The Indoor-Training Effect: Unexpected Gains from Distribution Shifts in the Transition Function

Serena Bono[1], Spandan Madan[2], Ishaan Grover[1], Mao Yasueda[3], Cynthia Breazeal[1], Hanspeter Pfister[2], Gabriel Kreiman[4]

[1]Massachusetts Institute of Technology
[2]Harvard University
[3]Yale University
[4]Harvard Medical School

sebono@mit.edu, spandan_madan@g.harvard.edu, igrover@mit.edu, mao.yasueda@yale.edu, cynthiab@media.mit.edu, pfister@g.harvard.edu, gabriel.kreiman@childrens.harvard.edu

Abstract

Is it better to perform tennis training in a pristine indoor environment or a noisy outdoor one? To model this problem, here we investigate whether shifts in the transition probabilities between the training and testing environments in reinforcement learning problems can lead to better performance under certain conditions. We generate new Markov Decision Processes (MDPs) starting from a given MDP, by adding quantifiable, parametric noise into the transition function. We refer to this process as Noise Injection and the resulting environments as δ-environments. This process allows us to create variations of the same environment with quantitative control over noise serving as a metric of distance between environments. Conventional wisdom suggests that training and testing on the same MDP should yield the best results. In stark contrast, we observe that agents can perform better when trained on the noise-free environment and tested on the noisy δ-environments, compared to training and testing on the same δ-environments. We confirm that this finding extends beyond noise variations: it is possible to showcase the same phenomenon in ATARI game variations including varying Ghost behavior in PacMan, and Paddle behavior in Pong. We demonstrate this intriguing behavior in 60 different variations of ATARI games, including PacMan, Pong, and Breakout. We refer to this phenomenon as the Indoor-Training Effect. Code to reproduce our experiments and to implement Noise Injection.

Code — https://github.com/serenabono/Pacman

Introduction

Consider the process of learning how to play tennis. You might think that the best way to prepare for an outdoor match is to train under the same outdoor conditions you will face during the match. However, training in a calm, noise-free indoor environment instead can help focus on mastering the fundamentals of tennis without the added challenge of sources of noise like wind. We refer to this phenomenon as the *Indoor-Training Effect*. Here we model this problem using reinforcement learning (RL) agents. Surprisingly, we found that under certain conditions, training in a noise-free environment can lead to better performance when tested in a noisy environment—just like tennis. This phenomenon challenges our intuitions about the standard way to train RL agents where conventional wisdom would suggest that the

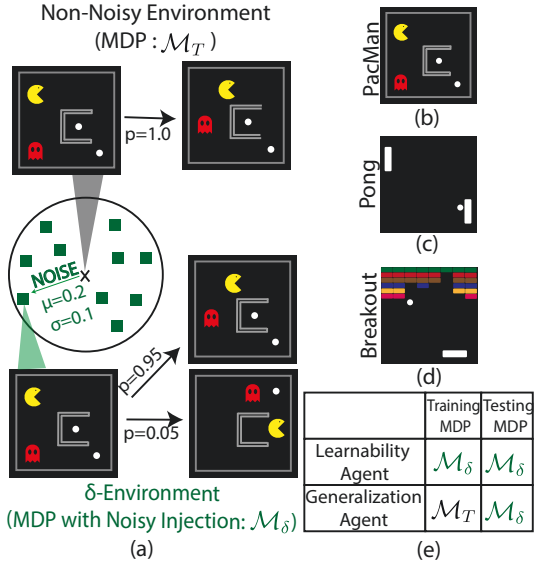

Figure 1: *ATARI games modified with Noise Injection.* (a) Noise Injection allows us to create multiple worlds in the vicinity of the original Target Environment($\mathcal{M}_T$) by adding controlled Gaussian noise (δ) to the original Transition Function (T). When the agent takes the action *right* in these $\delta-$environments, with a low probability the game may transition to a state which would not be possible in non-noisy PacMan. Experiments with noise injection are presented on three ATARI games—(b) PacMan, (c) Pong, and (d) Breakout. (e) We compare two agents with these environments—a Learnability agent trained and tested on the same target environment ($\mathcal{M}_\delta$), and a Generalization agent trained on a different MDP ($\mathcal{M}_T$) and tested on $\mathcal{M}_\delta$.

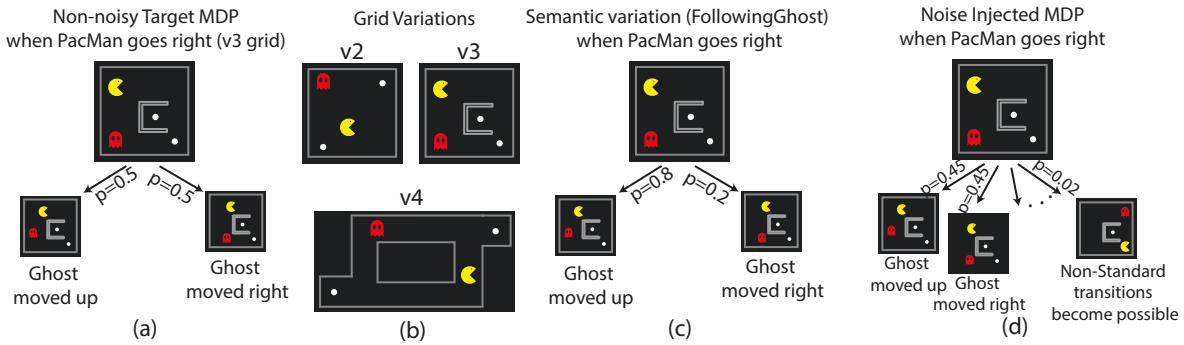

Figure 2: *Schematic illustration of variations for Pacman.*(a) Game dynamics when the agent picks the action *right* in a standard, non-noisy MDP for the v3 grid. (b) Grid variations for Pacman—v2, v3, and v4. These grids vary in size, positions of walls, and positions of food pellets. (c) Semantic variations whereby there is a meaningful change in the distribution of game elements. (d) Noise injected MDP generated by adding Gaussian noise to the standard transition function. Alongside states reachable by the ghost taking a legal move, non-standard transitions now become possible.

best approach to perform well on a target problem is to train an RL agent on the same test environment.

Environments in RL are usually described using Markov Decision Process (MDP). An MDP is defined by a State Space $\mathcal{S}$, an Action Space $\mathcal{A}$, a Transition Function $\mathcal{T}$, and a Reward Function $\mathcal{R}$. In practice, these parameters are assumed to be known or approximated with reasonable precision (Bäuerle and Glauner 2022; Goyal and Grand-Clement 2023). A significant challenge in RL is generalizing to environments that differ from the training environment (Cobbe et al. 2019; Kang et al. 2019; Devin et al. 2018). To address this, the RL community has focused on training agents capable of learning policies that perform well in novel, unseen environments at deployment time (Kirk et al. 2021; Moos et al. 2022a; OpenAI et al. 2019; Filos et al. 2020; Biedenkapp et al. 2020). The complexity of this task has called for ingenious ways of aligning the policy learned by the agent in training environments with the testing optimal policy. Notable approaches include using human feedback (Rummery and Niranjan 1994), using language (Tellex et al. 2011; Walter et al. 2013; Squire et al. 2015), and using vision (Guss et al. 2019; Osiński et al. 2020; Gopalan et al. 2017).

To study this, we explored zero-shot policy transfer where a policy trained in one environment is tested on a different environment. We extended past works which focused on uncertainty in the transition probabilities (Nilim and El Ghaoui 2005; Moos et al. 2022b; Goyal and Grand-Clement 2023), and propose a novel framework for studying zero-shot policy transfer in environments with controlled, quantifiable distribution shifts in the transition probabilities.

Our framework introduces these shifts by computing the transition function of an MDP, and adding small Gaussian noise to its entries. Starting with an environment ($\mathcal{M}_T$), noise is sampled and added to it to obtain a new MDP ($\mathcal{M}_\delta$). We refer to this approach as *Noise Injection* and the resulting new MDPs as δ*-environments* as in Fig. 1. Noise injection introduces several non-standard transitions, which had zero probability in the original MDP. Multiple such environments can be created by sampling noise and the noise serves as a metric of distance between environments. This approach allows us to create multiple worlds starting from the same MDP, with quantitative control over the variations in the transition probabilities. An increase in the standard deviation of the Gaussian noise results in increasingly perturbed MDPs. We report experiments with Noise Injection on multiple domains across three ATARI games—PacMan, Pong, and Breakout.

To study policy transfer we define two agents: a ***Learnability Agent*** ($\mathcal{L}_\delta$) which is **trained and tested on the same δ-*environment*** ($\mathcal{M}_\delta$), and a ***Generalization Agent*** ($\mathcal{G}_T$) which is **trained on the original noise-free environment** ($\mathcal{M}_T$) **but tested on the δ-*environment*** ($\mathcal{M}_\delta$). Conventional wisdom suggests that the Learnability Agent should perform better as it is trained and tested on the same environment. However, our study across 60 MDPs built on ATARI games reveals a surprising finding—there are several cases where the Generalization Agent outperformed the Learnability Agent. We confirmed that this finding extends beyond our setup of noise injection and δ-environments and also holds true for game variations including varying the Ghost behavior in PacMan, and Paddle behavior in Pong. We refer to these as semantic variations in MDPs.

In conclusion, to better understand this phenomenon, we analyzed the exploration patterns of the Learnability and Generalization Agents, and the corresponding policies learned by them. Our analyses revealed that $\mathcal{L}_\delta$ agents outperformed $\mathcal{G}_T$ agents, as expected from the literature, when $\mathcal{G}_T$ agents fail to explore the same State-Action pairs as the $\mathcal{L}_\delta$ agents. In contrast, when there were no large differences in their exploration patterns, the performance of $\mathcal{G}_T$ aligned or exceeded that of $\mathcal{L}_\delta$ agent.

Preliminaries: Reinforcement Learning

Similarly to (Cederborg et al. 2015), our work considers Reinforcement Learning (RL) as a group of algorithms designed to solve problems formulated as Markov Decision Processes (MDPs). A Markov Decision Process is characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \lambda)$, representing the collection of potential world states ($\mathcal{S}$), space of actions ($\mathcal{A}$), the transition function

ATARI Game	Grid Variations	Noise Injected Variations	Semantic Variation	Total
PacMan	v2, v3, v4	$\delta = 0$ (No Noise) $\delta \sim \mathcal{N}(0, 0.1)$ $\delta \sim \mathcal{N}(0, 0.5)$	RandomGhost FollowingGhost ($p = 0.3, 0.6$) TeleportingGhost ($p = 0.5, 0.2$)	33
Pong	p1, p2	$\delta = 0$ (No Noise) $\delta \sim \mathcal{N}(0, 0.1)$ $\delta \sim \mathcal{N}(0, 0.5)$	RandomPaddle FollowingPaddle ($p = 0.3, 0.6$)	18
Breakout	b1, b2, b3	$\delta = 0$ (No Noise) $\delta \sim \mathcal{N}(0, 0.1)$ $\delta \sim \mathcal{N}(0, 0.5)$	-	9

Table 1: **Overview of experimental protocol.** Our experiments include multiple variations of three ATARI games—PacMan, Pong, and Breakout. For each game, we have multiple grid variations. When introducing variations in these grids with noise injection, we report results for two levels of added noise—a low-noise setting: $\delta \sim \mathcal{N}(0, 0.1)$, and a high-noise setting: $\delta \sim \mathcal{N}(0, 0.5)$. Furthermore, for each grid we introduce further variations by modifying the distribution of the stochastic game element (ghost in PacMan, and the computer paddle in Pong). In all, we report results on 60 MDPs across these games.

($\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$), the reward function ($\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$), and a discount factor ($0 < \gamma \leq 1$). The objective is to identify policies ($\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$) that maximize cumulative rewards.

Q-learning (Watkins and Dayan 1992) and SARSA (Kaelbling, Littman, and Moore 1996) are two algorithms to learn such policies. Both Q-Learning and SARSA algorithms update the Q-values of state-action pairs, but they differ in their approaches. Q-Learning focuses on the maximum expected future rewards, and updates Q-values using the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \tag{1}$$

where α is the learning rate, γ is the discount factor, and s, s', a, a', r represent the current state, next state, current action, next action, and immediate reward, respectively.

On the other hand, SARSA updates Q-values based on the actual policy's actions with the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right] \tag{2}$$

Here the update incorporates both immediate rewards and the Q-value of the actual next action taken.

Agents must balance two critical aspects: exploration and exploitation. Exploration involves trying potentially less optimal actions to understand the environment better. Conversely, exploitation means choosing actions known to yield high rewards. We report results with the Boltzmann and the ϵ-greedy exploration strategies. Boltzmann exploration determines the probability of selecting an action as follows:

$$Pr_q(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}} \tag{3}$$

The constant τ is referred to as the temperature. On the other hand, the ϵ-greedy strategy is simpler and more direct—the agent selects a random action with probability ϵ, and the action with the highest Q-value with probability $1 - \epsilon$.

Related Works

Generalization benchmarks in RL involve training and testing across different subsets of tasks, levels, or environments. Recent years have seen several generalization benchmarks, which include variations in the state space (Hafner 2021), dynamics (Dulac-Arnold, Mankowitz, and Hester 2019), observation (Zhu et al. 2020), reward function (Bapst et al. 2019), and new game levels (Justesen et al. 2018), among others. There has also been recent work investigating generalization in Deep Reinforcement Learning (Zhu et al. 2023; Packer et al. 2018; Cobbe et al. 2019; Lyle et al. 2022). Combined, these tasks require explicit modeling of variations to effectively assess generalization, highlighting the need for robust evaluation protocols.

Contextual Markov Decision Processes (CMDP) provide a formal structure for this, where environments are sampled from a class of contexts, with agents trained on a subset and tested on a disjoint subset. These contexts are generated through two primary methods: Procedural Content Generation (PCG), which relies on a seed value for environment generation, and Controllable Environments (CE), which allow for manipulation of individual components. The integration of a suitable evaluation protocol with these contexts helps define the relationship between training and testing sets, which can range from interpolation to full extrapolation. Some examples of benchmarks using these frameworks include the OpenAI Procgen benchmark (Cobbe et al. 2020) and the Distracting Control Suite (Stone et al. 2021) for PCG (Ahmed et al. 2020) and RWRL (Dulac-Arnold, Mankowitz, and Hester 2019) for controllable environments. A major drawback in these benchmarks is the lack of a clearly defined metric for measuring how the distance between different contexts affects agent performance.

To solve this issue we draw inspiration from work studying generalization under controlled, quantifiable distribution shifts in computer vision. These studies include shifts in 3D rotation (Mondal, Dulberg, and Cohen 2022; Madan et al. 2023), category-viewpoint combinations (Madan et al. 2022a), incongruent scene context (Bomatter et al. 2021), novel light and viewpoint combinations (Sakai et al. 2022), object materials (Madan et al. 2022b) and texturess (Geirhos et al. 2018; Michaelis et al. 2019), and non-canonical viewpoints (Barbu et al. 2019), among others.

Generating MDPs for Investigating Generalization

We created 60 different MDPs across three ATARI games (PacMan, Pong, and Breakout) by varying grid layouts, distributions defining the stochasticity of different game elements, and modifying transition probabilities using Noise Injection (Fig. 2 and Table. 1). Here we outline these variations.

Domains We implemented all three ATARI games from scratch, building on the Berkeley PacMan Projects (DeNero, Klein, and Abbeel 2014). PacMan was modelled as an MDP characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \lambda)$.

State (s) and State Space ($\mathcal{S}$): We represented a grid of size $M \times N$ as a matrix of the same shape with the entries corresponding to the game element occupying the position in the grid—p (PacMan), g (Ghost), f (Food), w (Wall), or e (Empty). The state space $\mathcal{S}$ refers to the set of all possible states.

Action Space ($\mathcal{A}(s)$): Set of legal actions PacMan could take in state s. PacMan can move Left, Right, Up, or Down but not enter walls. Thus, when the PacMan is at the top left position the set of legal actions was only {Right, Down}.

Transition Matrix ($\mathcal{T}(s_i, a, s_j)$): Probability of moving to state s_j if the agent took action a at state s_i (Fig. 2a).

Reward Function ($\mathcal{R}(s)$): PacMan received +20 for eating a food pellet, -1 for every time step, -200 when it was killed, and +500 for finishing the game. (Cederborg et al. 2015).

Game Stochasticity: The motion of PacMan is deterministic—a left action (if legal) will ensure that PacMan moves left. However, ghosts move stochastically according to a prefixed distribution. For instance, a RandomGhost moves in all directions with equal probability (accounting for walls). Thus, the game is nondeterministic.

MDPs for Pong and Breakout are defined analogously. For additional details, please refer to Supplementary Section **Domains**.

Noise Injection Variation: Generating New Controlled Environments

We generate controlled variations of an original MDP by explicitly computing its Transition Function and then adding sampled noise to it.

Explicit enumeration of all states: States are defined by the position of the game elements. The probability of transitioning from one state to another is computed by multiplying the probability that each game element is able to reach the final configuration independently. Therefore, we visualize the game as a tree, each state is a node, and the edges represent the transition probabilities of the game elements independently reaching their final configuration. By rolling out all possible moves by each game elements at each step, we enumerate all possible reachable states.

Explicit computation of Transition Function: Once we have all possible states, we can calculate the transition function, denoted as $\mathcal{T}(s_i, a, s_j)$. This function is determined by calculating the probability of each game character moving from one state to another independently.

Creating $\delta-$environments: We introduce variations in the game environment by modifying the transition function to $\mathcal{T}_\delta = \mathcal{T} + \delta$. Here, δ is a variable that follows a normal distribution, randomly chosen before each game to add unpredictability (Fig. 2c). The modified transition function, $\mathcal{T}_\delta$, is then adjusted to make sure the total probability of moving from any state s_i using action a to any other state s_j sums to 1.

$$\mathcal{T}_\delta(s_j, a, s_i) = \frac{|\mathcal{S}|p_{i,j} + \delta_{i,j}}{|\mathcal{S}| + \sum_j \delta_{i,j}} \tag{4}$$

$|\mathcal{S}|$ denotes the number of states, and guarantees the probability of legal successors does not approach 0 as the state space grows. We investigated two settings—(i) *Low-Noise* with $\delta \sim \mathcal{N}(0, 0.1)$, where some non-standard transitions previously impossible without noise are now possible with a low probability. (ii) *High-Noise* with $\delta \sim \mathcal{N}(0, 0.5)$, where non-standard transitions are possible with higher probability. We further analyze minimum $\mathcal{N}(0, 0)$, and maximum $\mathcal{N}(0, 1)$ perturbation settings in the Supplement Sec. **Perturbation Bounds**.

Experimental Details

We compared the mean reward curve of Learnability and Generalization agents. An agent $\mathcal{G}_T$ is said to generalize well with respect to $\mathcal{M}_\delta$, if its mean reward is as good as the corresponding Learnability agent $\mathcal{L}_\delta$.

Agents are trained with both tabular Q-Learning (Watkins and Dayan 1992) and SARSA Q-learning (Kaelbling, Littman, and Moore 1996), using Boltzmann or ϵ-greedy exploration strategies. In particular, we trained agents for $1,000$ episodes and averaged results over 500 trained agents. After every 10 training episodes, agents were evaluated using 10 testing episodes. We report the mean reward curves at convergence. Hyperparameters were inherited from past work (Cederborg et al. 2015) and are available in the Supplement in Sec. **Training Parameters**. We extended the analysis to DQN (Cobbe et al. 2019) and reported the results in the Supplement in Sec. **DQN**. The experiments were conducted on a system with an Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz.

Results

We report findings from the Generalization and Learnability agents trained with the multiple variations of PacMan, Pong, and Breakout as described in Sec. **Generating MDPs for investigating generalization** and Table 1.

Generalization Agents Can Outperform Learnability Agents in Several Instances of the Indoor-Training Effect

The mean reward increased with training, as expected, (Fig. 3a), both for the Generalization agent (red) and for the Learnability agent (green). Also, as intuitively expected, both agents performed better under low-noise conditions (solid lines) compared to high-noise conditions (lines with '–' markers). Less intuitive was the relationship between Generalization and Learnability agents. Intriguingly, the Generalization agent consistently outperformed the Learnability

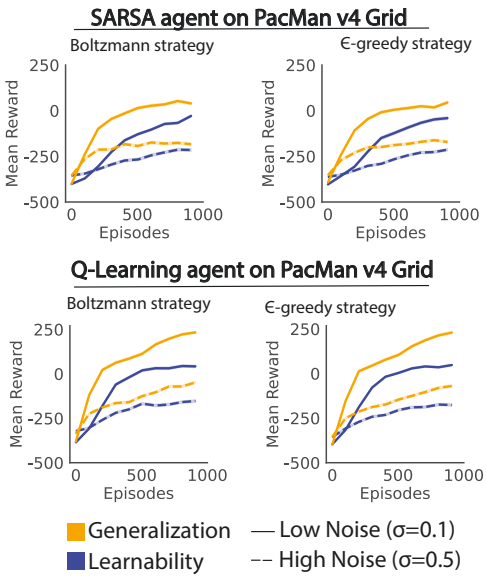

Figure 3: *Generalization agents can outperform Learnability agents.* Results for PacMan v4 grid with a RandomGhost reporting mean reward as a function of episode number. (a) SARSA agent trained with a Boltzmann exploration strategy for Target MDPs generated with both high (solid line) and low (line with 'x' markers) level noise injection. The Generalization Agent (red) beats the Learnability Agent (green) (two-sided t-test, p<0.001). (b) The same result holds for a SARSA agent trained with the $\epsilon-$greedy exploration strategy. This finding also holds for Q-Learning agents trained with (c) Boltzmann and (d) $\epsilon-$greedy exploration strategies. Standard deviation across the 500 agents is reported as the error bar in all figures. However, the standard deviation is too small for these error bars to be visible.

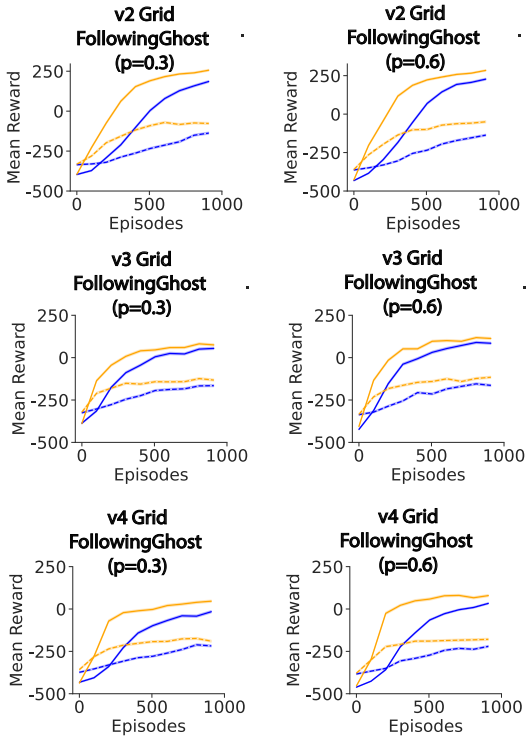

Figure 4: *Generalization can outperform Learnability across multiple variations of PacMan.* Format and conventions as in Fig. 3. Agents trained on v2, v3 and v4 with a DirectionalGhost with $p = 0.3$ and $p = 0.6$. All experiments are shown for SARSA agents trained with the $\epsilon-$greedy exploration strategy. Generalization agents consistently beat Learnability Agents (two-sided t-test, p<0.001).

agent (two-sided t-test, p<0.001). This gap continued until convergence at $1,000$ episodes, was observed both across low and high noise levels (solid lines versus '–' lines), when using a Boltzmann strategy (Fig. 3a, c) or an $\epsilon-$greedy strategy (Fig. 3b, d), and when using SARSA agents (Fig. 3a, b) or Q-Learning agents (Fig. 3c, d). Another metric commonly used to assess performance is the *Area Under the Curve* (AUC). We compute the ratio between the learnability and generalization agents' AUC and normalize it using regret. (see Supp. Sec. **Regret Normalization**).

To assess whether this observation was dependent on the target MDP, we replicated these findings on multiple PacMan grids and noise variations (Fig. 4). In Fig. 4, the Generalization agents beat the Learnability agents, for both low and high levels of noise (see Supp. Sec. **Additional Graphs Non-Semantic Variations**: Figs. Sup3- Sup5 for Boltzmann strategy and Q-learning results).

We also extended these findings to two additional ATARI games, Pong Fig. Sup1 and Breakout Fig. Sup2, to assess their applicability across different games (Fig. 5). Consistent with the results described for Pacman, the Generalization agent was on par with or better than the Learnability agent in Pong Fig. 5a,b) and Breakout Fig. 5c,d) (two-sided t-test, p<0.001; see Figs. Sup6- Sup12 for results with Q-Learning, Sarsa and different sampling strategies).

In sum, there exist several MDPs where it is better to train on a different MDP than the target. These results provide novel intriguing evidence suggesting that training on a different MDP can enable more efficient policy learning than training on the target environment.

Instances of the Indoor-Training Effect in Semantic Variations of ATARI Games

The results presented so far focused on altered MDPs generated by noise injection. Next, we evaluated *semantic* variations, where the changes are more meaningful and interpretable. Specifically, we modified the transition probabilities of Pacman so that ghosts could teleport to new locations, and Pong so that the paddle could follow the ball. We refer to these alternate semantic environments as $\mathcal{M}_{T'}$ (semantic noise), in contrast to $\mathcal{M}_{\delta}$ used for noise injection.

For PacMan, Learnability agents were trained and tested using TeleportingGhosts ($\mathcal{M}_{T'}$), while the Generalization agents were trained with PacMan with RandomGhosts ($\mathcal{M}_{T}$) and then tested on TeleportingGhosts ($\mathcal{M}_{T'}$) (Fig. 6a, b, Supp.

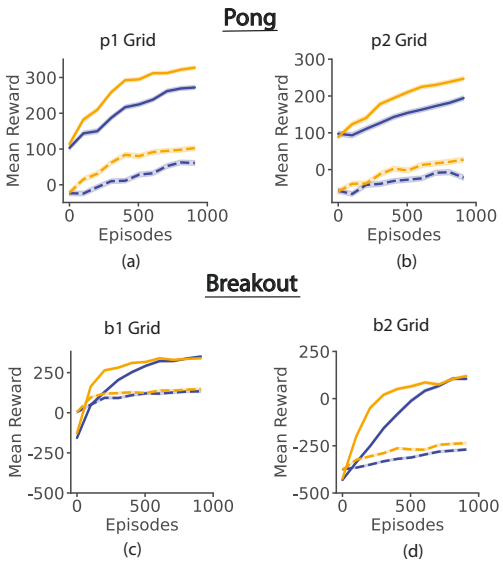

Figure 5: *Generalization agents outperform Learnability agents on Pong and Breakout as well.* Format and conventions as in Fig. 3. Performance of SARSA agents trained with an $\epsilon-$greedy exploration strategy on (a) Pong p1 grid, (b) Pong p2 grid, (c) Breakout b1 grid, and (d) Breakout b2 grid. The Generalization Agent consistently beats the Learnability Agent (two-sided t-test, p<0.001).

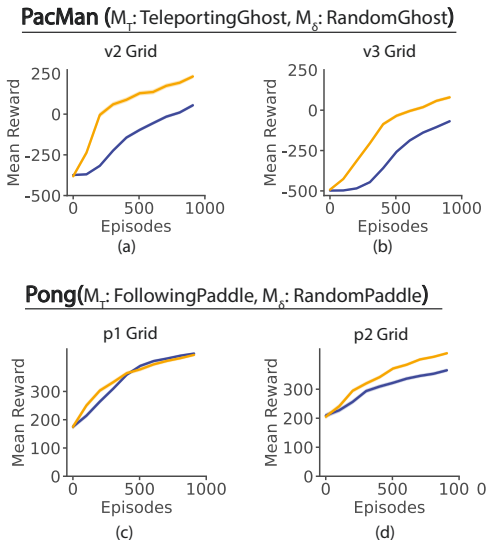

Figure 6: *Generalization agents outperform Learnability agents on semantic variations of PacMan and Pong as well.* Format and conventions as in Fig. 3. (a) Given the target PacMan MDP with the v2 grid and TeleportingGhost, the Generalization trained on the RandomGhost outperformed the Learnability agent that was trained and tested on the same Target MDP (TeleportingGhost) (two-sided t-test, p<0.001). (b) This finding extends to TeleportingGhost and RandomGhost MDPs with the PacMan v3 Grid as well. (c) For the Pong p1 grid, Generalization agents trained on an MDP with DirectionalPaddle performed better on the RandomPaddle MDP during testing, as compared to the Learnability Agent trained and tested on the RandomPaddle MDP. (d) The same finding extends to the p2 grid as well.

Sec. **Additional Graphs Non-Semantic Variations**). Even under these semantic noise conditions, Generalization agents outperformed Learnability agents (two-sided t-test, p<0.001; see Figs. Sup13-Sup15 for results with Q-Learning, Sarsa and different sampling strategies). In the case of Pong, we report analogous results with $\mathcal{M}_{T'}$ set to FollowingPaddle, and $\mathcal{M}_T$ set to RandomPaddle. Generalization agents also outperformed Learnability agents (by a smaller margin) in both the p1 and p2 grids (Fig. 6c,d). Analogous results for Pong with Q-Learning and other exploration strategies are reported in Figs. Sup16- Sup23.

The Exploration Patterns of State-Action Pairs Can Predict Differences Between Generalization and Learning Agents

To better understand how Generalization agents could outperform Learnability agents, we investigated the exploration patterns for $\mathcal{L}_\delta$ and $\mathcal{G}_T$. We enumerated all State ($\mathcal{S}$)-Action ($\mathcal{A}$) Pairs, and divided them into three groups—(i) Percentage of $\mathcal{S}$-$\mathcal{A}$ pairs explored by both agents (P_{LG}), (ii) Percentage of pairs explored only by the Learnability agent (P_L), and (iii) Percentage of pairs explored only by the Generalization agent (P_G). Thus, $P_{LG} + P_L + P_G = 100$. We defined $D_{LG} = P_L + P_G$, the divergence in the exploration patterns between these two agents.

In Fig. 7 we visualize D_{LG} for grids where $\mathcal{G}_T$ outperformed $\mathcal{L}_\delta$ agents and compare it to cases where it did not. Fig. 7a shows an agent trained with Q-Learning and Boltzmann exploration strategy for the Pacman v3 grid with RandomGhost stochasticity, where $\mathcal{G}_T$ beat the $\mathcal{L}_\delta$ agent. The corresponding panel Fig. 7b depicts D_{LG} —each entry of this grid represents an $\mathcal{S}$-$\mathcal{A}$ pair. We refer to this plot as the *exploration grid* for these agents. The exploration grid shows that most $\mathcal{S}$-$\mathcal{A}$ pairs were explored by both agents, with almost no pairs explored only by one type of agent and therefore no significant differences in their exploration patterns. In contrast, Fig. 7c, d report $\mathcal{S}$-$\mathcal{A}$ pairs for PacMan v2, here the $\mathcal{G}_T$ agent performs worse than the $\mathcal{L}_\delta$ agent. The exploration grid reveals that there is a high fraction of $\mathcal{S}$-$\mathcal{A}$ pairs explored either by one *or* the other agent but not both.

We grouped all the cases where $\mathcal{L}_\delta > \mathcal{G}_T$ (Fig. 7e, brown) and all the cases where $\mathcal{L}_\delta < \mathcal{G}_T$ (Fig. 7e, gray) and computed D_{LG}. On average, D_{LG} was significantly higher in MDPs where $\mathcal{L}_\delta > \mathcal{G}_T$ (two-sided t-test, $p < 0.05$). The same result holds true for Pong MDPs as reported in Fig. 7f. Exploration grids and additional results for variations of PacMan (Sup25-Sup32), Pong (Sup33-Sup40), and Breakout (Sup41-Sup44) can be found in the Supplement in Sec. **Additional Graphs State-Action Pairs**. Instead of grouping MDPs, we also conducted a correlation analysis. We defined the Reward Gap: $R_{LG} = R_G - R_L$. The Spearman correlation coefficient between D_{LG} and R_{LG} was 0.43 ($p < 0.005$)

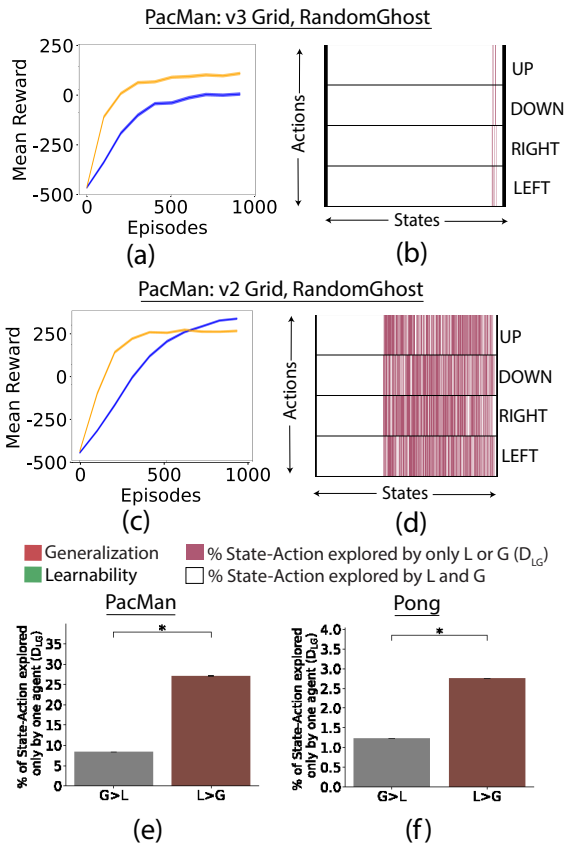

Figure 7: *The exploration patterns predict the reward gap between $\mathcal{L}_\delta$ and $\mathcal{G}_T$.* (a) Reward for agents trained on Pacman v3, where $\mathcal{G}_T$ outperforms $\mathcal{L}_\delta$ (format as in Fig. 3). (b) *Exploration grid* visualizing the difference in State-Action ($\mathcal{S}$-$\mathcal{A}$) pairs explored by these agents (D_{LG}). The grid shows Sates on the x-axis and Actions on the y-axis. The black lines separate the Actions for clarity. Each cell corresponds to one $\mathcal{S}$-$\mathcal{A}$ pair. In this case, a negligible fraction of $\mathcal{S}$-$\mathcal{A}$ pairs were visited only by one agent (pink). (c) Rewards for agents trained on PacMan v3. Here, $\mathcal{G}_T$ performs worse than $\mathcal{L}_\delta$ at the end of training. (d) A large fraction of pairs were only visited by either one or the other agent but not both (contrast with part (b)). (e) D_{LG} averaged over PacMan grids where $\mathcal{G}_T$ outperformed $\mathcal{L}_\delta$ (gray) and vice-versa (brown) (f) D_{LG} averaged over Pong grids. The "*" is for statistical significance (two-sided t-test, p<0.001).

for PacMan and 0.26 ($p < 0.005$) for Pong. Combined, these analyses show that the *Indoor-Training Effect* is associated with similar exploration patterns in the training and testing environments.

Discussion

In this work, our objective is to understand the paradoxical *Indoor-Training Effect* - where agents perform better when trained in a noise-free environment and tested in noisy δ-environments, compared to being trained and tested in the same δ-environments. Similarly to how training in a quiet, noise-free indoor environment helps athletes focus on mastering the fundamentals of tennis, we explore whether training in certain environments is more conducive to learning than training on the same testing environment.

To investigate this, we propose a new methodology to generate modified MDPs from a given MDP, along with a metric to quantify the distance between different environments. We demonstrate the Indoor-Training Effect across various algorithms and exploration strategies (Fig. 3), grid layouts and game stochasticity (Fig. 4), and multiple ATARI games (Fig. 5). We also showed that this phenomenon extends beyond Noise Injected environments, and can also occur when semantic changes are introduced in the game elements (Fig. 6).

To gain deeper insights into these environments, we examine the exploration patterns of agents under different transition probabilities. Similarly to a tennis player who has never encountered a smash serve during their training and develops an optimal playing style that does not anticipate or respond to such powerful shots, the suboptimal performance of the agents could be caused by a divergence in exploration patterns. We show that the performance gap between agents is indeed correlated with their exploration patterns under different transition probabilities (Fig. 7).

The Indoor-Training Effect is particularly relevant to robotics, where robots often operate in complex, dynamic environments. The Indoor-Training Effect opens new avenues of research, whereby robotic systems could be trained in simplified, controlled settings to master essential skills without the interference of noise. This finding could also enhance their ability to adapt and perform in real-world conditions where unpredictability and noise are prevalent. Such training strategies could lead to more robust, adaptable robots capable of navigating and executing tasks effectively in diverse and challenging environments.

We note that these findings are reminiscent of results with biological agents. For example, recent experiments with the *C. Elegans* worm have shown that biological agents perform best when cross-trained on different environments as compared to being tested on the environments they were trained on (Li, Kreiman, and Ramanathan 2024).

Despite the evidence provided in this study, we would like to highlight two main limitations. Firstly, our experiments were conducted solely in the context of ATARI games. We hope that future research can extend and examine the findings in real-world environments. Secondly, it will be interesting to assess whether the conclusions drawn from classical Reinforcement Learning methods extend to deep RL approaches.

These findings raise fundamental questions about our understanding of RL algorithms. Typically, RL practitioners have strived to train agents in environments that closely resemble their deployment conditions. This approach assumes that matching the training and testing environments is critical for optimal performance. However, the Indoor-Training Effect challenges this assumption by showing that agents trained in noise-free environments can sometimes outperform those trained in more chaotic, realistic settings when faced with noisy, unpredictable scenarios during testing.

References

Ahmed, O.; Träuble, F.; Goyal, A.; Neitz, A.; Wüthrich, M.; Bengio, Y.; Schölkopf, B.; and Bauer, S. 2020. CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning. *CoRR*, abs/2010.04296.

Bapst, V.; Sanchez-Gonzalez, A.; Doersch, C.; Stachenfeld, K. L.; Kohli, P.; Battaglia, P. W.; and Hamrick, J. B. 2019. Structured agents for physical construction. *CoRR*, abs/1904.03177.

Barbu, A.; Mayo, D.; Alverio, J.; Luo, W.; Wang, C.; Gutfreund, D.; Tenenbaum, J.; and Katz, B. 2019. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32.

Bäuerle, N.; and Glauner, A. 2022. Distributionally robust Markov decision processes and their connection to risk measures. *Mathematics of Operations Research*, 47(3): 1757–1780.

Biedenkapp, A.; Bozkurt, H. F.; Eimer, T.; Hutter, F.; and Lindauer, M. 2020. Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. In *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*.

Bomatter, P.; Zhang, M.; Karev, D.; Madan, S.; Tseng, C.; and Kreiman, G. 2021. When Pigs Fly: Contextual Reasoning in Synthetic and Natural Scenes. arXiv:2104.02215.

Cederborg, T.; Grover, I.; Isbell, C. L.; and Thomaz, A. L. 2015. Policy Shaping with Human Teachers. In *International Joint Conference on Artificial Intelligence*.

Cobbe, K.; Hesse, C.; Hilton, J.; and Schulman, J. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. arXiv:1912.01588.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, 1282–1289. PMLR.

DeNero, J.; Klein, D.; and Abbeel, P. 2014. CS188: Berkeley Pacman Projects. http://ai.berkeley.edu/home.html (Spring 2014).

Devin, C.; Abbeel, P.; Darrell, T.; and Levine, S. 2018. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 7111–7118. IEEE.

Dulac-Arnold, G.; Mankowitz, D. J.; and Hester, T. 2019. Challenges of Real-World Reinforcement Learning. *CoRR*, abs/1904.12901.

Filos, A.; Tigas, P.; McAllister, R.; Rhinehart, N.; Levine, S.; and Gal, Y. 2020. Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts? *CoRR*, abs/2006.14911.

Geirhos, R.; Rubisch, P.; Michaelis, C.; Bethge, M.; Wichmann, F. A.; and Brendel, W. 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

Gopalan, N.; Littman, M.; MacGlashan, J.; Squire, S.; Tellex, S.; Winder, J.; Wong, L.; et al. 2017. Planning with abstract Markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, 480–488.

Goyal, V.; and Grand-Clement, J. 2023. Robust Markov decision processes: Beyond rectangularity. *Mathematics of Operations Research*, 48(1): 203–226.

Guss, W. H.; Codel, C.; Hofmann, K.; Houghton, B.; Kuno, N.; Milani, S.; Mohanty, S.; Liebana, D. P.; Salakhutdinov, R.; Topin, N.; et al. 2019. The MineRL 2019 competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*.

Hafner, D. 2021. Benchmarking the Spectrum of Agent Capabilities. *CoRR*, abs/2109.06780.

Justesen, N.; Torrado, R. R.; Bontrager, P.; Khalifa, A.; Togelius, J.; and Risi, S. 2018. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.*, 4: 237–285.

Kang, K.; Belkhale, S.; Kahn, G.; Abbeel, P.; and Levine, S. 2019. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)*, 6008–6014. IEEE.

Kirk, R.; Zhang, A.; Grefenstette, E.; and Rocktäschel, T. 2021. A Survey of Generalisation in Deep Reinforcement Learning. *CoRR*, abs/2111.09794.

Li, C.; Kreiman, G.; and Ramanathan, S. 2024. Discovering neural policies to drive behavior by integrating deep reinforcement learning agents with biological neural networks. *Nature Machine Intelligence*, In Press.

Lyle, C.; Rowland, M.; Dabney, W.; Kwiatkowska, M.; and Gal, Y. 2022. Learning dynamics and generalization in deep reinforcement learning. In *International Conference on Machine Learning*, 14560–14581. PMLR.

Madan, S.; Henry, T.; Dozier, J.; Ho, H.; Bhandari, N.; Sasaki, T.; Durand, F.; Pfister, H.; and Boix, X. 2022a. When and how convolutional neural networks generalize to out-of-distribution category–viewpoint combinations. *Nature Machine Intelligence*, 4(2): 146–153.

Madan, S.; Sasaki, T.; Pfister, H.; Li, T.-M.; and Boix, X. 2023. Adversarial examples within the training distribution: A widespread challenge. arXiv:2106.16198.

Madan, S.; You, L.; Zhang, M.; Pfister, H.; and Kreiman, G. 2022b. What makes domain generalization hard? arXiv:2206.07802.

Michaelis, C.; Mitzkus, B.; Geirhos, R.; Rusak, E.; Bringmann, O.; Ecker, A. S.; Bethge, M.; and Brendel, W. 2019. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*.

Mondal, S. S.; Dulberg, Z.; and Cohen, J. 2022. Generalization to Out-of-Distribution transformations.

Moos, J.; Hansel, K.; Abdulsamad, H.; Stark, S.; Clever, D.; and Peters, J. 2022a. Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Machine Learning and Knowledge Extraction*, 4(1): 276–315.

Moos, J.; Hansel, K.; Abdulsamad, H.; Stark, S.; Clever, D.; and Peters, J. 2022b. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1): 276–315.

Nilim, A.; and El Ghaoui, L. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5): 780–798.

OpenAI; Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; Schneider, J.; Tezak, N.; Tworek, J.; Welinder, P.; Weng, L.; Yuan, Q.; Zaremba, W.; and Zhang, L. 2019. Solving Rubik's Cube with a Robot Hand. *CoRR*, abs/1910.07113.

Osiński, B.; Jakubowski, A.; Zięcina, P.; Miłoś, P.; Galias, C.; Homoceanu, S.; and Michalewski, H. 2020. Simulation-based reinforcement learning for real-world autonomous driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 6411–6418. IEEE.

Packer, C.; Gao, K.; Kos, J.; Krähenbühl, P.; Koltun, V.; and Song, D. 2018. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*.

Rummery, G. A.; and Niranjan, M. 1994. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

Sakai, A.; Sunagawa, T.; Madan, S.; Suzuki, K.; Katoh, T.; Kobashi, H.; Pfister, H.; Sinha, P.; Boix, X.; and Sasaki, T. 2022. Three approaches to facilitate invariant neurons and generalization to out-of-distribution orientations and illuminations. *Neural Networks*, 155: 119–143.

Squire, S.; Tellex, S.; Arumugam, D.; and Yang, L. 2015. Grounding English commands to reward functions. In *Robotics: Science and Systems*.

Stone, A.; Ramirez, O.; Konolige, K.; and Jonschkowski, R. 2021. The Distracting Control Suite – A Challenging Benchmark for Reinforcement Learning from Pixels. arXiv:2101.02722.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M.; Banerjee, A.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 1507–1514.

Walter, M. R.; Hemachandra, S. M.; Homberg, B. S.; Tellex, S.; and Teller, S. 2013. Learning semantic maps from natural language descriptions. Robotics: Science and Systems.

Watkins, C. J. C. H.; and Dayan, P. 1992. Q-learning. *Machine Learning*, 8(3): 279–292.

Zhu, Y.; Wong, J.; Mandlekar, A.; and Martín-Martín, R. 2020. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *CoRR*, abs/2009.12293.

Zhu, Z.; Lin, K.; Jain, A. K.; and Zhou, J. 2023. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.