Abstract of "Building an Explainable Reinforcement Learning Agent" by Jun Ki Lee, Ph.D., Brown University, January 2020.

The past five years have seen rapid advances in reinforcement learning. Using deep reinforcement-learning techniques, agents can play many Atari games at a superhuman level and win against human masters in Go and Starcraft. Despite these achievements, there is relatively little work on the problem of generating explanations of agent behavior. After training an agent using a technique called Deep Q Network (DQN), I analyzed the robustness of the learned value network. I measured robustness in terms of whether it could select sensible decisions in a range of semantically different states. I found that generalization was quite poor and, as a result, generating explanations for the learned behavior was impossible. Moreover, based on these and related results from the literature, evidence that deep networks learn high-level explainable structures that can convey insights to human users is quite weak. Judea Pearl claims that causality plays an essential role in human explanations. In my proposed work, I will develop a method for an agent to learn to make decisions by discovering the underlying causal structures in its environment, creating a "causal" reinforcement learning agent. I will then show that the behavior of such an agent can be meaningfully explained, leading to more trustworthy and transparent AI.

# Building an Explainable Reinforcement Learning Agent

Jun Ki Lee

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

**Thesis Statement** An attempt to generate explanations only using value and policy networks for reinforcement learning agents shows little success due to both its nature and the algorithms' inability to generalize. Judea Pearl suggests that causality plays an essential role in the human explanation. By adopting structural causal models within sequential decision making, learned agents can be better explained to human users.

With the invention of fast graphical processing units (GPUs) [37] in combination with deep learning techniques [32, 31, 14], there has been rapid progress in reinforcement learning. Learned agents can play Atari games at a superhuman level [45, 11, 12] and win human experts in games like Go, Poker, and Starcraft [64, 47, 72]. However, there is still little understanding of how these systems work from the perspectives of general audiences other than the experts. Since this thesis is especially interested in sequential decision making systems which are learned by reinforcement learning algorithms, I revisit the current developments in this area and investigate the meaning of explainable AI (XAI) for explaining the reinforcement learning agents.

The following part consists of my work relating to probing the generalization capacity of a deep RL algorithm called deep Q networks (DQN) [77]. Investigating what happens inside a deep artificial neural network after its training can be considered as a post-hoc method. Moreover, the semantic state perturbations are especially designed to promote counterfactual reasoning in explaining the learned agent. The results shows that how the algorithm genuinely fails to generalize with respect to these semantic perturbations in its input states, thus disabling the counterfactual reasoning between actual states and imaginary states. Given the lack of evidence that high-level explainable structures are found within the network, users are still left with weak insights on how the agents work.

According to Pearl [54], causality play an essential role in human explanation. I introduce a structural causal model as a mean to enhance explainability of a sequential decision making process. In my proposed work, I will develop a method for an agent to learn to make decisions by discovering the underlying causal structures in its environment, creating a "causal" reinforcement learning agent. I will then show that the behavior of such an agent can be meaningfully explained, leading to more trustworthy and transparent AI.

# Chapter 2

# Backgrounds

This chapter defines and describes underlying major concepts behind the main idea of this thesis. First, it explains what reinforcement learning is and the definition of Markov decision processes. The descriptions of most recent deep versions of reinforcement learning techniques are followed. Second, it tries to disambiguate the meaning of explainable AI and investigates recent related work in the context of supervised learning and reinforcement learning. Lastly, causality is introduced due to its importantness in human explanations and studies of explanations are discussed.

## 2.1 Reinforcement Learning

What distinguishes reinforcement learning from machine learning is that it learns by acting in an environment and receiving its reward in a delayed manner [67]. Its decision making are in general modeled using Markov Decision Processes.

A Markov Decision Process (MDP) is defined by 4-tuple $\langle S, A, T, R \rangle$. From all possible states $S$ in an environment, the agent starts from a state $s_0 \in S_0$ from a set of possible start states $S_0 \subset S$. $A$ contains all possible actions that an agent can take in each time step. The transition function $T : S, A \to S$ maps a state and action pair to its next state. The reward function $R : S \to \mathbb{R}$ determines the reward an agent receives when it encounters a state. The overall objective of an agent is to maximize the accumulated sum of rewards.

A policy, $\pi : s \to a$, is a mapping from states to actions, fully characterizing the behavior of an agent. The Q-value of a state–action pair, $q_\pi(s, a)$, is the expected return for following $\pi$ from $s$ after taking action $a$, $\mathbb{E}_\pi \left[ \sum_{k=1}^{\infty} \gamma^k R(s_{t+k}) \mid s_t = s, a_t = a \right]$, where $\gamma$ is the discount rate. The value of a state, $v_\pi(s)$, is the expected return by following $\pi$ from $s$, $q_\pi(s, \pi(s))$. The optimal policy $\pi^*$ is the policy $\pi$ that maximizes $v_\pi(s), \forall s \in S$, which is equivalent to maximizing $q_\pi(s, a) \forall s, a \in S, A$.

A widely used class of methods for specifying policies in RL is to construct an approximation of the value function represented as a function from a state–action pair to a value, $\hat{q}(s, a)$, and then select the action that maximizes $\hat{q}(s, a)$ at each timestep [67]. Deep Q-networks (DQNs) are one such method, using multi-layer artificial neural networks as a nonlinear function approximation for $\hat{q}(s, a)$

[45]. Another widely used class of method for specifying polices is to set a parameterized function, $\pi_\theta(s,a)$, that can be used to choose an action without looking up a value function. The only criteria for the parameterized function is that it needs to be differentiable with respect to the parameters in order to be trained with the policy gradient algorithm [68]. If the action space is continuous, the function can directly output actions and for the discrete action spaces the parameterized numerical preferences, $h_\theta(s,a) \in \mathbb{R}$, can be learned and used to give the actions with the highest preferences the highest probabilities by utilizing the exponential soft-max distribution, $\pi(s,a) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$.

Deep RL algorithms generally require a differentiable state-value parameterization function, $\hat{v}(s,\theta_v)$ and a differentiable policy parameterization function, $\pi(s,a,\theta_\pi)$, is additionally required for policy gradient algorithms [67]. The common testbed for the following algorithms is the Arcade Learning Environment (ALE) [5] which has boasting 55 ATARI games. Two games were later added and a task learning all 57 games in one network structure is called, Atari-57. The initial DQN algorithms used prioritized experience replay, double Q learning, and dueling networks [59, 70, 73]. Later algorithms use distributional value function citebellemare2017distributional, noisy DQN, and n-step bootstrapping. The combination of these algorithm is known to show much higher performances [21]. The distributed version of these algorithms such as Ape-X, Gorilla, and R2D2[24, 49] show much improved performances than the single threaded counterparts. Compared with the above value-based algorithms, the policy gradient algorithms such as A3C [46], TRPO [62], ACER [74], ACKTR [78], PPO [63], IMPALA [11], and SEED [12] use a structure called actor critic which maintains a value function approximation to be consulted to learn the policy function. The actor critic method all use distributed actors to play multiple games simultaneously whether or no it is single or multi threaded. Since policy-based methods can handle continuous action spaces, These set of algorithms are also tested on the environments where actions can be given as a set of vector with real numbers [7]. Model-based deep RL algorithms are soon followed after the seminal DQN work [45]. Solving Atari by learning a full model of a world still remain inferior to the model-free based method mentioned above [17, 26]. The predictron did learn to predict values without actions [65]. The model still takes a full MDP, but its representation can be an internal hidden layer values of a deep network. TreeQN [13] and value interation network [69] learns an abstract and local MDP model respectively and take tree-structure and value interation into the deep neural networks. Value prediction networks learn an MDP grounded on real actions while its representations are not the original inputs [52]. MuZero [61] is a successor to the value prediction networks algorithm in that it learns an MDP with real actions and also learns reward and values functions together. It has score the highest in the Atari-57 task.

## 2.2 Explainable Artificial Intelligence (XAI)

As our artificial intelligence systems are more widely used in our society, the interests in explaining these systems grew rapidly especially in the areas of decision making systems and supervised learning systems [15, 43, 48, 44, 38, 56, 10, 22, 58]. There is yet no general consensus in what *explainable AI* is and researchers are interchangeably using two terms, *explainability* and *interpretability*. In this paper, *explainability* will be regarded equally as *interpretability*.

### 2.2.1 Interpretability and Explainability

Lipton [38] provides the desiderata of the *interpretable* machine learning (ML): trust, causality, transferability, informativeness, and fair and ethical decision making. Lipton [38] also distinguishes two properties of the *interpretable ML*: *transparency* and *post-hoc explanations*.

*Transparency* in this context refers to asking a question how does a model work? A model here can be your choice of machine learning models such as deep neural networks, decision trees, and linear regression. This requires the knowledge of how a specific choice of machine learning model work. *Transparency* can be achieved by looking a model at the level of an entire model (simulatability), at the level of subcomponents (decomposability), or at the level of an algorithm (algorithmic transparency).

*The post-hoc explanations* are given to users by extracting information from learned models while a given model is assumed as a blackbox. Such explanations include natural language explanations, visualization of learned representations, and explanations by example. Miller [43] adopts *explanations* are *post-hoc interpretability*. Miller [43] and Biran and Cotton [6] also makes a distinction between *explanations* and *justifications*. *Justifications* can explain why a given model is making good choices, but cannot explain individual decisions.

In most of cases, deep neural networks are considered to be less transparent due to its choice of network parameters in ad-hoc and heuritic manners despite the fact that all of its numerical calculation processes are fully uncovered [56]. There are numerous work in building alternative models sitting besides the original model to explain deep neural networks in a more interpretable way. Such examples include a linear proxy model called LIME [55] and a decision tree method called DeepRED [81]. Visualizing learned features in deep neural networks at different levels of their hierarchies has already been examined by many researchers in the field. Researchers have seen signs of textures, shape patterns, and parts of objects [50]. Kim et al. [29]'s work is an early attempt to systematize the process of finding these features through concept activation vectors (CAV) and translating into meaning explanations. In terms of generating images from text inputs using a Generative Adversarial Network (GAN), Hong et al. [23] used a semantically hierarchical network structure instead of a end-to-end network for the network's transparency. To better explain deep neural networks decision making, pixel level visualization using heatmaps, sometimes called saliency maps, and semantic segmentation have also been used extensively [56].

There is much less work in explaining a RL agent. Wang et al. [73], Greydanus et al. [16], and Weitkamp et al. [75] developed methods to display saliency maps which is claimed to show the focus of attention in a learned deep neural network. Anderson et al. [1] combined saliency map with the reward decomposition method and conducted a user study to verify human users' understanding of the given information by checking a user's ability to predict the next outcomes. Mnih et al. [45] visualized the distribution of the internal representation values with respect to the distances between input vectors using the t-SNE method.

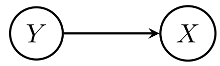


Figure 2.1: A causal diagram showing $Y$ causes $X$.

### 2.2.2 Causality

Before jumping into explaining explanations, I will first examine what causality is in order to provide the basis for the latter discussion of explanations.

Based on Hume's regularity theory [25], if one type of events always occurs before the other, there is a causal relationship between two types of events. However, like the fact that a rooster crows before sunrise does not indicate a rooster is the cause of a sunrise, a mere association between two events is not sufficient to claim that one event is the cause of the other event. To overcome this situation, Pearl and Mackenzie [54] suggests that causal reasoning should be performed on at least three levels: association, intervention, counterfactuals. Pearl named a structure containing these three levels the Ladder of Causation.

The first ladder is the *association* ladder. This level of reasoning can be understood by *seeing* and *observing*. Most of the current statistics and supervised learning techniques fall into this category. The reasoning at this level of the ladder can answer questions like how are two variables are related? and How would seeing an event X can change my belief in event Y?

The second ladder is the *intervention* ladder. Activities relating to this level of reasoning is *doing* and *intervening*. The causal reasoning at this level can answer questions like what is the difference in the expected outcome when I do A instead of B? What action is required to make Y happen? Reinforcement learning and learning causal Bayesian models fall into this category according to Bareinboim [3].

The third and highest ladder is the *counterfactuals* ladder. *Imagining*, *retrospection*, and *understanding* activities fall into this category. What if and why questions are in fact counterfactual questions. Counterfactual cases are the hypothetical events that does not cause an event which is to be explained [43]. I will later disambiguate this with contrastive explanations.

Halpern and Pearl [18] formally defines an *actual cause* of an event $X = x$ as a set of event $E$ (an in dividual variable is expressed as a form of $Y = y$) if the following criteria holds.

1. In a real situation, both $X$ and $E$ have to be true.

2. If $E$ had some *counterfactual* value, then the event $X = x$ would not have been true.

3. $E$ has to be minimal. This means that $E$ does not contain any irrelevant variable.

Causal relationships can be represented as a graph, each node represents variables and a directed edge from node $Y$ pointing to node $X$ means $Y$ causes $X$ as in Figure 2.1.

In Rubin [57]'s definition, a potential outcome of a variable $Y$ when $X$ is assigned with value $x$ is $Y_{X=x}$. A probability of a potential outcome of $Y$ holding a specific value $y$ when $X$ is holding value $x$ is defined as $P(Y_{X=x} = y)$. Using this notation, both the probability of necessity and sufficiency can be defined [54]. When the goal is to prove the causation between $X$ and $Y$, both the probabilities of necessity ($PN$) and sufficiency ($PS$) are defined as follows:

- Probability of Necessity (PN): $P(Y_{X=0} = 0|X = 1, Y = 1)$.
- Probability of Sufficiency (PS): $P(Y_{X=1} = 1|X = 0, Y = 0)$.

When there are more than one causes for a single outcome, comparison of these two values for two different causes can give us insights on which cause is more important than the other. The probability of necessity tells us that when the cause is not present, how likely the expected event do not occur given the cause led to the desired outcome in the real situation. The probability of sufficiency tells us that when the cause is present, how likely the outcome is expected to occur given that the lack of cause led to no desired outcome. Pearl and Mackenzie [54] states that this mechanism can play an important role when determining the most probable cause in autonomous systems.

Lastly, Pearl and Mackenzie [54] introduces *do*-operator to properly acknowledge intervention in probabilisties. While $P(X|Y)$ only captures mere association between two events, $P(X \mid do(Y))$ represents an intervention on $Y$ to influence $X$ and also inhibition of all other effects directing to $Y$.

### 2.2.3 Explanations

Pearl and Mackenzie [54] claims that the causation plays an essential role in human explanation. Every time when we encounter our world we always ask questions such as why events happen in particular ways, why objects have certain properties, and why people behave in such a way [39]. Lewis [35] defines explanation as to provide information about an event's causal history.

According to Gilpin et al. [15], a good explanation can be dependent on the question and pays particular interests to two types of why-questions: why and why-should. He also claims that a good explanation in general come from a good inference, but it can also be the use of abductive reasoning: finding all possible causes of an effect and finding the best one. The extensive use of Pearl and Mackenzie [54]'s Ladder of Causation can be turned into answering questions like what happened, how it happened, and why it happened at each respective level.

Lombrozo [39] notes that explanation is a product and a process at the same time. Gilpin et al. [15] argue that there are two processes and a product in explaining: a cognitive process, a product, and a social process. If explanation is a product, then explaining has to go thorugh both a cognitive and social processes. In this thesis I focus on the cognitive process and the product first in order to generate explanations and then for the future work seek a possibility to consider explanations as a tool for social communication. In terms of generating explanations directly from a policy of a learned agent, Hayes and Shah [20] developed a systematic algorithm to generate explanations for agents' decisions.

# Chapter 3

# Difficulty in Explaining a Deep RL Agent

*Portions of this chapter have appeared in the earlier paper, "Measuring and Characterizing Generalization in Deep Reinforcement Learning" [77] with Sam Witty, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen.*

## 3.1 The role of generalization in explaining

As discussed in the background chapter, the main key in explaining a learned agent is within its ability to reason in counterfactual situations and their outcomes. In this work, I and my colleagues have developed a method to perturb input states in a semantically meaningful way to create counterfactual situations. Although the true intention of this setup is to see how an agent behaves in counterfactual situations in order to explain the rationale behind its actions, the performance of an agent after perturbation is far too inferior to that of non-perturbed situations. I claim that this phenomena is mainly due to the lack of generalization in current Deep RL training method.

Deep RL methods have achieved remarkable performance on challenging control tasks. Observations of the resulting behavior give the impression that the agent has constructed a generalized representation (a semantic representation) that supports insightful action decisions. We re-examine what is meant by generalization in RL, and propose several definitions based on an agent's performance in on-policy, off-policy, and unreachable states. We propose a set of practical methods for evaluating agents with these definitions of generalization. We demonstrate these techniques on a common benchmark task for deep RL, and we show that the learned networks make poor decisions for states that differ only slightly from on-policy states, even though those states are not selected adversarially. Taken together, these results call into question the extent to which deep Q-networks learn generalized representations, and suggest that more experimentation and analysis is necessary before claims of representation learning can be supported.

**Prior Work on Generalization in RL.** Generalization has long been a concern in RL [66]. Somewhat more recently, Kakade [27] provided a theoretical framework for bounding the amount of training data needed for a discrete state and action RL agent to achieve near optimal reward. Nouri et al. [51], Zhang et al. [80] discuss how to apply the idea of a training/testing split from supervised learning in the context of offline policy evaluation with batch data in RL. Generalization has been cast as avoiding overfitting to a particular training environment, implying that sampling from diverse environments is necessary for generalization [76, 80]. Other work has focused on generalization as improved performance in off-policy states, a framework much closer to standard approaches in supervised learning. Techniques such as adding stochasticity to the policy [19], having the agent take random steps, no-ops, steps from human play [49], or probabilistically repeating the agent's previous action [41], all force the agent to transition to off-policy states.

These existing methods diversify the training data via exposure to on-policy and off-policy states, but none discuss generalization over states that are logically plausible but unreachable. The prior focus has been on generalization as a method for preventing overfitting, rather than as a capability of a trained agent.

**Generalization vs. Memorization.** Generalization is often contrasted with memorization and there have been recent efforts to understand their respective roles in deep learning. For instance, with an operationalized view of memorization as the behavior of deep networks trained on noise, Arpit et al. [2] showed that the same architectures that memorize noise can learn generalized behaviour on real data. By contrast, we assess generalization via unreachable states, which differs from this operationalized view of memorization. For instance, Zhang et al. [79] empirically demonstrated the capacity of deep networks to memorize an entire dataset and fit random data, questioning their generalization ability. Extending their work, with an operationalized view of memorization as being the behavior of deep networks trained on noise, Arpit et al. [2] showed that deep networks do not just memorize but the optimization process detects patterns and is content-aware. Most recently, Cohen et al. [8] empirically showed for the first time that these two concepts are in fact complementary. In contrast, we focus on generalization via interpolation and extrapolation, which differs from this operationalized view of memorization.

**Adversarial Attacks on Deep Networks.** While related to adversarial attacks on deep networks, this work differs in two important ways: (1) interventions are not adversarially selected and, (2) interventions operate on latent states, not on the agent's perception. Mandlekar et al. [42] attempted to make agents robust to random high-level perturbations on the input. That is, for the domain they explore, MuJoCo physics simulator, the inputs are at the resolution of human-understandable concepts. Yet, this work does not address questions of alignment between meaningful real world high-level perturbations and learned representations by the network.

## 3.2 Recasting generalization

Using existing notions of generalization, such as held-out set performance, is complicated when applied to RL for two reasons: (1) training data is dependent on the agent's policy; and (2) the vastness of the state space in real-world applications means it is likely for novel states to be encountered at deployment time.

One could imagine a procedure in RL that directly mimics evaluation on held-out samples by omitting some subset of training data from any learning steps. However, this methodology only evaluates the ability of a model to *use* data after it is collected, and ignores the effect of exploration on generalization. Using this definition, we could incorrectly claim that an agent has learned a general policy, even if this policy performs well on a very small subset of states. Instead, we focus on a definition that encapsulates the trained agent as a standalone entity, agnostic to the specific data it encountered during training.

**Generalization via State-Space Partitioning.** We partition the universe of possible input states to a trained agent into two sets, according to how the agent can encounter them following its learned policy $\pi$ from $s_0 \in S_0$. Here, $\Pi$ is the set of all policy functions, and $\alpha$, $\delta$, and $\beta$ are some small positive values close to 0. We can think of $\delta$ and $\beta$ as thresholds on estimation accuracy and optimality performance. The set of reachable states, $S_{\text{reachable}}$, is the set of states that an agent encounters with probability greater than $\alpha$ by following any $\pi' \in \Pi$.[1]

**Definition 1** (Interpolation)**.** An RL agent has high interpolation performance, $G_I$, if $\delta > |\hat{q}(s,a) - q_\pi(s,a)|$ and $\beta > q^*(s,a) - q_\pi(s,a)$, $\forall s \in S_{\text{off}}, a \in A$. The set of off-policy states, $S_{\text{off}}$, is defined as $S_{\text{reachable}} \setminus S_{\text{on}}$.

**Definition 2** (Extrapolation)**.** An RL agent has high extrapolation performance, $G_E$, if $\delta > |\hat{q}(s,a) - q_\pi(s,a)|$ and $\beta > q^*(s,a) - q_\pi(s,a)$, $\forall s \in S_{\text{unreachable}}, a \in A$. The set of unreachable states, $S_{\text{unreachable}}$, is defined as $S \setminus S_{\text{reachable}}$.

Note that $S$ only includes states that are in the domain of $T(s,a,s')$. In other words, specification of the transition function implicitly defines $S$, and by extension $S_{\text{unreachable}}$. This definition is particularly important in the context of deep RL, as the dimensionality of the observable input space is typically much larger than $|S|$. If we wish to demonstrate that an agent generalizes well for Amidar, $T(s,a,s')$ would need to be well defined with respect to latent state variables in the Amidar game, such as player and enemy position. If we wish to demonstrate that an agent generalizes well for all Atari games, we would need $T(s,a,s')$ to be well defined with respect to latent state variables in other Atari games as well, such as the paddle position in Breakout. Given any reasonable bound on the MDP, we would not expect the agent to perform well when exposed to random configurations of pixels.[2]

[1] These definitions can be customized with alternative metrics for value estimation and optimality, such as replacing $|\hat{v}(s) - v_\pi(s)|$ with $(\hat{v}(s) - v_\pi(s))^2$.

[2] Modifications to the transition function itself are better described as transfer learning [53].

Note that a large body of work implicitly uses $G_R$ as a criteria for performance, even though this is the weakest of generalization capabilities. It is what you get when testing a learned policy in the environment in which it was trained. Some readers may doubt that it is possible to learn policies that extrapolate well. However, Kansky et al. [28] show that, with an appropriate representation, reinforcement learning can produce policies that extrapolate well under similar conditions to what we describe in this paper. What has not been shown to date is that deep RL agents can learn policies that generalize well from pixel-level input.

We demonstrate a simple example of this state-space partition in Figure 3.1, a classic GridWorld benchmark. In this environment, the agent begins each episode in a deterministic start position, can take actions *right*, *right and up*, and *right and down*, and obtains a reward of $+1$ when it arrives at the goal state, $s_g$. Note that the agent must move right at every step, therefore there are three regions that are unreachable from the agent's fixed start position: the upper left corner, the lower left corner, and the lower left corner after the wall. While unreachable, the upper left corner is a valid state that does not restrict the agent's ability to reach the goal state and obtain a large reward.

Note that an agent interacting in the GridWorld environment learns tabular Q-values, therefore we should not expect it to satisfy any reasonable definition of generalization. However, given an adequate exploration strategy, an agent could conceivably visit every off-policy state during training, resulting in $\hat{v}(s)$ converging to $v^*(s), \forall s \in S_{\text{reachable}}$. This agent would satisfy $G_R$ and $G_I$ for arbitrarily small values of $\delta$ and $\beta$. Despite this positive outcome, most observers would not say that this agent "generalizes", because it lacks any function-approximation method. Only the definition $G_E$ is consistent with this conclusion.

With the emergence of RL-as-a-service[3] and concerns over propriety RL technology, evaluators may not have access to an agent's training episodes, even if they have access to the training environments. In this context, the distinction between $G_I$ and $G_E$ is particularly important when measuring an agent's generalization performance, as off-policy states may have unknowingly been visited during training.

**Quantifying Generalization Error.** Generalization in Q-value-based RL can be encapsulated by two measurements for off-policy and unreachable states, one that accounts for the condition $\delta > |\hat{q}(s,a) - q_\pi(s,a)|$—whether the agent's estimate is close to the actual Q-value after executing $\pi$—and another for the condition $\gamma > q^*(s,a) - q_\pi(s,a)$—whether the actual Q-value is close to the optimal Q-value. In our work, we use value estimate error, $\text{VEE}_\pi(s) = \hat{v}(s) - v_\pi(s)$, and total accumulated reward, $\text{TAR}_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} R(s_{t+k}) \mid s_t = s, a_t = a\right]$, respectively.

In most situations, $q^*(s,a)$ is not known explicitly; however, $\text{TAR}_\pi(s)$ can be used to evaluate the *relative* generalization ability between two agents, as the optimal value for a given state is fixed by definition.

Unlike $\text{TAR}_\pi(s)$, which, when measured in isolation can depend on the inherent difficulty of $s$, $\text{VEE}_\pi(s)$ has the advantage of consistency. For example, if an agent is placed in a state such that

[3]e.g., https://portal.ds.microsoft.com

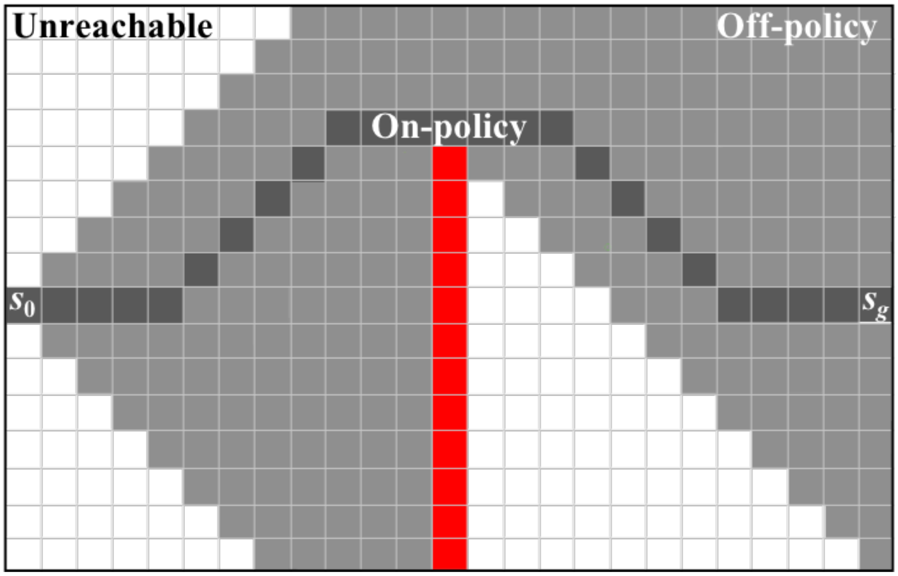



Figure 3.1: Examples of on-policy, off-policy, and unreachable states in GridWorld.

$v^*(s) = 0$, $\text{TAR}_\pi(s)$ alone does not capture the model's ability to generalize. $\text{VEE}_\pi(s)$ may, however, if $\hat{v}(s) \approx 0$. We address this limitation of $\text{TAR}_\pi(s)$ in our experiment by training benchmark (BM) agents on each of the evaluation conditions.

## 3.3 Methodology

In this section, we describe specific techniques for producing off-policy states and a general methodology for producing unreachable states based on parameterized simulators and controlled experiments.

### 3.3.1 Off-Policy States

It is helpful to think of off-policy states as the set of states that a particular agent *could* encounter, but *doesn't* when executing its policy from $s_0$. Framed in this way, the task of generating off-policy states in practice is equivalent to finding agents with policies that differ from the policy of the agent under inspection. We present three distinct categories of alternative policies for producing off-policy states, which we believe to encapsulate a broad set of historical methods for measuring generalization in RL.[4]

**Stochasticity.** One method for producing off-policy states is to introduce stochasticity into the policy of the agent under inspection [41]. We present a representative method we call *k off-policy actions* (k-OPA), which causes the agent to execute some sequence of on-policy actions and then take $k$ random actions to place the agent in an off-policy state. This method is scalable to large and

[4]We encourage readers to think critically about whether their strategy for generating off-policy states does in fact differ from the agent's policy, as this deviation may be difficult to measure.

complex environments, but careful consideration must be made to avoid overlap between states, as well as to ensure that the episode does not terminate before $k$ actions are completed. It is easy to imagine other variations, where the $k$ actions are not selected randomly but according to some other mechanism inconsistent with greedy-action selection.

**Human Agents.** The use of human agents has become a standard method in evaluating the generalization capabilities of RL agents. The most common method is known as human starts (HS) and is defined as exposing the agent to a state recorded by a human user interacting with an interface to the MDP environment [45]. One could easily imagine desirable variations on human starts within this general category, such as passing control back and forth between an agent and a human user. Human agents differ from other alternative agents in that they may not be motivated by the explicit reward function specified in the MDP, instead focusing on novelty or entertainment.

**Synthetic Agents.** Synthetic agents are commonly used during training in multiagent scenarios, although to our knowledge have not been used previously to evaluate an agent's generalization ability. We present a representative method we call *agent swaps* (AS), where the agent is exposed to a state midway through an alternative agent's trajectory. This method has the potential to be significantly more scalable than human starts in large and complex environments, but attention must be paid to avoiding overlap between the alternative agents and the agent under inspection. This method may also be useful in applications not amenable to a user interface or otherwise challenging to gather human data.

### 3.3.2 Unreachable States

Unreachable states are unlike off-policy states, which can be produced using carefully selected alternative agents. By definition, unreachable states require some modification to the training environment. We propose a methodology that is particularly well suited for applications of deep RL, where agents often only have access to low-level *observable effects*, rather than what we would typically describe as a semantically meaningful or high-level representation. In the case of AMIDAR and other Atari games, for example, the position of individual entities can be described as latent state and the rendered pixels are their observable effects.

**Intervening on Latent State.** We present two distinct classes of interventions on latent state: existential, adding or removing entities, and parameterized, varying the value of an input parameter for an entity. The particular design of intervention categories and magnitude should be based on expected sources of variation in the deployment environment, and will likely need to be customized for individual benchmarks.

To facilitate this kind of intervention on latent state, we implemented INTERVENIDAR, an AMIDAR simulator. INTERVENIDAR closely mimics the Atari 2600 AMIDAR's behavior,[5] while allowing users to modify board configurations, sprite positions, enemy movement behavior, and other features of

[5] Readers familiar with AMIDAR will know that there are other features of gameplay not listed here; although INTERVENIDAR reproduces them, they are not important to the training regimens, nor the overall results of this paper.

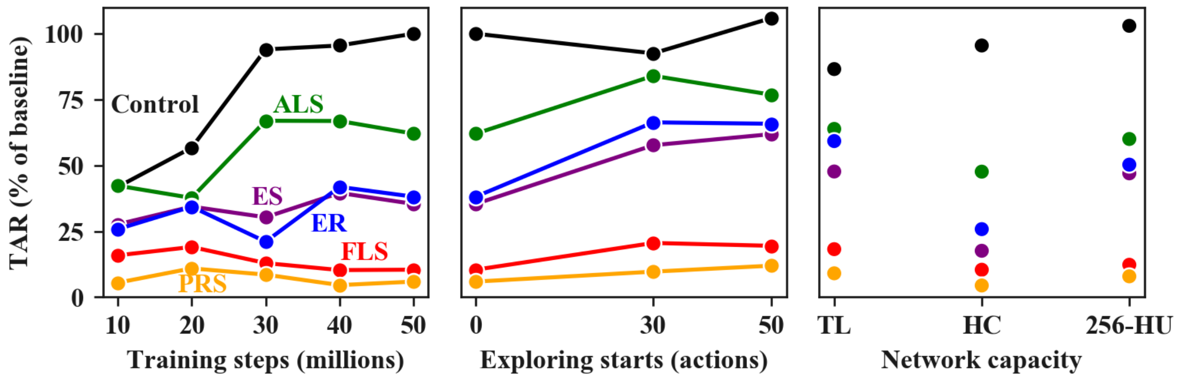



Figure 3.2: Average total accumulated reward (TAR) from various unreachable states for each of the trained agents. The benchmark agents trained using ALS, ES, ER, FLS, and PRS configurations respectively achieved average TARs of 94, 74, 14, 77, and 90 percent of the baseline TAR.

gameplay without modifying Intervenidar source code. Some manipulable features that we use in our experiments are:

*Enemy existence and movement.* The five enemies in Amidar move at a constant speed along a fixed track. By default, Intervenidar also has five enemies whose movement behavior is a time-based lookup table that mimics enemy position and speed in Amidar. Other distinct enemy movement behaviors include following the perimeter and the alternative movement protocols. These enemy behaviors are implemented as functions of the enemy's local board configuration and are used for our transfer learning experiments.

*Line segment existence and predicates.* A line segment is any piece of track that intersects with another piece of track at both endpoints. Line segments may be filled or unfilled; the player's objective is to fill all of them. In Intervenidar, users may specify which of the 88 line segments are filled at any timestep. Furthermore, Intervenidar allows users to customize the quantity and position of line segments.

*Player/enemy positions.* Player and enemy entities always begin a game in the same start positions during Amidar, but they may be moved to arbitrary locations at any point in Intervenidar.

We included these features in the experiments because they encapsulate what we believe to be the fundamental components of Amidar gameplay, avoiding death and navigating the board to accumulate reward. The scale of these interventions were selected to reflect a small change from the original environment, and are detailed in the case-study section.

**Control.** In addition to producing unreachable states, parameterizable simulators enable fine control of experiments, informing researchers and practitioners about *where* agents fail to generalize, not simply that they fail macroscopically. One limitation of using exclusively off-policy states is that multiple components of latent state may be confounded, making it challenging to disentangle the *causes* of brittleness from other differences between on-policy and off-policy states. Controlled experiments avoid this problem of confounding by modifying only a single component of latent state.

| | C | 10-OPA | | 20-OPA | | AS | | HS | | ALS | | ER | | ES | | FLS | | PRS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAR | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE | TAR | VEE |
| BL-50 | 1.00 | 1.00 | -0.01 | 0.94 | -0.03 | 0.15 | -0.31 | 0.09 | -0.67 | 0.62 | -0.01 | 0.38 | -0.08 | 0.35 | -0.08 | 0.10 | -0.49 | 0.06 | -1.43 |
| BL-40 | 0.96 | 0.91 | -0.01 | 0.89 | -0.03 | **0.15** | **-0.29** | 0.09 | -0.71 | **0.67** | **-0.01** | **0.42** | **-0.07** | **0.40** | **-0.08** | 0.10 | -0.49 | 0.05 | -1.75 |
| BL-30 | 0.94 | 0.85 | -0.02 | 0.55 | -0.08 | **0.15** | -0.32 | **0.11** | **-0.59** | **0.67** | **0.00** | 0.21 | -0.16 | 0.30 | -0.11 | **0.13** | **-0.36** | **0.09** | **-0.89** |
| BL-20 | 0.57 | 0.17 | -0.27 | 0.15 | -0.40 | **0.17** | **-0.29** | **0.11** | **-0.58** | 0.38 | -0.07 | 0.34 | **-0.05** | 0.34 | **-0.06** | **0.19** | **-0.18** | **0.11** | **-0.54** |
| BL-10 | 0.42 | 0.22 | -0.05 | 0.17 | -0.18 | 0.12 | **-0.27** | 0.06 | -0.93 | 0.42 | **0.00** | 0.26 | **-0.05** | 0.28 | **-0.05** | **0.16** | **-0.18** | 0.05 | **-1.03** |
| 256HU | **1.03** | **1.10** | **-0.01** | **0.97** | -0.04 | **0.31** | **-0.19** | **0.10** | **-0.51** | 0.60 | **-0.01** | **0.50** | **-0.03** | **0.47** | **-0.03** | **0.12** | **-0.27** | **0.08** | **-0.79** |
| HC | 0.96 | 0.98 | -0.01 | 0.41 | -0.13 | **0.24** | **-0.15** | 0.08 | **-0.60** | 0.48 | -0.02 | 0.26 | -0.08 | 0.18 | -0.14 | 0.10 | **-0.34** | 0.04 | **-1.39** |
| TL | 0.87 | 0.74 | -0.03 | 0.60 | -0.08 | **0.18** | **-0.29** | **0.09** | -1.00 | **0.64** | **-0.01** | **0.59** | **-0.03** | **0.48** | **-0.05** | **0.18** | **-0.32** | **0.09** | **-0.98** |
| 30RA | 0.93 | 0.96 | **-0.01** | **0.95** | **-0.03** | 0.15 | **-0.28** | **0.14** | **-0.35** | **0.84** | **-0.01** | **0.66** | **-0.03** | **0.58** | **-0.03** | **0.21** | **-0.21** | **0.10** | **-0.76** |
| 50RA | **1.06** | **1.13** | **-0.01** | **1.12** | **-0.02** | **0.24** | **-0.13** | **0.13** | **-0.49** | **0.77** | **-0.01** | **0.66** | **-0.02** | **0.62** | **-0.02** | **0.19** | **-0.28** | **0.12** | **-0.63** |
| BM | | | | | | | | | | 0.95 | 0.00 | 0.14 | -0.01 | 0.75 | 0.01 | 0.77 | 0.00 | 0.90 | 0.00 |

Table 3.1: TAR and VEE for all of the trained agents and experimental configurations. Alternative agents that perform better than the baseline state-of-the-art agent are shown in **bold**. TAR values are normalized by the TAR of the control. VEE values are normalized by their respective TARs. Since all VEE values for control (C) are zero or very near zero, we omit this column.

## 3.4 Case Study: Amidar

We trained a suite of agents and evaluated them on a series of on-policy, off-policy, and unreachable INTERVENIDAR states. Using our proposed partitioning of states and empirical methodology, we ran a series of experiments on these agents' ability to generalize. In this section, we discuss how we generated off-policy and unreachable states for the AMIDAR problem domain.

We used the standard AMIDAR MDP specification for state: a three-dimensional tensor composed of greyscale pixel values for the current, and three previous, frames during gameplay [45]. There are five movement actions. The transition function is deterministic, and entirely encapsulated by the AMIDAR game. The reward function is the difference between succesive scores, and is truncated such that positive differences in score result in a reward of 1. There are no negative rewards, and state transitions with no change in score result in a reward of 0.

We trained all agents using the state-of-the-art dueling network architecture, double Q-loss function, and prioritized experience replay [71, 73, 60]. All of the training sessions in this paper used the same hyperparameters as in Mnih et al.'s work and we use the OpenAI's baselines implementation [9].

**AMIDAR Agents.** We explored three types of modifications on network architecture and training regimens in an attempt to produce more generalized agents: (1) increasing dataset size by increasing training time; (2) broadening the support of the training data by increasing exploration at the start of each episode; and (3) reducing model capacity by decreasing network size and number of layers. To establish performance benchmarks for unreachable states, we trained an agent on each of the experimental extrapolation configurations.

*Training Time.* To understand the effect of training-set size on generalization performance, we saved checkpoints of the parameters for the baseline DQN after 10, 20, 30, and 40 million training actions before the model's training reward converged at approximately 50 million actions. This process differs from increasing training dataset size in prediction tasks in that increasing the number of training episodes simulataneously changes the distribution of states in the agent's experience replay.

*Exploring Starts.* To increase the diversity of the agent's experience, we trained agents with 30 and 50 random actions at the beginning of each training episode before returning to the agent's standard $\epsilon$-greedy exploration strategy.

*Model Capacity.* To reduce the capacity of the Q-value function, we explored three architectural variations from the state-of-the-art dueling architecture: (1) reducing the size of the fully connected layers by half (256-HU), (2) reducing the number of channels in each of the three convolutional filters by half respectively (HC), and (3) removing the last convolutional layer of the network (TL). Recent work on deep networks for computer vision suggest that deeper architectures produce more heirarchical representations, enabling a higher degree of generalization [30].

**Off-policy States.** We employed three strategies to generate off-policy states for an agent: human starts, agent swaps, and $k$-OPA. None of these methods require the INTERVENIDAR system. In each case, we ran an agent nine times, for $n$ steps, where $n \in \{100, 200, \ldots, 900\}$.

*Human starts.* Four individuals played 30 INTERVENIDAR games each. We randomly selected 75 action sequences lasting more than 1000 steps and extracted 9 states, taken at each of the $n$ time steps [49].

*Agent swaps.* We designated five of the trained agents as *alternative agents*: (1) the baseline agent, (2) the agent that starts with 50 random actions, (3) the agent with half of the convolutional channels as the original architecture, (4) the agent with only two convolutional layers, and (5) the agent with 256 hidden units. We chose these agents with the belief that their policies would be sufficienctly different from each other to provide some variation in off-policy states.[6]

*$k$-OPA.* Unlike the previous two cases where states came from sources external to the agent, in this case we had every agent play the game for $n$ steps before taking $k$ random actions, where $k$ was set to 10 and 20.

**Unreachable States.** With INTERVENIDAR, we generated unreachable states, guaranteeing that the agent begins an episode in a state it has never encountered during training. All modifications to the board happen before gameplay.

*Modifications to enemies.* We make one existential and one parameterized modification to enemies: We randomly remove between one and four enemies from the board (ER), and we shift one randomly selected enemy by $n$ steps along its path, where $n$ is drawn randomly between 1 and 20 (ES).

*Modifications to line segments.* We make one existential and one parameterized modification to line segments: We add one new vertical line segment to a random location on the board (ALS) and we randomly fill between one and four non-adjacent unfilled line segments (FLS).

*Modification to player start position.* We start the player in a randomly chosen unoccupied tile location that has at least one tile of buffer between the player and any enemies (PRS).

**Transfer Learning: Assessing Representations.** We conducted a series of transfer learning experiments [53], freezing the convolutional layers and retraining the fully connected layers for 25 million steps. We use these results to understand how learned representations in the convolutional layers relates to overall generalization performance. We train each of the agents using the alternative enemy movement protocol so that enemies move on the basis of local track features, rather than using a lookup table. If an agent has learned useful representations in the convolutional layers, then we expect that agent to learn a new policy using those representations for the alternative movement protocol.[7]

## 3.5 Results

Our experiments demonstrate that: (1) the state-of-the-art DQN has poor generalization performance for AMIDAR gameplay; (2) distance in the network's learned representation is strongly anti-correlated

[6] When evaluating any of the alternative agents, we only used states from the remaining four to generate off-policy states.

[7] We distinguish this transfer learning experiment from our extrapolation experiments in that the transfer learning experiment modifies the transition function $T(s, a, s')$ and by extension $q^*(s, a)$. In the extrapolation experiments, an agent can later encounter states it has observed during training and effectively use its learned policy, which is not necessarily true if the transition function changed.

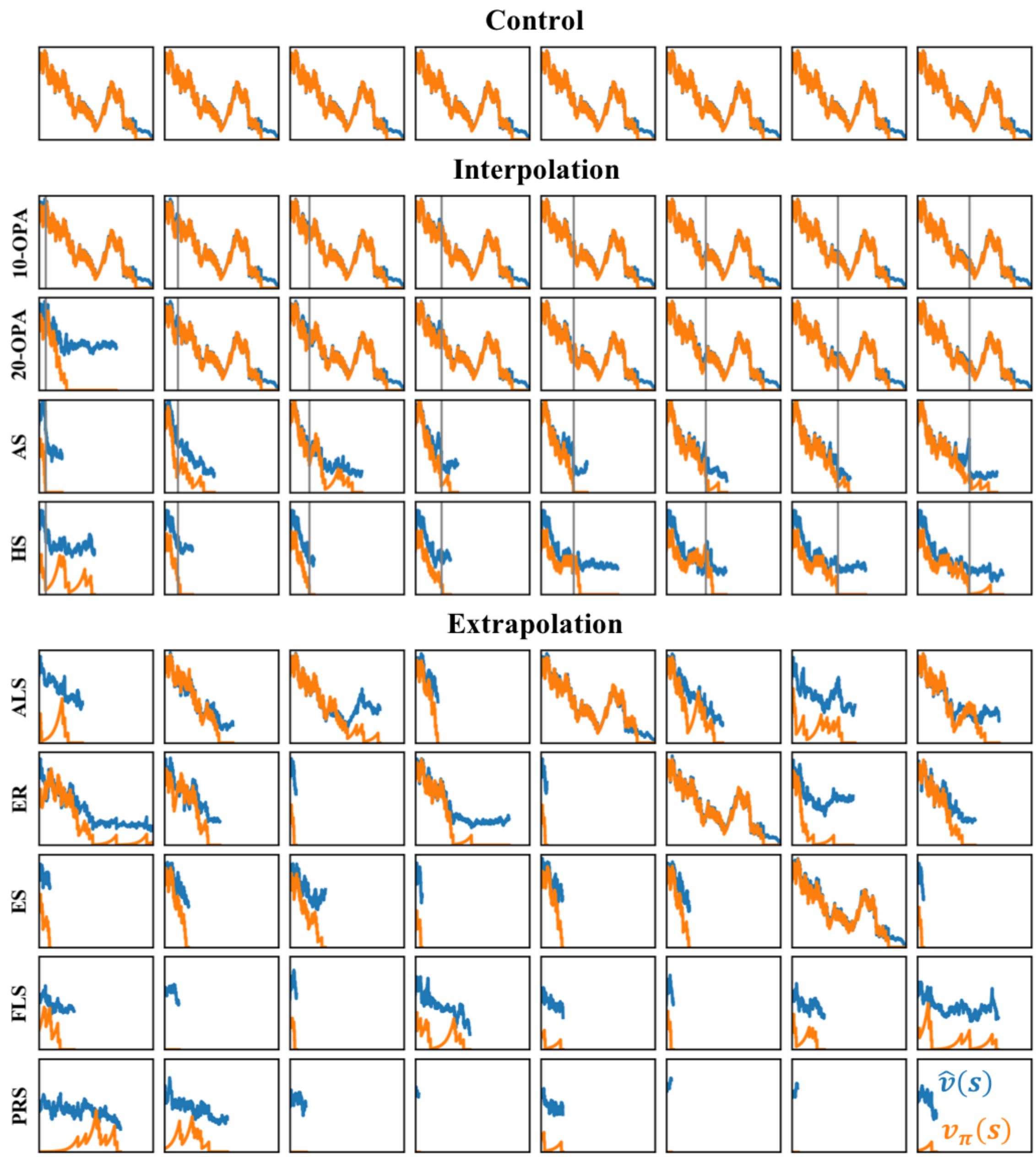



Figure 3.3: $\hat{v}(s)$ and $v_\pi(s)$ for replicated trajectories for all experiments. Each subplot is a single independent trial. For the interpolation experiments, the vertical grey line shows the point where the agent takes random actions (in the k-OPA experiments) or regains control (in the agent swaps and human-starts experiments). The length of each episode is consistently lower and the difference between $\hat{v}(s)$ and $v_\pi(s)$ is consistently higher for the extrapolation experiments.

with generalization performance; (3) modifications to training volume, model capacity, and exploration have minor and sometimes counterintuitive effects on generalization performance; and

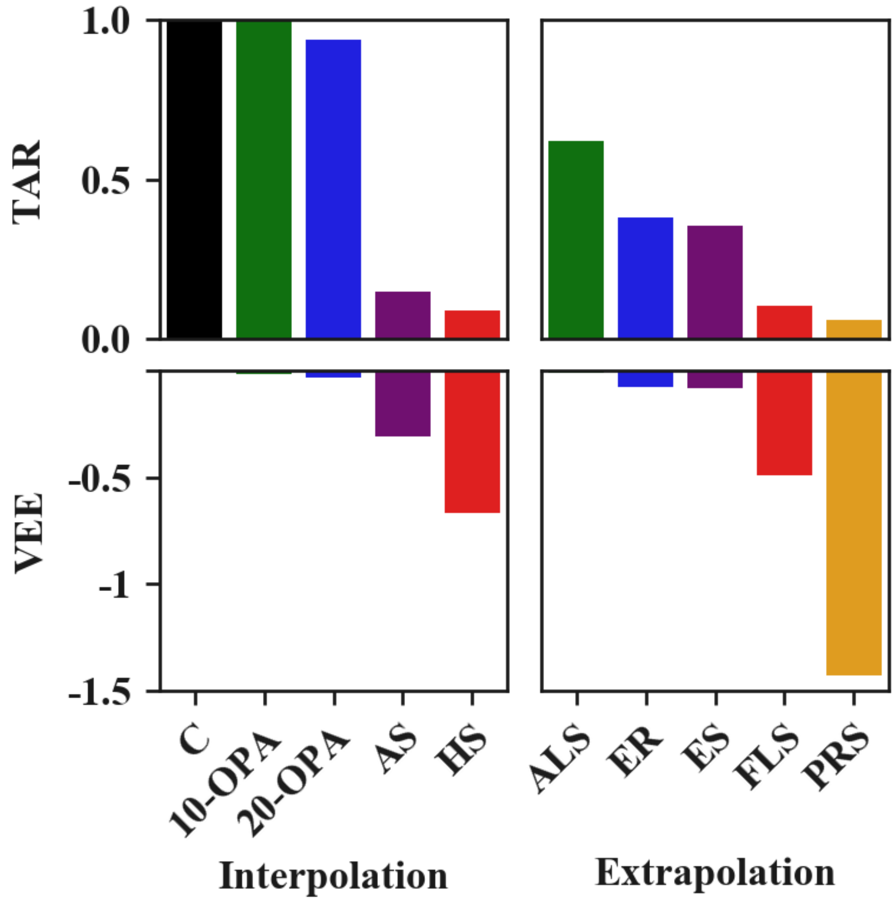



Figure 3.4: TAR and average VEE for control, extrapolation, and interpolation experiments. The agent consistently overestimates the state value. TAR and VEE are strongly anti-correlated. All TAR bars are normalized by the TAR of the control condition. All VEE bars are normalized by their respective TAR.

(4) generalization performance does not necessarily correlate with an agent's ability to transfer representations to a new environment.

**Poor Generalization Performance.** Figures 3.3 and 3.4 show that the fully trained state-of-the-art DQN dueling architecture produces a policy that is exceptionally brittle to small non-adversarial changes in the environment. The most egregious examples can be seen in Figure 3.4, in the filling line segments (FLS) and player random starts (PRS) interventions. Visual inspection of the action sequences proceeeding these states showed the agent predominantly remaining stationary, often terminating the epsisode without traversing a single line segment. This behavior can be seen in Figure 3.3, where PRS and FLS episodes terminate prematurely. Videos displaying this behaviour can be found in the supplementary materials.

Furthermore, Figure 3.4 shows that VEE and TAR are very highly anti-correlated across the

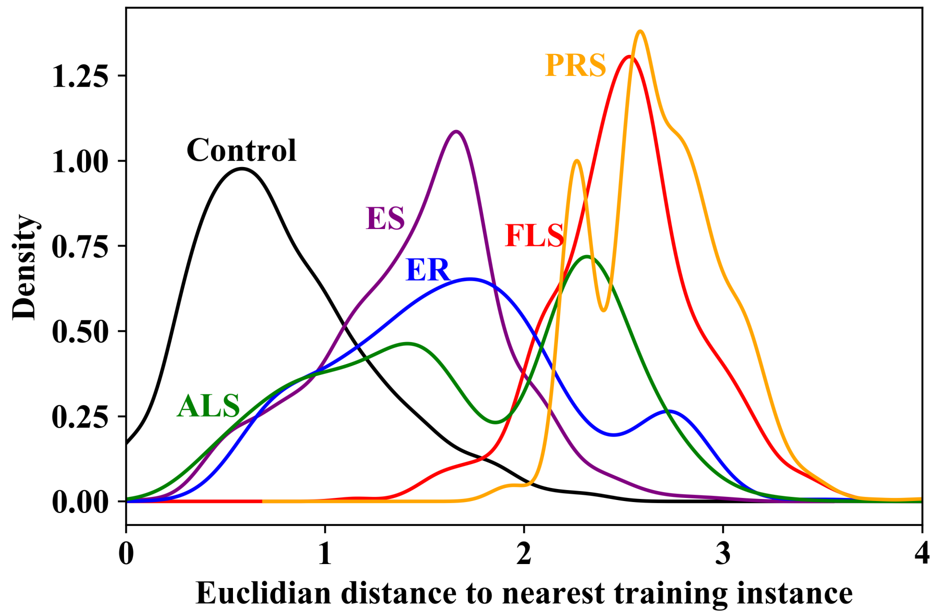



Figure 3.5: Smoothed empirical distributions of the distances between the test points of the extrapolation experiments and the training data. Generalization performance is anti-correlated with distance from previously seen states.

experiments, indicating that the agent's ability to select appropriate actions is related to its ability to correctly measure the value of a particular state. We observe that the model always overestimates the value of off-policy and unreachable states. In contrast, the agent's value estimates are small and approximately symetrically distributed around 0 in the control condition.

**Distance in Representation.** By extracting the activations of the last layer of the DQN, we are able to observe the distance between training and evaluation states with respect to the network's learned representation. Figure 3.5 depicts the density estimates for the distribution of these distances. We find that the agent does not "recognize" the unreachable states where generalization is the worst, such as PRS and FLS, implying that the learned representation is inconsistent with these components of latent state. Alternatively, one could imagine a network that performs poorly by conflating states that are meaningfully different. Using the activations of the last layers, both the relative distances between each state's internal representation and its Q-valuecan be depicted in a two dimensional graph using t-SNE [40] as in Figure 3.6. The perturbed states, such as FLS, ER, ALS, stayed close together in terms of their internal representations. However, each state's temporal correlation seems to play a more important role in combining each state's internal representation.

**Training Agents for Generalization.** We take inspiration from well-established methods in supervised learning; increasing training set size, broadening the support of the training distribution, and reducing model capacity. We propose the following analogs to each of these methods, respectively; increasing the number of training episodes, introducing additional exploration, and removing layers and nodes.

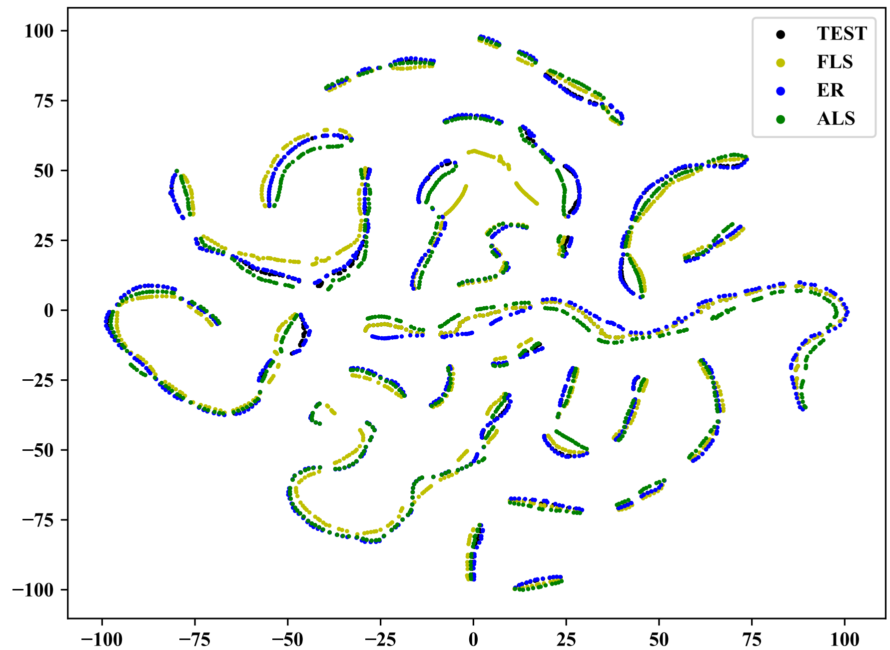


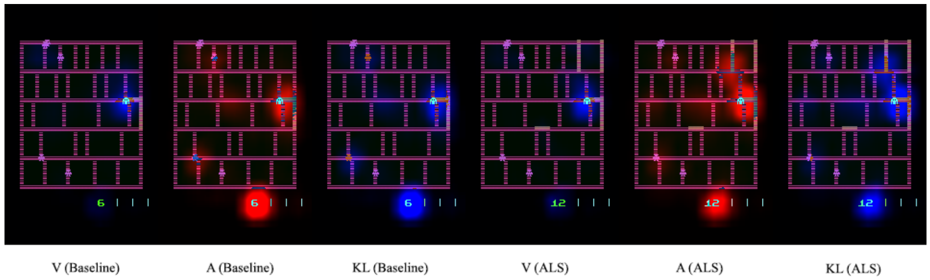

Figure 3.6: t-SNE plot of a single run. t-SNE plot is showing the relationship between the relative distances of each input state's internal representation and each state's Q-value. The data is collected from a single test run of the game. Although perturbed states stay close together in terms of their internal representations, each state's temporal correlation seems to play a more important role in combining each state's internal representation.

These experiments indicate that: (1) naïvely increasing the number of training episodes until training set performance converges reduces generalization; (2) some reductions to model capacity induce improvements to generalization; and (3) increasing exploration and otherwise diversifying training experience results in more generalized policies. These results are shown in figure 3.2.

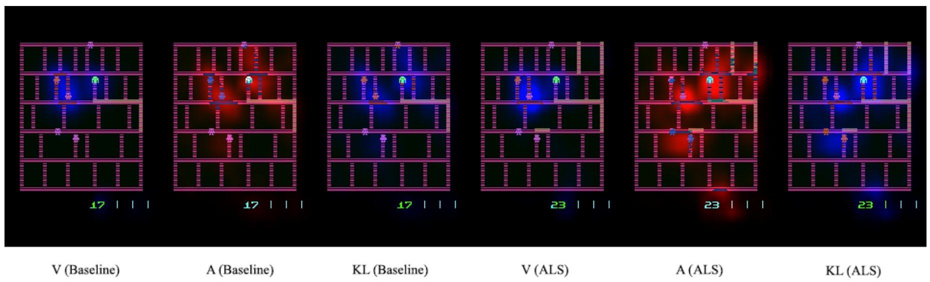
*Training Episodes.* While increasing training time clearly increases the total accumulated reward in the control condition, shorter training times appear to contribute to increased generalization ability. This increase is minimal, but it does illustrate that naïvely increasing training time until converge of training rewards may not be the best strategy for producing generalized agents.

*Model Capacity.* Of the reductions to model capacity, we find that shrinking the size of the fully-connected layers results in the greatest increase in generalization performance across perturbations. Reducing the number of convolutional layers also results in improvements in generalization performance, particularly for the enemy perturbation experiments.

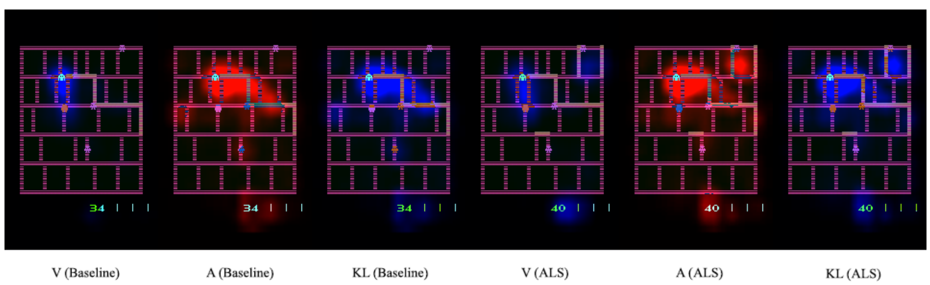
*Exploration Starts.* We find that increasing the diversity of training experience has the greatest effect on generalization performance, particularly for the agent with 50 random actions. This agent experiences almost a twofold increase in total accumulated reward for human starts and all of the extrapolation experiments. This agent outperforms the baseline agent in every condition. Of particular interest is the agent's performance on the enemy shift experiments, where the agents' total




(a) At frame 30.




(b) At frame 86.




(c) At frame 127.

Figure 3.7: Saliency Maps of AMIDAR. The value function (V), advantage function (A) and KL divergence (KL) values are depicted in heatmaps. Baseline and ALS states are compared.

accumulated reward approaches the reward achieved by an agent trained entirely in that scenario.

**Hierarchical Representations and Generalization.** While the agents with increased exploration demonstrate a clear improvement in generalization ability over baseline, it is not consistent with their ability to accumulate large reward with the alternative enemy-movement protocol after retraining. This finding contradicts those of work on representations in computer vision, where transferability of representations directly corresponds to generalization ability.

**Saliency Maps.** Using Greydanus et al. [16]'s method, the value function (V), advantage function (A) and KL divergence (KL) values are depicted in heatmaps. Baseline and ALS states are compared in Figure 3.7.

# Chapter 4

# Proposed Work: Causal RL Agent

Following the researchers who adopted model-based approaches to interpretable ML, I will propose building a RL framework which adopts causal reasoning from its basis. There are two components in this approach: a direct adaptation of a structural causal model into the state and reward mechanism and an adaptation of the Ladder of Causation into both its learning and testing processes.

## 4.1 Causal Multi-armed Bandit

Multi-armed Bandit (MAB) problems have been researchers' first choices in understanding the learning mechanism of evaluative feedback in sequential decision making systems in their simplest form [67]. Recently, Bareinboim et al. brought the concept of causal reasoning into MAB problems [4, 33, 34]. Causal understanding of a given problem helps solve problems more efficiently and possibly give powers to understand the solutions better [33, 54]. In this proposal, we pose a problem of finding the right structural causal model (SCM) in a MAB setting and propose a KWIK(know what it knows) [36] algorithm as a solution.

## 4.2 Structural Causal Bandit Problem

The structural causal bandit(SCB) problem is defined by placing an agent in an environment where at each time step an agent pulls one of the $K$ available arms and receives a reward as in Figure **??**. The goal is to maximize the accumulated reward after $T$ rounds.

The performance of a proposed algorithm is measured using a cumulative regret $Reg_T = \max_i \sum_{t=1}^{T} \mathbb{E}[Y_{A_i}] - \mathbb{E}[Y_{A_t}] = \max_i \sum_{t=1}^{T} \mu_i - \mathbb{E}[\mu_{A_t}] = \sum_{i=1}^{K} \Delta_i \mathbb{E}[T_i(T)]$ where $T_i(T)$ means the number of arm $i$ played after $T$ rounds and $A_t$ means the arm player at time $t$, and $\Delta_i = \max_j \mu_j - \mu_i$.

We define a SCM $M$ as a tuple $\langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}) \rangle$ where $\mathbf{U}$ is a set of unobserved confounders called exogenous variables, $\mathbf{V}$ is a set of observed variables called endogenous variables, and $\mathbf{F}$ is a set of deterministic functions which sets variables $V_i \in \mathbf{V}$ based on both endogenous and exogenous

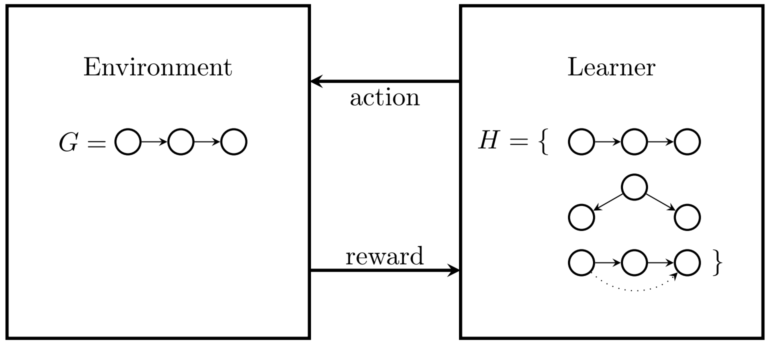



Figure 4.1: Environment and agent settings in SCB.

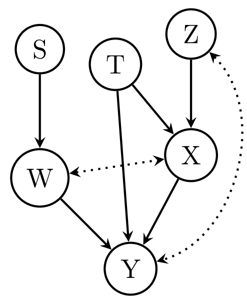



Figure 4.2: An example of a structural Causal model (SCM)

variables. Such a relation in each function can be formulated as a graph structure $G = \langle \mathbf{V}, \mathbf{E} \rangle$ where there are two different kinds of edges. A directed edge $V_i \to V_j \in \mathbb{E}$ means $V_i$ is a parent of $V_j$ and $V_j$'s value is determined by $V_i$. If there is a bidirectional edge between two exogenous variables $V_i$ and $V_j$ then they share an unobserved confounder $\mathbf{U}_i \cap \mathbf{U}_j \neq \emptyset$. A sample causal structure is depicted in Figure 4.2. $P$ provides probability distributions for unobserved confounders $\mathbf{U}$.

In order to pose a structural causal model as a bandit problem, we define an endogenous variable $Y$ as a real valued reward and assume all other endogenous $\mathbf{V} \setminus Y$ variables are manipulable and consider a combination of its value as an individual arm choice.

While an environment only holds a target SCM $M$, in KWIK setting, a learner agent can hold as many SCMs as possible in its hypothesis space $\mathbf{H}$.

## 4.3 Algorithm

Although the final goal is to build a KWIK algorithm for find a correct causal model of the environment, I first propose a naïve KWIK algorithm in solving SCB problem which is completely oblivious about the underlying causal model. Therefore, it runs two-step KWIK bandit algorithm first on the set of possible intervenable variables and second on its values. An an example, the SCM in Figure 4.2, all the possible intervenable sets are as follows:

**Algorithm 1** Naïve KWIK Algorithm

**function** NaïveKWIK($\mathbf{V}$)
  **for** $t = 1, 2, ...$ **do**
    $\mathbf{I} \leftarrow$ Intervenable($\mathbf{V}$)
    **for** $\mathbf{I}_i \in \mathbf{I}$ **do**
      $\mathbf{X}_i \leftarrow$ KWIK($\mathbf{I}_i$)
    **end for**
    **if** $\perp \notin \mathbf{X}$ **then**
      $i \leftarrow \max_i \mathbf{X}_i$
    **else**
      $i \leftarrow$ randomly choose from $\forall \mathbf{X}_i,\ \ \mathbf{X}_i = \perp$
    **end if**
    $\mathbf{W}_i \leftarrow$ Values($\mathbf{I}_i$)
    $\mathbf{R}_i \leftarrow$ KWIK($\mathbf{W}_i$)
    **if** $\perp \notin \mathbf{R}_i$ **then**
      $j \leftarrow \max_j \mathbf{R}_{i,j}$
      **return** $\mathbf{W}_{i,j}$
    **else**
      $j \leftarrow$ randomly choose from $\forall \mathbf{W}_{i,j},\ \ \mathbf{R}_{i,j} = \perp$
    **end if**
  **end for**
**end function**

$$\{\{X\}, \{Z\}, \{W\}, \{X, Z\}, \{X, W\}, \{Z, W\}, \{X, Z, W\}\}$$

.

Since the two-step KWIK bandit algorithm is no different than making it one-step KWIK bandit algorithm by cross-producting two different arm choice steps, given the number of intervenable sets as $n$ and the maximum value variation as $m$, the KWIK bound is $B(\epsilon, \delta) = O(\frac{nm}{\epsilon^2} \ln \frac{nm}{\delta})$.

The proposed naïve KWIK algorithm is listed in Algorithm 1.

# Chapter 5

# Timeline

1. **2/15** Proposal
2. ∼**5/23** (NeurIPS 2020) Causal Bandit Work on learning causal structures in Causal MAB setting
3. ∼**9/1** (AAAI 2021) Explainable Causal RL - Application to RL (full MDP) setting
4. **9/1**∼**12/1** Writing the Thesis
5. **12/1** Defense

# Bibliography


[1] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. Explaining reinforcement learning to mere mortals: An empirical study. *arXiv preprint arXiv:1903.09708*, 2019.

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

[3] Elias Bareinboim. Causal reinforcement learning. URL `https://www.youtube.com/watch?v=bwz3NpVfz6k`.

[4] Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. In *Advances in Neural Information Processing Systems*, pages 1342–1350, 2015.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[6] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, page 1, 2017.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[8] Gilad Cohen, Raja Giryes, and Guillermo Sapiro. Dnn or $k$-nn: That is the generalize vs. memorize question. *arXiv preprint arXiv:1805.06822*, 2018.

[9] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

[10] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[11] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[12] Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. *arXiv preprint arXiv:1910.06591*, 2019.

[13] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*, 2017.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *IET Conference Proceedings*, pages 850–855(5), January 1999.

[15] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[16] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*, 2017.

[17] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[18] Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 56(4):843–887, 2005.

[19] Matthew J Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. In *AAAI Workshop: Learning for General Competency in Video Games*, 2015.

[20] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE, 2017.

[21] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainabilty of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1312, 2019.

[23] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Interpretable text-to-image synthesis with hierarchical semantic layout generation. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 77–95. Springer, 2019.

[24] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

[25] David Hume. An enquiry concerning human understanding. In *Seven masterpieces of philosophy*. Routledge, 2016.

[26] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[27] Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[28] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*, 2017.

[29] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[31] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[33] Sanghack Lee and Elias Bareinboim. Structural causal bandits: Where to intervene? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2568–2578. Curran Associates, Inc., 2018.

[34] Sanghack Lee and Elias Bareinboim. Structural causal bandits with non-manipulable variables. Technical report, Technical Report R-40, Purdue AI Lab, Department of Computer Science, Purdue University, 2019.

[35] David K Lewis. Causal explanation. *Philosophical Papers 2*, pages 214–240, 1986.

[36] Lihong Li, Michael L Littman, Thomas J Walsh, and Alexander L Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.

[37] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2):39–55, 2008.

[38] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[39] Tania Lombrozo. The structure and function of explanations. *Trends in cognitive sciences*, 10 (10):464–470, 2006.

[40] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[41] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv preprint arXiv:1709.06009*, 2017.

[42] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3932–3939. IEEE, 2017.

[43] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[44] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 279–288. ACM, 2019.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[46] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[47] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[48] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[49] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[50] Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 55–76. Springer, 2019.

[51] Ali Nouri, Michael L Littman, Lihong Li, Ronald Parr, Christopher Painter-Wakefield, and Gavin Taylor. A novel benchmark methodology and data repository for real-life reinforcement learning. In *Proceedings of the 26th international conference on machine learning*, 2009.

[52] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6118–6128, 2017.

[53] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[54] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[56] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *arXiv preprint arXiv:1905.08883*, 2019.

[57] Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

[58] Wojciech Samek. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

[59] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[60] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

[61] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[62] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[63] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[64] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[65] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3191–3199. JMLR. org, 2017.

[66] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[67] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[68] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[69] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[70] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[71] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[72] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[73] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[74] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[75] Laurens Weitkamp, Elise van der Pol, and Zeynep Akata. Visual rationalizations in deep reinforcement learning for atari games. In *Benelux Conference on Artificial Intelligence*, pages 151–165. Springer, 2018.

[76] Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 120–127. IEEE, 2011.

[77] Sam Witty, Jun Ki Lee, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

[78] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[79] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[80] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[81] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016.