ÉCOLE UNIVERSITAIRE
DE PHYSIQUE ET D'INGÉNIERIE
Université Clermont Auvergne

M2 Project Report

Evaluation of Transfer Learning Applied on 2D Robotic Arms

Jérôme NORTIER

Master 2 - Perception Artificielle et Robotique

October 2025 - March 2026

Abstract

This report presents a preliminary study on transfer learning between robots with differing morphologies, conducted in preparation for an upcoming internship on human-to-robot skill transfer. The study investigates a 'Full-Policy Transfer' approach using trajectory correspondence between two distinct entities: a 2-Degree-of-Freedom (2DoF) source and a 3-Degree-of-Freedom (3DoF) target robotic arm.

Using a 2D planar manipulator performing a reaching task as a benchmark, the agent learns incremental joint control policies via Proximal Policy Optimization (PPO), establishing a stable baseline for transfer. Aligned trajectories from this task enable the mapping of observations and actions across morphologies. This approach leverages cycle-consistency to ensure compatibility between disparate state-action spaces, laying the groundwork for transferring complex behaviors such as object manipulation.

While full transfer experiments are ongoing, the current results validate the environment design, the training framework, and the trajectory alignment methodology. These findings provide a robust foundation for extending the framework to human demonstrations, aiming to reduce training time and computational costs in real-world robotic applications.

Contents

I - Introduction

I.1 Context

Deep reinforcement learning (RL) enables autonomous agents to acquire complex behaviors through interaction with their environment [2]. However, RL training is computationally expensive and time-intensive, requiring extensive data collection and careful hyperparameter tuning. To address these challenges, transfer learning can accelerate learning by reusing knowledge from one context in another.
Included in a study on human-to-robot transfer (pursued in my upcoming internship), this project focuses on robot-to-robot transfer learning between 2-Degree-of-Freedom (2DoF) and 3-Degree-of-Freedom (3DoF) planar robotic arms. The concrete objective is to transfer a full policy trained with Proximal Policy Optimization (PPO) on a 2DoF source robot to a 3DoF target robot via trajectory correspondence. We expect the transferred policy to reach comparable task performance on the target (measured by success rate and final cumulative reward) within significantly fewer environment interactions than training from scratch. The proposed method emphasizes trajectory alignment and cycle-consistent observation/action mappings to preserve task-relevant structure across morphologies.

I.2 Objectives

The primary objective of this work is to evaluate the effectiveness of transfer learning methods applied to 2D robotic manipulators. Specifically, the project aims to:

- Transfer a policy learned by a 2DoF robotic arm using Proximal Policy Optimization (PPO) to a 3DoF robotic arm and report quantitative metrics (final success rate, cumulative reward, and sample-efficiency relative to a from-scratch baseline).
- Validate the trajectory alignment methodology (observation mapping T_s and action mapping T_a) to enable full-policy transfer without retraining the policy head.
- Measure the performance of the transferred policy in terms of task success rates, cumulative reward, and reduction in required environment interactions (sample efficiency).

These objectives lay the groundwork for future research on human-to-robot skill transfer, with the potential to reduce training costs and enhance the adaptability of robotic systems.

II - Theoretical background

II.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of **Machine Learning** where an agent learns to make decisions by interacting with an environment. Unlike supervised learning, RL does not rely on labeled input-output pairs; instead, the agent receives feedback in the form of *rewards* based on the actions it takes. The goal of the agent is to learn a policy that maximizes the cumulative reward over time.

In the RL framework, the environment is typically modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the set of states representing the environment's configuration.
- $\mathcal{A}$ is the set of actions the agent can take.
- $P(s'|s, a)$ is the transition probability from state s to s' given action a.
- $R(s, a)$ is the reward received after taking action a in state s.
- $\gamma \in [0, 1]$ is the discount factor, weighting future rewards.

The agent observes the current state of the environment through the observations it receives, selects an action according to its policy $\pi(a|s)$, receives a reward, and transitions to a new state. This interaction loop is illustrated in Figure 1.

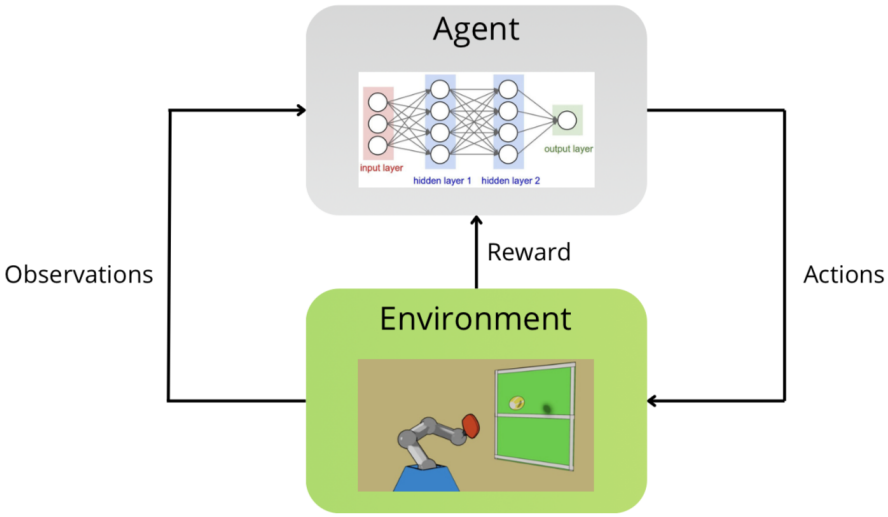

Figure 1: Schematic representation of a reinforcement learning setup. The agent (MLP) receives observations from the environment (robot playing tennis), selects actions, and receives rewards to guide learning.

Formally, the agent aims to find an optimal policy π^* that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big].$$

Reinforcement learning can be applied to discrete or continuous action spaces and is widely used in robotics, games, and control tasks. In robotics, the states often include sensor readings or joint configurations, actions correspond to motor commands, and the reward reflects task performance, such as reaching a target or successfully manipulating an object.

II.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a state-of-the-art **policy gradient** method in reinforcement learning introduced by Schulman et al. [1]. It is designed to improve the stability and efficiency of policy updates while avoiding large destructive changes that can occur with naive policy gradient methods.

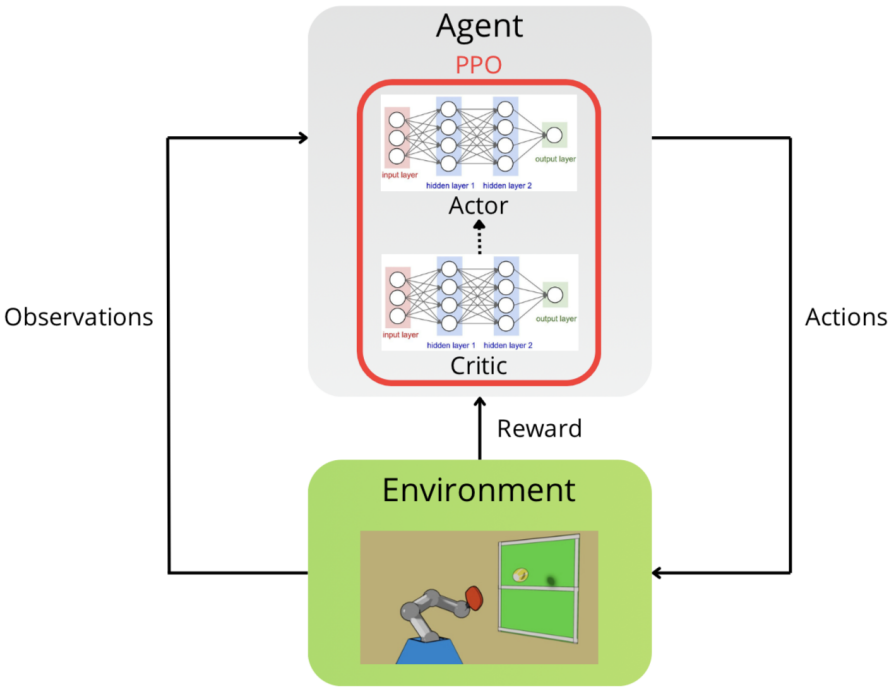

Figure 2: Proximal Policy Optimization in a reinforcement learning setup. The agent uses a PPO policy where the critic evaluate the action/state to help the actor improves the policy.

Like standard RL, PPO operates within the agent-environment loop (see Figure 2), but it introduces a clipped objective function that constrains policy updates. Formally, the PPO objective for a single timestep is:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big],$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies.

- $\hat{A}_t$ is the advantage estimate at timestep t.
- ϵ is a small hyperparameter controlling the clipping range.

In reinforcement learning, the agent cannot directly backpropagate through the environment because the dynamics are unknown and the reward signal is often sparse. Actor-Critic methods, such as PPO, address this by introducing a critic that estimates the value of states or advantages. This stabilizes learning: the actor focuses on improving the policy, while the critic provides a reliable evaluation of expected returns, guiding updates and reducing variance.

PPO further constrains policy updates through clipping, ensuring stable and efficient learning even in continuous or complex tasks. Therefore, it is well-suited for systems performing dynamic tasks, involving multiple actuators and continuous control. It can handle nonlinear system
and learn coordinated actions across several degrees of freedom (e.g., manipulation, locomotion, or multi-joint coordination) in robotic systems.

II.3 State of Transfer Learning

Training deep reinforcement learning agents is often **time-consuming and computationally intensive**, especially for complex robotic tasks. Some tasks may require millions of interactions with the environment to reach acceptable performance. Transfer learning offers a solution to reduce or even eliminate the need for full training on a new task, leveraging knowledge acquired from previous experiences to accelerate learning.

The main goal of transfer learning is to **reuse and adapt prior knowledge** to a related target task. This can involve fine-tuning a pre-trained policy, reusing learned representations, or guiding exploration, all aimed at reducing training time and computational cost, while maintaining or improving task performance.
An overview of different transfer learning approaches is illustrated in Figure 3, organized by the type of knowledge being transferred.

The main approaches can be summarized as follows:

- **Learning from Demonstration (LfD):** This approach leverages trajectories or behaviors demonstrated by an expert (human or pre-trained agent) to guide the learning process of the target agent. By initializing the policy closer to successful behaviors and constraining exploration to relevant regions of the state-action space, LfD can significantly reduce the number of interactions required to reach effective performance. It is particularly useful in robotics where unsafe exploration can be costly or damaging.
- **Policy Transfer:** Policy transfer involves directly reusing a pre-trained policy from a source task, optionally followed by fine-tuning in the target task. This approach assumes that the source and target tasks share similar dynamics or objectives, allowing the agent to start from a knowledgeable policy rather than random initialization. Fine-tuning helps adapt to differences in the target environment while preserving useful prior behaviors.

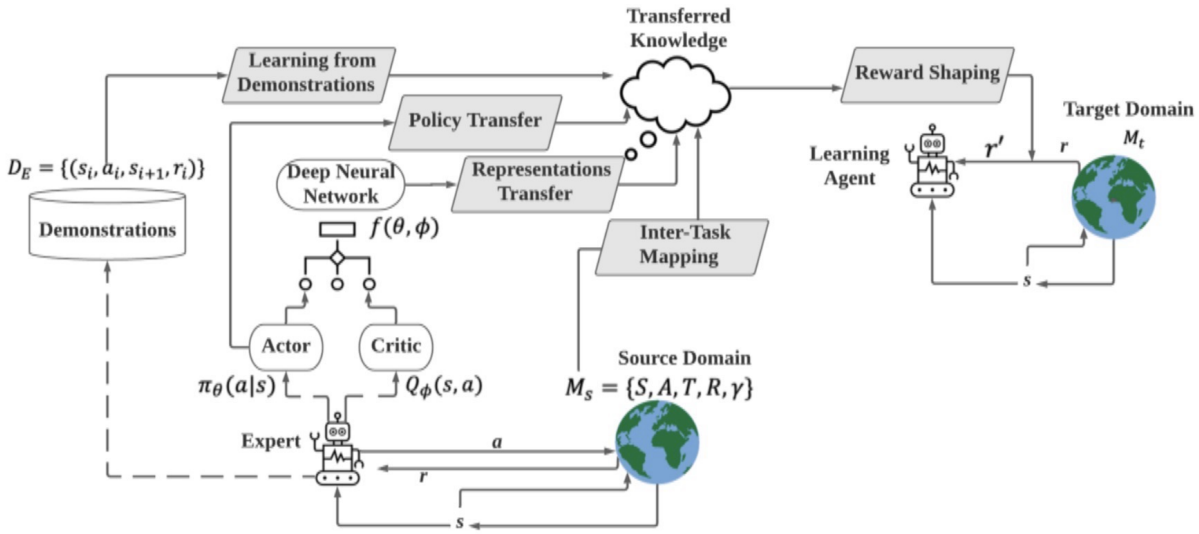

Figure 3: An overview of different transfer learning approaches, organized by the format of transferred knowledge.

- **Representation Transfer:** In this method, learned features, latent embeddings, or state representations from the source task are transferred to the target task. By providing a compact and informative representation of the environment, representation transfer reduces the learning complexity in the target domain, especially when observations are high-dimensional or partially observable, as is common in robotic vision or sensor-based control.

- **Reward Shaping:** Reward shaping uses knowledge from the source task to design additional or auxiliary reward signals in the target task. By providing denser and more informative feedback, it guides the agent toward desired behaviors faster than relying solely on sparse or delayed rewards. This approach can accelerate convergence and improve exploration efficiency, particularly in tasks where the main reward is difficult to obtain.

- **Inter-Task Mapping:** Inter-task mapping establishes explicit correspondences between the states and actions of source and target domains, allowing knowledge to be transferred between agents with different morphologies or kinematics. Mapping functions can be learned or manually defined, and they enable transfer across structurally distinct robots, facilitating applications such as robot-to-robot or human-to-robot transfer.

These approaches provide a flexible toolkit to adapt prior knowledge across tasks or domains, enabling more efficient learning and, in some cases, immediate transfer without additional training (zero-shot transfer).

II.4 Universal Neural Network (UNN) and State Abstraction

The Universal Neural Network (UNN) framework is based on the idea that transfer between robots with different morphologies can be achieved through a shared *abstract state space*.

Instead of directly transferring policies or raw state representations, the method introduces an intermediate latent representation that captures task-relevant features while filtering out morphology-specific details.

Formally, let $s_i \in \mathcal{S}_i$ denote the state of robot i. UNN defines an abstraction function:

$$\phi_i : \mathcal{S}_i \to \mathcal{Z}$$

where $\mathcal{Z}$ is a shared latent space across robots. The key assumption is that, although the robots may differ in kinematics, degrees of freedom, or actuation, they can share a common representation of the task in $\mathcal{Z}$.

The policy is then defined in the abstract space:

$$\pi : \mathcal{Z} \to \mathcal{A}_i$$

This decouples morphology-specific state encoding from task-level decision making. As a result, knowledge acquired on one robot can be transferred to another by learning only the abstraction function ϕ_j for the new robot, while keeping the shared task-level structure.

In practice, UNN framework operates through three main components:

1. A robot-specific encoder ϕ_i mapping raw states to the shared latent space.
2. A shared task module operating in the latent space (UNN).
3. A robot-specific decoder or policy head producing actions.

During training on a source robot, both the robot-specific encoder and decoder, along with the shared task module, are optimized jointly. When transferring to a target robot, the shared task module is reused, while only the target encoder (state abstraction) and the final action layer are trained or fine-tuned.

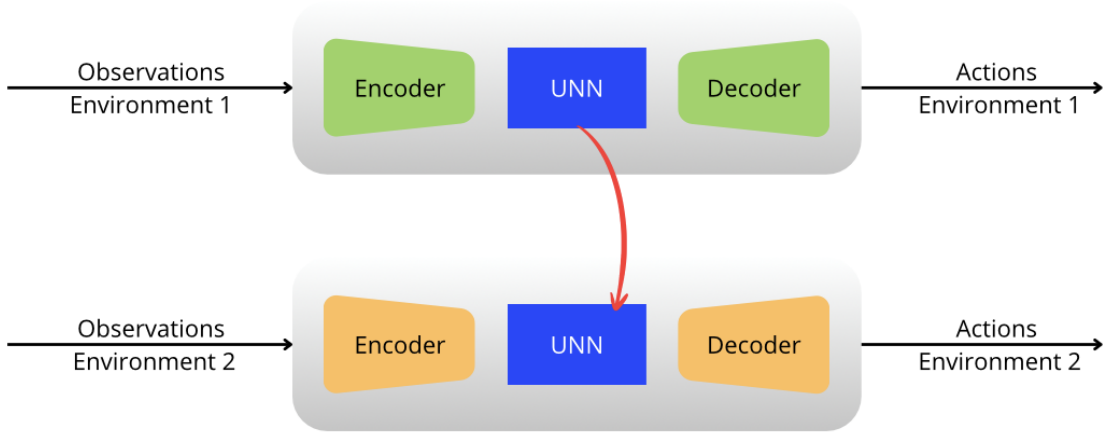

Figure 4: UNN architecture illustrating robot-specific encoders, shared latent task space (UNN), and robot-specific action heads.

The central idea is that the latent space enforces structural consistency between robots at the task level, allowing transfer even when their state and action dimensions differ. This abstraction reduces the mismatch between source and target domains and avoids direct mapping of heterogeneous state vectors.

The main advantage of UNN lies in its ability to handle robots with different morphologies through state abstraction, making it particularly relevant for robot-to-robot transfer. It promotes modularity and reduces the need to retrain a full policy from scratch.

However, its performance strongly depends on the quality of the learned latent space. If the abstraction fails to preserve task-relevant state variables and control-related dynamics, transfer performance may degrade. Additionally, learning appropriate encoders for significantly different robot morphologies may still require non-negligible data and training effort.

III - Methodology: Full-Policy Transfer via Trajectory Correspondence

III.1 Theoretical Framework

Our approach aims to directly transfer a full PPO-trained agent from a source robot to a target robot with a different morphology, without learning an intermediate latent space. The key idea is to leverage trajectory correspondences between the two environments, allowing observations and actions of one robot to be projected onto the other while preserving task-relevant dynamics.

Formally, let $\mathcal{S}_i$ and $\mathcal{A}_i$ denote the observation and action spaces of robot i, and let π_θ be the policy trained on the source robot. We define two mappings:

$$T_s : \mathcal{S}_{target} \to \mathcal{S}_{source}, \quad T_a : \mathcal{A}_{source} \to \mathcal{A}_{target}$$

These mappings are derived from aligned trajectory pairs obtained in a fundamental task (reaching). The transferred policy is then applied as follows:

$$\tilde{s}_{source} = T_s(s_{target}), \quad a_{source} = \pi_\theta(\tilde{s}_{source}), \quad a_{target} = T_a(a_{source})$$

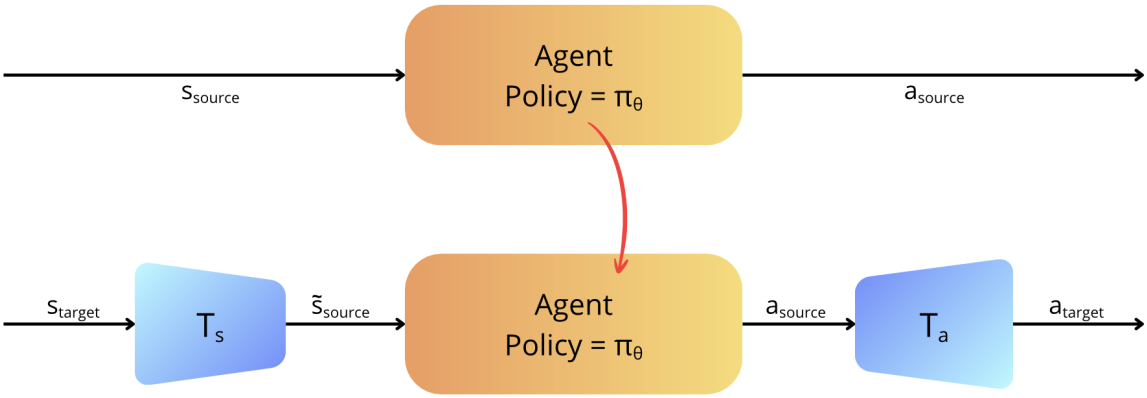

Figure 5: Full-policy transfer pipeline illustrating trajectory alignment and cycle-consistent observation/action mappings between source and target robots.

III.2 Trajectory Alignment Procedure

Fundamental Task: Reaching Both source (2-DoF) and target (3-DoF) robots are trained on a reaching task in identical environments. The following alignment is performed:

- The same random seed ensures identical target positions across environments.
- Timesteps are synchronized between the two robots.
- Joint angles are normalized to allow correspondence between differing morphologies.

From this, we record trajectory pairs:

$$(s_t^{\text{2DoF}}, s_t^{\text{3DoF}}), \quad (a_t^{\text{2DoF}}, a_t^{\text{3DoF}})$$

with the following observations and actions:

- Observations (Reaching): joint angles, end-effector position, target position, distance end-effector/target.
- Actions (Reaching): joint angular velocities.

These trajectories form the basis for constructing the observation and action mappings T_s and T_a. We enforce cycle-consistency during mapping construction: projecting a target state to the source and back should approximately reconstruct the original state, i.e. $T_s^{-1}(T_s(s_{target})) \approx s_{target}$ (and similarly for actions T_a). This constraint improves invertibility and preserves task-relevant structure across morphologies.

Transfer Task: Pushing Once trajectory correspondences are established, we apply the method to the more complex pushing task. Observations and actions are as follows:

- Observations (Pushing): joint angles, end-effector position, ball position, target position, distance end-effector/ball, distance ball/target.
- Actions (Pushing): joint angular velocities.

The expert policy trained on the source robot is applied to the target robot via the mappings:

$$\tilde{s}_{\text{2DoF}} = T_s(s_{\text{3DoF}}), \quad a_{\text{2DoF}} = \pi_\theta(\tilde{s}_{\text{2DoF}}), \quad a_{\text{3DoF}} = T_a(a_{\text{2DoF}})$$

This allows the target robot to execute the task using the policy learned by the source robot, while accounting for morphological differences.

IV - Experimental Setup

IV.1 Overview of the Experimental Framework

The objective of this experimental study is to evaluate a transfer learning framework between morphologically distinct robotic manipulators. The approach relies on learning a control policy on a source robot and transferring the acquired skill to a structurally different target robot through trajectory correspondence.

The experimental protocol follows a progressive and structured sequence:

- First, a 2DoF planar manipulator is trained on a reaching task. This task serves as a fundamental skill, allowing the policy to implicitly learn inverse kinematics and spatial goal-directed control through reinforcement learning.
- Second, the framework is extended to a 3DoF manipulator with the objective of generating equivalent trajectories for the same reaching behavior. The additional degree of freedom introduces kinematic redundancy, enabling the study of structural correspondence between robots with different morphologies.
- Third, a more complex manipulation task is introduced. This task corresponds to the desired target application for transfer.
- Finally, the trajectories generated by both robots are used to construct a transfer mapping. The objective is to align behaviors in a task-consistent manner and evaluate whether structural correspondences enable skill transfer across different kinematic configurations.

All experiments are conducted in a purely kinematic simulation environment. Dynamics, inertia, and contact forces are intentionally excluded to focus on the geometric structure of the robots, representation alignment, and policy behavior. The reaching task, while simple, provides a minimal yet non-trivial benchmark that captures essential aspects of robotic control, including nonlinear kinematics, joint coupling, and goal-directed motion. It also establishes a well-defined basis for trajectory alignment, which is required for subsequent transfer experiments.

IV.1.1 Robot Kinematic Model

The source robot is a planar open-chain manipulator composed of two revolute joints. The system operates in a two-dimensional workspace and is modeled under purely kinematic assumptions.

Let the joint configuration be defined as:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

where q_1 and q_2 denote the shoulder and elbow joint angles respectively.

The forward kinematics mapping from joint space to Cartesian end-effector position is given by:

$$\mathbf{x}(\mathbf{q}) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{bmatrix}$$

where l_1 and l_2 represent the link lengths.

This nonlinear mapping introduces joint coupling effects: the position of the second link depends on both joint angles, which creates a non-trivial inverse kinematics problem. The reinforcement learning agent must therefore implicitly approximate the inverse mapping:

$$\mathbf{q} = f^{-1}(\mathbf{x})$$

without explicit supervision.
Joint limits are imposed as:

$$q_i \in [-\pi, \pi]$$

which ensures full rotational capability while preventing discontinuities in angle representation.

The manipulator is modeled without dynamics (no mass, inertia, or velocity integration). This choice isolates geometric structure and focuses learning on spatial control rather than torque-level regulation. It also ensures that any observed difficulty originates from kinematic complexity rather than dynamic instability.

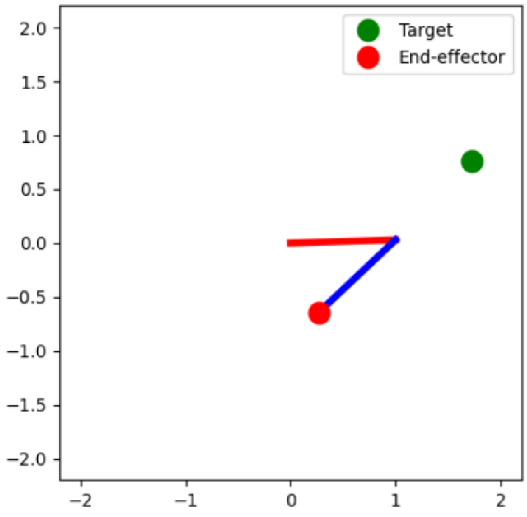

Figure 6: Environment where the 2DoF arm realizes the reaching task. The arm's segments are represented red and blue, the end-effector with a red dot and the target with a green dot.

IV.1.2 Observation Design

The observation vector provided to the agent is designed to capture both the robot configuration and the task objective while remaining minimal and structurally transferable.

The state is defined as:

$$\mathbf{s} = \begin{bmatrix} q_1 \\ q_2 \\ x_t - x_e \\ y_t - y_e \end{bmatrix}$$

where:

- q_1, q_2 are the joint angles,
- (x_e, y_e) is the end-effector position computed through forward kinematics,
- (x_t, y_t) is the sampled target position.

Instead of providing absolute target coordinates, the observation includes the relative Cartesian error:

$$\mathbf{e} = \begin{bmatrix} x_t - x_e \\ y_t - y_e \end{bmatrix}$$

This design choice has several important consequences:

- It centers the learning problem around goal-directed behavior rather than global positioning.
- It introduces translational invariance, since the agent only needs to minimize a relative displacement.
- It facilitates future transfer, as relative task representations are more robust to morphological differences than absolute spatial encodings.

All observation components are normalized to comparable ranges to stabilize training. Joint angles are scaled by π, and positional errors are normalized by the maximum reachable workspace radius:

$$r_{max} = l_1 + l_2$$

Normalization ensures that gradients remain balanced across state dimensions, preventing domination of learning updates by larger-magnitude variables.

The resulting observation space remains low-dimensional and fully observable, which simplifies analysis while preserving nonlinearity in the underlying control problem.

IV.1.3 Action Parameterization

The agent outputs incremental joint commands rather than absolute joint positions. The action vector is defined as:

$$\mathbf{a} = \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \end{bmatrix}$$

where each component represents a small variation applied to the current joint configuration.

The joint update rule is:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \mathbf{a}_t$$

This incremental formulation provides several advantages:

- It transforms the control problem into a local regulation task, which is easier for policy optimization.
- It ensures smoother trajectories compared to direct absolute position control.
- It improves stability, since large configuration jumps are naturally avoided.

The action components are bounded as:

$$\Delta q_i \in [-\Delta q_{\max}, \Delta q_{\max}]$$

where $\Delta q_{\max}$ is chosen to limit angular velocity and prevent unrealistic motion. This bound effectively controls exploration amplitude and prevents divergence during early training.

Using incremental actions also makes the learned behavior closer to velocity control, which is more consistent with real robotic actuation than direct position assignment.

The action space is continuous, which aligns with the policy parameterization of PPO and allows gradient-based optimization over smooth control signals.

IV.1.4 Reward Function Design

The reward function is designed to encourage progressive error reduction while maintaining smooth and stable joint motions.

Let the Euclidean distance between the end-effector and the target be:

$$d = \|\mathbf{e}\|_2 = \sqrt{(x_t - x_e)^2 + (y_t - y_e)^2}$$

The primary reward term promotes distance minimization:

$$r_{distance} = -d$$

This negative distance formulation ensures that the agent is continuously incentivized to reduce positional error at every timestep, rather than receiving sparse feedback only upon success.

To encourage smooth control and avoid excessive joint oscillations, an action regularization term is introduced:

$$r_{control} = -\lambda \|\mathbf{a}\|_2^2$$

where $\lambda > 0$ controls the trade-off between precision and smoothness. This term penalizes large joint increments and stabilizes learning by discouraging unstable exploration.

A success bonus is added when the end-effector enters a tolerance region around the target:

$$r_{success} = \begin{cases} r_s & \text{if } d < \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

where ε defines the precision threshold and r_s is a positive constant. This sparse bonus accelerates convergence by reinforcing accurate reaching behaviors.

The total reward is therefore:

$$r = -d - \lambda \|\mathbf{a}\|_2^2 + r_{success}$$

This structure ensures:

- Continuous gradient information through the distance term,
- Motion regularization through action penalization,
- Explicit reinforcement of task completion through the success bonus.

The reward remains dense and well-shaped, which is critical for stable PPO optimization and for generating consistent trajectories that can later be used for structural transfer.

IV.1.5 Episode Structure

Each training episode is limited to a maximum number of timesteps $T_{\max}$, ensuring consistent evaluation across learning runs:

$$T_{\max} = 200$$

An episode terminates either when the end-effector reaches the target within a success threshold ε or when the maximum timestep limit is reached:

$$\text{done} = \begin{cases} \text{True} & \text{if } d < \varepsilon \text{ or } t \geq T_{\max} \\ \text{False} & \text{otherwise} \end{cases}$$

This structure provides:

- Sufficient time for the agent to explore the workspace while avoiding excessively long episodes that slow down learning.

- Consistent episode lengths, which stabilize PPO's advantage estimation and training dynamics.
- Clear success signals for reward shaping and evaluation metrics.

At the end of each episode, statistics are logged, including:

- Final distance to target
- Cumulative reward
- Episode length

These metrics are essential for monitoring learning progress and for later comparisons across robot morphologies during transfer experiments.

IV.2 Extension to 3DoF Robot

After training the 2DoF manipulator on the reaching task, the framework is extended to a 3DoF planar manipulator. This introduces kinematic redundancy, which increases the dimensionality of the action space and the complexity of the inverse kinematics problem.

The observation vector for the 3DoF robot includes:

$$\mathbf{s}_{\text{3DoF}} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ x_t - x_e \\ y_t - y_e \end{bmatrix}$$

where q_3 is the additional joint angle, and (x_e, y_e) is computed from the 3-link forward kinematics.

The action vector is similarly extended:

$$\mathbf{a}_{\text{3DoF}} = \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \end{bmatrix}$$

Training on the 3DoF robot serves two main purposes:

- It enables the generation of structurally equivalent trajectories, which are necessary for transferring policies from 2DoF to 3DoF robots.
- It allows evaluation of how well the transfer method handles increased redundancy and morphological differences.

Episode structure, reward function, and observation/action normalization remain consistent with the 2DoF setup to facilitate correspondence between the environments.

Figure 7 illustrates the 3DoF reaching environment, including the extended kinematic chain and target placement.

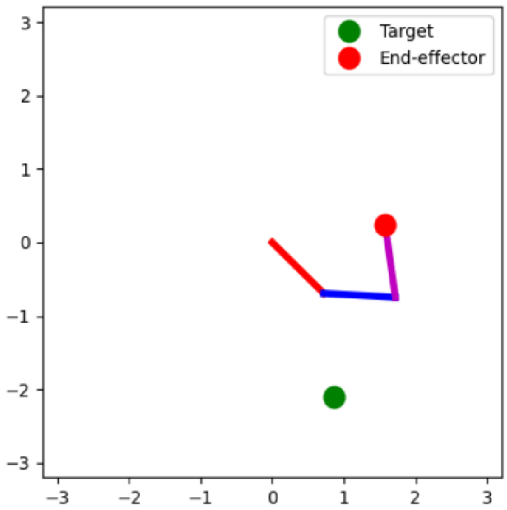

Figure 7: Environment where the 3DoF arm realizes the reaching task. The arm's segments are represented red, blue and purple, the end-effector with a red dot and the target with a green dot.

IV.3 Training on the Push-Ball Task

Following the fundamental reaching task, the environment is extended to a more complex manipulation scenario: pushing a ball to a target location. This task introduces additional elements in both the observation and action spaces:

- Observations include joint angles (q_i), end-effector position (x_e, y_e), ball position (x_b, y_b), target position (x_t, y_t), distance from end-effector to ball, and distance from ball to target.
- Actions remain incremental joint commands Δq_i, extended to match the robot's degrees of freedom (2DoF or 3DoF).

The reward function is designed to capture multiple objectives:

- Reducing distance between the end-effector and the ball.
- Reducing distance between the ball and the target.
- Smooth and stable joint control via penalization of large increments.
- Success bonus when the ball reaches the target within a tolerance threshold.

Formally, if d_{eb} is the end-effector to ball distance, and d_{bt} the ball to target distance:

$$r = -\alpha d_{eb} - \beta d_{bt} - \lambda \|\mathbf{a}\|_2^2 + r_{success}$$

where α, β, λ are scaling factors and $r_{success}$ is awarded when the ball is within ε of the target.

Training the push-ball task serves to:

- Prepare a learned policy that will be applied to the target robot via the trajectory alignment procedure described in Section 3.

Figure 8 illustrates the push-ball environment, showing the end-effector, ball, and target, as well as the robot configuration.

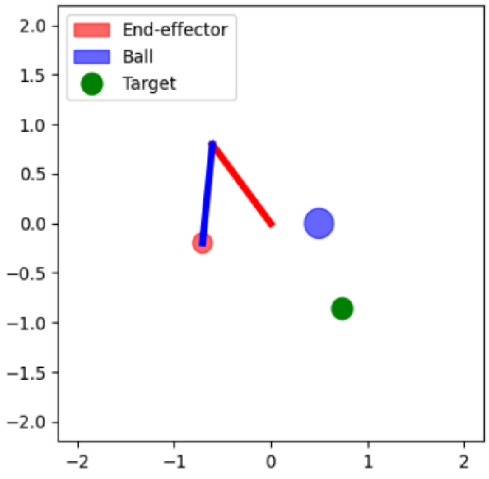

Figure 8: Environment where the 2DoF arm realizes the push ball task. The arm's segments are represented in red and blue, the end-effector with a red dot, the ball with a bigger blue dot, and the target with a green dot.

IV.4 Trajectory Generation for Transfer

The transfer framework relies on generating trajectories that capture correspondences between the source and target robots. Trajectories are recorded from the trained agent performing the fundamental task (reaching) in both environments. Observations and actions are aligned timestep-wise to create pairs, which will later be used to define mappings between the two robots.

Formally, let $\mathbf{s}_{\text{source}}(t)$ and $\mathbf{a}_{\text{source}}(t)$ denote the observation and action of the source robot at timestep t, and $\mathbf{s}_{\text{target}}(t)$ the corresponding observation in the target robot environment. These aligned pairs will serve as the basis for constructing mapping functions:

$$T_s : \mathbf{s}_{\text{target}} \mapsto \mathbf{s}_{\text{source}}, \quad T_a : \mathbf{a}_{\text{source}} \mapsto \mathbf{a}_{\text{target}}$$

These mappings, once built, will allow the source policy to be applied to the target robot while respecting its kinematic structure. Enforcing cycle-consistency during trajectory alignment helps ensure that the structural dynamics of the task are preserved between robots.

These trajectories form the foundation for transferring more complex tasks, such as the push-ball task, once the mappings are fully defined.

IV.5 Summary of Experimental Setup

The experimental setup combines environment design, task progression, and trajectory collection to prepare for transfer learning. The process can be summarized as follows:

1. **Fundamental task training:** A 2DoF planar manipulator is trained on a reaching task, allowing the agent to learn basic inverse kinematics and interaction with the environment.
2. **Extension to target morphology:** The setup is extended to a 3DoF manipulator, increasing kinematic complexity. The goal is to generate aligned trajectory pairs between source and target robots to facilitate later transfer.
3. **Complex task training:** A more complex manipulation task, pushing a ball to a target, is implemented and trained on the source robot. Observations and actions are richer to capture multi-objective interactions.
4. **Trajectory generation:** During the reaching task, aligned observations and actions are recorded for both robots. These will be used to construct the mappings necessary for transferring policies to the target robot.

This setup ensures that both source and target robots share comparable reference frames and timing, allowing the transfer framework to leverage previously learned behaviors. While the mappings for transfer are not yet applied, the collected trajectories provide the foundation for subsequent experiments.

V - Results

V.1 2DoF Reaching Task

The primary results available at this stage concern the 2DoF reaching task. Figure 9 shows an example of the learning curve, illustrating the evolution of cumulative reward across training episodes. The curve demonstrates the progressive improvement of the PPO agent in reaching the target with increasing accuracy.

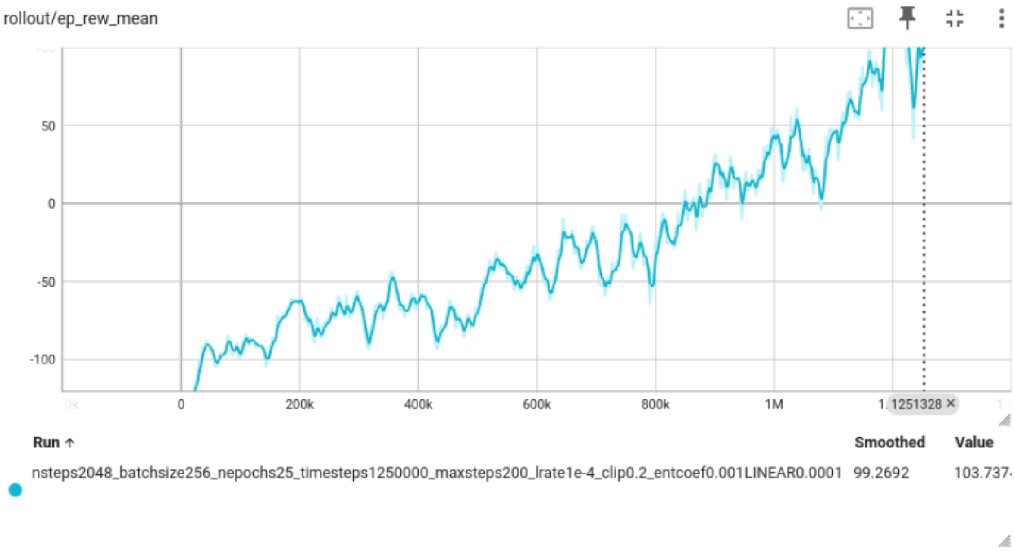

Figure 9: Evolution of the reward's value over the entire reaching task learning process of the 2DoF arm.

Key metrics observed include:

- Reduction of average end-effector to target distance over episodes.
- Increased success rate of reaching the target within the defined tolerance.

Although only the 2DoF reaching task has been completed, these results provide a baseline for evaluating the structural transfer to the 3DoF manipulator.

V.2 Discussion on Preliminary Results

While the main push-ball task and transfer experiments have not yet been completed, the 2DoF reaching results confirm that:

- PPO is able to learn effective joint control policies in a planar manipulator.
- The training framework, including environment design and reward shaping, is functional.
- The aligned trajectories obtained from the reaching task can be expected to form a valid basis for transfer once the mapping procedure is implemented.

These preliminary findings establish a foundation for the transfer experiments planned during the internship.

VI - Project Planning & Timeline

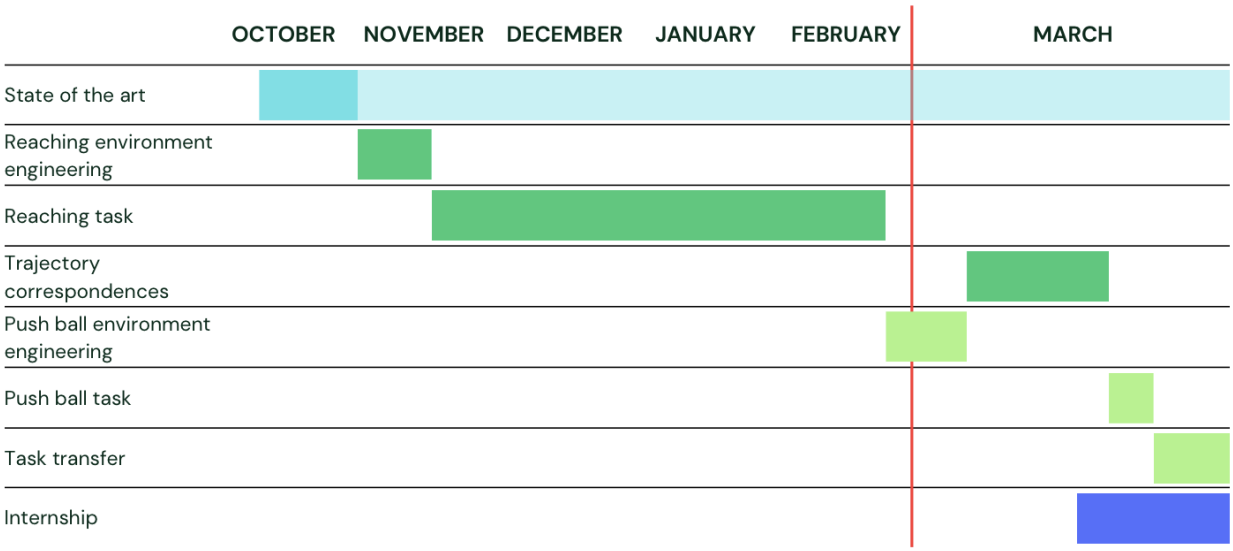

Figure 10: Gantt chart summarizing the project timeline, including completed and remaining tasks.

The Gantt chart in Figure 10 summarizes the timeline of the project. The initial phases included literature review. These tasks appear extended on the chart because bibliographic work was ongoing throughout the project as part of continuous learning.

The engineering of the 2DoF reaching environment and development and training of the reaching task required the longest period, reflecting the iterative nature of environment setup, PPO training, and policy validation. Following this, the development of the push-ball environment began. From this point onward, the timeline transitions from completed work to planned tasks. Until now, project progress was balanced with coursework, which explains the longer durations shown in the chart.

The next immediate step is the generation of trajectories for transfer. Starting March 20, during the internship, the remaining tasks—finalizing trajectory generation, completing the push-ball task, and performing the transfer—will be carried out at full-time pace. Due to the skills acquired so far, these tasks are expected to be completed more efficiently, with a target completion by the end of March.

VII - Discussion & Perspectives

VII.1 Challenges and Limitations

The full-policy transfer approach presented in this work relies on reusing the entire agent trained on the source robot. While this allows leveraging previously acquired skills, it has limitations. The transferred policy may perform poorly if the source agent has exploited environment-specific shortcuts rather than learning the true underlying task dynamics. Even with PPO stabilizing the learning process, there is no guarantee that the agent has fully captured the task dynamics; successful task execution does not necessarily imply complete understanding of the system's physics or optimal strategies.

Another limitation concerns the sensitivity to hyperparameters. PPO training requires careful tuning of parameters such as learning rate, batch size, horizon, and clipping epsilon. Misconfigured hyperparameters can lead to unstable learning or suboptimal policies. Additionally, aligning trajectories between source and target robots requires synchronized environments and careful preprocessing, which adds practical complexity.

VII.2 Learning Experience and Skill Development

Throughout this project, a significant portion of time was dedicated to understanding PPO in depth. The actor-critic architecture, the role of the critic in providing a stable learning signal, and the clipping mechanism were critical concepts to grasp. Experiments on simplified environments such as CartPole helped in developing intuition on policy stability and hyperparameter effects. This iterative process improved familiarity with Gymnasium, Stable-Baselines3, and the practical aspects of reinforcement learning for robotic applications.

VII.3 Perspectives and Future Work

This study provides a foundation for the upcoming internship on human-to-robot skill transfer. Future work will focus on extending the current framework to more complex tasks while leveraging cycle-consistency-based trajectory alignment to enable human skill transfer.

VIII - Conclusion

This project has established a solid foundation for studying transfer learning between robots of different morphologies. Through the implementation of a 2DoF planar manipulator performing a reaching task, the PPO agent successfully learned to control joint movements to reach targets, validating the environment design, reward shaping, and training framework.

Aligned trajectories obtained during this fundamental task provide the basis for transferring policies to a 3DoF manipulator; the cycle-consistency approach ensures that observation-action correspondences are maintained between source and target robots.

Although more complex tasks, such as pushing a ball to a target, and the full transfer procedure are yet to be completed, the current results confirm the feasibility of the approach and set the stage for future experiments.

Outlook: The methods and insights developed in this project will directly inform the upcoming internship on human-to-robot skill transfer. Extending the current framework to handle human demonstrations and more complex robotic tasks will allow leveraging prior experience efficiently, reducing training time and computational cost in real-world applications. This work represents a crucial step toward enabling robots to acquire and generalize skills from humans.

Declaration on the Use of Generative AI Tools

Generative AI tools were used during this project as supportive instruments. They assisted in formatting, language revision, and reformulation of sections of the report in order to improve clarity and structure.

AI tools were also occasionally used to help identify relevant documentation, understand theoretical concepts, and debug programming errors encountered during development.

All scientific decisions, methodological choices, implementation steps, experimental design, and analysis of results were carried out independently. The use of AI tools was limited to support tasks and did not replace critical reasoning or technical work.

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv preprint arXiv:1707.06347, 2017.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd edition, MIT Press, 2018.

[3] A. Anwar, A. Raychowdhury, *A Practical Guide to Proximal Policy Optimization*, arXiv preprint arXiv:2205.12729, 2022.

[4] Z. Zhu, K. Lin, A. Jain, J. Zhou, *Transfer Learning in Deep Reinforcement Learning: A Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 13344–13362, 2023.

[5] S. Beaussant, *Transfer Learning between Robots with State Abstraction*, Thèse de Doctorat, Université Clermont Auvergne, 2023. Soutenue publiquement le 20 Septembre 2023 devant le Jury : Serena Ivaldi, David Filliat, Stéphane Doncieux, Olivier Stasse, Benoît Thuilot, Sébastien Lengagne. https://hal.science/tel-04598434v1

[6] M. Mounsif, *Exploration of Teacher-Centered and Task-Centered paradigms for efficient transfer of skills between morphologically distinct robots*, Thèse de Doctorat, Université Clermont Auvergne, 2020. Soutenue publiquement devant le Jury : David Filliat, Olivier Stasse, Gentiane Venture, Sébastien Druon, Lounis Adouane, Benoît Thuilot, Sébastien Lengagne.

[7] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, *Stable-Baselines3*, 2021. https://github.com/DLR-RM/stable-baselines3

[8] M. Towers, J. Terry, A. Kwiatkowski, et al., *Gymnasium*, 2023. https://gymnasium.farama.org/index.html

[9] R. Zhu, T. Dai, O. Celiktutan, *Cross Domain Policy Transfer with Effect Cycle-Consistency*, arXiv preprint arXiv:2403.02018, 2024.

[10] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, X. Wang, *Learning Cross-Domain Correspondence for Control with Dynamics Cycle-Consistency*, arXiv preprint arXiv:2012.09811, 2020.