

# SetPoint Learning : Toward an approach to control them all

Fabien FURFARO\*, Joëlle FERZLY, Boubker TABITI<sup>†1</sup>

<sup>1</sup>*Capgemini Engineering, 12 rue de la Verrerie, 92190 Meudon, France*

---

## Abstract

Deep learning has enabled significant advancements in the field of dynamic system control, particularly in robotics, video games, navigation, and energy management, especially for non-linear systems. However, this type of learning does not allow for generalization to all dynamic systems, and sometimes, controllers based on system modeling (MPC, PID, MOR) are more relevant than methods based on stochastic data, especially for cases where the system is linearizable, and the target setpoint is well-known, which corresponds to most cases in the electronic, mechanical, and thermal domains. We introduce a new learning method that is generalizable to all linear dynamic systems where the setpoint is known and adaptive. We show that when targeting a setpoint, neural networks adapt towards an objective and would be a promising approach for the use of generative diffusion models for direct interaction with multiple real-world environments in parallel. This study has led to the creation of a set of simulation environments that can serve as a basis for other reinforcement or data-based approaches. Environment implementation is publicly available in <https://github.com/fabienfrfr>.

*Keywords:* Reinforcement Learning, Adaptive control, Gym environment, Time invariant linear systems.

---

## 1 Introduction

Reinforcement learning (RL) has emerged as a powerful paradigm for planning decision-making in control environments, often surpassing traditional linear or data-based control methods [6]. In linearizable systems, planning can be reduced to trajectory optimization, achievable through controllers [4]. Notable RL algorithms include value-based approaches like Deep Q-Networks (DQN) and policy-based methods such as Proximal Policy Optimization (PPO) [8].

While these methods have demonstrated effectiveness in specific scenarios, they often require substantial effort to study the system and optimize parameters for each case [optimization-challenges]. This lack of versatility has sparked interest in approaches that can create adaptable agents capable of functioning across diverse environments [adaptable-agents]. The ability of an RL agent to generalize and perform well in various settings remains a significant challenge in the field [5].

To address these limitations, researchers have proposed several alternatives, including aiding techniques, navigation with human feedback, planning diffusion, and more recently joint embedding predictive architectures (JEPA) [3]. However, these advanced techniques often introduce additional complexity and may not offer universal generalization across all physical systems.

In this work, we present a novel, simplified approach that enables generalization in controlling linearizable systems and a set of control environments adaptable to various learning algorithms. Our main hypothesis diverges from the conventional assumption that learning models should always strive for optimal results. Instead, we posit that targeting (setpoint) intermediate results with lower scores can lead to improved overall performance, analogous to a human trainer adjusting their level for a student. By incorporating this concept into ML, we aim to develop more robust and adaptable agents capable of handling a wider range of environments and tasks. Our method not only simplifies the learning process but also potentially improves the generalization capabilities of agents, addressing a critical challenge in the field.

---

\*Corresponding author : [fabien.furfaro@capgemini.com](mailto:fabien.furfaro@capgemini.com)

<sup>†</sup>Equal contribution.

## 2 Background and contribution

Our approach is an analogue of past work in behavioral synthesis using trajectory optimization [(MPC, Witkin Kass, 1988; Tassa et al., 2012)]. But we propose here an approach where the variables are always the same whatever the physical system and where the objective to be achieved is defined in the controller input. This approach is also compatible for the use of generative models.

Our method relies on modifying the states of the simulation environment through the use of a Wrapper or an environment we developed. This enables our approach to interact more dynamically with the system.

## 3 Problem Setting

Consider a system governed by discrete-time dynamics

$$S_{t+1} = f(S_t, a_t), \quad (1)$$

where  $S_t$  is the state of the system at time  $t$ ,  $a_t$  is the control action at time  $t$ , and  $f$  represents the system dynamics function. For a linear system, the dynamics can be expressed by the following equation:

$$y(t) = s(t) * h(t), \quad (2)$$

where  $y(t)$  is the output signal,  $s(t)$  is the input action from the controlling system,  $h(t)$  represents the impulse response of the system, and  $*$  denotes the convolution operation. In this context, the function  $f$  in (1) is the convolution operation  $f(S_t, a_t) = a_t * h(t)$ . The objective is to achieve the optimal control input  $u^*(t)$  that minimizes a given cost function  $J(S)$ :

$$u^*(t) = \arg \min J(S). \quad (3)$$

The main challenge lies in designing a model that can solve problem (1) regardless of the specific physical system.

## 4 Setpoint Learning method

In this work, we present a setpoint learning approach designed to improve setpoint tracking by incorporating a target guideline. This method mainly relies on selecting a set of primary input variables from initial observations, according to their relevance in understanding system dynamics and achieving predefined targets. Additionally, this approach uses a target setpoint as a reference to guide the learning process. The key variables include:

- $a_{t-1}$  : the action taken in the preceding state,
- $s_{t-1}$  : the state of the system in the previous timestep,
- $s_t$  : the current state of the system
- $(s_{sp})_{t+1}$  : the setpoint, or target state, to be achieved at the next timestep.

By incorporating the previous action  $a_{t-1}$ , we introduce a temporal dimension, allowing the models to learn from past actions and adapt to changing dynamics across different contexts. By directing actions towards the convergence of the current state  $s_t$  to the target state  $s_{sp}$  at the next timestep, this method ensures goal-oriented behavior while maintaining adaptability, as presented in Figure 1.

The setpoint learning method relies on selecting  $a_{(i,t-1)}$  from the action space  $A$ , the states  $s_{(j,t-1)}$  and  $s_{(j,t)}$  from the state space  $S$ , and the target setpoint  $s_{sp(j,t+1)}$  to be achieved at the next timestep, resulting in the state space  $\tilde{S}_t$  at time  $t$ , as describes in the following equation

$$\{a_{(i,t-1)}, s_{(j,t-1)}, s_{(j,t)}, s_{sp(j,t+1)}\} \longrightarrow \tilde{S}_t \quad (4)$$

Here,  $i$  represents the index of the action  $a$  within the set  $A$ , and  $j$  represents the index of the state  $s$  within the set  $S$ .

This work aims to demonstrate two key points: (i) a model trained on an environment using the variables defined in equation (4) can control any other linear environment that incorporates the same set of variables; and (ii) this model is capable of tracking any given setpoint.

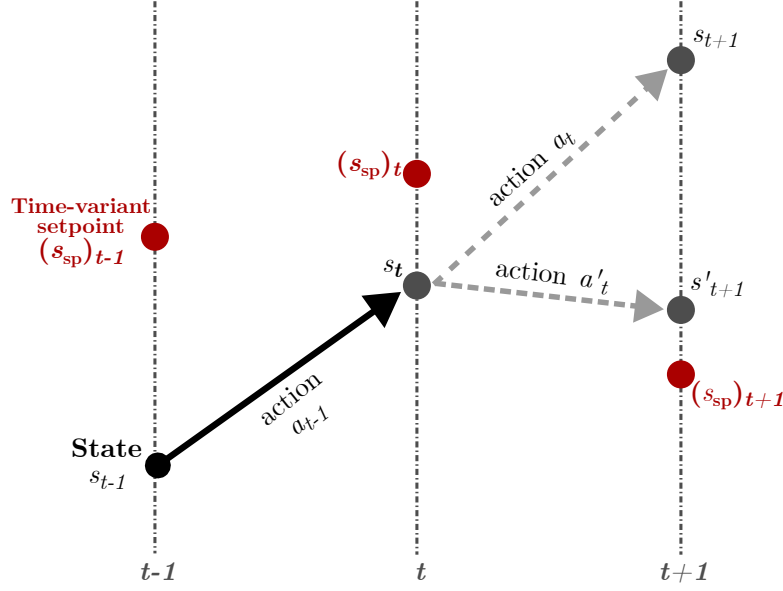


Figure 1: Representative diagram of the target setpoint concept. At time  $t$ , the agent in state  $s_t$  should choose the action in order to get as close as possible to the target state (setpoint  $s_{sp}$ ) at time  $t + 1$ .

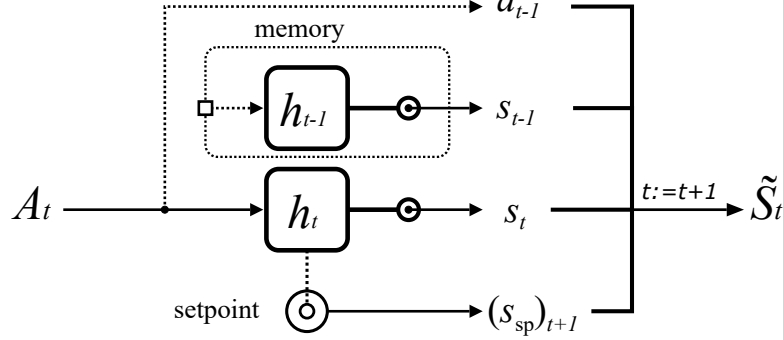


Figure 2: Representative scheme of the setpoint learning environment. Here,  $A_t$  represents the action space,  $h_{t-1}$  a linearized system or wrapped environment stored in memory and  $h_t$  the present system. At  $t := t + 1$ ,  $\tilde{S}_t$  denotes the resulting state space.

This method is designed to be applied in two distinct frameworks: 1) reinforcement learning, presented in Section 5, where the goal is for the agent to learn to follow a setpoint and adapt to different environments, and 2) diffusion models, detailed in Section 6, where each image represents state-action spaces.

## 5 Reinforcement learning case

The proposed approach can be employed in a reinforcement learning framework, leading to the Sepoint Reinforcement Learning approach (SPRL). It mainly enables agents to adapt to different setpoints across various environments without the need for specific trainings.

The environment integrating the SPRL approach is presented in Figure 3. The reward system is redefined based on the deviation from the setpoint to precisely adjust the agent's objective. Thus, the trained agent can choose an action  $a_t$  based on the current state  $s_t$ , aiming to closely approach the target setpoint  $s_{sp}$  at time  $t + 1$ , as illustrated in Figure 1.

As already mentioned, ...

### 5.1 Application to the CartPole problem

To validate our approach on a simple model before tackling more complex problems, we first apply it to the inverted pendulum problem (CartPole). This environment is chosen due to its simple reward system and the number of independent variables, which are comparable to those in our proposed method.

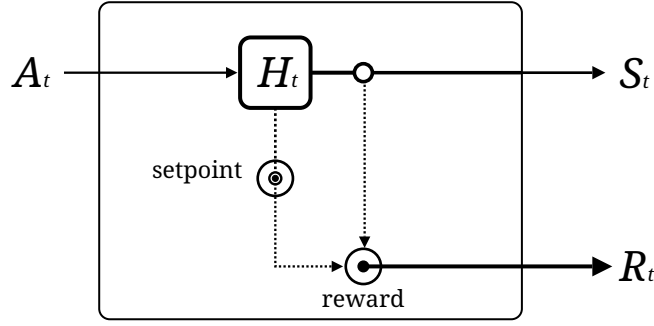


Figure 3: Scheme of the Setpoint Reinforcement Learning environment (SPRL). Here,  $A_t$  represents the action space,  $H_t$  a linearized system or wrapped environment. At  $t := t+1$ ,  $R_t$  denotes the reward received (calculated based on the system's response and the setpoint), and  $S_t$  the resulting state space.

### 5.1.1 Cartpole environment

We briefly recall the structure of the Cartpole environment and then present the modified environment.

#### Classical Cartpole environment

The classical CartPole environment as introduced in [1] provides four observations that allow us to define the set of states  $S$  at time  $t$  as

$$S_t = \{x_t, v_t, \theta_t, \omega_t\}, \quad (5)$$

where  $x_t$  denotes the cart position,  $v_t$  the cart velocity,  $\theta_t$  the pole angle, and  $\omega_t$  the pole angular velocity. The reward function at time  $t$  is defined as follows:

$$R_t(\theta_t) = 1 \text{ if the episode is not terminated and } \theta_t \text{ is in the range } (-0.2095, 0.2095) \text{ rad.} \quad (6)$$

#### Modified Cartpole environment

In the design of our modified environment, the pole angle  $\theta$  is selected as the primary variable due to its key role in maintaining equilibrium in the classical CartPole setup. Focusing on the angle provides the agent with relevant information to capture system dynamics and adjust its actions accordingly. The following table summarizes the observations provided by the modified environment.

Variable	Observation	min	max
$a_{t-1}$	Previous action	0	1
$\theta_{t-1}$	Previous pole angle	$-0.2095$ rad	$0.2095$ rad
$\theta_t$	Pole angle	$-0.2095$ rad	$0.2095$ rad
$(\theta_{\text{sp}})_{t+1}$	Target pole angle	$-0.2095$ rad	$0.2095$ rad

Table 1: Observations provided by the modified CartPole environment.

This leads us to define the set of states  $\tilde{S}$  of the modified environment at time  $t$  as follows:

$$\tilde{S}_t = \{a_{t-1}, \theta_{t-1}, \theta_t, (\theta_{\text{sp}})_{t+1}\}, \quad (7)$$

where  $\theta_{\text{sp}}$  is the target angle we aim to reach at time  $t+1$ .

Moreover, the introduction of a reward function based on the setpoint reinforces the understanding of the objective by concentrating high rewards around the setpoint, as shown in Figure 4. This reward function at time  $t$  is defined as follows, with  $Z = (\theta_{\text{sp}} - \frac{\theta_{\Omega}}{2}, \theta_{\text{sp}} + \frac{\theta_{\Omega}}{2})$  rad :

$$R_{\theta_{\text{sp}}}(\theta_t) = \begin{cases} 1 & \text{if } \theta_t \text{ is within the range } Z \\ 0.1 & \text{if } \theta_t \text{ is not in } Z \text{ and } \theta_t \text{ is within } (-(\theta_{\text{sp}})_{\text{max}}, (\theta_{\text{sp}})_{\text{max}}) \text{ rad.} \end{cases} \quad (8)$$

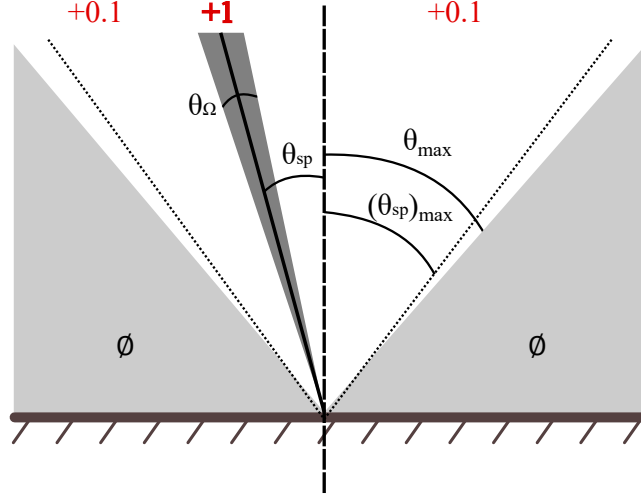


Figure 4: Distribution of the reward (8) in the modified CartPole environment. Around the setpoint, in the range  $Z$  (dark gray zone), the reward is equal to 1. Otherwise, the reward is 0.1 within the range  $(-(\theta_{sp})_{max}, (\theta_{sp})_{max})$  rad.

### 5.1.2 Experimental Setup

In order to evaluate the adaptability of the SPRL approach, we train agents using the SPRL approach with different target setpoint value  $\theta_{sp}$  (zero, random constant  $X$ , and variable random function  $f(X)$ ). We also consider three different scenarios as references trainings, each distinguished by the space state (5) or (7) and the reward function (6) or (8). The configuration of the total six scenarios is resumed in Table 2.

We use the "CartPole-v1" version for agent training as well as for the tests. The constraint on the cart's position is removed, allowing the agent to learn to follow setpoints with more or less off-centered angles. The only constraint remains the pole angle. The episode is terminated when one of the stopping conditions is met, and truncated if the episode ends before the threshold defined by the version of the environment used. For more details, refer to [1].

The training was realized over 1000 episodes using the Dual Double DQN algorithm, implemented in the RLlib library [2]. During the training, we save the point at which the moving average of the last 100 scores is maximal, enabling us to compare the performance of the different models as shown in Figure 5.

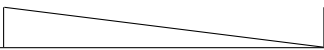
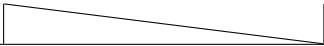
Training model	Environment	Space state	Reward	Setpoint $\theta_{sp}$	Goal
$T_{\tilde{S}, R_{\theta_0}}$	SPRL	$\tilde{S}$ (7)	$R_{\theta_{sp}}$ (8)	0	Train SPRL approach with three different setpoints
$T_{\tilde{S}, R_{\theta_X}}$	SPRL	$\tilde{S}$ (7)	$R_{\theta_{sp}}$ (8)	random variable, constant per episode	
$T_{\tilde{S}, R_{\theta_{f(X)}}}$	SPRL	$\tilde{S}$ (7)	$R_{\theta_{sp}}$ (8)	random, time-varying function	
$T_{S, R}$	classical	$S$ (5)	$R$ (6)		Reference training
$T_{\tilde{S}, R}$	SPRL	$\tilde{S}$ (7)	$R$ (6)	0	
$T_{S, R_{\theta_0}}$	classical	$S$ (5)	$R_{\theta_{sp}}$ (8)		

Table 2: Overview of the six training models, each defined by the approach used (classical or SPRL), the reward function ( $R$  or  $R_{\theta_{sp}}$ ), and the associated setpoint.

### 5.1.3 Results

We begin by presenting training results for both the SPRL and classic CartPole environments. Figure 5 illustrates agent performance, displaying rewards trajectories across episodes for each model detailed in Table 2.

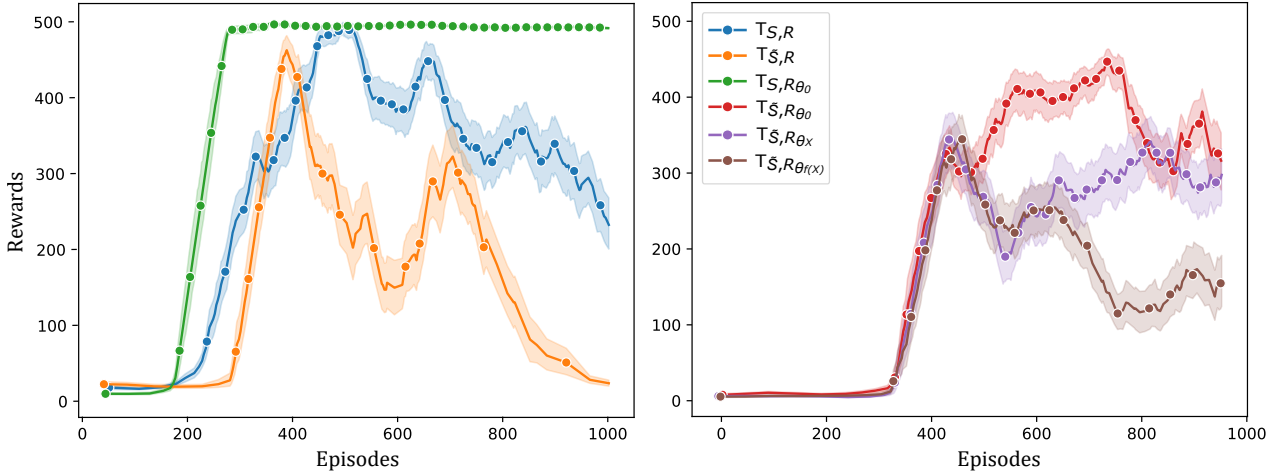


Figure 5: Performance of the six trained agents following the models described in Table 2.

Tests \ Trainings						
	$T_{S,R}$	$T_{S,R_{\theta_0}}$	$T_{\tilde{S},R}$	$T_{\tilde{S},R_{\theta_0}}$	$T_{\tilde{S},R_{\theta_X}}$	$T_{\tilde{S},R_{\theta_{f(X)}}}$
$R_{\theta_0}$	0.910542	0.994389	0.993675	<b>0.997996</b>	0.758116	0.894289
$R_{\theta_X}$	0.151506	0.100000	0.109036	0.100000	0.737048	<b>0.869880</b>
$R_{\theta_{f(X)}}$	0.134337	0.100000	0.403614	0.437952	0.682530	<b>0.859036</b>

Table 3: Adaptability test results for agents trained in three distinct test environments (null setpoint, random constant per episode, and random time-varying function), expressed by median over 100 iterations.

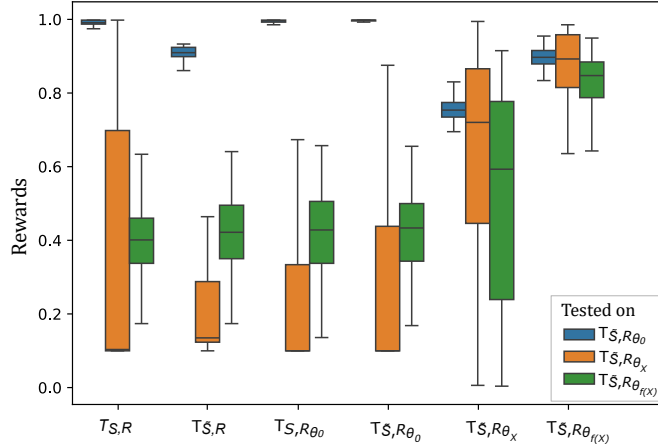


Figure 6: Box plot of the six models tested on the three scenarios respectively integrating a zero setpoint (blue), random constant (orange), and random function (green).

The results of this synthesis, as depicted in table 3, highlight the remarkable adaptability of the  $T_{\tilde{S},R_{\theta_{f(X)}}}$  model. Evaluating  $T_{\tilde{S},R_{\theta_{f(X)}}}$  across three tests shows good performance in following the setpoint, whether constant or varying over time. This performance even surpassed that of the  $T_{\tilde{S},R_{\theta_X}}$  model, specifically trained to follow a constant setpoint  $X$ .

## 5.2 Linear Time-invariant system

In order to improve generalization performance of the approach and evaluate its adaptability, we apply it to linear time-invariant (LTI) systems, which are widely used in engineering and physical sciences.

### 5.2.1 Developed environment

The developed environment facilitates the application of the Setpoint RL approach for controlling linear dynamic systems. Given the absence of such an environment in existing libraries, we designed the `LinearSystemControl` environment, inspired by the prior development of the `SetpointWrapper` environment, as described in Section 5.1.1. The `LinearSystemControl` class inherits from the `Gymnasium.Env` base class, ensuring compatibility with reinforcement learning agents.

This environment offers several essential features for the simulation and control of linear dynamic systems. These features are defined by a set of class attributes and methods. The class attributes allow users to configure the environment according to their needs. Among these attributes are the selection of the operating mode (`env_mode`), setpoint updates (`update_setpoint`), specification of the system's initial state (`reset_X_start`), and other parameters influencing the environment's behavior. These attributes are provided to the class via a dictionary named `config` during instantiation.

### 5.2.2 Experimental Setup

In this section, we present the setup for our numerical experiments. We first examine the different training models, providing a detailed overview of the various scenarios used to train our agents. This includes specific conditions and parameters for each model.

We trained multiple models and evaluated them in different environments based on sets of transfer functions. The transfer functions used during training are all first-order, characterized by a gain within the interval  $[-1, 1]$  and a fixed time constant of 1. This approach allows us to assess each model's adaptability to control setpoints based on the environment in which it was trained. We address this problem using a discrete action space. Finally, the agents are trained using the Dual Double DQN reinforcement learning algorithm, as presented in [10] and [9].

#### Scenario 1

In the first scenario, we consider the system represented by the transfer function  $G(s, 1) = \frac{1}{s+1}$ . This function is commonly used to model first-order linear systems and is suitable for analyzing simple dynamic systems. The agent is trained in an environment defined by  $\{\frac{-1}{s+1}\}$ . To precisely analyze the agent's performance, we divide this scenario into three sub-tests, each characterized by a specific setpoint for the duration of the episode: A fixed setpoint of  $\frac{1}{2}$ , a constant random setpoint  $X$ , a variable random setpoint  $f(X)$ .

#### Scenario 2

In the second scenario, we introduce the negative transfer function of  $G(s, 1)$  representing the same system as in Scenario 1. The transfer function  $-G(s, 1) = \{\frac{1}{s+1}\}$  is included to allow the agent to explore an environment where the effect of actions is reversed. The agent is trained in an environment defined by  $\{\frac{1}{s+1}, \frac{-1}{s+1}\}$ . Each test is defined by a different setpoint: A fixed setpoint of  $\frac{1}{2}$ , a constant random setpoint  $X$ , a variable random setpoint  $f(X)$ .

#### Scenario 3

The third scenario aims to evaluate the agent with the transfer function  $G(s, 1)$  by varying the gain. The transfer function in this scenario is  $\frac{a}{s+1}$ , where  $a \in [-1, 1]$ . This allows modification of the system's response amplitude without affecting its dynamics. The agent is trained in an environment defined by  $\frac{a}{s+1}$ . Since the gain variation can be less than  $\frac{1}{2}$ , only two tests are performed: constant random setpoint  $X$  for the duration of the episode, a variable random setpoint  $f(X)$  for the duration of the episode.

### 5.2.3 Results

We present in this section the training results and analyze the performance of the agents under different training configurations. To illustrate the concept of generalization, we extend the tests to the different training environments used for the eight models. Additionally, for a more effective comparison, we introduce three agents:

- a constant agent, named `Const`, which only applies a null action
- a random agent named `Random`

- a ToR agent corresponding to the Bang-bang controller

In our case, the Bang-bang controller is optimized to act in case of system inversion. The direction of the next action is determined by the sign of the difference between the target state and the current state, taking into account the possible inversion of the environment.

We summarize the performance of these agents in the table in Figure 7, presenting the results of a test conducted with 100 episodes per agent across different environments. The results are expressed as normalized averages. The results are color-coded with a gradient where the best performances are shown in red and the worst in blue.

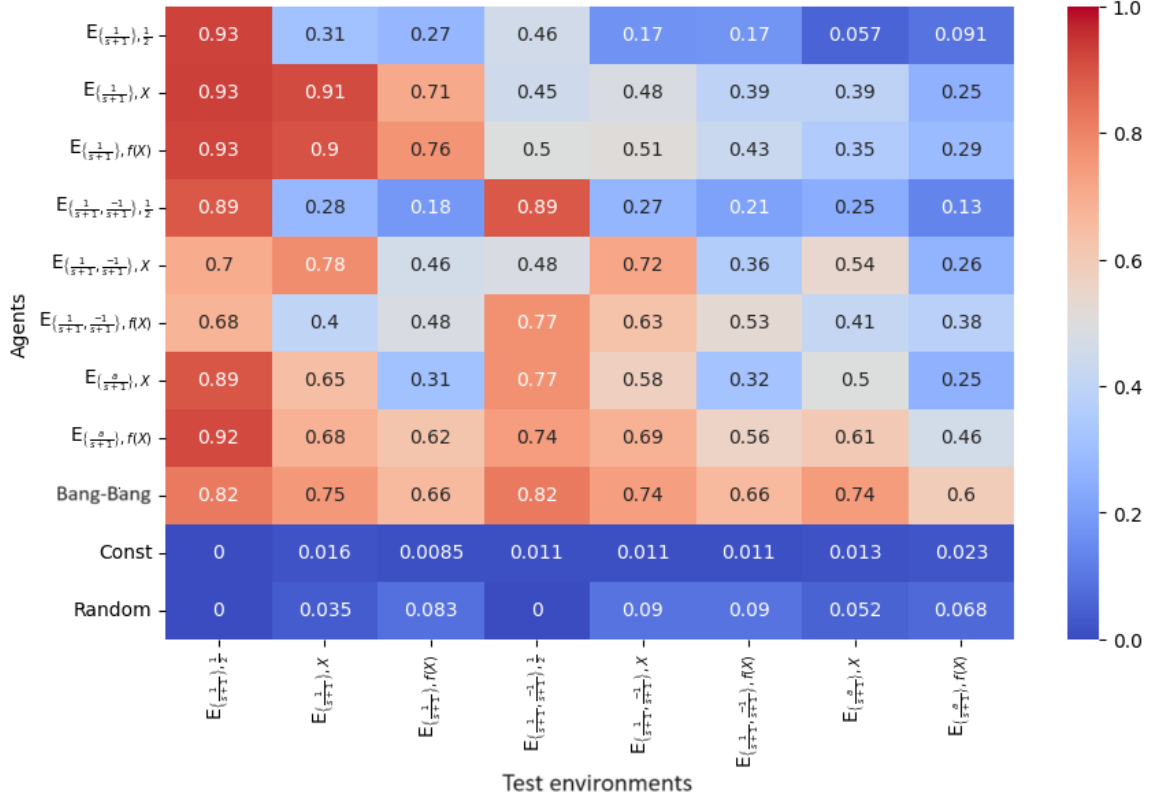


Figure 7: Comparison of agent performance as a function of evaluation environment over 100 episodes, expressed by normalized means.

## 6 Diffusion learning case

Numerous applications demonstrate that reinforcement learning enables agents to control manageable physical environments [6]. However, when agents must control multiple different environments in parallel, reinforcement learning has shown several limitations in terms of generalization [kirk2021survey]. Currently, one of the approaches that allows generalization across multiple environments is the diffusion method [janner2022planning]. To address this, we have developed a parallel environment generator where we test a diffusive approach based on our setpoint method.

### 6.1 Multiple LTI system data generation

In our multiple LTI environment, we explore four distinct configurations, labeled (a), (b), (c), and (d) (see Figure 8). It's important to note that in this context,  $A_t$  yields  $S_{t+1p}$ , representing the output state. Configurations (b), (c), and (d) corresponds to the division of configuration (a) into  $N \times N$  transfer functions. For that, we use interconnected system techniques in python-control.



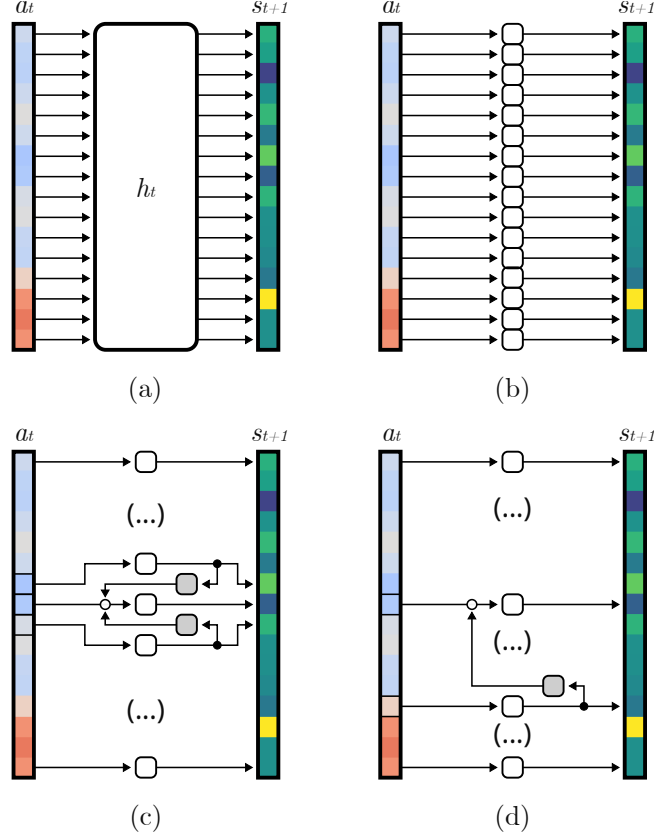


Figure 8: Scheme of multiple LTI system data generation. (a) One transfer function with  $N \times N$  input and output, (b), (c), and (d)  $N \times N$  transfer function with  $N \times N$  input and output. (b) Independent transfer function, (c) 2D local diffusion interconnection, (d) Randomized link between transfer function.

The initial inputs and state variables can also be divided independently of the configurations. This flexibility allows for a more adaptable approach to various environmental complexities. These benefits align with recent trends in reinforcement learning that focus on efficient scaling and generalization across multiple environments [espeholt2018impala]. The ability to independently divide initial inputs and state variables offers:

1. Greater control over the granularity of the input space
2. Flexibility in handling environments of varying complexities
3. Potential for hierarchical learning approaches

To implement our approach with multiple environments, we generated a set of images with dimensions (1, 5, 32, 32) to incorporate into a diffusive model. This constructed database contains, respectively for 5 channel images:

$$\tilde{S}_{D,t} = \{s_{t-1}, a_{t-1}, s_t, a_t, s_{t+1}\} \quad (9)$$

Where  $s_{t-1}$ ,  $s_t$  and  $s_{t+1}$  are the system states at times  $t-1$ ,  $t$  and  $t+1$  respectively, and  $a_{t-1}$  represents the previous action before getting the next state  $s_t$ . This structure ensures consistency with our reinforcement learning approach while being adaptable for the diffusive approach. The use of image-based representations allows us to leverage the power of convolutional neural networks in processing spatial information, which has shown great success in various learning tasks [6].

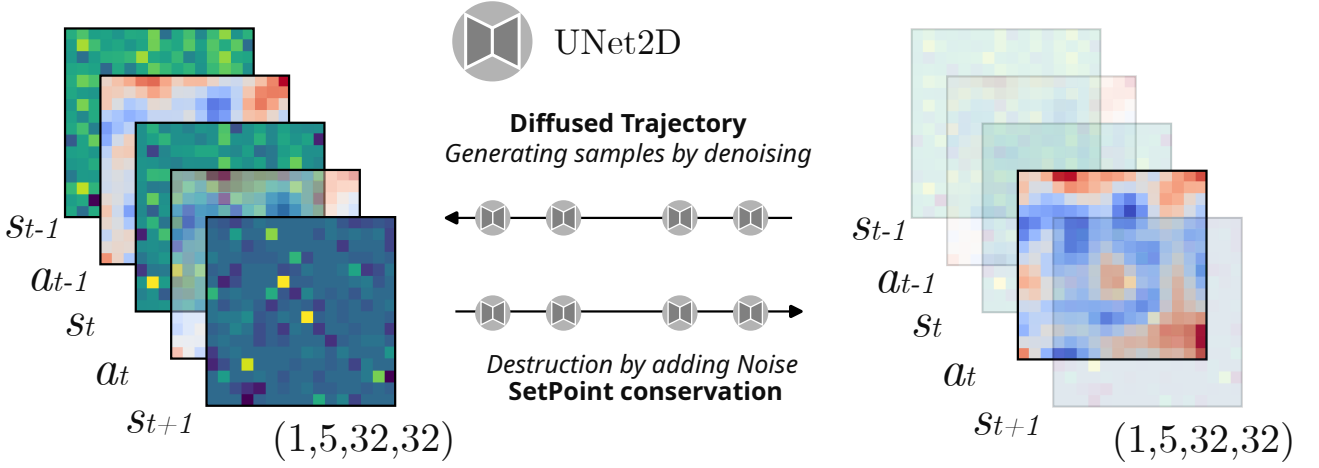


Figure 9: Algorithm scheme of DDPMS-Approach. Je separerais les figures ? pour bien separer la partie "génération des environnement multiple" et l'algorithme proposé pour la génération de la base de données...

The integration of diffusion models with reinforcement learning has recently gained attention due to its potential in improving sample efficiency and exploration [janer2022planning](#). Our approach of using a structured database that includes both state and action information aligns with recent work on offline reinforcement learning with diffusion models (see Figure 9).

## 6.2 Multiple LTI System Training

In our SPRL (Setpoint Reinforcement Learning) approach, we initially aimed to target the setpoint at time  $t + 1$ . However, to align with the diffusive approach, we slightly modified the state representation to form an image of dimensions  $(1, 5, 32, 32)$ . In this configuration, and given that the data is pre-generated, each channel of the image can represent a setpoint. To test our approach with an  $N \times N$  environment, we employed three distinct configurations:

- **Direct generation from complete noise:** In this configuration, environment configurations are directly generated from complete noise, following the classical diffusion model approach [ho2020denoising](#).
- **Generation with a fixed "target" channel:** Here, environment configurations are generated with a fixed "target" channel, which remains unchanged throughout the generation process. This approach aligns with the concept of conditional generation in diffusion models [song2020score](#).
- **Generation with a noisy "target" channel:** In this final configuration, environment configurations are generated with a noisy "target" channel, meaning this channel is perturbed by noise before the generation process. This method can be seen as a form of data augmentation, potentially improving the robustness of the model [11].

These configurations allow us to explore different aspects of the diffusion process in the context of control of multiple LTI systems. By varying the nature of the target channel, we can investigate the model's ability to generate coherent environment configurations under different constraints.

We implemented these methods using a state-of-the-art diffusion model architecture, similar to that proposed by [ho2020denoising](#) with HuggingFace tools, but adapted for our specific control task. The model was trained on a diverse dataset of LTI system trajectories, encompassing a wide range of initial conditions and setpoints. We used a batch size of 64 and trained for 100,000 iterations using the Adam optimizer with a learning rate of  $1e-4$ . The diffusion process consisted of 1000 noise-prediction steps, following the approach outlined in [song2020score](#).

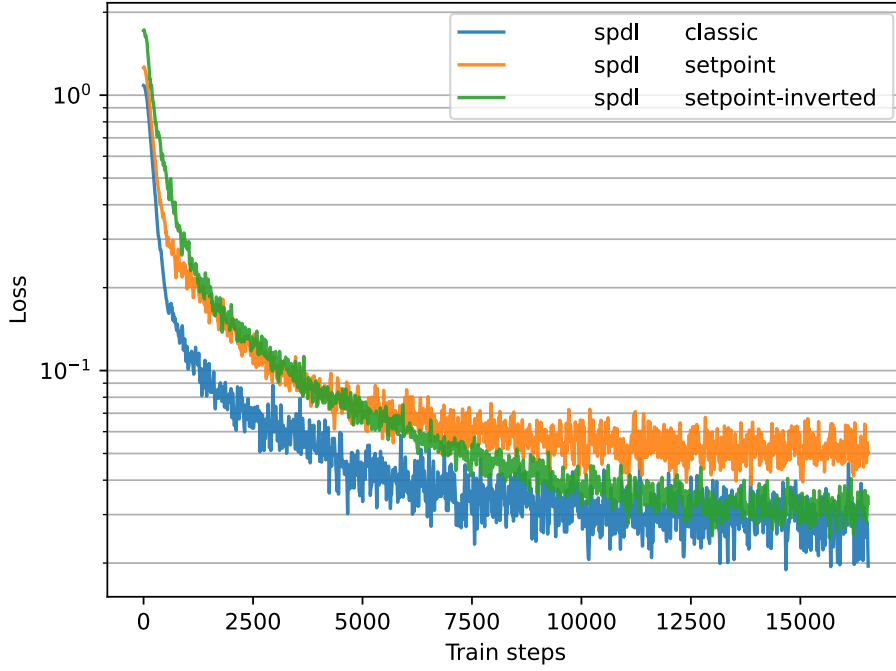


Figure 10: Train curves of DDPMS-Approach. In A...

Figure 10 illustrates the training curves for each of the three configurations, providing valuable insights into their respective learning dynamics and convergence properties. Our analysis reveals several key observations:

- **Convergence:** All three training curves demonstrate convergence, indicating the stability of our approach across different configurations.
- **Complete Generation Efficiency:** The complete generation method exhibits the highest efficiency, as evidenced by its lowest loss values. This aligns with findings in recent literature suggesting that unconstrained diffusion models often achieve superior performance [ho2020denoising](#).
- **Channel Generation Performance:** The channel generation training shows convergence comparable to full generation, albeit requiring more training steps. This trade-off between performance and computational cost is consistent with observations in conditional generation tasks [song2020score](#).
- **Partial Channel Generation:** Training with the generation of all channels except one exhibits lower convergence compared to full generation for the same training duration. This phenomenon may be attributed to the increased complexity of learning with partially fixed inputs, a challenge noted in recent work on constrained generative models [dhariwal2021diffusion](#).

These results suggest that while complete generation offers the best performance in terms of final loss, there may be practical scenarios where the other configurations provide valuable trade-offs between performance, training time, and specific task requirements. The observed differences in convergence rates and final performance across configurations underscore the importance of carefully selecting the generation strategy based on the specific requirements of the learning task at hand [janner2022planning](#).

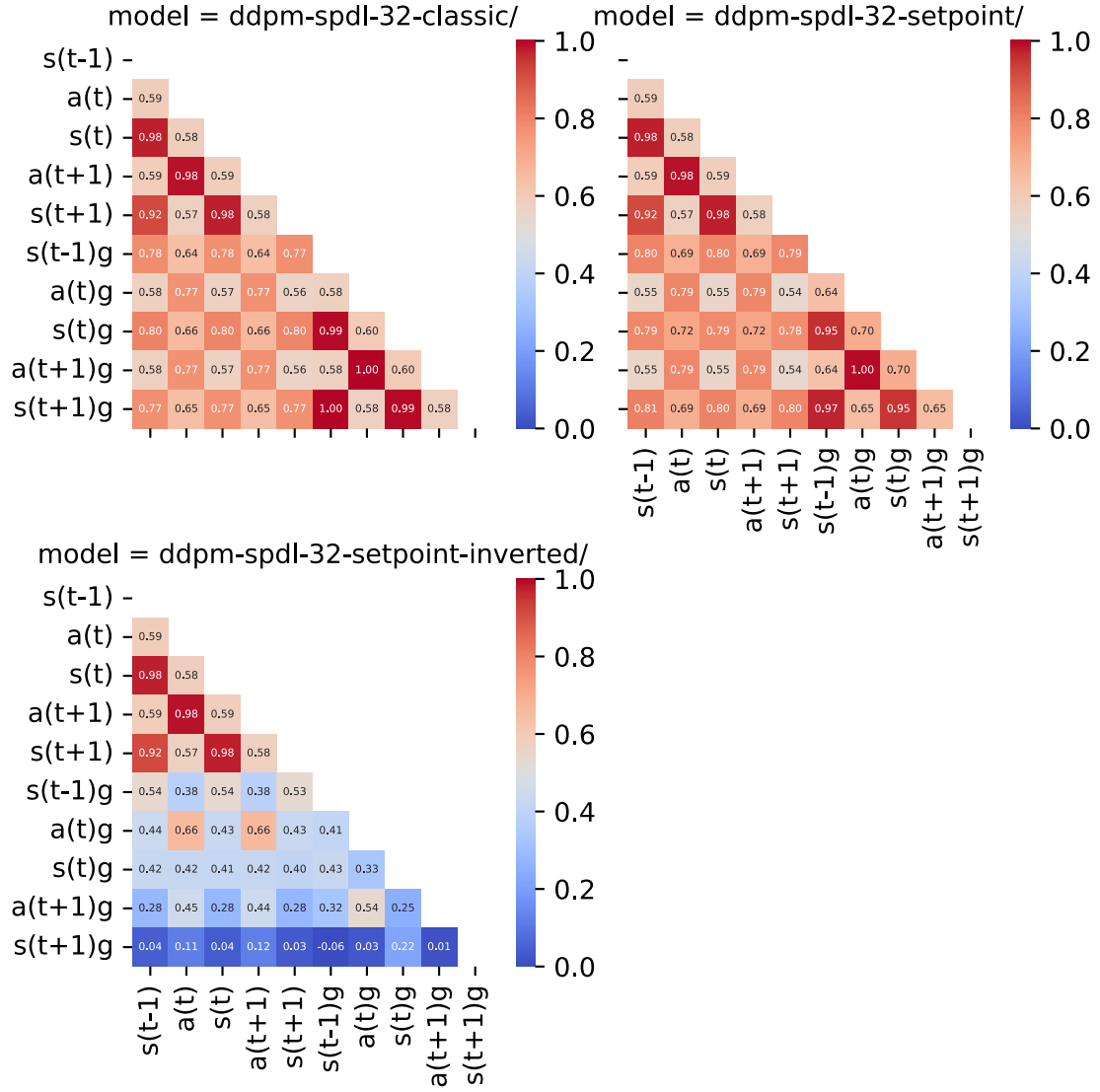


Figure 11: Correlation matrix of DDPMS prediction. In A... METTRE LES VALEUR DES CORRELATION DANS LE TABLEUX

### 6.3 Evaluation of Diffusion Methods for LTI Control

To assess the efficacy of our diffusion-based approach in controlling multiple LTI environments, we conducted a comprehensive evaluation of the three generation methods: Complete Generation, Partial Channel Generation, and Pure Setpoint Learning. Our evaluation metrics focused on correlation scores, which measure the alignment between the generated control signals and the observed control trajectories, as well as precision, which quantifies the accuracy of the control actions. Our analysis of the diffusion methods yielded several significant insights:

- **Correlation Scores:** We measured positive correlation scores for each generation method. Notably, Complete Generation and Partial Channel Generation demonstrated the highest scores, indicating their superior performance in controlling multiple LTI environments in parallel across various configurations. This aligns with recent findings on the versatility of diffusion models in handling complex, multi-dimensional tasks [janmer2022planning](#).
- **Precision Limitations:** Despite the positive results, the precision achieved (approximately 80%) is not sufficiently high for direct application in efficient multiple control problems. This level of precision is comparable to bang-bang control, which, while useful in certain contexts, lacks the nuance required for fine-tuned control in complex systems [kirk2004optimal](#).
- **Setpoint Learning Challenges:** We observed that pure setpoint learning was ineffective. Generating only one channel resulted in asymmetrical correlation tables, highlighting the importance of considering the

full state space in diffusion-based control strategies. This observation is consistent with recent work emphasizing the need for comprehensive state representation in reinforcement learning tasks [wang2023diffusion](#).

These results suggest that while setpoint diffusion methods show promise for controlling multiple LTI environments simultaneously, there is room for improvement, particularly in terms of precision. The challenges encountered with pure setpoint learning and single-channel generation underscore the complexity of applying diffusion models to control problems and the need for careful consideration of state representation.

## 7 Conclusions and perspective

Our study explores two primary axes of investigation: Reinforcement Learning (RL) and Diffusion Models, demonstrating a setpoint approach that offers adaptable and generalizable control for linear systems. The control environment constructed in this paper provides a foundation for developing more sophisticated models with additional hyperparameters.

The four configurations we’ve explored, particularly the divisible approach, represent a novel method to address generalization challenges in RL across multiple environments. This approach aligns with recent work on multi-scale reinforcement learning, where agents learn to operate at different levels of abstraction [\[7\]](#). By allowing for flexible division of both transfer functions and input/state variables, our method potentially offers a more robust approach to handling complex, multi-environment scenarios.

While our Denoising Diffusion Probabilistic Models (DDPMs) experiments show promising outcomes, they have not yet achieved optimal control. To address this limitation, future work could explore the use of State Space Models (SSMs) such as S4 [\[gu2022efficiently\]](#) or Vision-Mamba [\[zhu2023vision\]](#). These models could potentially allow for efficient prediction of actions from image sequences while minimizing computational requirements. However, it’s decisive to consider their limitations, including illusions, repetitions, and permutation-composition issues as highlighted in recent research [\[liu2024unveiling\]](#).

The integration of Transformer models to predict actions (TD) for a set of wrapped environments presents another promising possibility. Recent advancements in decision-making models, such as Decision Mamba and Decision Transformer [\[lee2024decision\]](#), offer potential for enhancing the adaptability and efficiency of control systems across diverse environments.

An direction for future research is the exploration of end-to-end control of physical environments using Large Language Models (LLMs). This approach could serve as an alternative to function calling techniques that demand significant computational power, such as those used in Eureka [\[shin2023eureka\]](#). Recent work like OpenVLA or RoboMamba have shown superior performance compared to RT-2 or Octo in generating actions for standard robots [\[li2024openvla\]](#), [\[RoboMamba\]](#).

Future work should focus on:

1. Exploring optimal division strategies for different types of environments.
2. Investigating the potential for combining our approach with other generalization techniques in reinforcement learning.
3. Integrating advanced models like SSMs and LLMs to improve robustness, efficiency, and generalizability.
4. Addressing the limitations of current models, particularly in handling complex, multi-environment scenarios.

For items 3, you can transform the problem of setpoint with encoder and decoder :

$$\begin{aligned} \{a_{(i,t-1)}, s_{(j,t-1)}, s_{(j,t)}\} &\longrightarrow \tilde{S}_{encoder} \\ \{\tilde{F}_{encoder}, s_{sp(j,t+1)}\} &\longrightarrow \tilde{S}_{decoder} \end{aligned} \quad (10)$$

The  $(\tilde{F}_{encoder}, s_{sp(j,t+1)})$  can be directly the output of encoder network, or the alignment output of LLM. In this problem, there can be a model specialized in the control of a physical model to keep it in equilibrium (e.g.: humanoid robot stabilized via RL) where the output of the LLM by the setpoint approach functions as an equilibrium disruptor to do a specific task (e.g.: move your arm to grab an object). Another more expensive approach in terms of hardware resources would be to train several modalities simultaneously to optimize the alignment and understanding of the backbone to a control problem, as exists with images and text [\[bao2022vlmo\]](#).

In conclusion, while our current approach demonstrates significant potential for generalized control in linear systems, the rapidly evolving landscape of AI and machine learning continues to offer new tools and methodologies. By leveraging these advancements, we can further improve the adaptability and efficiency of control systems across a wide range of applications, paving the way for more sophisticated and versatile AI-driven control solutions.

## References

- [1] *Gymnasium documentation*, [https://gymnasium.farama.org/environments/classic\\_control/cart\\_pole/](https://gymnasium.farama.org/environments/classic_control/cart_pole/).
- [2] *Rllib documentation*, <https://docs.ray.io/en/latest/rllib/index.html>.
- [3] M. ASSRAN, Q. DUVAL, I. MISRA, P. BOJANOWSKI, P. VINCENT, M. RABBAT, Y. LECUN, AND N. BALLAS, *Self-supervised learning from images with a joint-embedding predictive architecture*, 2023, <https://arxiv.org/abs/2301.08243>, <https://arxiv.org/abs/2301.08243>.
- [4] A. BEMPORAD AND M. MORARI, *Robust model predictive control: A survey*, in Robustness in Identification and Control, A. Garulli, A. Tesi, and A. Vicino, eds., vol. 245 of Lecture Notes in Control and Information Sciences, Springer-Verlag, 1999, pp. 207–226, <https://doi.org/10.1007/BFb0109870>.
- [5] K. COBBE, O. KLIMOV, C. HESSE, T. KIM, AND J. SCHULMAN, *Quantifying generalization in reinforcement learning*, (2019), <https://arxiv.org/abs/1812.02341>.
- [6] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, ET AL., *Human-level control through deep reinforcement learning*, *nature*, 518 (2015), pp. 529–533, <https://doi.org/https://doi.org/10.1038/nature14236>.
- [7] O. NACHUM, S. GU, H. LEE, AND S. LEVINE, *Data-efficient hierarchical reinforcement learning*, 2018, <https://arxiv.org/pdf/1805.08296.pdf>.
- [8] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, 2017, <https://arxiv.org/abs/1707.06347>.
- [9] H. VAN HASSELT, A. GUEZ, AND D. SILVER, *Deep reinforcement learning with double Q-learning*, in Proceedings of the AAAI conference on artificial intelligence, vol. 30, 2016, <https://arxiv.org/abs/1509.06461>.
- [10] Z. WANG, T. SCHAUL, M. HESSEL, H. HASSELT, M. LANCTOT, AND N. FREITAS, *Dueling network architectures for deep reinforcement learning*, in International conference on machine learning, PMLR, 2016, pp. 1995–2003, <https://arxiv.org/abs/1511.06581>.
- [11] Z. XIAO, K. KREIS, AND A. VAHDAT, *Tackling the generative learning trilemma with denoising diffusion gans*, 2022, <https://arxiv.org/abs/2112.07804>, <https://arxiv.org/abs/2112.07804>.