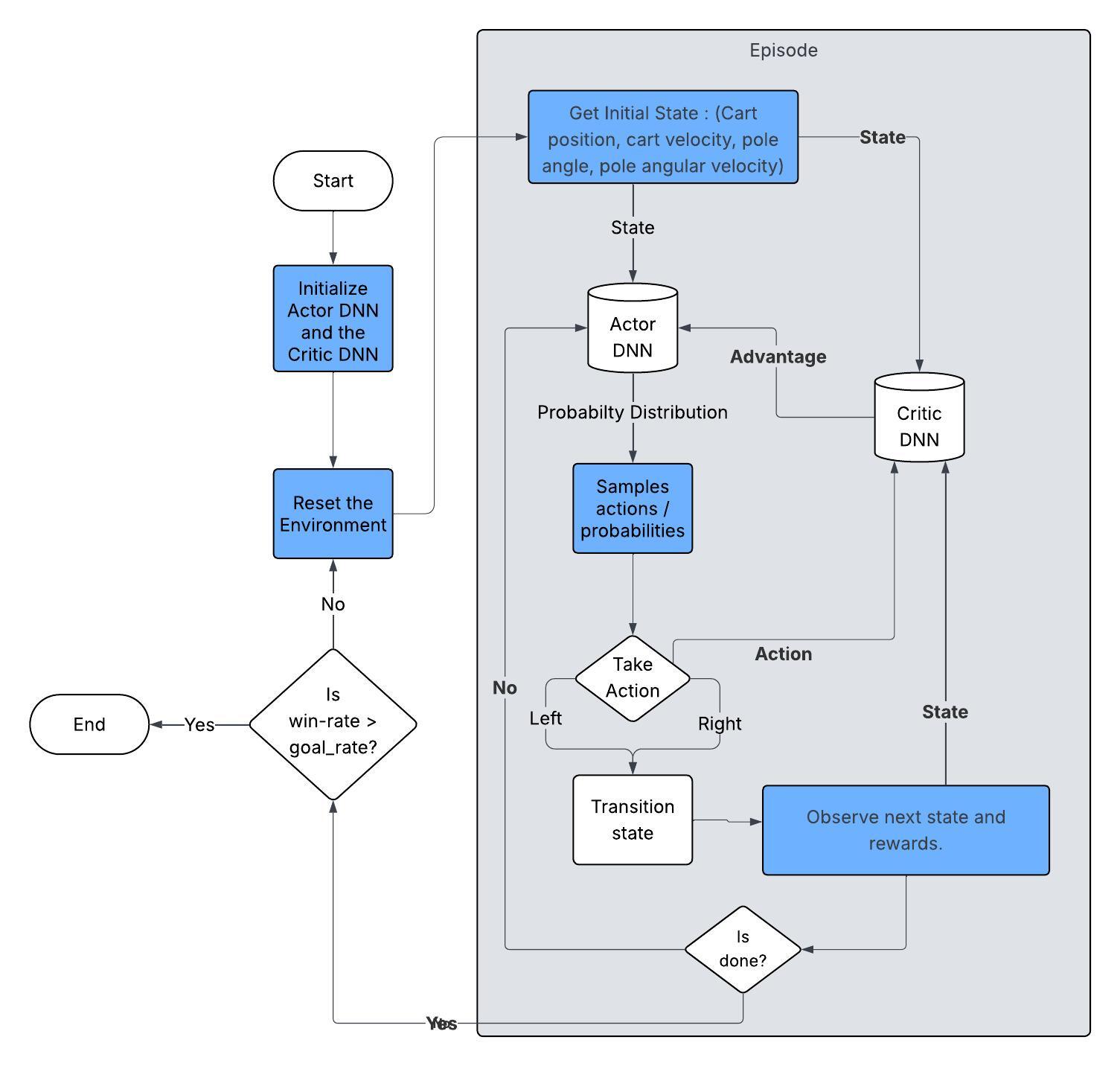
Display completion message

break

**Solving the cartpole problem with A2C.**

“In the field of Reinforcement Learning, the Advantage Actor Critic (A2C) algorithm combines two types of Reinforcement Learning algorithms (Policy Based and Value Based) together.” (Wang, 2021). To solve the cartpole problem using the A2C algorithm, one would create a DNN for the actor as well as one for the critic. The actor network is responsible for taking the current state as input and generating a probability distribution as output. The actor samples this distribution and chooses an action based on the probabilities. The critic network is responsible for checking the success of the actor’s decision by estimating “the expected cumulative reward starting from state *s*.” (*Actor-critic algorithm,* 2025). The critic estimates a TD target or value for the current state (estimated reward for the current state plus the discounted value of future states). The advantage function then compares the advantage of taking action *a* in state *s* over the average or expected value of being in state *s* with the current policy. In this manner, the critic gives feedback on how much better or worse the action taken was in comparison to the average action taken, thus improving learning. (Wang, 2021). In contrast to the REINFORCE algorithm, the critic in the A2C algorithm can be updated after each step. (*Actor-critic algorithm,* 2025).

A depiction of the solution to the cartpole problem utilizing the A2C algorithm is displayed on the following page.



In pseudocode, this A2C solution for cartpole would appear as:

Initialize environment

Initialize Actor DNN

Initialize Critic DNN

For each episode in the maximum number of episodes or until proficiency is obtained:

For each step in the episode until the maximum number of steps or the game status is done:

Choose an action using policy in the actor network.

Take the action and observe the next state, reward, and action.

Calculate the advantage for action taken using the critic DNN.

Calculate the actor and critic loss.

Update the actor and critic networks using respective losses and learning rates.

**How value-based approaches differ from policy gradient approaches.**

Q-learning populates a table with the Q-value, or expected future rewards, for each action in each state. The Q-value can be either calculated and the table updated manually, or a DNN can estimate the Q-values. After each step, the Q-values are updated for continuous refinement. In Q-learning, an agent uses a policy to determine whether to choose a random action or the best action, with random action exploration decreasing over time. However, policy gradient algorithms directly map probability distributions of actions for each state. After each episode is complete, the stored actions, rewards, and states are evaluated to determine the value of each step. The value is then used to calculate probabilities, and the policy is refined by updating the probability distributions. (Walkerastro, 2024). The table below highlights the key differences in the two approaches.

|  |  |  |
| --- | --- | --- |
|  | Value-based approaches | Policy Gradient |
| Optimization Method | Update Q-function (state-action value). | Update policy parameters (either map states to actions or probability distributions to actions). |
| Exploration | Controlled by epsilon, typically starts high and decays over time. | Sampling actions from probability distributions encourages exploration. |
| Best for | Simple deterministic environments | Complex stochastic environments. |

**How actor-critic approaches differ from value- and policy-based approaches**.

As previously discussed, actor-critic approaches combine value-based and policy-based approaches. (Wang, 2021). In actor-critic algorithms, the actor maps the probability distributions of a state to actions, while the critic evaluates how good the action was compared to average actions. The key difference between the actor-critic approach and the value-based approach is the addition of directly updating the policy with the actor network. The key difference in policy-based algorithms with actor-critic algorithms is the addition of a critic network to evaluate the quality of the action taken after each step. These differences allow for improved sample efficiency, faster convergence, versatility across action spaces, and off-policy learning. (*Actor-critic algorithm,* 2025).

References

GeeksforGeeks. (2025, October 9). *Actor-critic algorithm in Reinforcement Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/actor-critic-algorithm-in-reinforcement-learning/

Miller, T. (2023). *Policy gradients*. Mastering Reinforcement Learning. <https://gibberblot.github.io/rl-notes/single-agent/policy-gradients.html>

Walkerastro. (2024, August 28). *Policy gradient methods vs Q-learning*. Medium. <https://medium.com/@walkerastro41/policy-gradient-methods-vs-q-learning-c9f513f63d3d>

Wang, M. (2021, January 22). *Advantage actor critic tutorial: MINA2C*. Towards Data Science. <https://towardsdatascience.com/advantage-actor-critic-tutorial-mina2c-7a3249962fc8/>

*What is a policy in RL?*. Milvus. (2025b). <https://milvus.io/ai-quick-reference/what-is-a-policy-in-rl>

Yoon, C. (2019, July 17). *Understanding actor-critic methods*. Medium. Retrieved February 10, 2023, from <https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f>