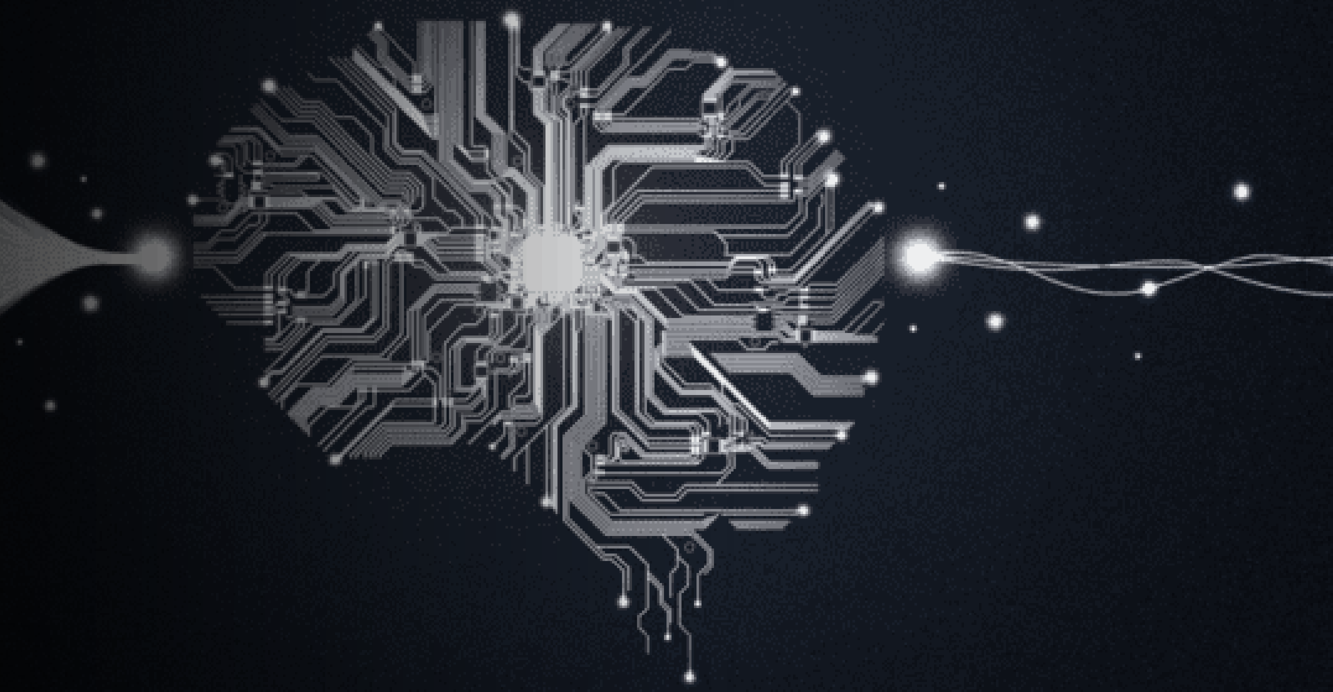
Exploring Generalization in Deep Reinforcement Learning for Control Tasks

Universitat Pompeu Fabra Barcelona

Àlex Montoya Pérez

Anders Jonsson & Sergio Calo Oliveira

Grau en Enginyeria en Informàtica



Universitat Pompeu Fabra Barcelona

OVERVIEW

01 INTRODUCTION

02 BACKGROUND THEORY

03 METHODOLOGY

04 EXPERIMENT EVALUATION

05 DISCUSSIONS, CONCLUSION & FUTURE WORK

OVERVIEW

01 **INTRODUCTION**

02 **BACKGROUND THEORY**

03 **METHODOLOGY**

04 **EXPERIMENT EVALUATION**

05 **DISCUSSIONS, CONCLUSION & FUTURE WORK**

OVERVIEW

01	02	03	04	05
INTRODUCTION	BACKGROUND THEORY	METHODOLOGY	EXPERIMENT EVALUATION	DISCUSSIONS, CONCLUSION & FUTURE WORK

Universitat Pompeu Fabra Barcelona

OVERVIEW

01
INTRODUCTION

02
BACKGROUND THEORY

03
METHODOLOGY

04
EXPERIMENT EVALUATION

05
DISCUSSIONS, CONCLUSION & FUTURE WORK

OVERVIEW

01 INTRODUCTION

02 BACKGROUND THEORY

03 METHODOLOGY

04 EXPERIMENT EVALUATION

05 DISCUSSIONS, CONCLUSION & FUTURE WORK

OVERVIEW

01 INTRODUCTION

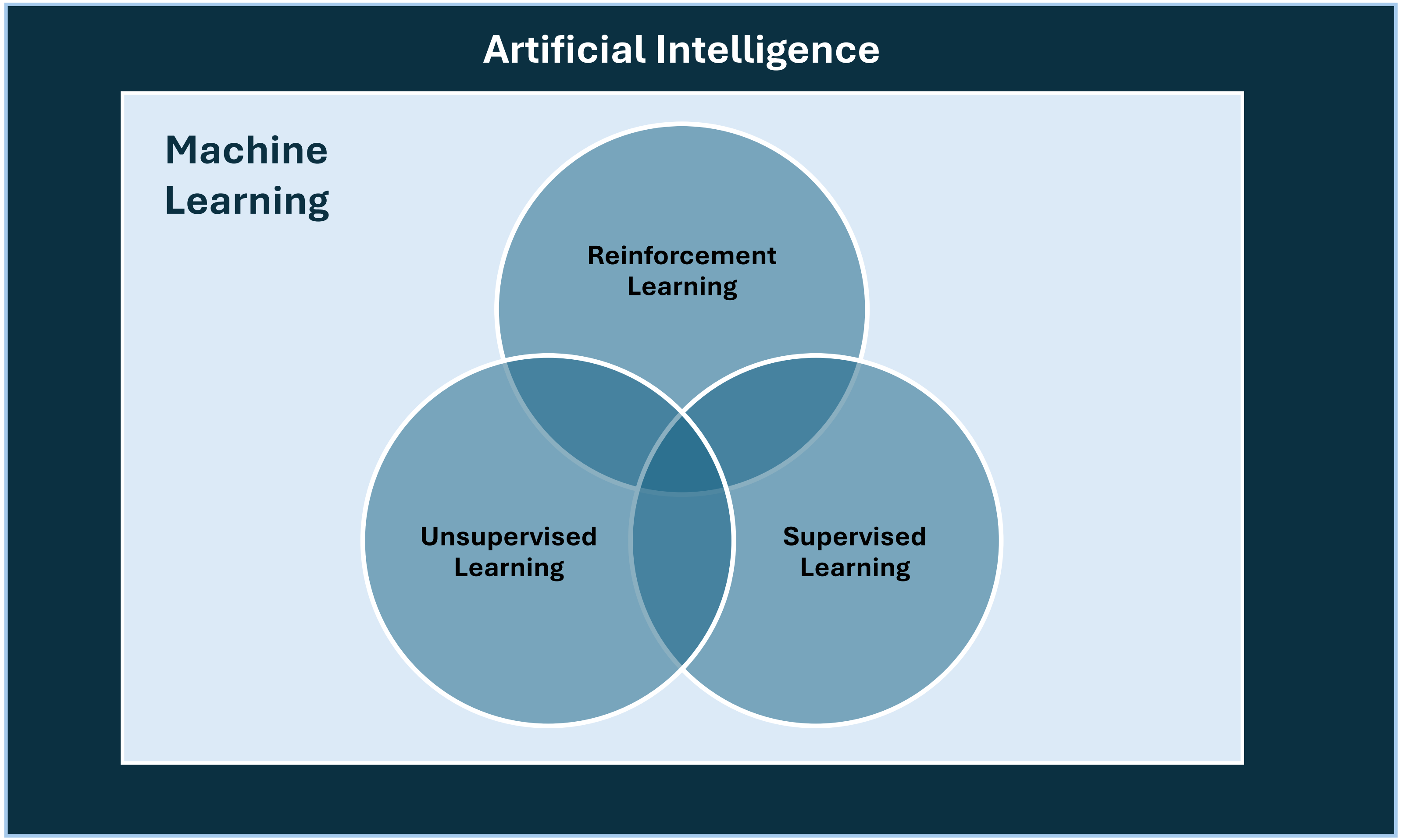
02 BACKGROUND THEORY

03 METHODOLOGY

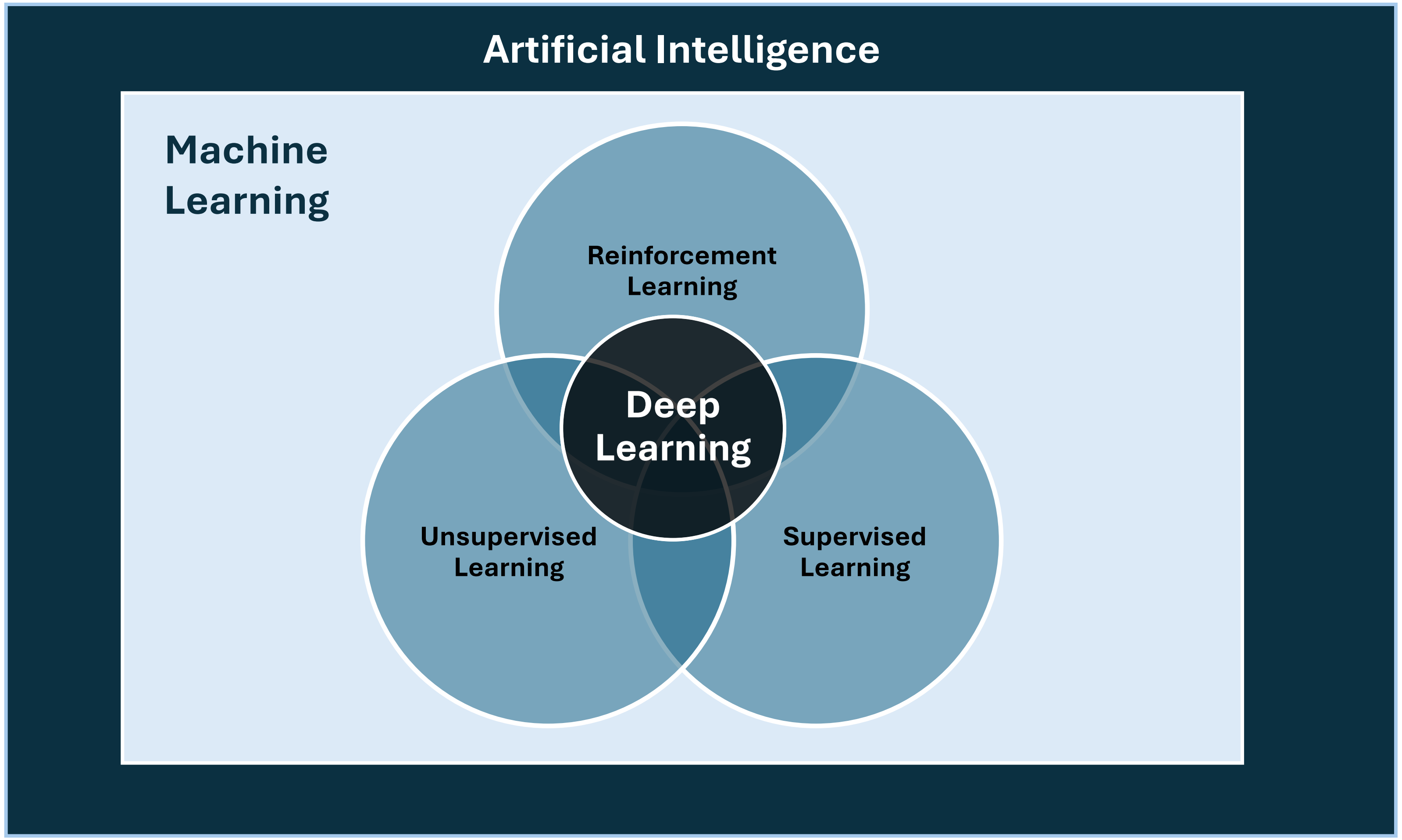
04 EXPERIMENT EVALUATION

05 DISCUSSIONS, CONCLUSION & FUTURE WORK

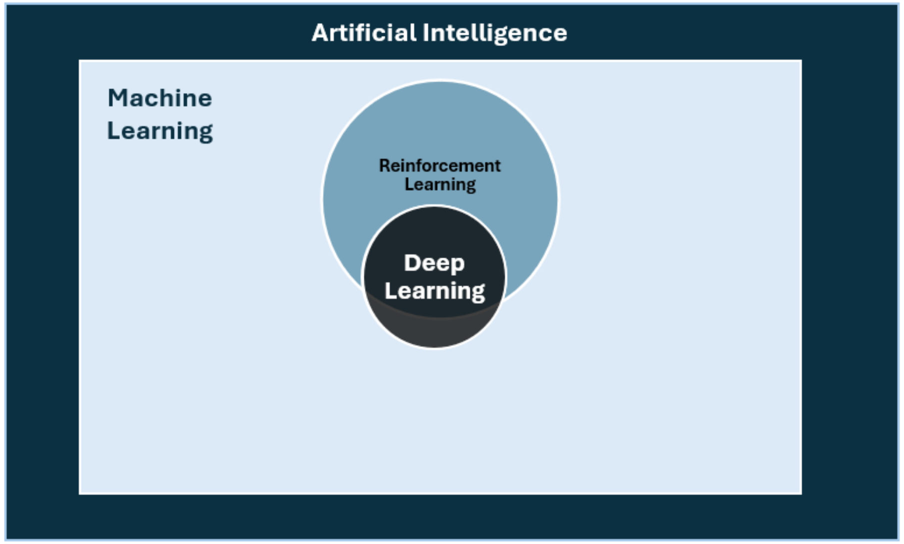
1.1 INTRODUCTION



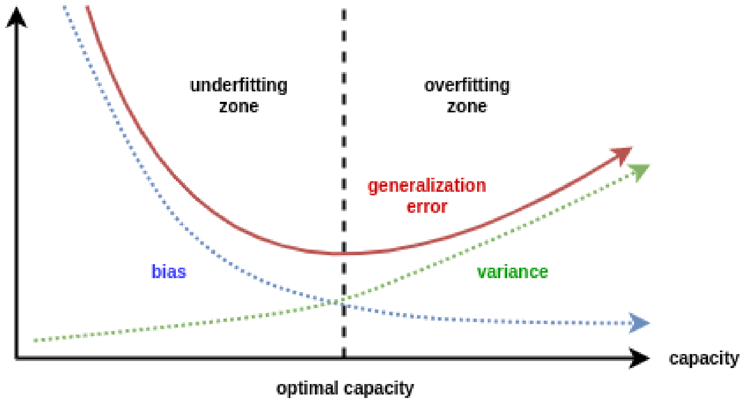
1.1 INTRODUCTION



1.2 OBJECTIVE



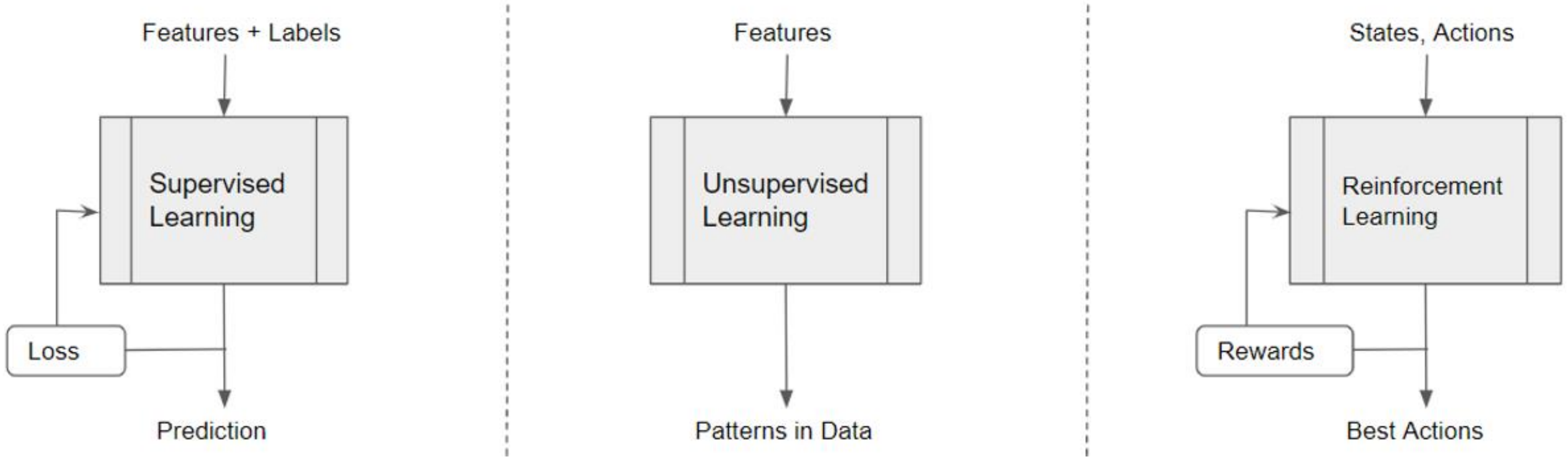
1. **Training**: Learning Efficiency.
2. **Generalization**: Adaptability to different environments.



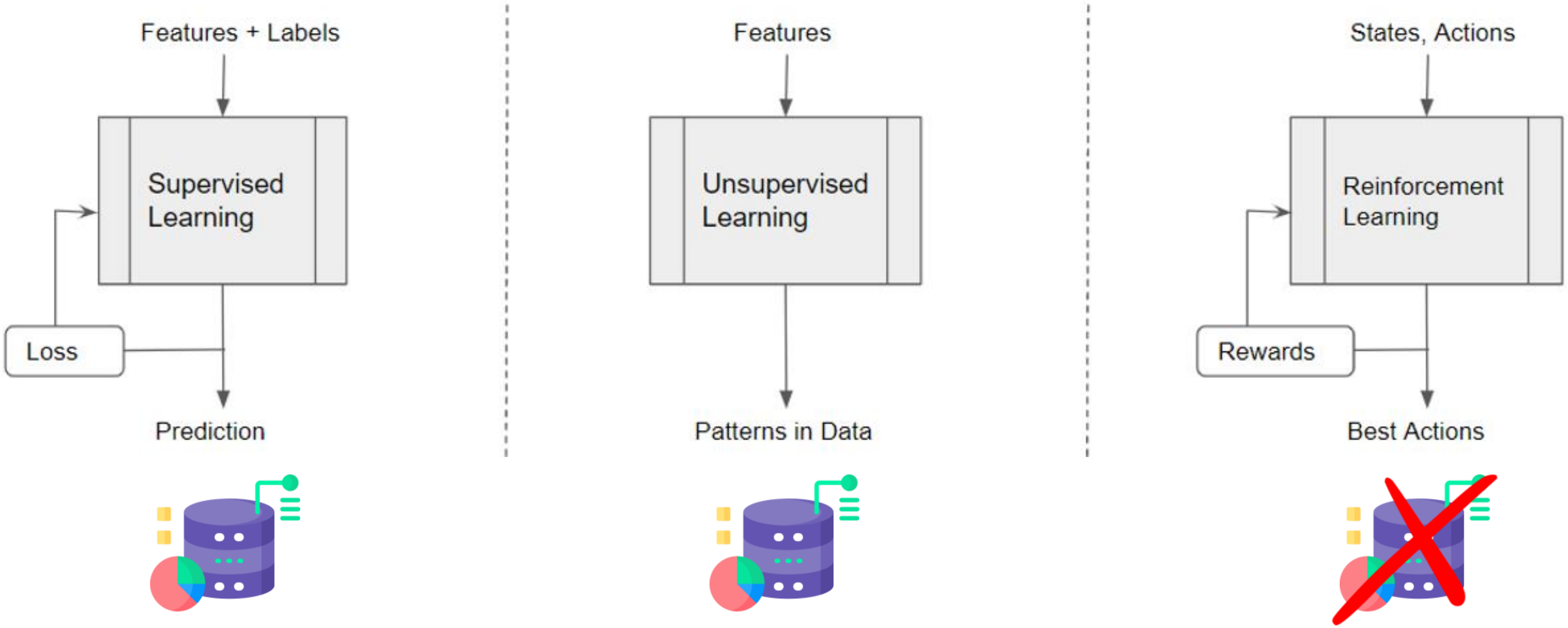
OVERVIEW

01	02	03	04	05
INTRODUCTION	BACKGROUND THEORY	METHODOLOGY	EXPERIMENT EVALUATION	DISCUSSIONS, CONCLUSION & FUTURE WORK

2.1 Reinforcement Learning



2.1 Reinforcement Learning

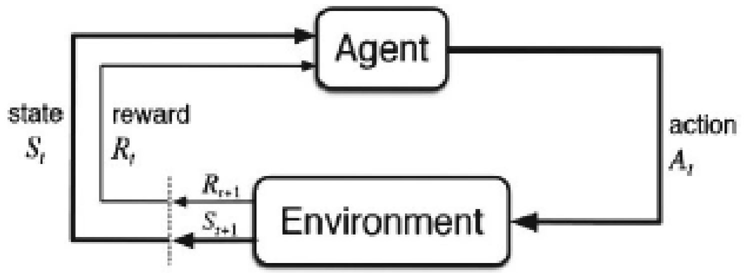


2.1 Reinforcement Learning

Markov Decision Processes (MDP)

MDPs are tuples $(\mathrm{S}, \mathrm{A}, \mathrm{T}, \mathrm{R}, \gamma)$:

- **States (S):** Finite set of possible states
- **Actions (A):** Set of actions applicable in a specific state $s \in S$
- **Transition Function (T):** specifies the probability distribution of the next state given the current state and action
- **Reward Function (R):** Assigns rewards to states or actions.
- **Discount Factor (γ):** Represents the difference in importance between short and long term rewards. $\gamma \in (0, 1]$.



2.1 Reinforcement Learning

Policies and Value Functions

A policy π represents a strategy that dictates the agent's behavior in an environment.

State Value Function ($V^{\pi}(s)$):
Expected return when starting in state s and following policy π.

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t \mid s_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right]$$
$$= \sum_{s_{t+1}} T(s, \pi(s), s_{t+1})\Big(R(s, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1})\Big).$$

State-Action Value Function ($Q^{\pi}(s,a)$) :
Expected return when starting in state s, taking action a and following policy π.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t \mid s_t = s, a_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right]$$
$$= \sum_{s_{t+1}} T(s, a, s_{t+1})\Big(R(s, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1})\Big).$$

Goal in MDP: Find the best policy, maximizing the value function for all states

$$V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s) = \max_{a} Q^{\pi^*}(s,a),$$
$$Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a).$$

2.1 Reinforcement Learning

Universitat Pompeu Fabra Barcelona

Bellman Optimality Equation

π*(s): for any state s is the action a that maximizes the sum of the immediate reward and the discounted value of the next state, averaged over all possible next states s_{t+1}

$$\pi^*(s) = \arg\max_a \sum_{s \in \mathcal{S}} T(s, a, s_{t+1})\Big(R(s, a, s_{t+1}) + \gamma V^*(s_{t+1})\Big) = \arg\max_a Q^{\pi^*}(s, a).$$

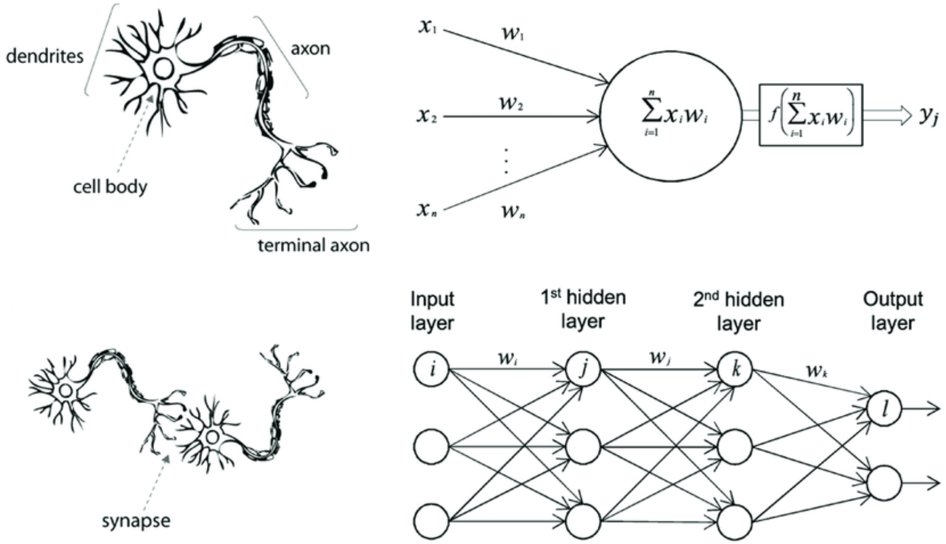
This can also be expressed using the optimal Q-value:

$$Q^*(s, a) = \sum_{s_{t+1} \in \mathcal{S}} T(s, a, s_{t+1})[R(s, a, s_{t+1}) + \gamma V^*(s_{t+1})]$$

$$V^*(s) = \max_a Q^*(s, a).$$

This allows for learning Q-functions instead of V-functions in model-free approaches when transition and reward functions are unknown.

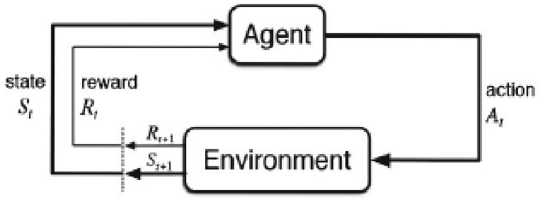
2.2 Deep Learning

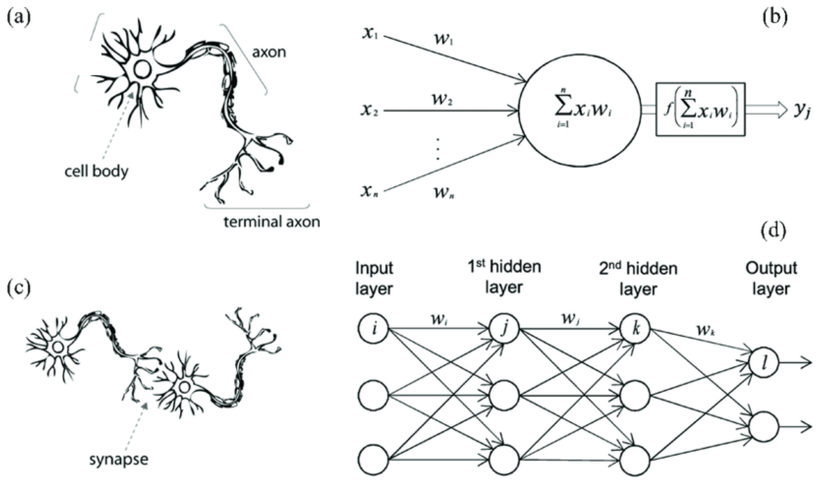


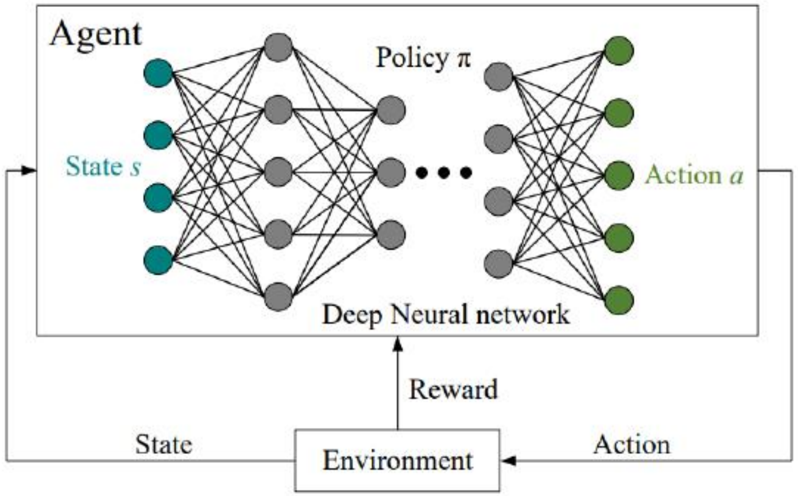
Training Procedure

1. **Initialization:** Randomly initialize the θ.
2. **Forward Propagation** $y_i^l = \sigma\left(\sum_{j=1}^{n^{l-1}} w_{ij}^l y_j^{l-1} + b_i^l\right)$
3. **Backward Propagation** $w_{ij}^l \leftarrow w_{ij}^l - \alpha \frac{\partial \mathcal{L}}{\partial w_{ij}^l},$
4. **Repeat steps 2 and 3**

2.3 Deep Reinforcement Learning



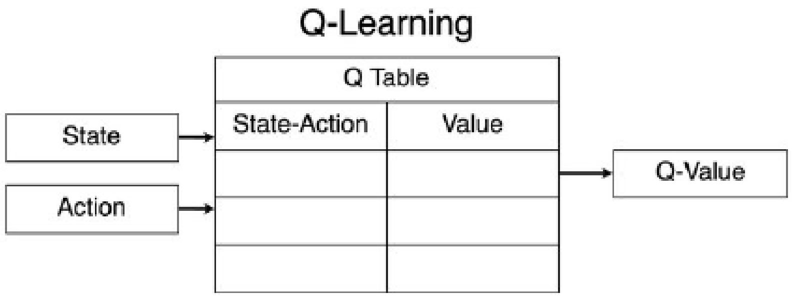




2.3 Deep Reinforcement Learning

Q-Learning

Goal: Learn a policy that maximizes long-term accumulated rewards by estimating Q-values.



2.3 Deep Reinforcement Learning

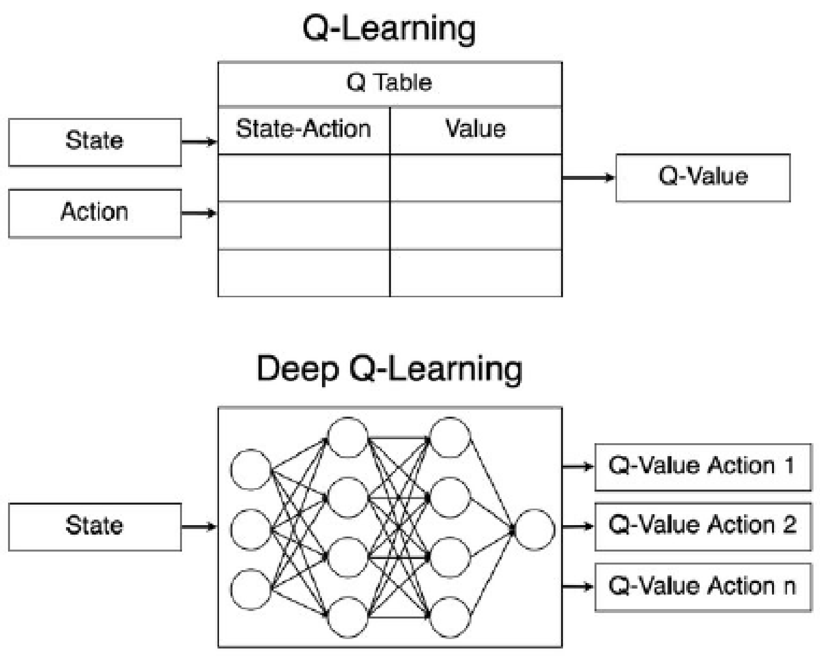
Q-Learning

Goal: Learn a policy that maximizes long-term accumulated rewards by estimating Q-values.

DQN

The DQN algorithm employs two multilayer perceptron ANN:

- **Eval-net:** ANN responsible for calculating the actual Q-Value based on the current state-action pairs.
- **Target-net:** A replica of the Eval-net, updated at regular intervals to maintain stability during training by providing consistent Q-value targets



2.3 Deep Reinforcement Learning

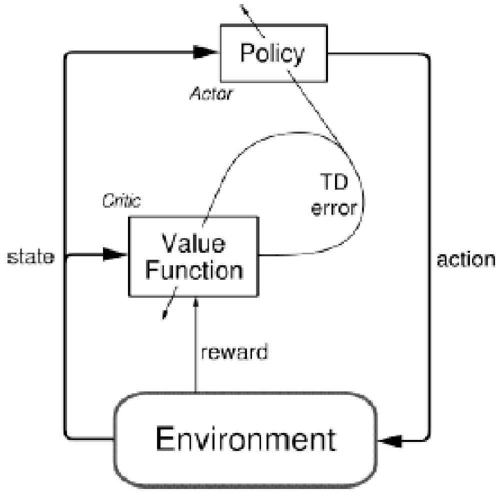
Actor-Critic

Unlike traditional TD methods where a single value function represents both policy and value estimates, AC methods maintain separate structures for these components.

- **Actor:** Responsible for decision-making, the actor selects actions according to the prevailing policy
- **Critic:** Assesses the actor's decisions, the critic estimates the value function and evaluates the actor's actions. offering feedback on their efficacy**.**

Advantage Function: primary feedback signal, guiding learning for both the actor and the critic.

$$A_t(s_t, a_t) = R(s_t, a_t) + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t).$$



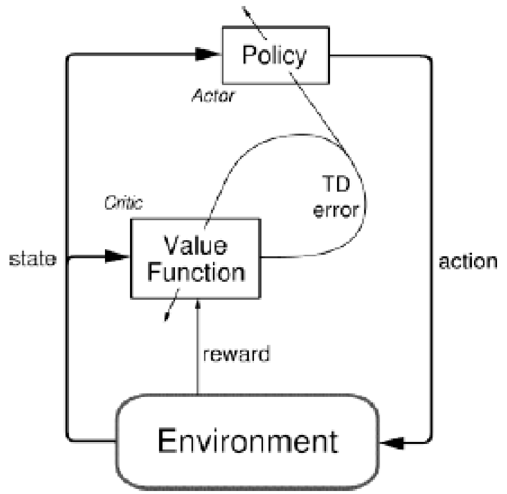
2.3 Deep Reinforcement Learning

Proximal Policy Optimization (PPO)

An Actor-Critic algorithm that iteratively refines a policy parameterized by θ (π_θ) through environment interaction and data-driven updates to maximize expected rewards.

$$\theta_{\text{new}} = \underset{\theta}{\operatorname{argmax}} \sum_{t=0}^{T} \mathbb{E}\left[\min\left(r_t(\theta)\ A^{\pi_{\theta_{\text{old}}}}(s_t, a_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\right)\right]$$

Probability ratio between the new and old policies $r_t(\theta) = \dfrac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{\text{old}}}(a_t, s_t)}.$



Clipped Objective Function:

PPO maintains stability by restricting policy changes within a small range using a clipping mechanism

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A^{\pi_{\theta_{\text{old}}}}(s_t, a_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A^{\pi_{\theta_{\text{old}}}}(s_t, a_t)\right)\right],$$

OVERVIEW

01	02	03	04	05
INTRODUCTION	BACKGROUND THEORY	METHODOLOGY	EXPERIMENT EVALUATION	DISCUSSIONS, CONCLUSION & FUTURE WORK

Universitat Pompeu Fabra Barcelona

3.1 Environment

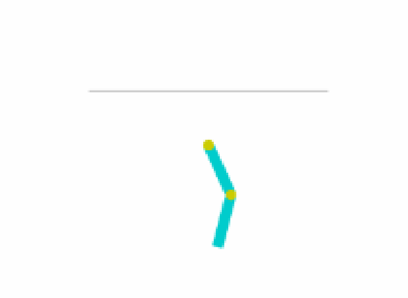
python ⟺ Gymnasium / Gymnasium-Robotics ⟺ MuJoCo

Advanced physics simulation

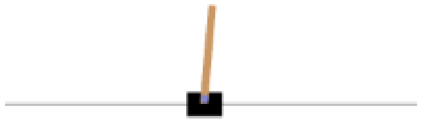
3.1 Environment

Control Tasks

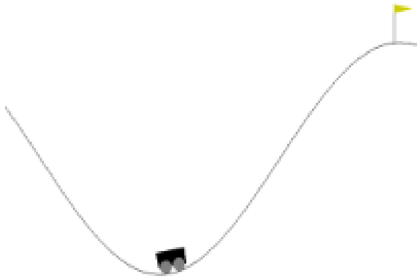
Gymnasium



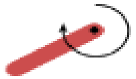
Acrobot



CartPole



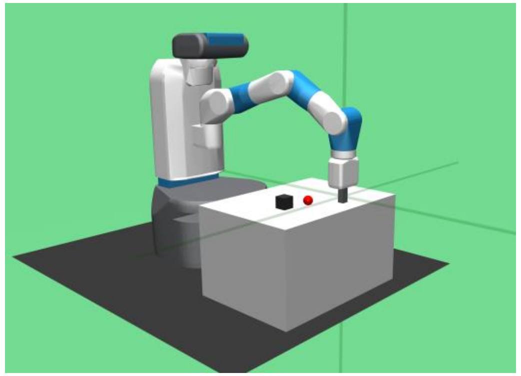
Mountain Car Continuous



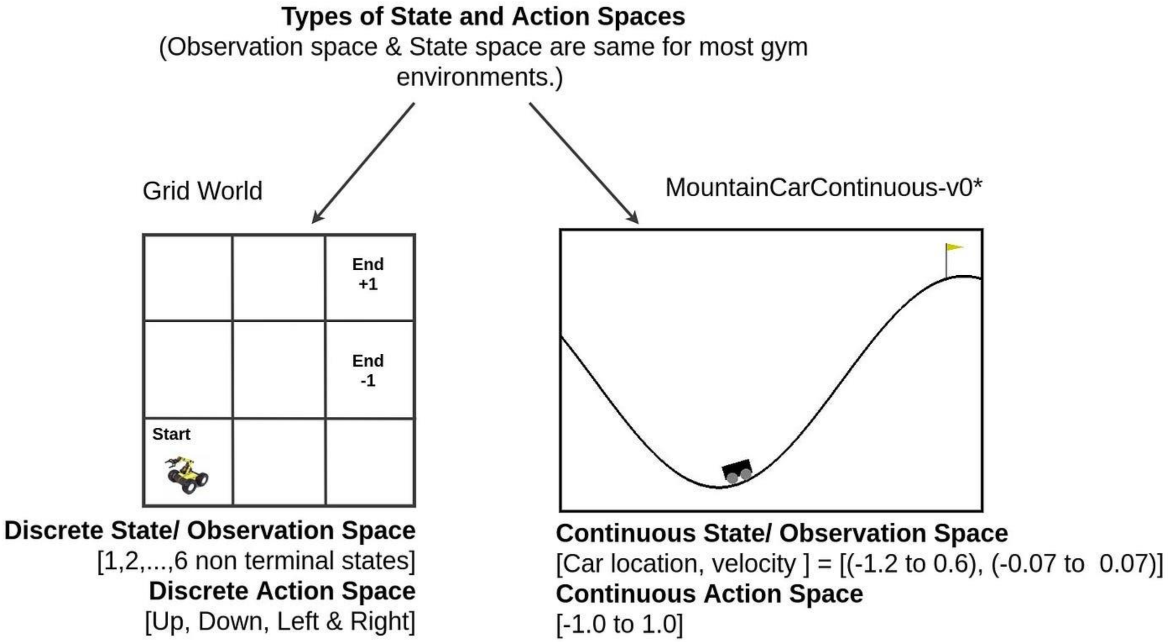
Pendulum

Custom Pick And Place

Gymnasium-Robotics

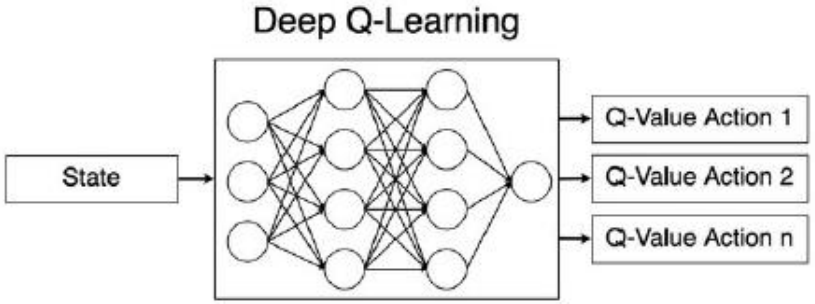


Thesis Orientation



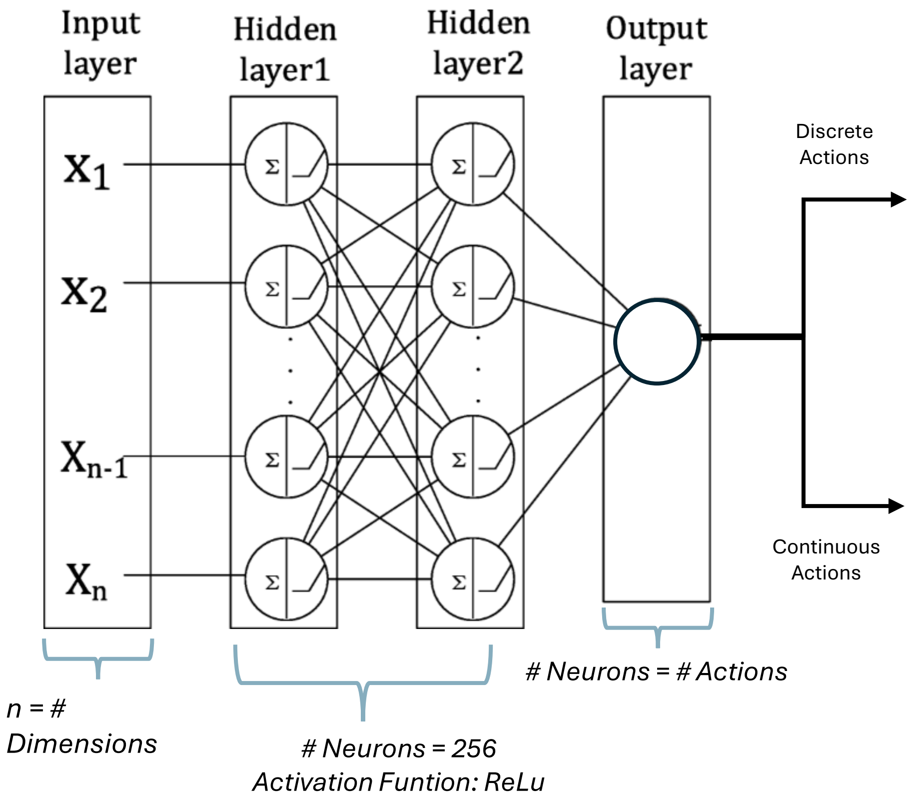
Problem with DQN and Continuous Actions

Q-value functions assigns an expected value to each state-action pair. In a continuous action space, the number of possible actions is infinite.



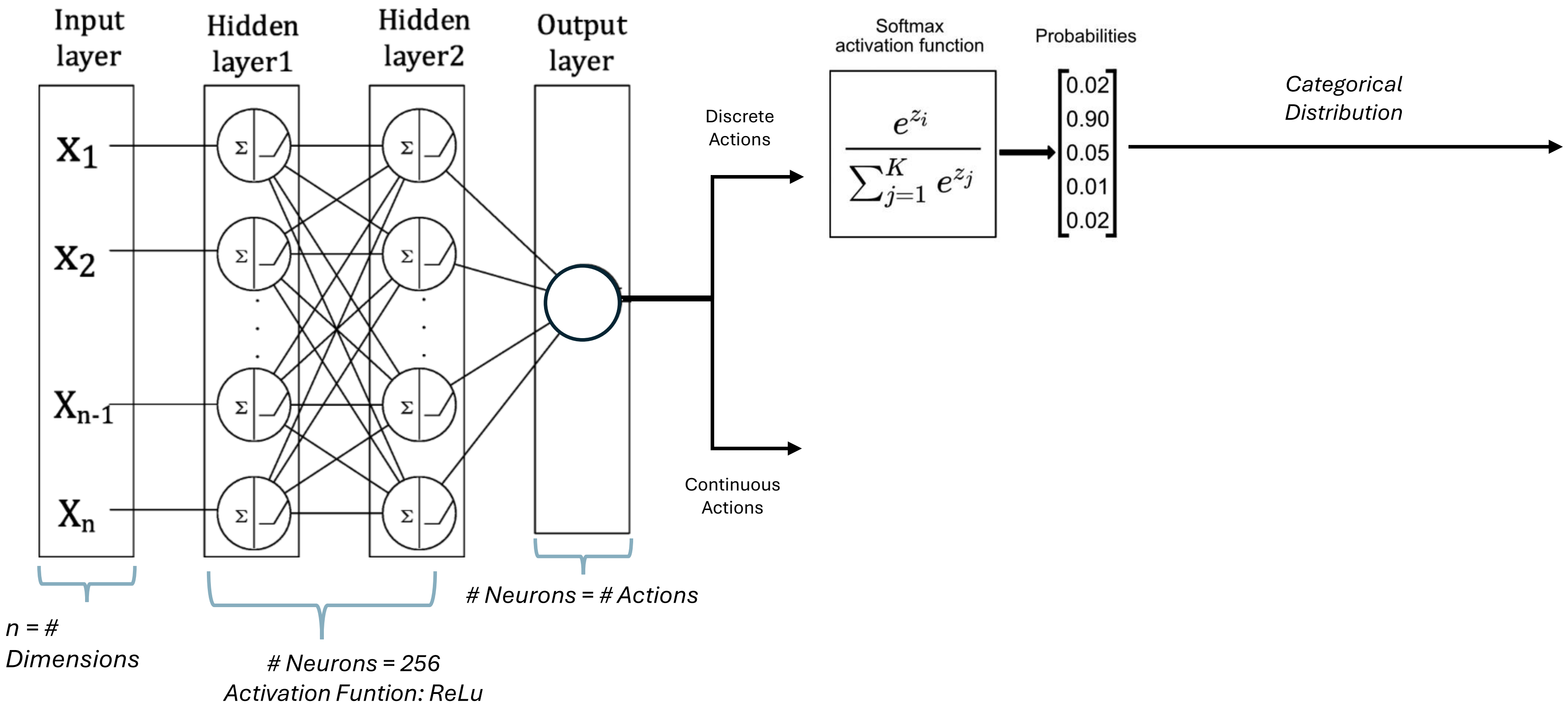
3.2 Deep Reinforcement Learning Agent

Actor and Critic Networks: *Actor*



3.2 Deep Reinforcement Learning Agent

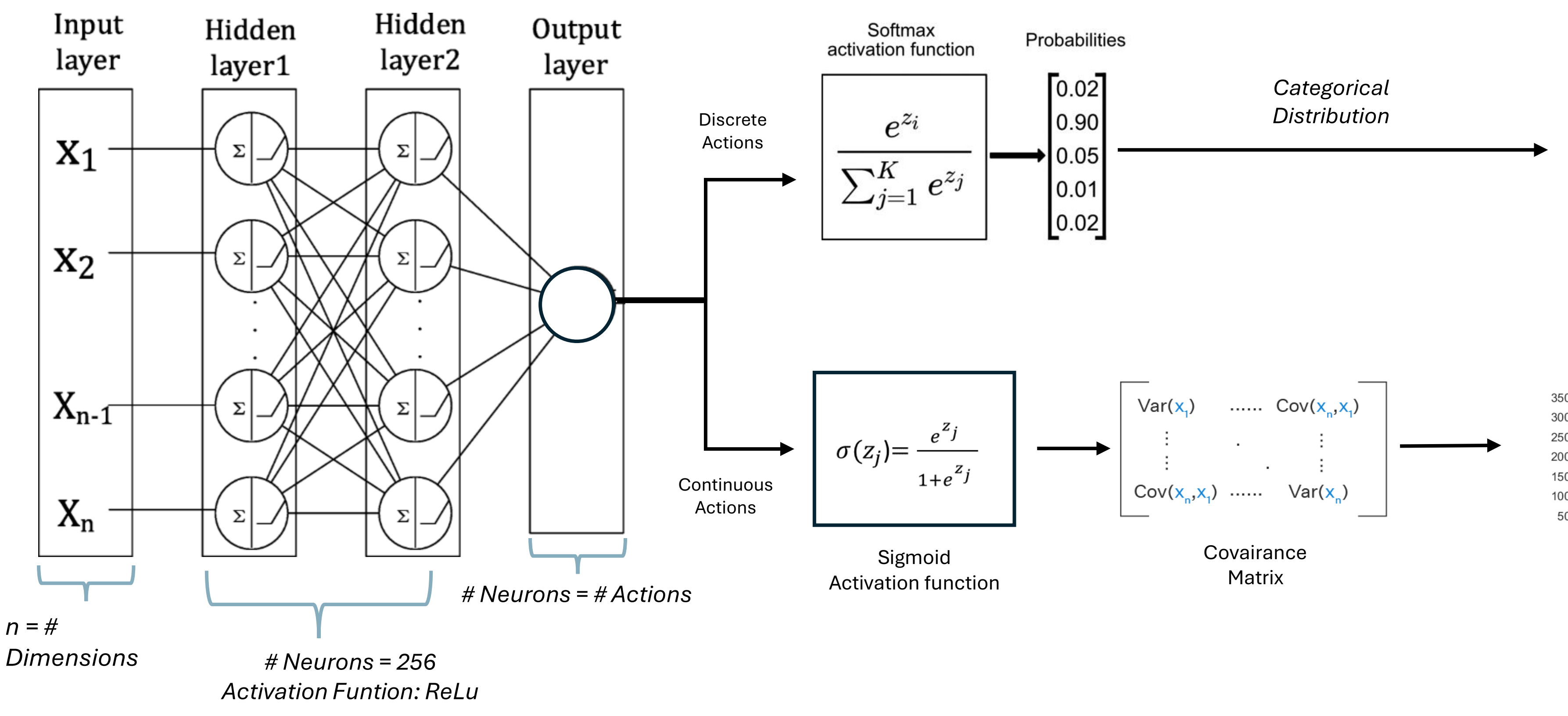
Actor and Critic Networks: *Actor*



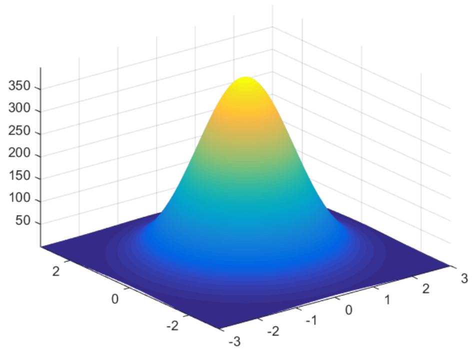
Action	Probability
Left	1/10
Right	3/10
Up	5/10
Down	1/10

3.2 Deep Reinforcement Learning Agent

Actor and Critic Networks: *Actor*



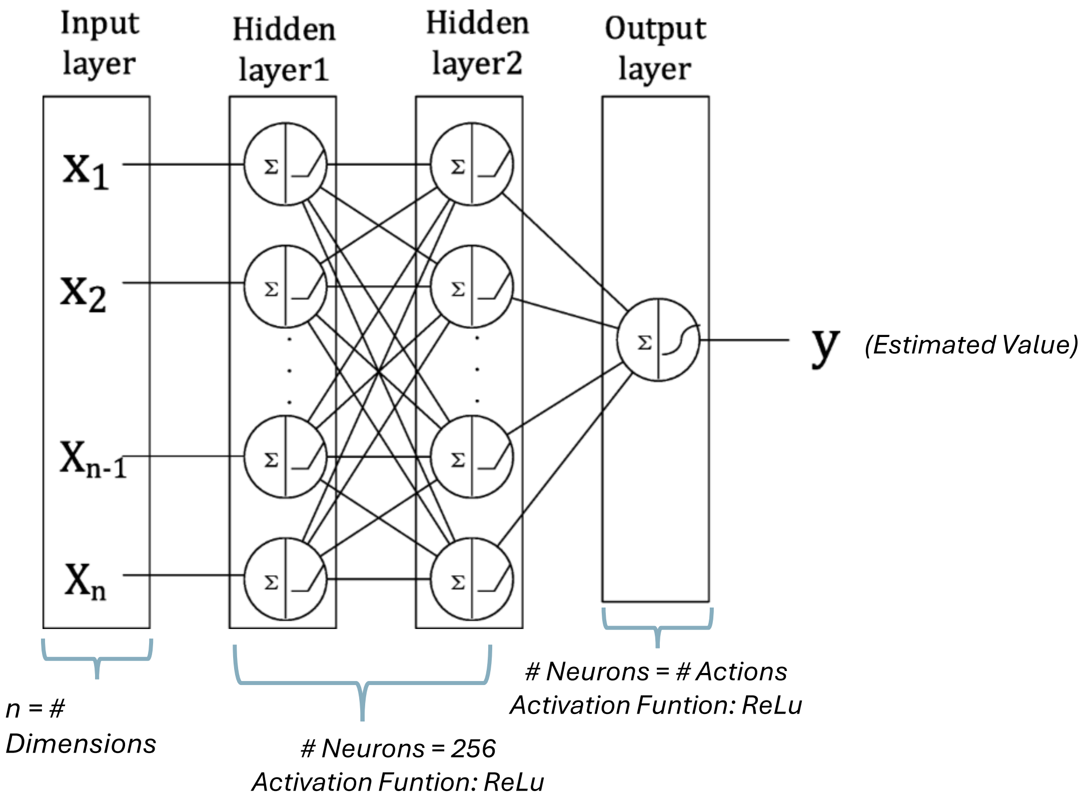
Action	Probability
Left	1/10
Right	3/10
Up	5/10
Down	1/10



Multivariate Normal Distribution

3.2 Deep Reinforcement Learning Agent

Actor and Critic Networks: *Critic*



- Simpler due to its straightforward task of estimating state value, requiring only a single scalar output.

3.2 Deep Reinforcement Learning Agent

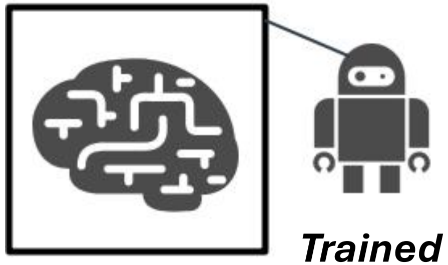
Hyperparameters Fine-Tuning

- **Timesteps per Batch:** Number of timesteps in each batch.
- **Max Timesteps per Episode:** Maximum number of timesteps allowed for each episode before termination.
- **Gamma (γ):** Importance of future rewards in the agent's decision-making process.
- **Epsilon (ϵ):** Clipping parameter to limit policy updates.
- **Alpha (α):** Learning rate, controlling the magnitude of parameter updates during optimization.
- **Training Cycles per Batch**: The number of training cycles executed per batch,

Hyperparameter	Control Task
Timesteps per Batch (TB)	500, 750, 1000
Max Timesteps per Episode (MTE)	750, 1000
Gamma (γ)	0.99
Epsilon (ϵ)	0.1, 0.2
Alpha (α)	0.01, 0.001, 0.0001, 0.00001
Training Cycles per Batch (TCB)	5, 10

Hyperparameter	Pick and Place
Timesteps per Batch (TB)	10K, 50K, 100K, 150K, 200K
Max Timesteps per Episode (MTE	1K, 5K, 10K, 15K
Gamma (γ)	0.99
Epsilon (ϵ)	0.1, 0.2
Alpha (α)	0.01, 0.001, 0.0001, 0.00001
Training Cycles per Batch (TCB)	5, 10

3.3 Generalization



Trained Agent

Zero-Shot Generalization

TE	MF	TV	VT	SV	MS
Acrobot-v1	Center of Mass	0.5	[0.1, 3.0]	0.2	400
	Link Length	1.0	[0.25, 4.0]	0.25	400
	Link Mass	1.0	[0.25, 4.5]	0.25	400
CartPole-v1	Force Magnitude	10	[5.0, 15.0]	1.0	10000
	Gravity	9.8	[5.0, 15.0]	1.0	10000
	Pole Length	0.5	[0.2, 0.8]	0.1	10000
	Pole Mass	0.1	[0.1, 0.6]	0.05	10000
MountainCar Continuous-v0	Car Friction	0.0025	[0.001, 0.005]	0.0005	300
	Initial Position	[-0.04, -0-06]	[-0.6, 0.3]	0.1	300
	Car Power	0.0015	[0.0005, 0.003]	0.0005	300
Pendulum-v1	Delta Time	0.05	[0.01, 0.15]	0.01	8000
	Gravity	9.8	[5, 15]	1	8000
	Pendulum Length	1.0	[0.5, 1.6]	0.1	8000
	Pendulum Mass	1.0	[0.25, 2.50]	2.5	8000

Table 3.2: Summary of modifications made to test generalization. MF = Modified Feature of the trained environment, TV = Trained Value, VT = Values Tested with the modified feature, SV = Step size between the values tested, MS = Maximum number of Steps to reach the goal.

OVERVIEW

01

INTRODUCTION

02

BACKGROUND THEORY

03

METHODOLOGY

04

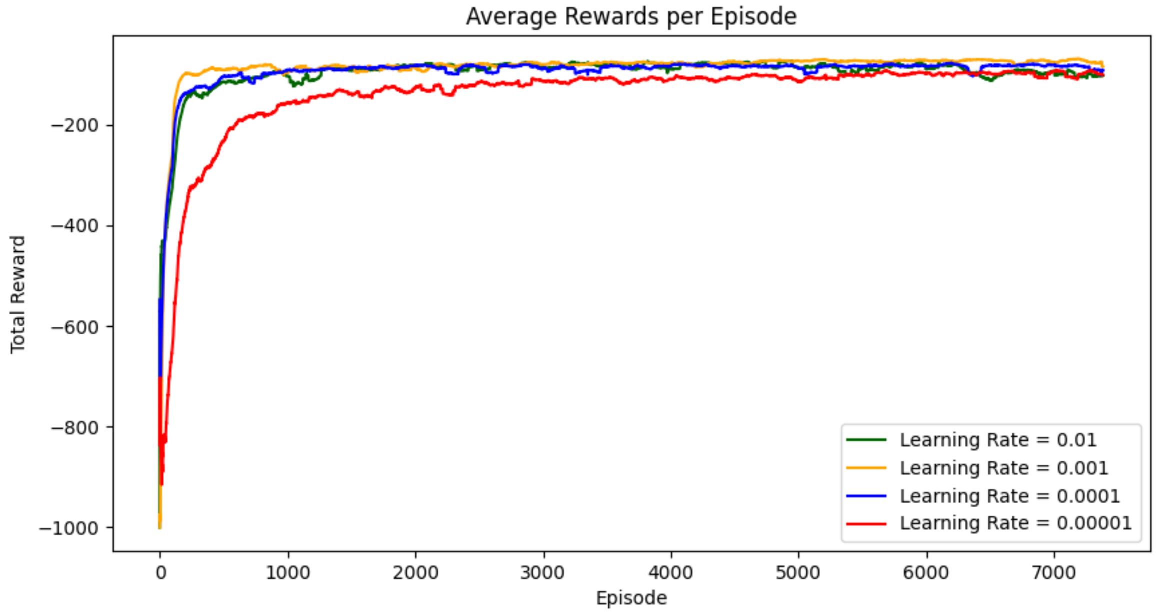
EXPERIMENT EVALUATION

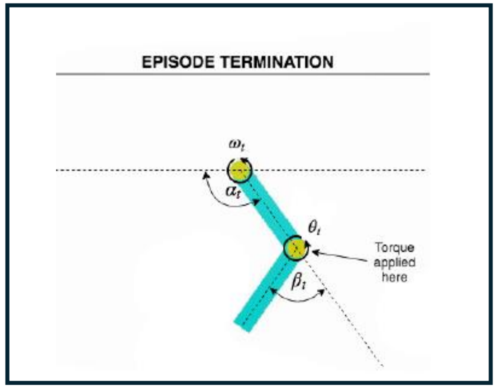
05

DISCUSSIONS, CONCLUSION & FUTURE WORK

4.1 Training Results

Acrobot





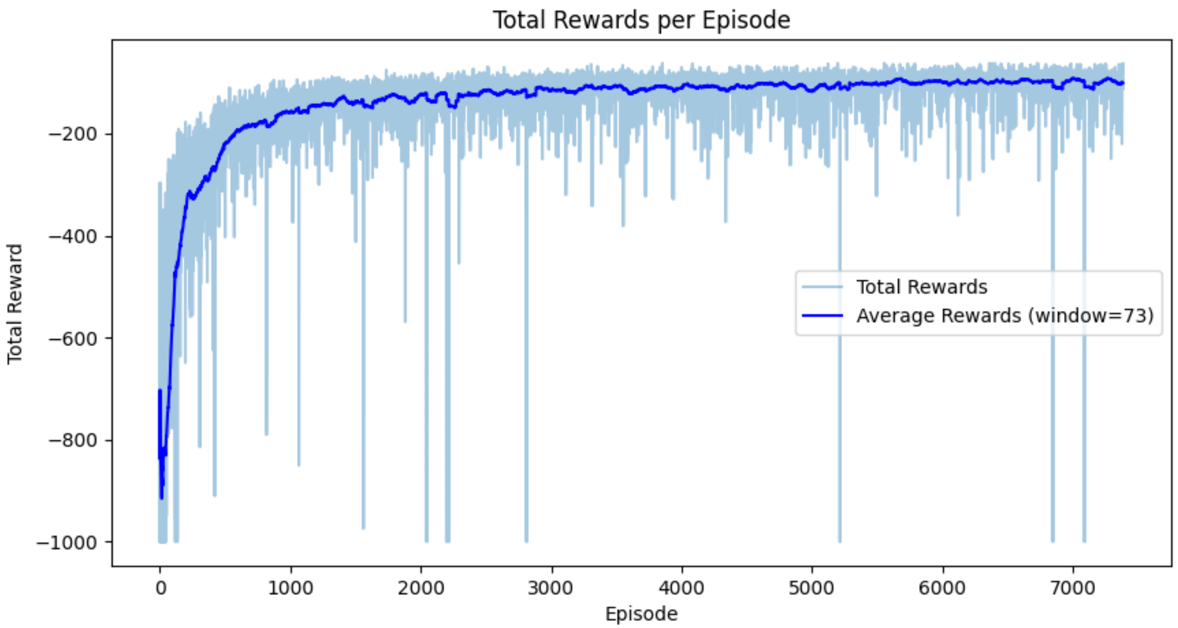
Goal: The free end reaches the target height.

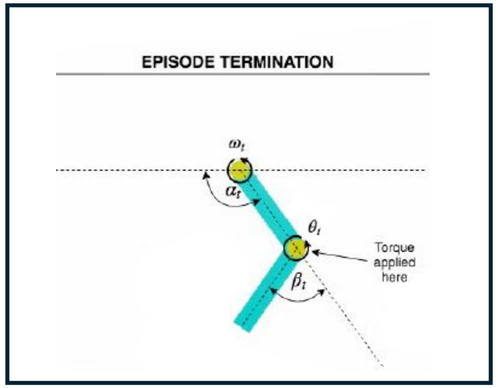
Reward: Each step incurs a reward of -1 until the goal is achieved.

Actions:
- Apply -1 torque: Exerts a negative torque on the actuated joint.
- Apply 0 torque: Exerts no torque on the actuated joint.
- Apply 1 torque: Exerts a positive torque on the actuated joint

4.1 Training Results

Acrobot





Environment	Timesteps	α	Time
Acrobot-v1	1M	0.00001	0:11:33

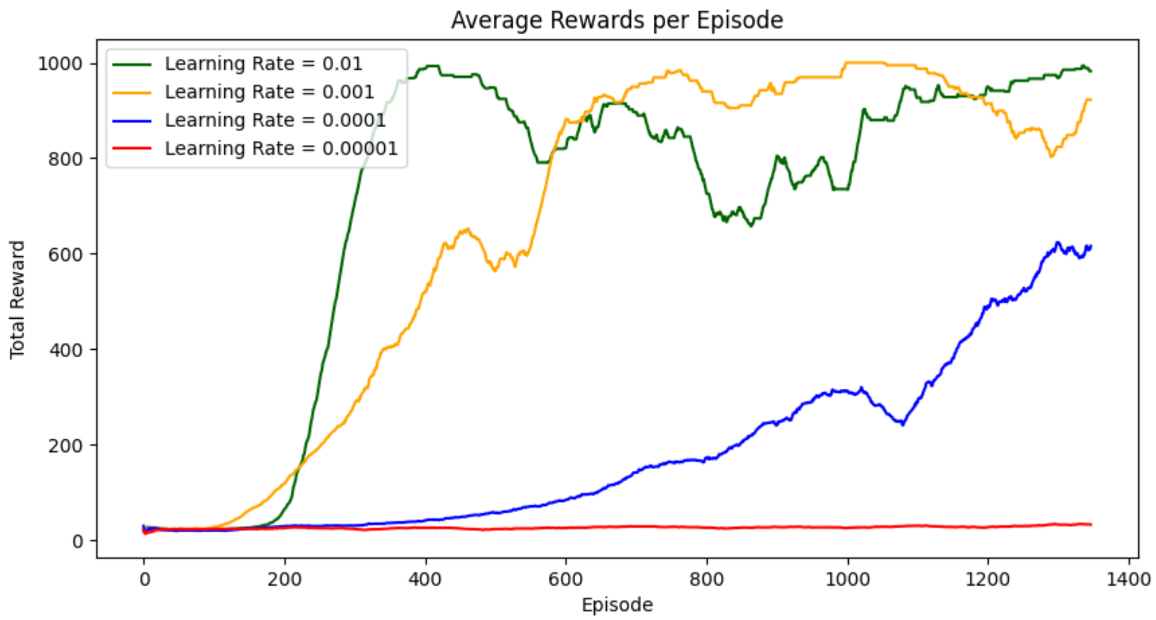
Reward: Each step incurs a reward of -1 until the goal is achieved.

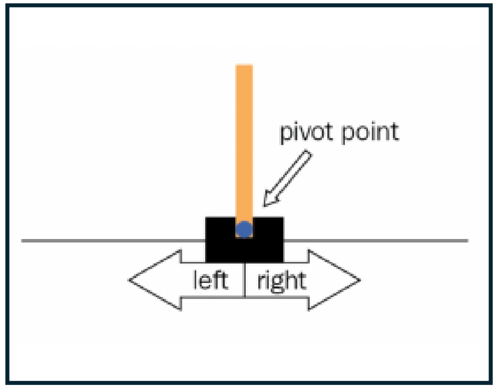
Actions:

- Apply -1 torque: Exerts a negative torque on the actuated joint.
- Apply 0 torque: Exerts no torque on the actuated joint.
- Apply 1 torque: Exerts a positive torque on the actuated joint

4.1 Training Results

CartPole





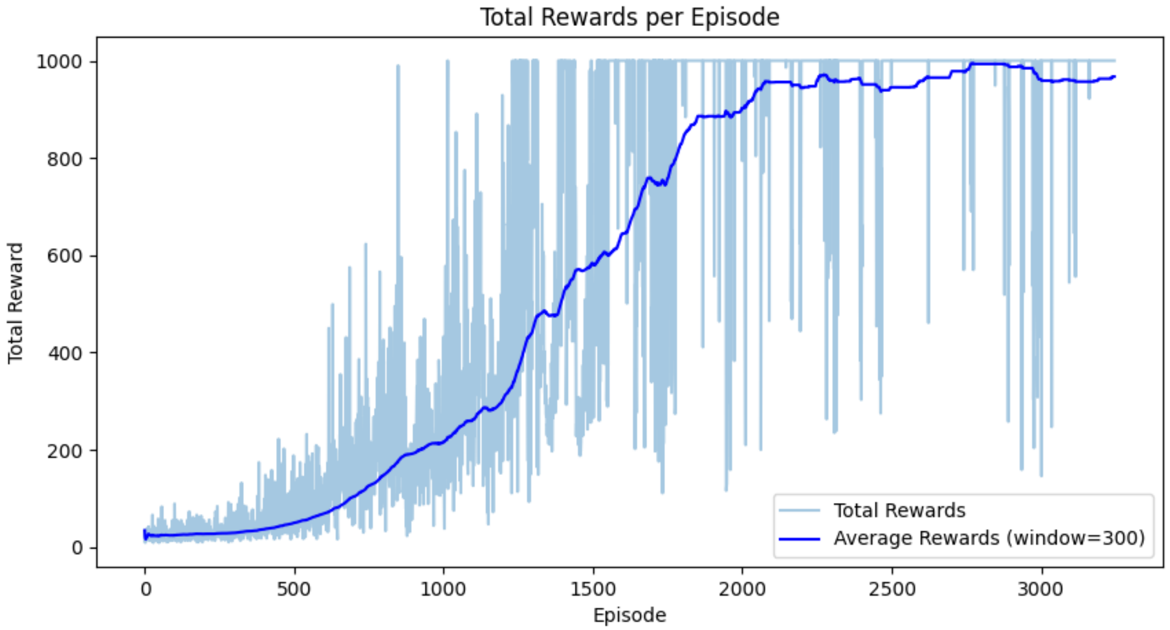
Goal: keep the pole upright for as long as possible.

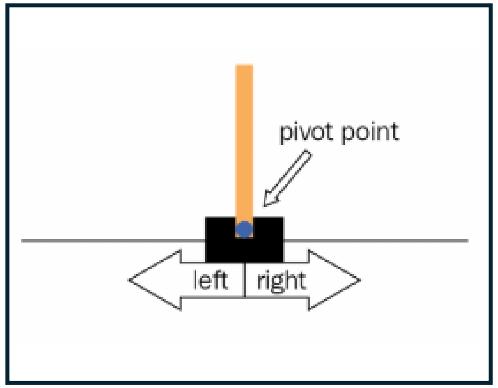
Reward: A reward of +1 is given for every step taken.

Actions:
- Action 0: Push cart to the left.
- Action 1: Push cart to the right.

4.1 Training Results

CartPole





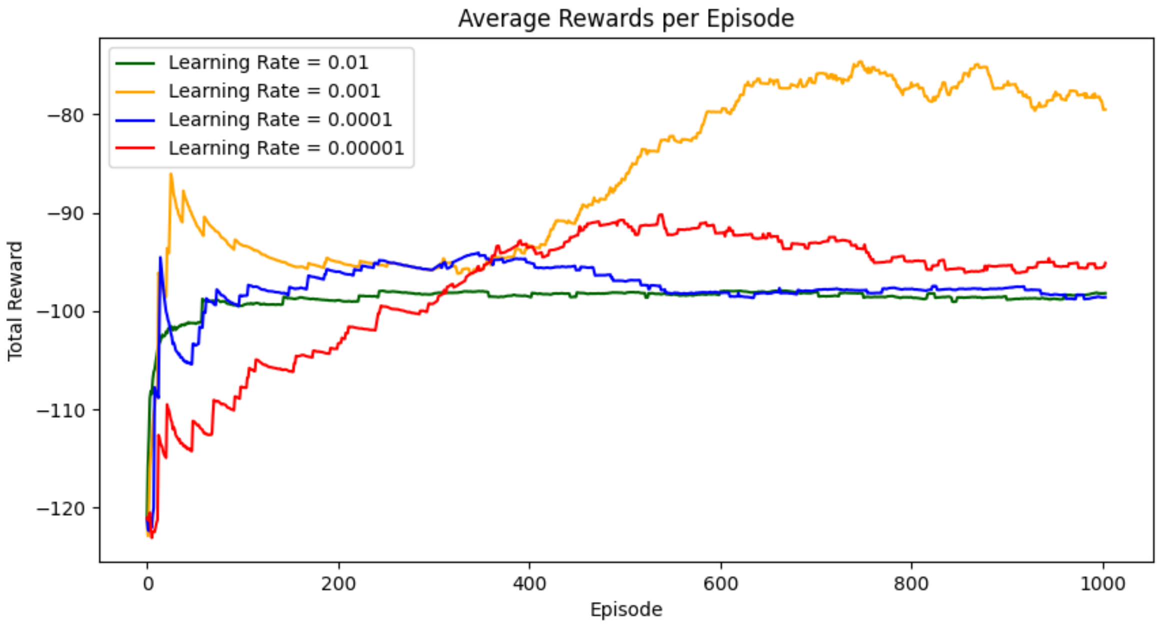
Environment	Timesteps	α	Time
CartPole	2M	0.001	0:20:10

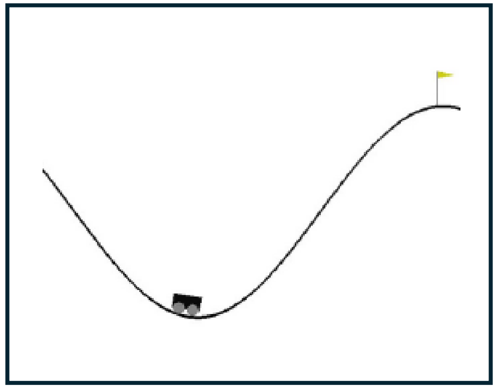
Reward: A reward of +1 is given for every step taken.

Actions:
- Action 0: Push cart to the left.
- Action 1: Push cart to the right.

4.1 Training Results

Mountain Car Continuous





Goal: Reach the flag (goal position) with the car.

Reward: A negative reward of $-0.1 * action^2$ is received at each timestep to penalize large magnitude actions. A positive reward of +100 is added if (position ≥ 0.45)

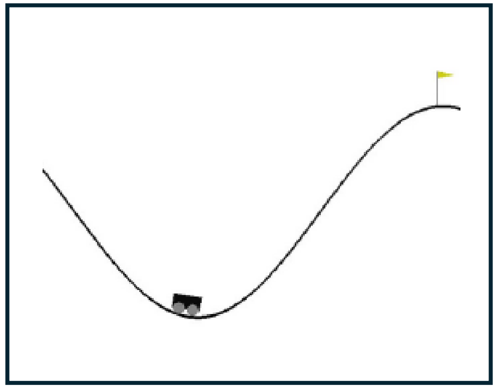
$$\text{velocity}_{t+1} = \text{velocity}_t + \text{force} \times \text{power} - 0.0025 \times \cos(3 \times \text{position}_t)$$

$$\text{position}_{t+1} = \text{position}_t + \text{velocity}_{t+1}$$

Action: Force ∈ [−1, 1]

4.1 Training Results

Mountain Car Continuous



Environment	Timesteps	α	Time
Mountain Car Continuous	3M	0.001	0:32:05

Reward: A negative reward of $-0.1 * action^2$ is received at each timestep to penalize large magnitude actions. A positive reward of +100 is added if (position ≥ 0.45)

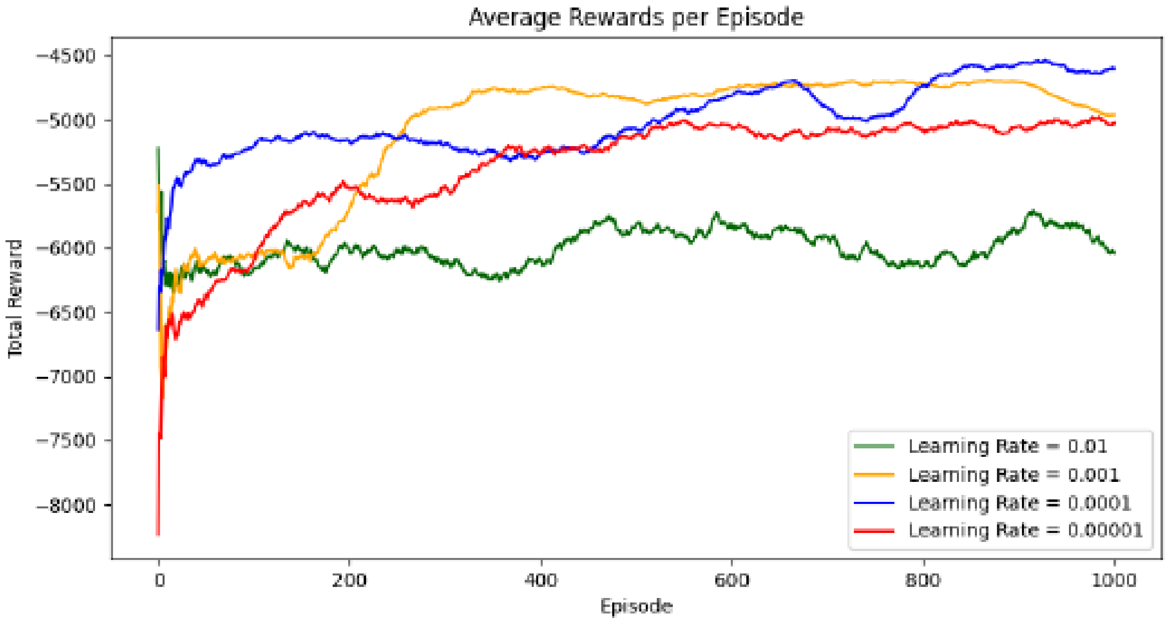
$$\text{velocity}_{t+1} = \text{velocity}_t + \text{force} \times \text{power} - 0.0025 \times \cos(3 \times \text{position}_t)$$

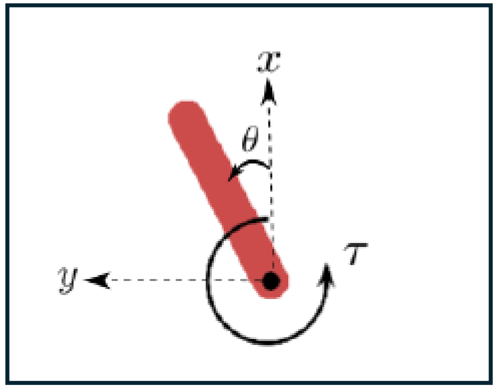
$$\text{position}_{t+1} = \text{position}_t + \text{velocity}_{t+1}$$

Action: Force ∈ [−1, 1]

4.1 Training Results

Pendulum





Goal: Apply torque on the free end to swing it into an upright position.

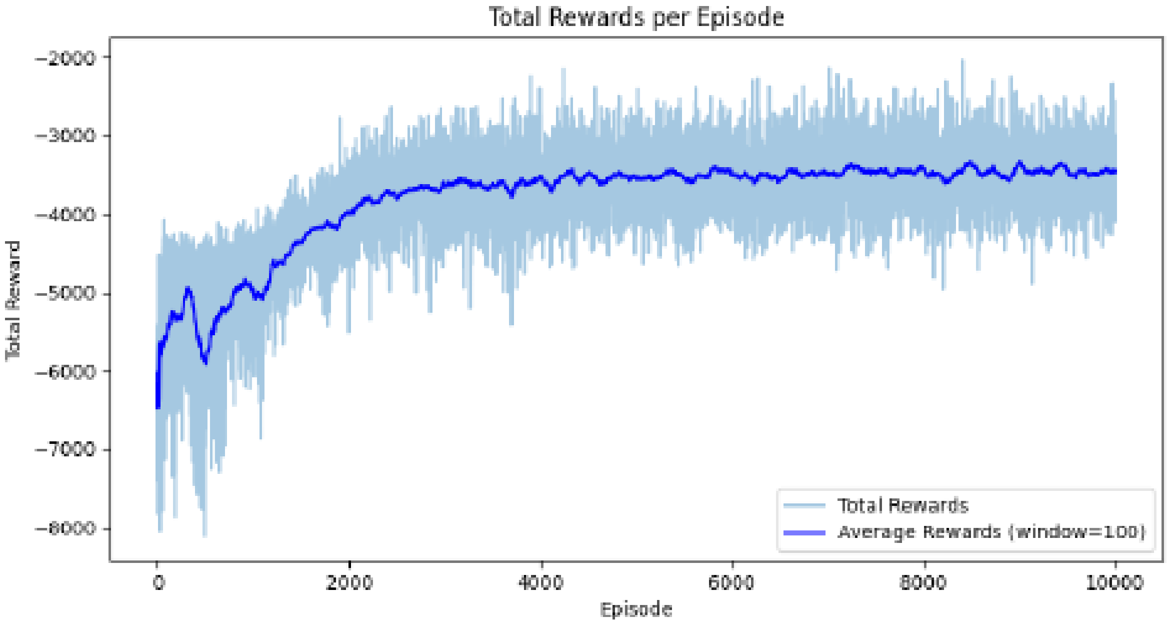
Reward: The reward function penalizes the square of the angle, the square of the angular velocity, and the square of the applied torque

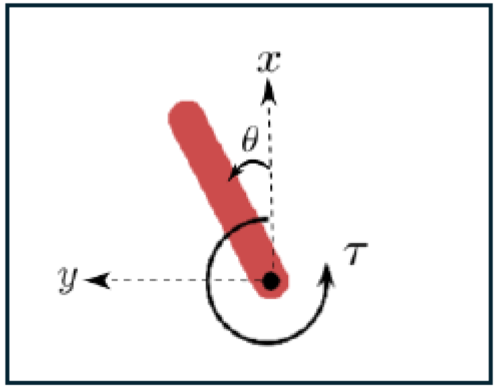
$$r = -(\theta^2 + 0.1 \cdot \theta_{\mathrm{dt}}^2 + 0.001 \cdot \mathrm{torque}^2)$$

Action: Torque ∈ [−1, 1]

4.1 Training Results

Pendulum





Environment	Timesteps	α	Time
Pendulum	4M	0.0001	1:04:39

Reward: The reward function penalizes the square of the angle, the square of the angular velocity, and the square of the applied torque

$$r = -(\theta^2 + 0.1 \cdot \theta_{\mathrm{dt}}^2 + 0.001 \cdot \mathrm{torque}^2)$$

Action: Torque ∈ [−1, 1]

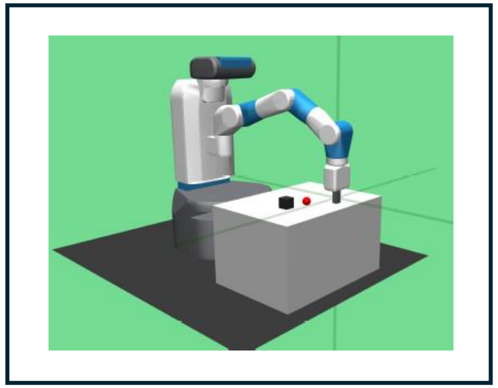
4.1 Training Results

Custom Pick And Place

HPC Cluster (SNOW)

Reward Types:

- *Sparse*: provide a binary feedback (-1 or 0).
- *Dense*: provide a continuous feedback based on the Euclidean distance between the achieved and desired goal positions.



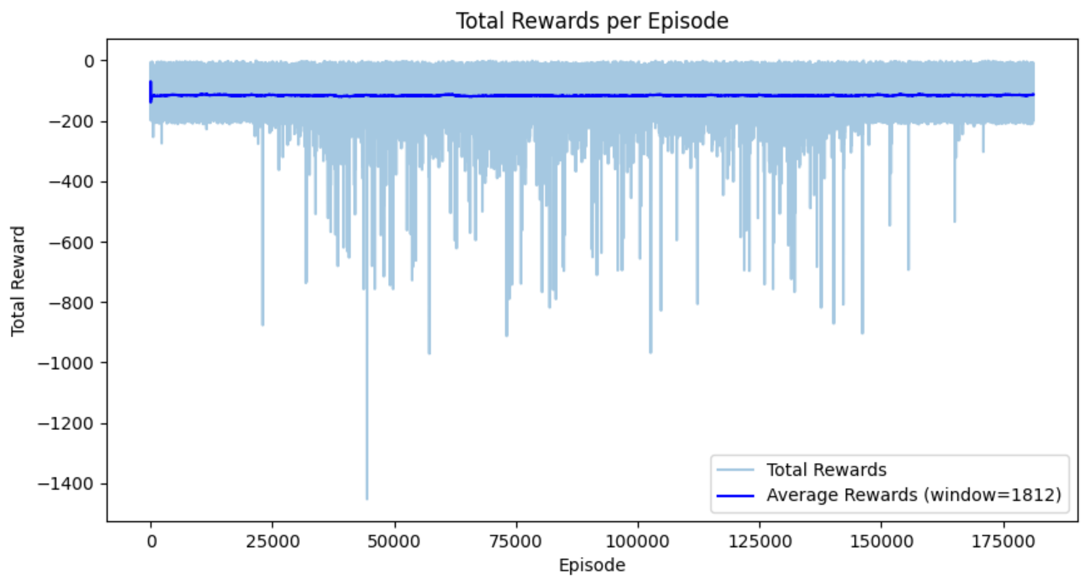
Goal: Pick up an object from a random location and place it in a random target location.

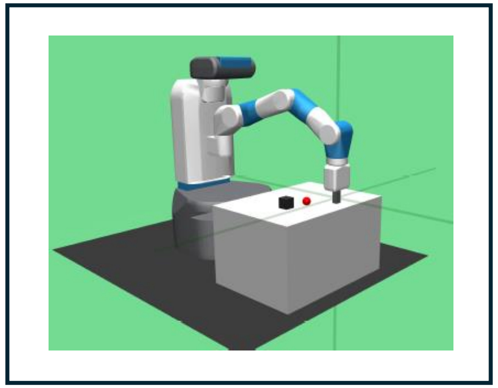
Actions:

- Displacement in the x direction (dx) ∈ [-1, 1]
- Displacement in the y direction (dy) ∈ [-1, 1]
- Displacement in the z direction (dz) ∈ [-1, 1]
- Gripper control ∈ [-1, 1]

4.1 Training Results

Custom Pick And Place





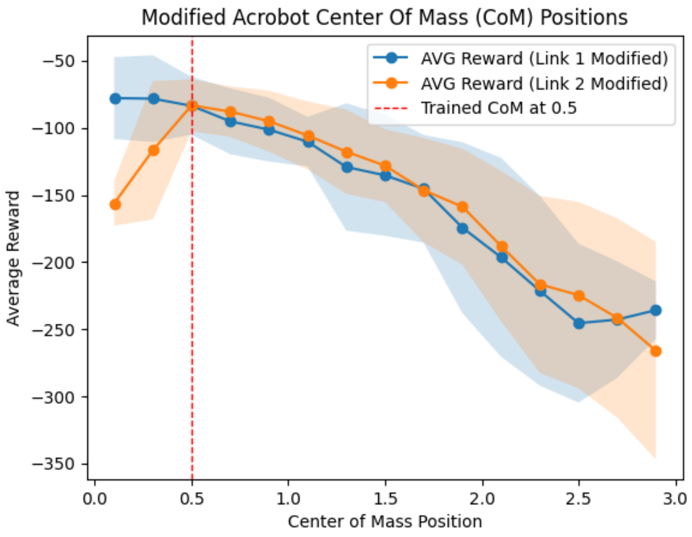
Environment	Timesteps	α	Time
Custom Pick and Place	200M	0.001	14 days

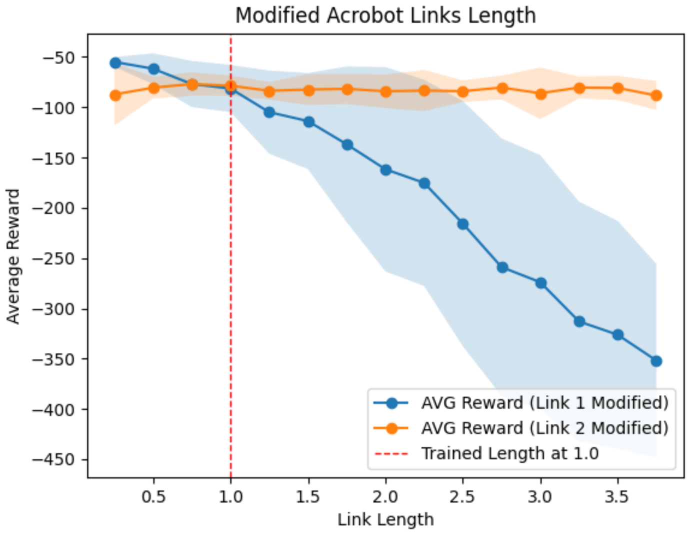
Dense Reward Types: provide a continuous feedback based on the Euclidean distance between the achieved and desired goal positions.

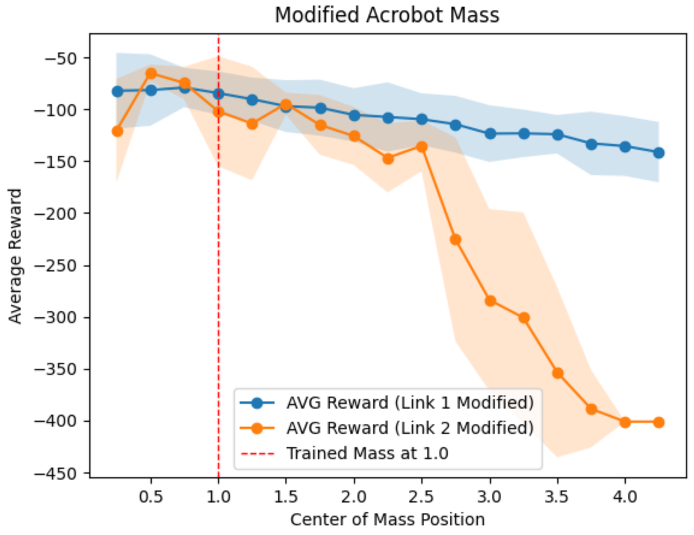
Actions:
- Displacement in the x direction (dx) ∈ [-1, 1]
- Displacement in the y direction (dy) ∈ [-1, 1]
- Displacement in the z direction (dz) ∈ [-1, 1]
- Gripper control ∈ [-1, 1]

4.2 Generalization Results

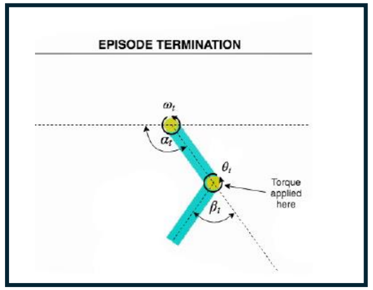
Custom Acrobot





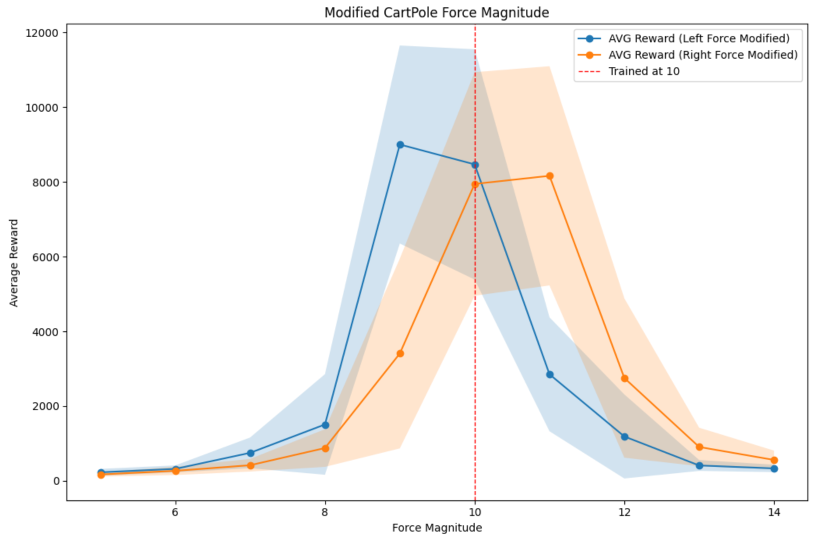


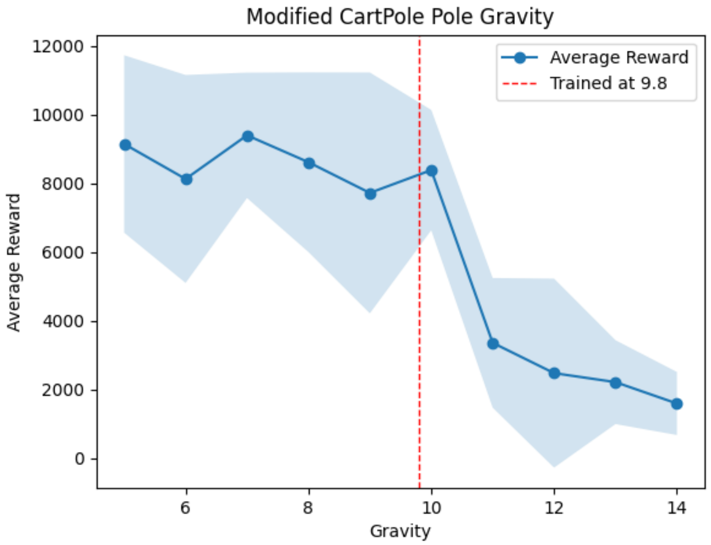
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Center of Mass	0.5	[0.1, 3.0]	0.2	400
Link Length	1.0	[0.25, 4.0]	0.25	400
Link Mass	1.0	[0.25, 4.5]	0.25	400



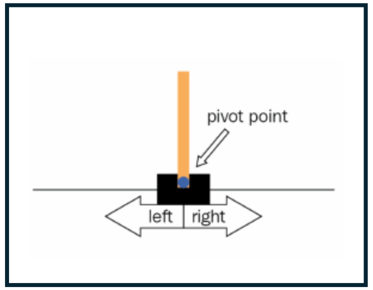
4.2 Generalization Results

Custom CartPole



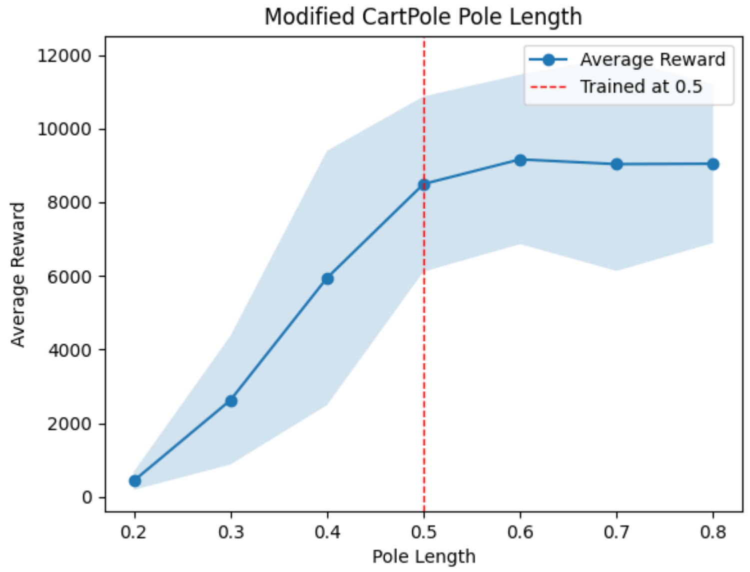


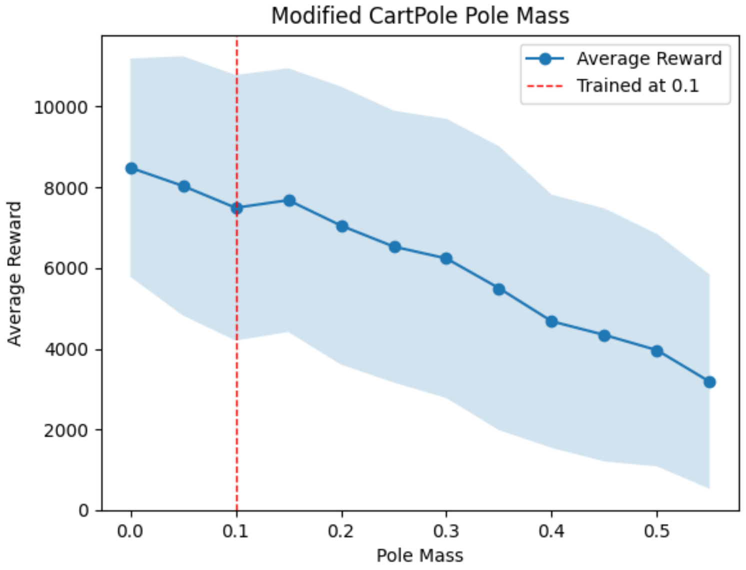
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Force Magnitude	**10**	**[5.0, 15.0]**	**1.0**	**10000**
Gravity	**9.8**	**[5.0, 15.0]**	**1.0**	**10000**
Pole Length	0.5	[0.2, 0.8]	0.1	10000
Pole Mass	0.1	[0.1, 0.6]	0.05	10000



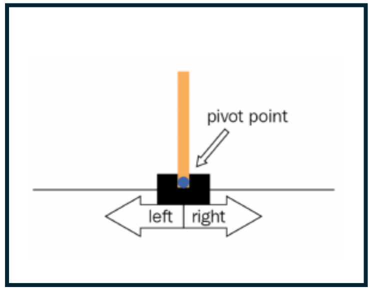
4.2 Generalization Results

Custom CartPole



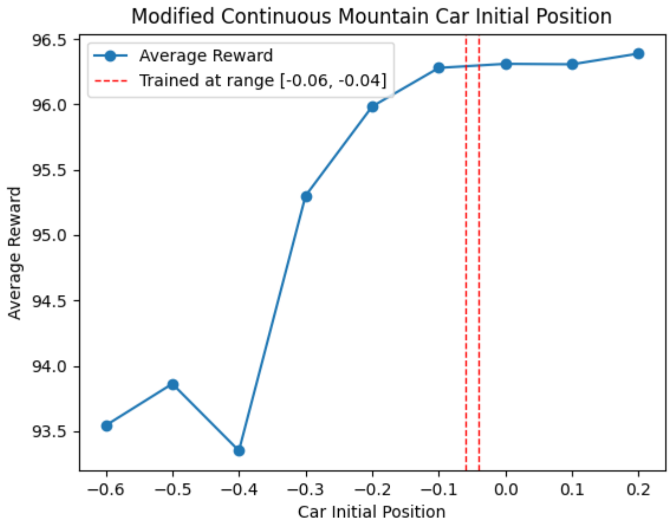


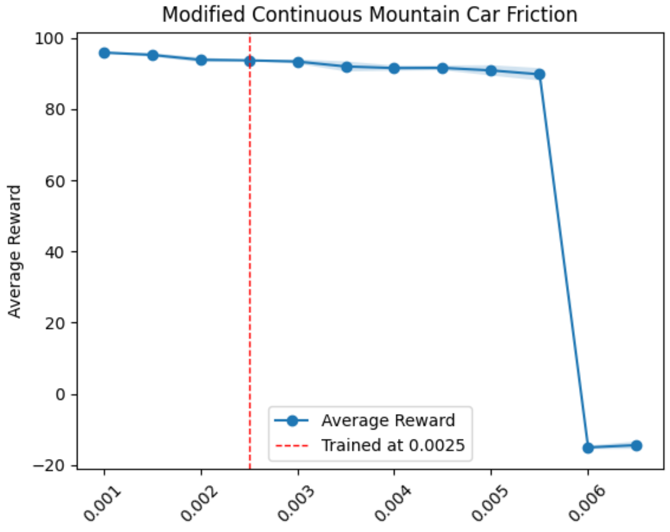
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Force Magnitude	10	[5.0, 15.0]	1.0	10000
Gravity	9.8	[5.0, 15.0]	1.0	10000
Pole Length	**0.5**	**[0.2, 0.8]**	**0.1**	**10000**
Pole Mass	**0.1**	**[0.1, 0.6]**	**0.05**	**10000**



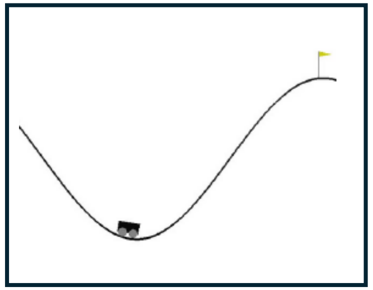
4.2 Generalization Results

Custom MountainCar Continuous



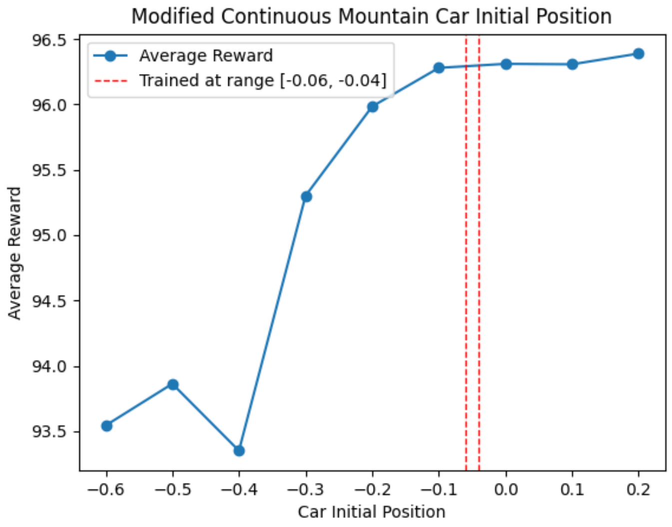


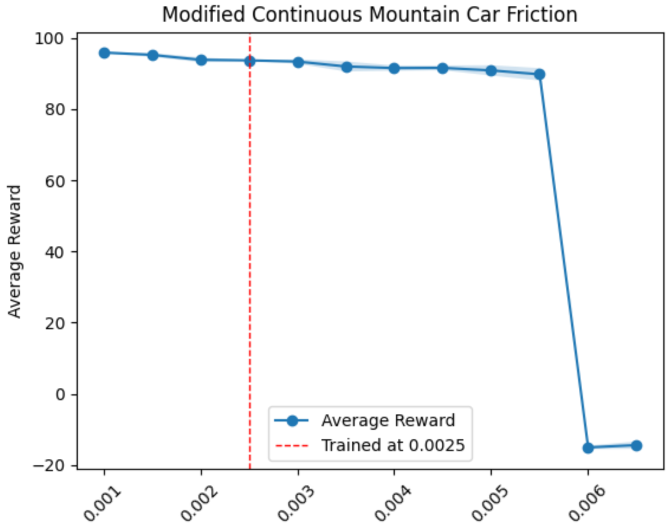
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Car Friction	**0.0025**	**[0.001, 0.005]**	**0.0005**	**300**
Initial Position	**[-0.04, -0.06]**	**[-0.6, 0.3]**	**0.1**	**300**
Car Power	0.0015	[0.0005, 0.003]	0.0005	300

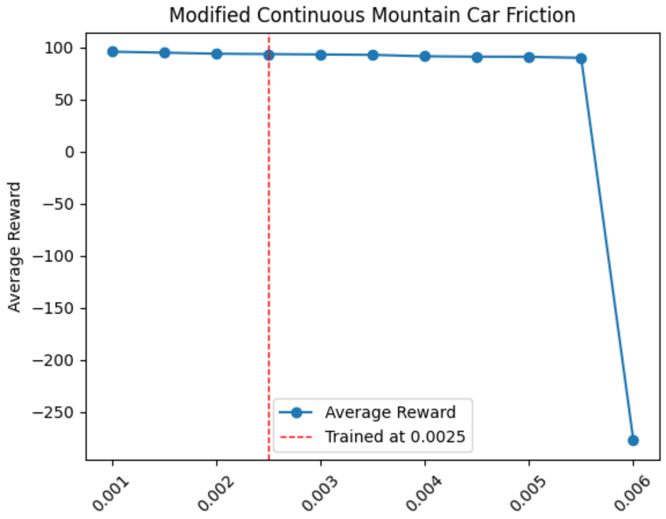


4.2 Generalization Results

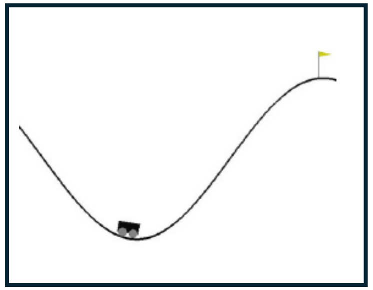
Custom MountainCar Continuous





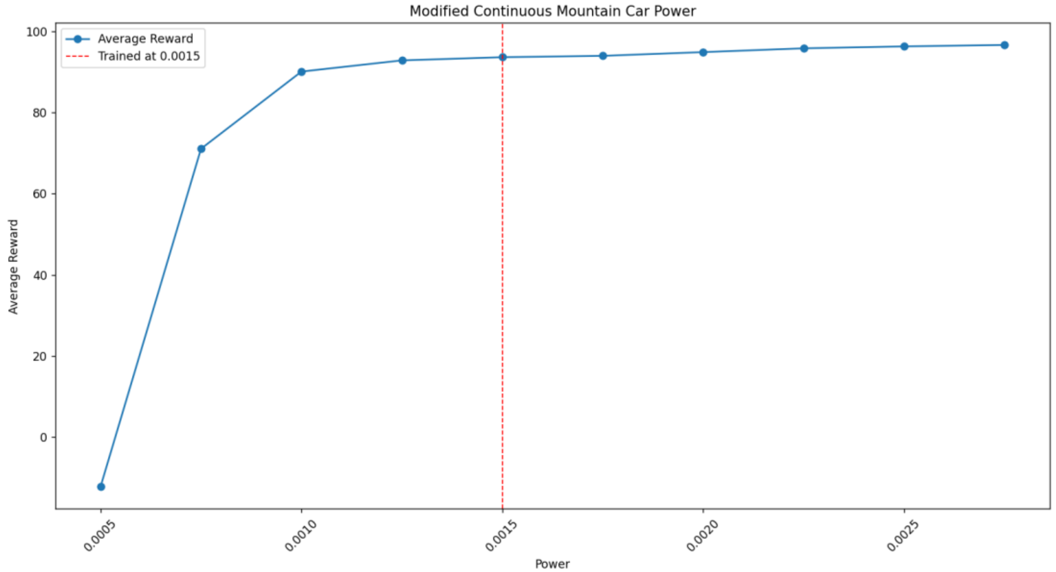


Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Car Friction	**0.0025**	**[0.001, 0.005]**	**0.0005**	**5000**
Initial Position	[-0.04, -0.06]	[-0.6, 0.3]	0.1	300
Car Power	0.0015	[0.0005, 0.003]	0.0005	300

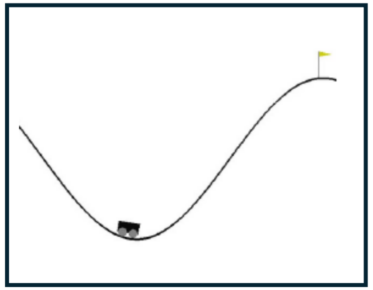


4.2 Generalization Results

Custom MountainCar Continuous

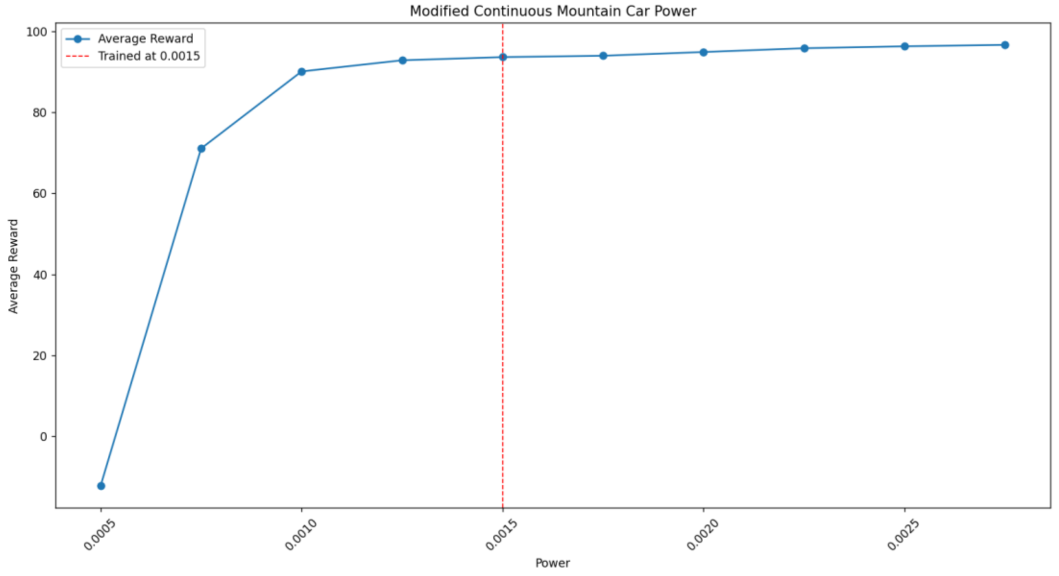


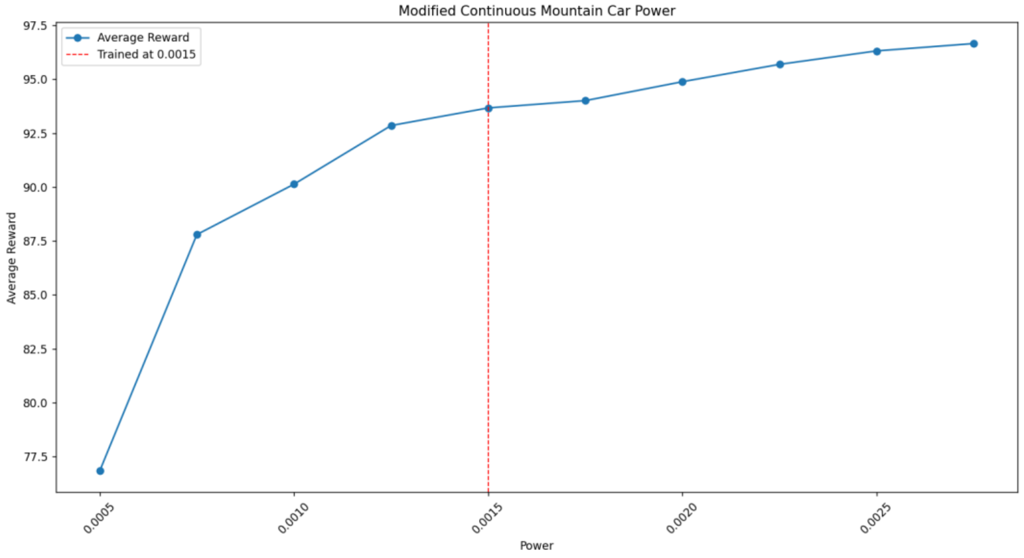
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Car Friction	0.0025	[0.001, 0.005]	0.0005	300
Initial Position	[-0.04, -0.06]	[-0.6, 0.3]	0.1	300
Car Power	**0.0015**	**[0.0005, 0.003]**	**0.0005**	**300**



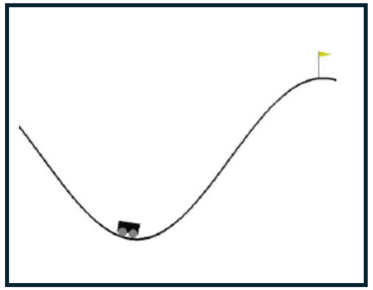
4.2 Generalization Results

Custom MountainCar Continuous



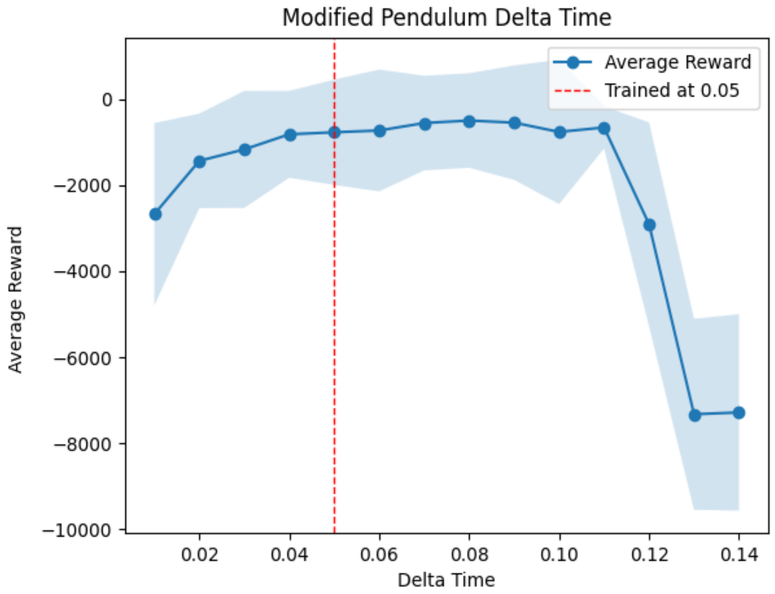


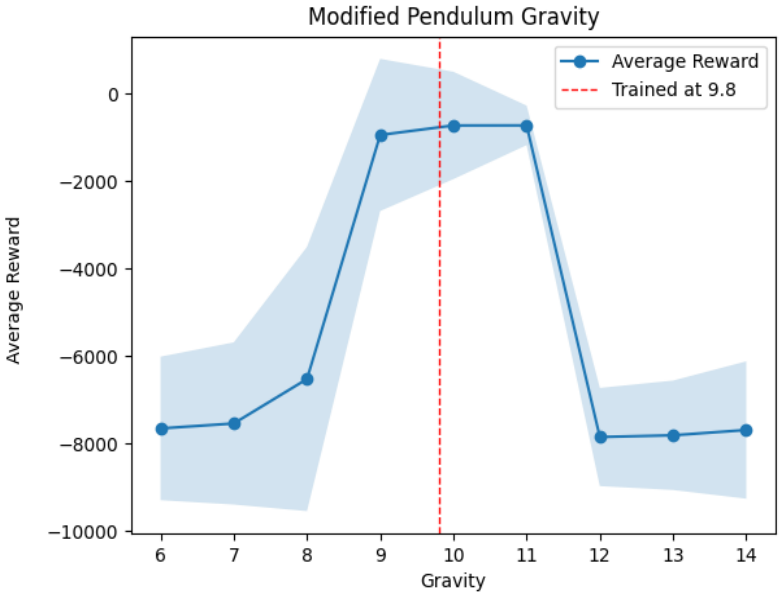
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Car Friction	0.0025	[0.001, 0.005]	0.0005	300
Initial Position	[-0.04, -0.06]	[-0.6, 0.3]	0.1	300
Car Power	**0.0015**	**[0.0005, 0.003]**	**0.0005**	**500**



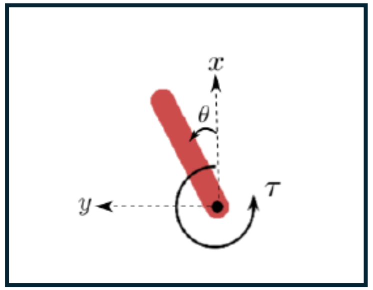
4.2 Generalization Results

Custom Pendulum



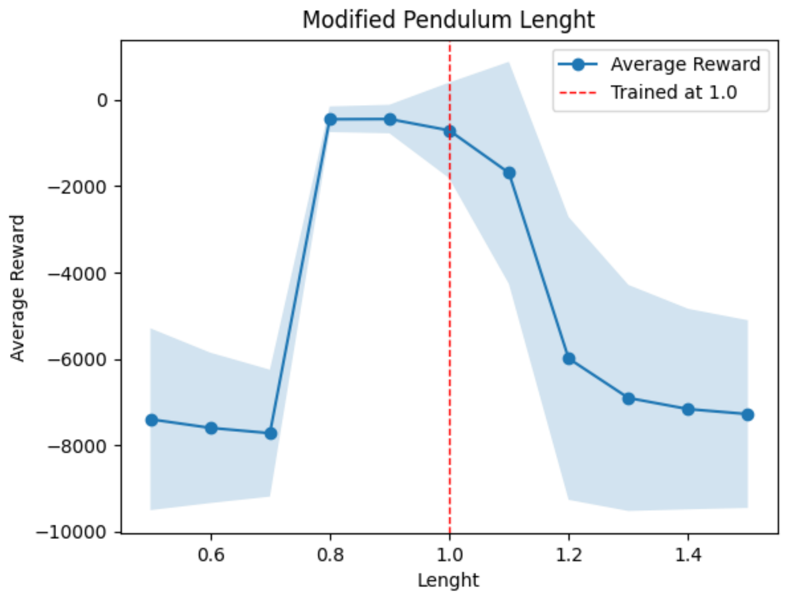


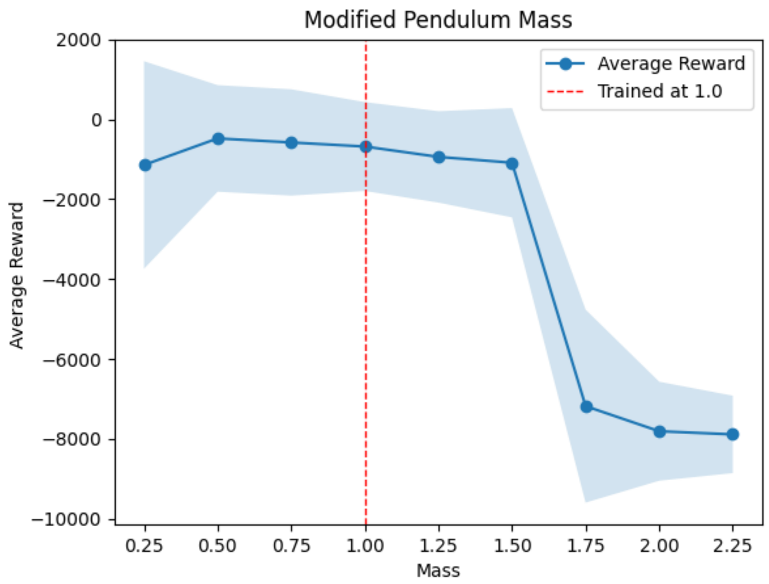
Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Delta Time	**0.05**	**[0.01, 0.15]**	**0.01**	**8000**
Gravity	**9.8**	**[5, 15]**	**1**	**8000**
Pendulum Length	1.0	[0.5, 1.6]	0.1	8000
Pendulum Mass	1.0	[0.25, 2.50]	2.5	8000



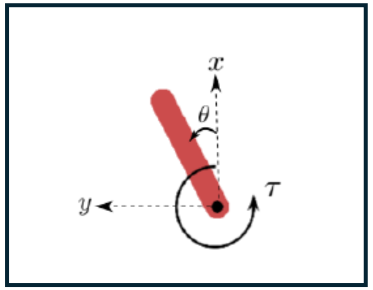
4.2 Generalization Results

Custom Pendulum





Feature Modified	Trained Value	Values Tested	Step Size	Max. Steps
Delta Time	0.05	[0.01, 0.15]	0.01	8000
Gravity	9.8	[5, 15]	1	8000
Pendulum Length	**1.0**	**[0.5, 1.6]**	**0.1**	**8000**
Pendulum Mass	**1.0**	**[0.25, 2.50]**	**2.5**	**8000**



OVERVIEW

01

INTRODUCTION

02

BACKGROUND THEORY

03

METHODOLOGY

04

EXPERIMENT EVALUATION

05

DISCUSSIONS, CONCLUSION & FUTURE WORK

5.1 Discussions & Conclusions

- **Initial Hipotesis**: Potential synergy between RL and DL, leveraging RL’s strengths in sequential decision making and DL’s capabilities in robust function approximation.

Control Tasks:

- **Performance**: DRL agents effectively learned and adapted using the PPO algorithm.
- **Key Factors:** Systematic hyperparameter tuning was crucial
- **Outcome:** High performance and consistent rewards.

Pick and Place Task:

- **Challenges:** High-dimensional state/action spaces, intricate physical interactions.
- **Results:** DRL agents struggled despite extensive training and environment simplifications.
- **Key Issues:** Difficulty in precise control and adaptability, large action space.

5.1 Discussions & Conclusions

- **Initial Hipotesis**: Potential synergy between RL and DL, leveraging RL's strengths in sequential decision making and DL's capabilities in robust function approximation.

<u>Control Tasks:</u>

- **Performance**: DRL agents effectively learned and adapted using the PPO algorithm.
- **Key Factors:** Systematic hyperparameter tuning was crucial
- **Outcome:** High performance and consistent rewards.

<u>Pick and Place Task:</u>

- **Challenges:** High-dimensional state/action spaces, intricate physical interactions.
- **Results:** DRL agents struggled despite extensive training and environment simplifications.
- **Key Issues:** Difficulty in precise control and adaptability, large action space.

<u>Generalization:</u>

•**Tests:** Agents faced parameter changes in trained environments.

•**Findings:** Mixed results; some robustness, but significant performance drops with substantial parameter alterations.

5.3 Future Work

Broader Environment Testing:

- Include more complex, dynamic scenarios.
- Expand beyond standard control tasks to environments with higher-dimensional state spaces, multi-agent interactions, and varied physical dynamics.

Advanced Learning Techniques:

- **Hierarchical Reinforcement Learning (HRL):** Decompose tasks into simpler sub-tasks.
- **Curriculum Learning:** Train on progressively more challenging tasks.
- **Meta-Learning:** "Learning to learn" for quicker adaptation to new tasks.
- **Domain Randomization:** Train in varied simulated environments to enhance robustness.

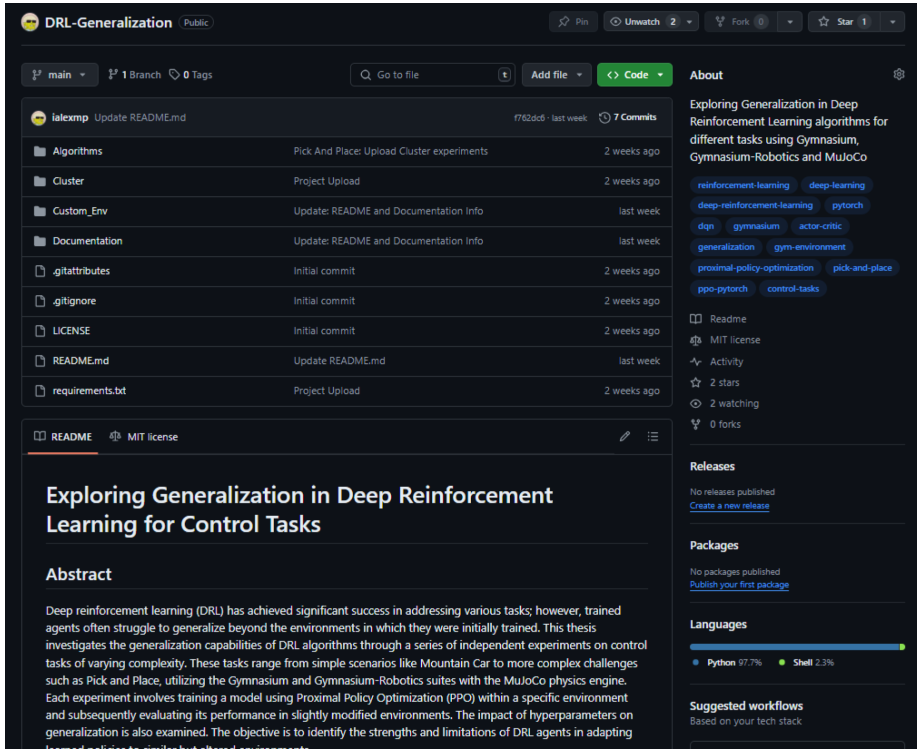
Transfer Learning:

- Leverage knowledge from one task/environment to improve performance on related tasks.

Hyperparameter Optimization:

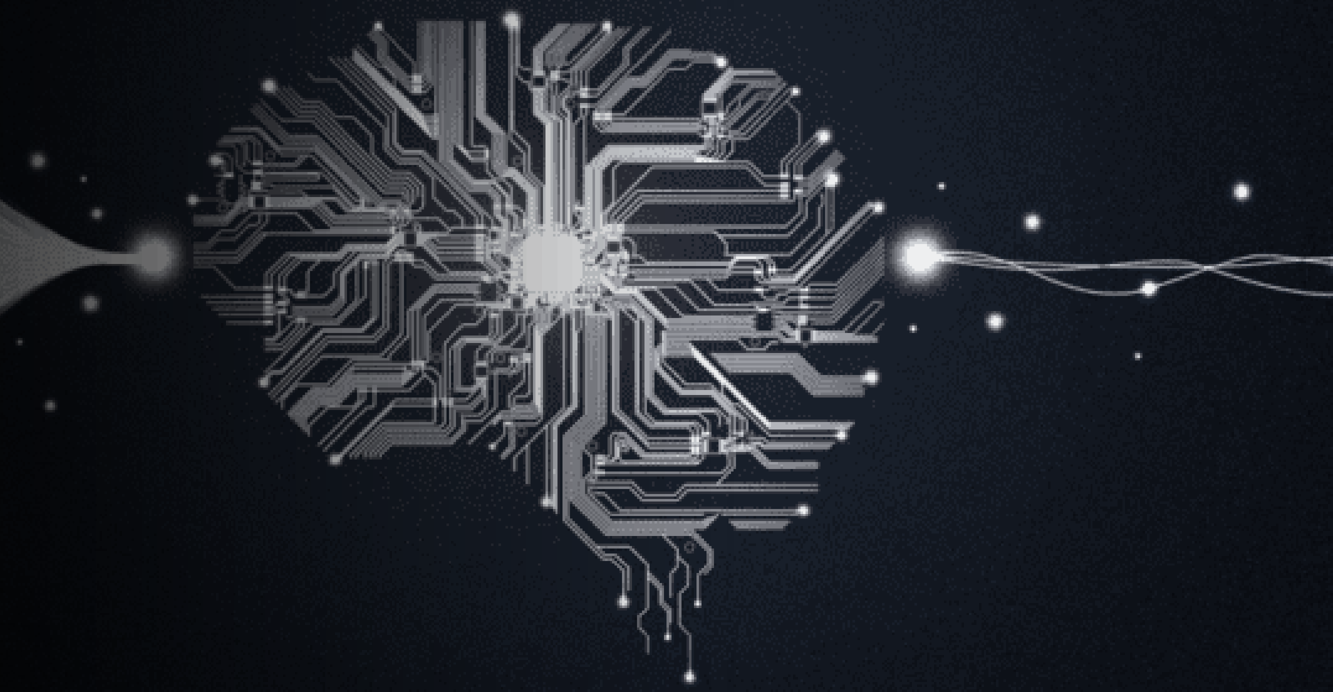
- Use **Bayesian optimization** and **AutoML** for efficient hyperparameter tuning.

Resources



Access to the repository : https://github.com/ialexmp/DRL-Generalization

Thanks For Your Attention



Àlex Montoya Pérez

Anders Jonsson & Sergio Calo Oliveira

Grau en Enginyeria en Informàtica

Universitat Pompeu Fabra Barcelona