Identifying, addressing, and understanding challenging cases in Machine Learning

by

Cinjon Resnick

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
May, 2022

Professor Kyunghyun Cho

Professor Joan Bruna

Science in the Service of Society

Acknowledgements

The end of every mile is marked by a numbered stone. Sometimes it's explicit, like on the highway. Usually, it is blank and filled in by your imagination. Regardless, a litany of people helped you reach that place on a frequently misshapen journey. With an almost half-decade of mile markers in my rear view mirror, it is here that I acknowledge more than just the Cheshire cats and mad hatters one meets along the way, but also the inspirations that shaped me before following the white rabbit.

It starts at birth, so a special thanks to my mom and dad for instilling characteristics and life events that allowed me to persevere to the end of journeys. Similarly to my brothers, who each have generously given their support in their own way; Cade, who showed me that I could get into an arena with anyone. Kallan - I'm still astonished by that dinner we had my sophomore year of college and how prescient it has remained. I will forever be grateful for all you did behind the scenes growing up and the regular check-ins that you persistently engage. Kai, with whom every great adventure has started and ended - my appreciation knows no bounds.

Professors Cho and Bruna, you two gave me an experience in grad school that is strikingly different from the one that people told me about, and better for me in just about every way. Was it because you understood that from the get go? Was it because you learned it along the way? Was it because that is who you are? I am not sure and it doesn't matter; the point is that you gave me what I needed to make this work and mature into a scientist. For that, I am eternally grateful and fully intend to pay it forward. On an individual level, Prof. Cho, you gave me a chance when

I just had a simple idea about autoencoders and translation and you supported that all the way to Google Brain and back again. I have regaled your research prowess to many, but what's most surprising is how you excel as a manager, a compliment that I understand is common to all of your advisees and a trait that I think you had from birth. Prof. Bruna, you actually convinced me to leave industry and join CILVR, which was not an easy feat and is as good an indicator of any of your ability to connect. I appreciate immensely not just your research guidance over the years, but also your personal touch and care.

Avi + Dima, you two silly shits who always make me laugh and with whom chaos feels permissible. I treasure that so much. Adam B, the conversation we had in China about me finishing the PhD still stands out, but then again so does most every conversation I have with you. Dino, you inspired me years ago with your ability to engender trust and today with your ability to see the core simple answer buried behind the complexity. Acrefoot, it meant a lot to me when you said you were proud that I completed this. Thanks for always being there, even for calls about mom.

Ken, you're the dependable rock of support that I've turned to so many times these past years. Alisa, you are the best training partner I could ever want. Angela, you're the most charming ray of sunshine in New York. Without you three, I would have left this city a long time back.

Martin, from starting a research haus in Brooklyn to starting a workshop in Spanish-speaking countries, we've done some cool shit and that counts for so much in my heart. I'm proud of that. But I'm also proud of you. Getting to watch your journey has been a privilege - thanks for letting me in. Will, I married you. Man, that was an honor. Thanks for being a rock for all those years, and thanks for indulging the absurdist in me - that's special af.

Adam, Jesse, Mohammad, Doug, and Andrew, I dedicated my first paper post GB to y'all and I'm dedicating this thesis to you as well. I've still never had as much fun working on anything than I did with you all on research. Thanks for being amazing mentors.

Hugo, in another life I would have been very happy working with you in Montreal. I sincerely

appreciate the ways you given me your time over the years and how much you seem to appreciate my antics.

Or, I think it's magical that we did work together! What a fortuitous connection that's been! One of these days, I'm going to show up to your haus with a bag of cashews. Amlan, wow it was fantastic getting to know you. You work so hard and you care so much. Keep being you and don't ever forget to look me up in NYC.

Roberta Raileanu and Ilia Kulikov, you were both phenomenal collaborators and made that first year so much more than it would have been otherwise. Thanks for taking a risk with me.

Victor Fomine and Sergei Volodin, you are almost surely never going to see this, but you should know that you have had an exponentially positive effect on my life through your coaching. It's akin to a love language for me and you both fill that role tremendously well. Please don't retire thanks.

Office Hours crew, while we never did enact the original motivation from Portugal, I've certainly gotten many hours worth of office from y'all. Much obliged.

Gleb Kuznetsov, you said something so off-hand years ago about me finishing the PhD, and I was a little annoyed every time I remembered it. However, I think I did finish it slightly to spite that comment.

Ambert Yeung + Sho Nakamori, thank you for gifting a year of this PhD experience the promise of huge potential and a return to what I was seeking before I started this journey.

Brian Victor + Debbie McWhinney, your support and trust has been invigorating. Thanks for letting me in to your circle. Phyllis + Dennis, thanks for making that possible (and so much more).

Kait, Jakob, Amy Z, thank you so much for being absurd with me. Jess, the warmth of your laser is always welcome. Ruchi + Aditya, one of these days I'm going to live up to the support you've had for me.

Abstract

Machine learning has advanced tremendously this past decade. Object detection systems routinely perform beyond human-level accuracy with no loss in speed [210], game-playing agents play at superhuman level in real time [177; 196; 178], and generative models write language useful enough for downstream products [26]. And yet, autonomous vehicles (AV) crash due to surprising mistakes, the best gaming agents lose to simple strategies, and our language models produce nonsensical utterances at a surprisingly high rate. I could have chosen examples from any field because these failures are not endemic to just vision, games, or language. There are always challenging cases remaining after training our system, and these cases are where the systems fail.

This thesis focuses on the challenging cases in a machine learning system in order to improve its overall capabilities. In the first part, we study methods for identifying the challenging cases, an important precursor for improving the system. In the second part, we then study methods for addressing the challenging cases, arguably the most important part of this thesis for real-world applicability. And in the third part, we study methods for understanding the root cause of challenging cases, an important step in attaining guarantees about our system's capabilities. As machine learning is practiced in many different settings, our study does too. We explore these questions in the context of computer vision, language learning, and task learning. The connecting thread among them is the drive towards creating a communicative and visually aware robot that can capably complete household tasks. In that context, we present in parallel the Machine Learning Application Framework that highlights where our contributions improve downstream

applications.

All together, this work studies how to identify, address, and understand the most challenging cases over a diverse array of machine learning systems. This research is imperative towards deploying many systems that we care about, including most autonomous vehicles and health assistants. Consequently, it represents an important step towards society's technological goals.

Contents

List of Figures

List of Tables

1 Introduction

It has been a long standing goal in society to have household robots that can safely and delightfully handle real world tasks. Rosie from the Jetsons, Irona from Richie Rich, and Robby from Hazel were all robot maids depicted in mass media from the early 1960s. With little need for human-guided teaching, they could learn and perform the expected servile tasks like folding laundry, gracefully dusting the shelves, and even cooking dinner. They did this while intelligently receiving instructions, replying coherently in conversational form, and never had any demonstrated issues with new objects in the scene or unusual patterns due to light or orientation. Either the fiction writers of the time had a disproportionate expectation of our technological pace or a diminutive ability to project the difficulties inherent in building such a complex agent.

Thiry years later, some of the top robotics minds established the iRobot company and debuted the residential floor cleaning robot ‘Roomba’ a decade later. Twenty years hence, there have been nine versions and 20 million units sold, however they are still incapable of doing much more than cleaning floors in an unobstructed space. They do not receive instruction, they cannot tell us when there is a failure or what is the failure, and they sweep up objects that should not be swept. It was only the most recent version in September 2021 that provided faculties for avoiding dog excrement. They still commonly trip over power cords and confuse rugs with black stripes for cliffs[1]. In the factory setting, there have been spectacular robots built by companies like Kuka. Their cost is justified in that they can complete challenging tasks with very high precision and

[1]https://twitter.com/DimaKrotov/status/1436150021792149506

for a long time without tiring. However, they can only perform rote mechanical tasks and must be adapted to each factory setting.

Little of the vision dreamed of more than half a century ago has come to fruition. We do not expect our robots to converse coherently; We expect them to have trouble generalizing their visual understanding of scenes; And while they can learn new tasks, doing so requires dense signal in order to reduce the sample complexity, an unattainable condition in most settings. These issues are not just a matter of reality being a more challenging environment than simulation [165; 86]; even there do we have these difficulties [87; 102; 170].

We see these challenges expressed clearly by top modern-day roboticists in the primacy given to the rather banal challenge of rearrangement. The Rearrangement Challenge [13] asks agents to bring a physical environment into a specified goal state and was presented recently as the next grand direction for Embodied AI. It is pragmatic and overarching, encompassing diverse settings like cleaning the bedroom, loading the dishwasher, picking and placing orders in a fulfillment center, and rearranging the furniture. It also is a generalization of many other scenarios: the Amazon Robotics Challenge [76] asks agents to move items from a shelf into a bag and RoboCup@Home [184] asks agents to perform tasks like opening and shutting a refrigerator door, both of which lay under the umbrella goal of rearrangement. That we cannot reliably solve even simple versions of this challenge demonstrates how limited we are in our research understanding and consequently our engineering capabilities.

1.1 The Most Challenging Cases

Enjoining together all of the tasks embedded in that early vision and trying to build a communicative and capably rearranging robot is a worthwhile (albeit Herculean) goal. However, we can also consider each of them in isolation as individual research questions. The common result when doing so is similar to the parable of the blind men and the elephant [207]:

A group of blind men heard that a strange animal had been brought to the town, but none were aware of its shape and form. Out of curiosity, they said: "We must inspect and know it by touch, of which we are capable". The first person, whose hand landed on the trunk, said, "This being is like a thick snake". For another whose hand reached its ear, it seemed like a fan. As for another, whose hand was upon its leg, the elephant is a pillar like a tree-trunk. The blind man who placed his hand upon its side said the elephant, "is a wall". Another who felt its tail described it as a rope. The last felt its tusk, stating the elephant is that which is hard, smooth and like a spear.

In other words, every discipline becomes ingrained in its own pursuit and the drawn borders make it difficult to see what is common amongst them. When we take a step back, we see that these individual research questions all share a similar phenomenon - a major aspect preventing their progress is the prominence of the most challenging cases. While I define this concept rigorously in Section 2.2, colloquially they are the data who hold membership in a semantically-defined group that is, on average, much more difficult for the model than data selected at random from commonly encountered examples. Additionally, group membership is defined according to semantics that humans care about, like where the can opener is positioned relative to the can rather than, say, the number of blue pixels in the image. We accordingly refer to the cases by the decision criteria of their most challenging group. The following enumeration describes how this applies to each isolated research goal.

- **Learning new sequential tasks** is typically a problem in the Reinforcement Learning domain. Performing this capability in a real world setting requires the agent to understand what behavior to change after a long chain of steps. In industry parlance, the problem is that there are long range dependencies in sparse reward episodic environments stemming from the agent's starting state being distant from the goal state. An example is the entire chain of events that leads to correctly locating a can opener lying on the counter and placing it into the proper draw.

The agent typically receives high reward in states close to the goal state. Reward diminishes as we move farther away and, coupled with the sparse signal, the learning agent's ability to correct their behavior diminishes. Consequently, the error rate increases inordinately, and we commonly see groups of states where the error patterns are similar. Returning to the can opener task, one challenging group might be the set of states encountered when locating the tool while next to the fridge; another may be those involved with placing it in the right drawer. Improving the ability of agents to address their hard cases (at the group level) in sparse environments would help the field make headway on a wide class of problems in sparse exploration.

- **Visual scene understanding** is usually under Supervised Learning. To reliably understand scenes, an agent would need to gauge where its model is weak. By identifying such cases, it can learn about them and their interactions before calamities occur. In vision, these cases are often related to the rarity of their joint class. Examples abound where the model has seen each aspect of a case independently but not in concert, for example motorcycles at night lit just so by the moon's reflection off of a puddle. The model may have low error detecting motorcycles, low error detecting vehicles in night weather, and similarly low error when there are puddles involved. However, the model may have difficulty when all three are expressed together, regardless of the model family. One reason why is because of biases expressed in the photography hardware, famously so for race [119; 176]. Another reason is due to machine learning training biases [187]. Identifying these challenging visual cases quickly, capably, and online would help us more confidently place agents in the field because they would have the faculties to gauge when they are in an uncertain regime.

- **Conversing coherently with humans** requires that the robot understands language to such a degree that the latency between utterances is small, the words expressed are meaningful in context, and that its communication expresses systematic generalization[2]. Our best modern example of talking robots are assistants like the Amazon Alexa, who have trouble with the first and third aspects.

We focus on systematic generalization, or compositionality, because the most challenging cases of conversation are frequently related to this attribute. As illuminated in Roller et al. [163], compositionality is at the core of the ability to transfer to new tasks, which is a fundamental open problem in machine learning today. Without this capability, our agents are limited in how quickly they can adapt to new conversational topics or even amend their prior built-in notions of language. An example of a challenging case are text with specific unknown words strewn amongst known words, e.g. if the model first encounters 'cyan' in a discussion about the ocean. This would have high error unless the model understood the word in context of the rest of the sentence. Agents trained to exhibit compositionality in their language understanding would see the new meaning as arising from a composition of concepts in context.

Understanding the root causes of compositionality in language and why they arise will help us build languages that exhibit this property. That will then help us address the challenging cases related to poor systematic generalization in language.

Of course, the issue of the challenging cases being inhibitory extends beyond the robot application we described. We know, from the No Free Lunch Theorem [175], that we cannot find one algorithm that will solve these issues for us; corner cases, tail events, and transfer learning is inherently hard and will always exist no matter how much compute we use. In light of these limitations, what we are asking in this thesis are the following questions:

- How can we advance our ability to identify semantic groups of challenging cases?

[2]Defined as the capacity to understand and produce a potentially infinite number of novel combinations from known components [131; 42]

- How can we better understand the underlying reasons for the existence of the challenging cases?
- How can we best address broad swaths of challenging cases without having the experience be akin to whack-a-mole?

In concert – how we can mitigate the problem of the challenging cases so that we can directly attain the benefits in downstream applications? Progress occurs when we figure out how to mitigate their effect. While the No Free Lunch Theorem implies that we must ask each of these questions in context of a singular application, the techniques that we develop would ideally be applicable across applications. With that in mind, advancing research that identifies, addresses, or understands the challenging cases will have a broad effect on the field of machine learning, both in theory and application.

1.2 Overview

Following Section 1.1, this thesis presents research work on the most challenging cases in machine learning. The work tries to either identify such cases, address them, or understand them on a deeper level. As this concern spans the field, so do our works. They range from task learning (with reinforcement learning) to language learning and vision.

I present this work through two parallel concepts. The first uses the machine learning application framework to both contextualize each work's results in practice, as well as unify them on the same view. The second shows how the works are, under the banner of engaging the challenging cases, all advancing a single important engineering goal - the aforementioned intent to create a communicative and visually aware robot that can learn a new rearrangement task. The thesis is organized as follows:

- Chapter 2 introduces background understanding for our settings and places the challenging cases in the context of the machine learning application framework. Towards this, we introduce rigorous notation for episodic reinforcement learning, multi-agent learning, and supervised learning, as well as properly defining the most challenging cases.

- In Part I, we seek to identify challenging cases in machine learning. Chapter 4 presents our work [161] focusing on object detection in vision that pseudo-automatically identifies challenging semantic groups of data for robotic systems, with an emphasis on autonomous vehicles. This is related to the robot directive in that it is integral that we are aware of a robot's visual shortcomings in advance of deploying them.

- In Part II, we study how to address the most challenging cases. Chapter 5 presents our work, an algorithm [162] for task learning via episodic reinforcement learning. It was originally a direct response to the challenging cases found in training agents for the Pommerman environment [158; 159], a benchmark designed for multi-agent research but also applicable to single-agent studies.

- Part III focuses on understanding the root cause of challenging cases, with an emphasis on language learning in multi-agent settings. Chapter 6 presents our work [160] illustrating the central role of agent capacity in determining whether the learned language is actually compositional, an important and desired attribute of the learned language that is correlated with the most challenging cases.

- Finally, Chapter 7 concludes by showing how we can synthesize the presented contributions as a new research direction in order to make effective progress on the rearrangement task.

2 The challenging cases in the context of the machine learning application framework

2.1 Notation

Work presented in this thesis spans across many disciplines within machine learning. In this section, we introduce notation to provide suitable background for each.

Reinforcement Learning commonly assumes that the environment is a Markov Decision Process (MDP) and learning occurs via episodic MDPs. An MDP consist of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ representing the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the discount factor $\gamma \in (0, 1]$, and the transition probability function $\mathcal{P}$. The domain of $\mathcal{P}$ is $\mathcal{S} \times \mathcal{A}$ and the range is a distribution on $\mathcal{S}$. The markovian assumption of an MDP implies that $\mathcal{P}$ satisfies $\mathcal{P}(s_t|s_{t-1}, \dots, s_0, a_{t-1}, \dots, a_0) = \mathcal{P}(s_t|s_{t-1}, a_{t-1})$. Observe that $\mathcal{R}$ could be either dense, meaning that reward is frequently given, or sparse, meaning that it is most often zero. In the latter case, reward is frequently given only at the end of a long sequence of actions (an episode). There is also an underlying probability distribution of the states determined by the transition probability function and the start state.

A policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is the conduit through which agents produce a distribution over actions from a state. The value of a state $s \in \mathcal{S}$ for a policy π is defined as the discounted sum of rewards when running that policy starting from s:

$$V_\pi(s) = \mathbb{E}\left[\sum_t^\infty \gamma^t R(s_t, a_t) \middle| S_0 = s, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\right] \tag{2.1}$$

When using policy gradients as our algorithm of choice for finding an optimal π^*, we traditionally maximize the following cumulative return function instantiated from a specific starting state s_0:

$$J(\pi_\theta) = V_{\pi_\theta}(s_0) = \mathbb{E}\left[\sum_t^\infty \gamma^t R(s_t, a_t) \middle| S_0 = s_0, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\right] \tag{2.2}$$

Ideally, the learned agent should be able to generalize to new environments, however most reinforcement learning setups test the agent on a very similar, if not the same, distribution upon which it was trained. For example, agents trained on the Arcade Learning Environment [16] will be tested on the same game. One reason that the (aforementioned) Rearrangement Challenge is so difficult is because of the ask that the agent generalizes to new environment settings [72].

Multi-Agent Learning extends the Reinforcement Learning setting to have more than one agent. The singular policy π becomes a vector $\pi = [\pi_0, \ldots, \pi_{n-1}]$ of agent policies π_i. Each π_i is drawn from a policy space $\prod_i$. The space $\prod$ is the joint policy of all $\prod_i$ and $\prod_{-i} = \prod \setminus \prod_i$ is the joint policy for all agents except agent i. Similarly, the vector π_{-i} is of policies in $\pi \setminus \pi_i$. The optimal π_i^* for a given agent is then given by:

$$\underset{\pi_i \in \prod_i}{\operatorname{argmax}} \int_{\pi_{-i} \in \prod_{-i}} \mathbb{E}\left[\sum_t^\infty \gamma^t R(s_t, a_t) \middle| S_0 = s_0, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\right] \tag{2.3}$$

There are two common approaches to solve multi-agent problems. If we assume that all of the other agents are static agents of the world that do not learn, then we really only care about the singular agent that we are training and the Markovian assumption remains true. Consequently, we can use the reinforcement learning framework.

Where the 'multi-agent' modifier is differentiating is when the agents have their own disjoint observations that influence their learning. This breaks the Markovian assumption because it is no longer true that the transition function depends only on the agent of interest's policy, but also on the observations, actions, and learning of all other agents. That this information is rarely in the pool of common knowledge is one of the central challenges of multi-agent learning and why it is so much more difficult than single-agent reinforcement learning. This naturally leads to the Partially Observable Markov Decision Process (POMDP), which modifies the MDP by adding in Ω and O representing, respectively, the set of observations and the set of conditional observation probabilities. At each turn, the agent does not receive the state but instead a *partial observation* $o \in \Omega$ of the new state s' with probability $O(o|s')$.

Supervised Learning operates in a very different regime that assumes independence in our training predictions. This is in contrast to Reinforcement Learning where the cumulative training reward at state s_t is dependent on the decisions taken up to time t.

Specifically, it assumes two random variables X, Y where $x \in X \subseteq \mathbb{R}^{d_X}$ are feature vectors for which there is an associated label $y \in Y \subseteq \mathbb{R}^{d_Y}$. For example, x could be an image of a number and y could be which of the ten digits is x. Just like the states in reinforcement learning, these also have accompanying distributions $p(X)$, $p(Y)$, and the joint distribution $p(X, Y)$. There is a training set T^{train} from which we sample training examples $\left\{(x_1, y_1), \ldots, (x_N, y_N)\right\} \sim T^{\text{train}}$, test sets T_i^{test} from which we sample test examples, and a loss function $\mathcal{L} : Y \times \hat{Y} \longrightarrow \mathbb{R}$ that measures the difference between the actual label and the predicted label. The individual training and test sets have their own probability distributions that may not be the same. The goal is to use the training set to learn a predictor $f : X \longrightarrow \hat{Y}$ such that the expected error for the held out test distributions, $\mathcal{R}^i(f) = \underset{(x,y)\sim T_i^{\text{test}}}{\mathbb{E}} \mathcal{L}(y, f(x))$, is low. This expression is referred to as the *risk*. It is challenging to compute the risk in general, so we use an empirical estimate $\mathcal{R}^i(f) = \frac{1}{M}\sum_{j=1}^{M} \mathcal{L}(y_j, f(x_j))$ where each $(x_j, y_j) \sim T_i^{\text{test}}$.

In training, we only have access to T^{train} and so we optimize the model towards its risk. That frequently overfits, and so we apply a regularizer C that penalizes model complexity, culminating in an objective to minimize the *regularized empirical risk* [191] where samples $(x_j, y_j) \sim T^{\text{train}}$:

$$\mathcal{R}(f) = \frac{1}{M}\sum_{j=1}^{M} \mathcal{L}(y_j, f(x_j)) + \lambda C(f) \tag{2.4}$$

Most commonly, there are three disjoint sets - a training set, a validation set, and a test set. While iterating, the validation set is used to gauge the model's generalization capabilities and its position on the bias-variance trade-off. The test set measures final performance. The goal is to have f generalize beyond the training set and learn an underlying representation of the data that allows it to do well on not just samples drawn from T^{train} but also on samples drawn from all T_i^{test}. This is exhibited in object detection for autonomous vehicles where we want the agent to see pedestrians all the same in clear California skies, foggy London, and snowy Norway settings.

Bridging the settings In order to bridge reinforcement learning and supervised learning, we define concepts $u_i \in U$, a predictor $f : U \rightarrow \hat{Y}$, and a function $\mathcal{Y}(u_i)$ representing the true value of u_i. In supervised learning, $X = U$, $\hat{Y}$ is our label prediction range, and the ground truth label $y_i = \mathcal{Y}(u_i)$. In reinforcement learning, $\mathcal{S} = U$, $\hat{Y}$ is our state value range, and the optimal value of the state $V_{\pi^\star}(u_i) = \mathcal{Y}(u_i)$. We retain the same underlying probability distributions.

Define a conceptual group $\mathcal{G}_k = \{u_i : 1 \geq \mathcal{W}_k(u_i) > \tau_k \geq 0\}$ where τ_k is a threshold parameter associated with a $\mathcal{W}_k$ function that decides membership. An example $\mathcal{W}_k$ decides whether an RGB image scene includes both a jeep and rain, in which case $\tau_k = 0$ because it is binary. Another example $\mathcal{W}_j$ decides whether an agent's state is a part of the kitchen. There, $\tau_j > 0$ because the state of being in a kitchen is not always binary in a house setting. Observe that $\mathcal{G}_k$ is a truncated set of concepts of the original set U. Where $p_U(x)$ was the probability density function for U, the probability density function for $\mathcal{G}_k$ is:

$$p_{\mathcal{G}_k}(x) \propto p_U(x) \mathbb{1}\{W_k(x) > \tau_k\} \tag{2.5}$$

2.2 Defining the most challenging cases

We now define the most challenging cases and how they coincide with each setting. Recall the colloquial definition presented in Chapter 1 – the most challenging cases are data (training examples or states) who hold membership in a semantically-defined group that is, on average, much more difficult for the model than is commonly encountered data selected at random.

For a target task, we consider a set U of either examples in supervised learning or states in reinforcement learning and consider a family of groups $\mathcal{G}_1, \ldots, \mathcal{G}_n$ where each associated $\mathcal{W}_i$ labels a notion of human semantics. Although it is not always a requirement, this often implies that $\mathcal{W}_i$ has a human in the loop as part of its process. If the task was for a reinforcement learning agent to navigate a constrained maze, then one family of groups would cover the positions of the maze according to location. If the task was supervised object detection, then a family of groups would cover the (large number of) combinatorial class interactions in the scenes.

Elements of the $\mathcal{G}_i$ are our *cases*. However, when we discuss them, we describe them in terms of their deciding function $\mathcal{W}_i$, which serves as an aggregating concept conveying the common feature amongst the cases. In practice, we would ask if the model has difficulty with a) Jeeps b) when it is raining c) at night. This combination would be its own group.

The most challenging cases are contained within groups whose risk is much greater than the risk over the original distribution U:

$$\bar{\mathcal{G}} = \{\mathcal{G}_k : \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i))|u_i \sim \mathcal{G}_k\right] \gg \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i))|u_i \sim U\right]\} \tag{2.6}$$

$$= \{\mathcal{G}_k : \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i))|u_i \sim \mathcal{G}_k\right] - \alpha_k = \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i))|u_i \sim U\right]\} \tag{2.7}$$

For $\mathcal{L}$, euclidean distance suffices in reinforcement learning. For supervised learning, we use the target task's evaluation criterion, commonly either euclidean distance or log loss.

We list the groups in descending order according to the positive α_k and find our most challenging cases listed first. Equation 2.5 shows that changing the cases from U to $\mathcal{G}_k$ changes the distribution, which, as we know from the No Free Lunch Theorem, will lead to performance difficulties on some group. Equation 2.6 then restricts the cases to those that are detrimental to the model. That these cases are semantically defined groups of data examples according to $\mathcal{W}_k$ and τ_k means that it is trivial for a human to describe the case. For the model, which only ever sees the individual examples and not the bigger picture grouping, the difficulties lay in the examples within the set $- \bigcup_{\mathcal{G}_k \in \bar{\mathcal{G}}} \{u \in \mathcal{G}_k\}$.

2.3 Uniting the settings with an application framework

In this section, we present the Machine Learning Application Framework in order to further unite the three settings under one umbrella view as well as help bridge theory into practice.

The framework posits a *model trained* to *optimize* results on *training data* and *evaluated* on held out *test data*. The model is improved by looping, wherein each loop we collect more data to address poor evaluation results and retrain or finetune the model accordingly. While each of the italicized parts are not independent - we are very unlikely to use a Support Vector Machine (SVM) [46] with Q-learning [205] - they are to a large extent modular. For example, the *model* could be an SVM or it could be a neural network, within which there are many choices as well;

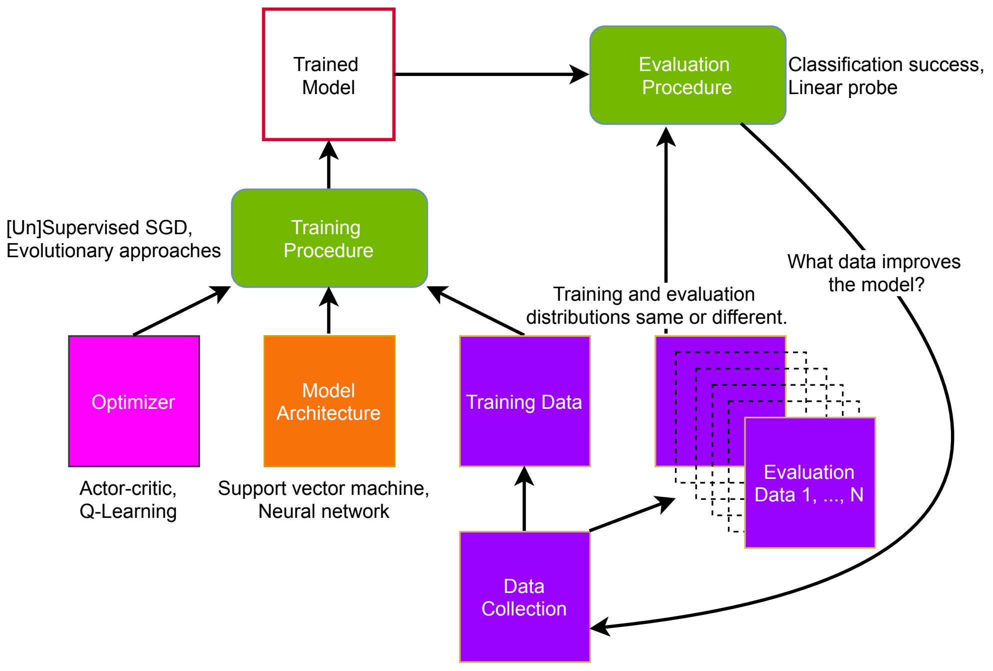
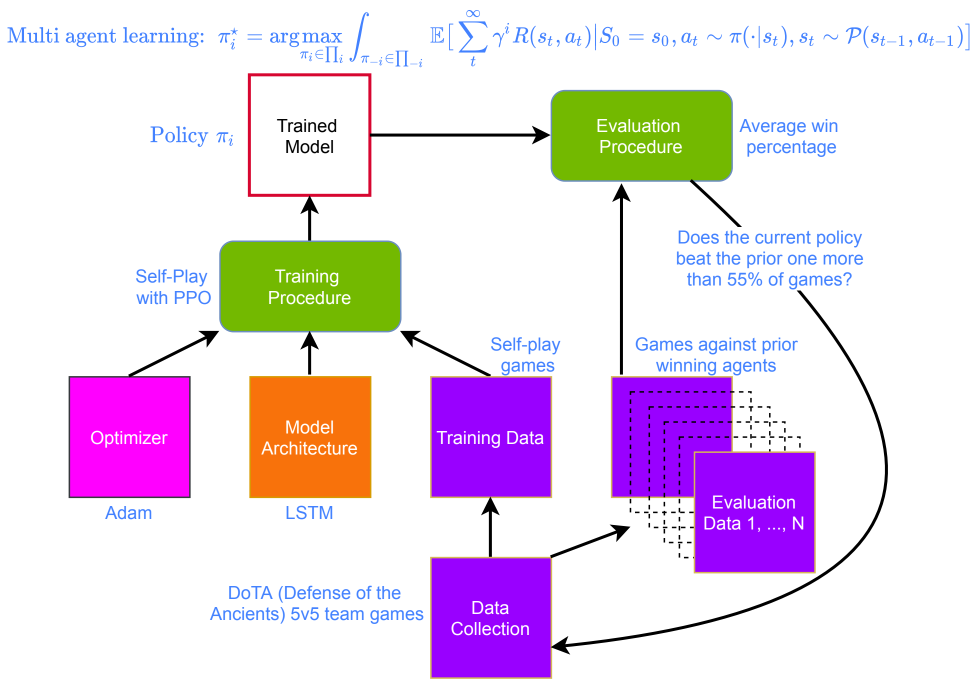

Trained Model
Evaluation Procedure
Classification success, Linear probe
[Un]Supervised SGD, Evolutionary approaches
Training Procedure
What data improves the model?
Training and evaluation distributions same or different.
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Actor-critic, Q-Learning
Support vector machine, Neural network
Data Collection

Figure 2.1: A graphic interpretation of the Machine Learning Application Framework. There is a data collection process that feeds both training and evaluation datasets. The training data, along with the model architecture and the optimizer, feed the training procedure and yields a trained model. That model is evaluated with the evaluation procedure, which is chosen independently from the training procedure. Finally, evaluation informs what new data to collect, which then results in more data and either retraining or fine-tuning, completing the loop. Observe that while the optimizer, the architecture, and the training/evaluation procedures are not completely independent, they can be mixed and matched. For example, we could use supervised or evolutionary approaches with neural networks or support vector machines.

the *training data* could come from the same distribution as the *test data* or they could differ; the *evaluation* procedure could be a linear probe on a model *trained* with self-supervision or they could both be results on the same game; the *optimizer* could be an Actor-Critic [103] method or it could be from the Q-learning family. See Figure 2.1 for a generic graphical representation.

An instantiation of the framework engenders a system, and these systems have improved dramatically over the past decade. And yet, as we outlined in Section 1, there remain copious and debilitating challenging cases upon which these models fail.

Figure 2.2 represents the framework for the multi-agent system that engineered OpenAI's recent DoTA agent [20]. This game-playing bot had more experience than all humans combined



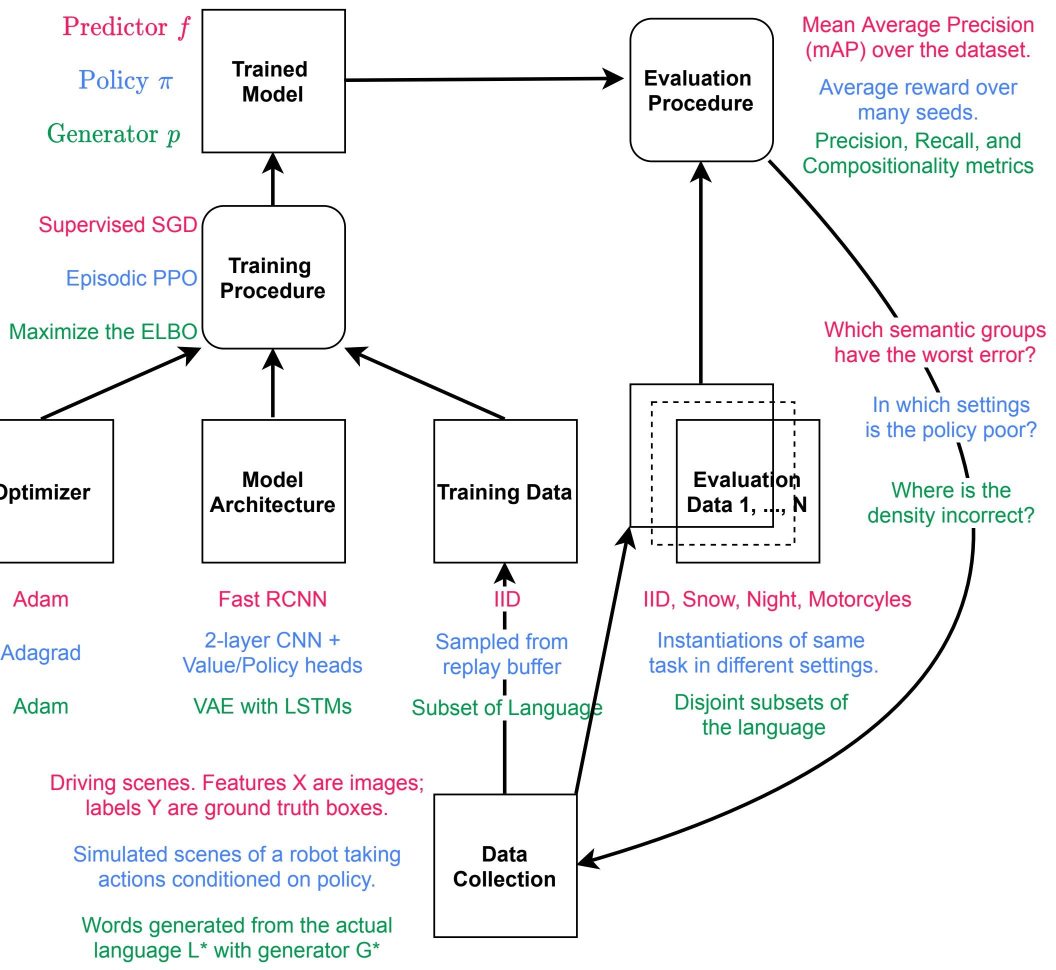
Multi agent learning: $\pi_i^\star = \arg\max_{\pi_i \in \prod_i} \int_{\pi_{-i} \in \prod_{-i}} \mathbb{E}\big[\sum_t^\infty \gamma^i R(s_t, a_t) | S_0 = s_0, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\big]$
Policy π_i
Trained Model
Evaluation Procedure
Average win percentage
Does the current policy beat the prior one more than 55% of games?
Self-Play with PPO
Training Procedure
Self-play games
Games against prior winning agents
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Adam
LSTM
DoTA (Defense of the Ancients) 5v5 team games
Data Collection

Figure 2.2: Machine Learning Application Framework for the algorithm powering the agent made by OpenAI to play Defense of the Ancients. While the agent was able to beat top-level players at the game, it was beat handily shortly after being released publicly.

and yet it still lost because of holes in its understanding and an inability to creatively adapt. The failure lies in that it was not exposed in training to opponents that could challenge it in the way that humans did. That, coupled with the expansive DoTA gamescape [12] meant that even amateur humans could find strategies that beat it. These strategies were the challenging cases where error was much higher than expected. In other words, the joint policy space $\prod_{-i}$ over which it tried to maximize its capabilities was too small.

Figure 2.3 shows another framework presenting a color coded interpretation of all three of the research directions involved in engineering a robot maid. Honing in on the system in red, the framework shown for the detection module is almost identical to what powers modern day autonomous L2 detection solutions as found in Tesla systems [Karpathy]. These modules are crucial to achieving autonomy and are fed more data than any human ever sees. Regardless, they still succumb to challenging cases and mistake the moon for a traffic light[1] or assume that traffic lights must remain static[2]. It is not an exaggeration to say that these challenging cases are preventing these applications from being used more widely. In AV, they are preventing the industry from reaching Level 5 and ushering in a large technological shift.

[1]https://twitter.com/JordanTeslaTech/status/1418413307862585344
[2]https://twitter.com/haltakov/status/1400797882891091970



Predictor f
Policy π
Generator p
Trained Model
Evaluation Procedure
Mean Average Precision (mAP) over the dataset.
Average reward over many seeds.
Precision, Recall, and Compositionality metrics
Supervised SGD
Episodic PPO
Maximize the ELBO
Training Procedure
Which semantic groups have the worst error?
In which settings is the policy poor?
Where is the density incorrect?
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Adam
Adagrad
Adam
Fast RCNN
2-layer CNN + Value/Policy heads
VAE with LSTMs
IID
Sampled from replay buffer
Subset of Language
IID, Snow, Night, Motorcyles
Instantiations of same task in different settings.
Disjoint subsets of the language
Driving scenes. Features X are images; labels Y are ground truth boxes.
Simulated scenes of a robot taking actions conditioned on policy.
Words generated from the actual language L* with generator G*
Data Collection

Figure 2.3: A Machine Learning Application Framework for all three of the tasks: Object Detection (red), Task Learning (blue), and Language Learning (green). This graphic highlights how we can perceive each of the three tasks as being of the same template. We investigate this and their connections to the hard examples further within each chapter. Note that the example approaches for each category have competing approaches not shown here, e.g. a competing object detection approach called DETR [33] uses hungarian matching in their training procedure and transformers for their model architecture.

Interpreting these problems overlaid on each other (like in Figure 2.3) within the context of the framework demonstrates how they are highly interrelated. Improving our research understanding of the challenging cases - the central goal of this thesis - directly improves the mechanisms of this loop and consequently catalyzes progress. In Chapter 4, Chapter 5, and Chapter 6, I return to this framework and show which parts are improved by my work.

2.4 Summary

We have now defined everything we need to understand the most challenging cases as well as the rest of this thesis, and can move on in the following chapters to how my work has advanced the field. In particular, I will present practical work on important applications like autonomous driving and language learning. Each chapter will focus on either identifying the most challenging cases, addressing them, or understanding their root causes.

3 Related Work

In this section, we provide background on related work from areas adjacent to the topic of the most challenging cases. This is in addition to the related work sections within each chapter, which focus on our specific contributions.

3.1 Out of Distribution Generalization

The most challenging cases are prominent in Out of Distribution Generalization (OOD). The classical definition for OOD generalization [199; 5] assumes that the test distribution belongs to an infinite family $\{\mathbb{P}^e\}$ where $e \in \xi$ are the set of environments, and $\mathcal{R}^e := \mathbb{E}\left[\mathcal{L}(y^e, f(x^e))|(x^e, y^e) \sim \mathbb{P}^e\right]$ is the risk for environment e. The goal is to minimize worst case risk over all test distributions: $\mathcal{R}(f) = \max_{e \in \xi} \mathcal{R}^e(f)$. Arjovsky [5] characterizes this definition as the simplest quantity that captures the relevant phenomena of the field, because it only requires high performance over all test distributions. Our topic dovetails with that notion as we seek to get high performance on the most challenging cases, which are the semantic groups with outsized risk.

In contrast to most work in this literature [56; 17], we are not trying to solve OOD generalization in theory and at large, but instead narrow the focus to practical applications and the cases to which we most want to generalize. While the larger goal of generalization would help us, e.g. learning to driving autonomously in California transferring to environments in India, we focus directly on those practical applications.

A subfield within OOD generalization - distributional robust optimization (DRO)[1] is more in line with this objective, with works like Sagawa et al. [166]; Oren et al. [140]. Sagawa et al. [166] assumes the groups have already been identified and then addresses them via strong regularization or group adjustments to explicitly handle the difference in group risk; that work is particularly of note in the modern machine learning error as it explores DRO in the context of highly over-parameterized models and shows that it compares favorably to empirical risk minimization. Oren et al. [140] focuses on language and proposes a procedure that minimizes training loss over the worst-case mixture of topics, where a topic is an umbrella group $\mathcal{G}_k$ containing utterances similar with respect to their subject matter, e.g. 'politics', 'business', or 'food'. They require that the topics have sufficient overlap with the training distribution and view the problem in the context of population shift at test time.

3.2 Fairness & Bias

There exists a common concern when deploying ML systems in the wild about how sensitive they are to protected attributes such as race, gender, or disabilities. This concern increases when those models are used in sensitive environments to make important life decisions. The infamous Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) software measured case-by-case recidivism, and the investigation [89] into this software found that it exhibited higher false positive rates for African-Americans on trial than for Caucasians. Unfortunately, due to the systemic biases in data, algorithms, and even the mechanisms in how we design algorithms, this calamity is not uncommon in machine learning.

The field of Fairness & Bias[2] focuses on these issues and surfaces relevant work addressing the most challenging cases where groups $\mathcal{G}_k$ are decided by W_k related to protected attributes. These works can be delineated by the type of bias involved or the type of fairness that we demand.

[1]See Rahimian and Mehrotra [152] for a thorough review of the DRO literature.
[2]See Mehrabi et al. [129] for a thorough review of the fairness literature.

Some example types of bias are:

- **Representation bias** arises when the training dataset is not representative of the diversity of the population. Examples are the commercial face recognition software biased against darker skin tones [28], which each serve as the challenging groups $\mathcal{G}_k$. This results in a poor recognition of dark-skinned populations throughout the world and is especially problematic for African countries.
- **Algorithmic Bias** occurs when bias is a consequence of the choices that influenced the model design, such as optimization, architecture, regularization, or feedback loops. An example are modern social feeds, which favor highly engaging news items. This has a direct effect on societal discourse and polarizes politics in a two-party system [99].
- **Historical bias** is the existing bias in society that seeps into our models even with perfect sampling in place. Example are plentiful in language where associations between names, careers, and beliefs have propagated from media into models through data [31]. Examples include women being more likely than men to be caretakers or nurses and much less likely than men to be doctors. This has a dangerous feedback loop in that it promotes incorrect views that certain subgroups are less capable than others.

3.3 Heterogenous Treatment Effects

In medicine, Randomized Control Trials (RCT) are the gold standard for comparing treatment efficacy. We want RCTs to generalize their results to broader parts of the population. However, frequently their results will not be true for patients that differ from those in the clinical trial [32; 93]. Generalization requires assuming the stable unit treatment value, which says that a medicinal outcome is unaffected by how the treatment was assigned or the treatment exposure of the individuals involved. To attain more statistical precision, practitioners usually turn to heterogeneity of treatment effect (HTE), an analysis that illuminates non-random variation in a treatment for population subgroups. Further, the ultimate goal of HTE is to predict optimal treatments for individuals. We thus see a qualitative connection between HTE and identifying the most challenging cases in that HTE strives to delineate which subgroups are the most challenging for the predictive model defined by the RCT.

For a thorough understanding of HTE, please see Velentgas et al. [193]; Varadhan et al. [192]. In short, it relies on subgroup analysis where the groups are selected based on the mechanism and their plausibility, taking into account priors related to the treatment itself. However, it is insufficient to do only that because we lose too much statistical power when splitting the samples into subgroups. One problem that arises is 'p-hacking' - even if *some* split suggests a strong statistical difference, this may be random noise. This follows as a result of there being many potential ways to split the sample and researchers generally striving to find something of consequence in the data. The common answer to this is to register a pre-analysis plan, but it is not a catch-all solution. Another problem is that it is challenging to pre-define splits that adequately capture the interactions, because the most affected group might be found at the intersection of multiple attributes.

HTE methods have recently begun incorporating machine learning [198; 40]. This is encapsulated well by the method proposed by Ali et al. [2], which is not exactly HTE but takes inspiration from it. They assume a fitted model $f : X \to Y$, a validation set T^{val}, a test set T^{test}, a class of subgroups $\mathcal{G}_k$, and an error measure $\mathcal{L}$. Observe that both the regions defined by $\mathcal{G}_k$ are known in advance as well as which of the elements of the validation and test sets are in each $\mathcal{G}_k$. The core of their algorithm is as follows:

1. For each $(x, y) \in T^{\text{val}} \cup T^{\text{test}}$, compute $\mathcal{L}_f(x, y)$.

2. For each group $\mathcal{G}_k$, compute $W(\mathcal{G}_k) = \sum_{\substack{(x_t,y_t)\in T^{\text{test}} \\ (x_v,y_v)\in T^{\text{val}}}} T(\mathcal{L}_f(x_t, y_t), \mathcal{L}_f(x_v, y_v))$ where $T(a, b) = 1$ if $a < b$, $T(a, b) = \frac{1}{2}$ if $a \approx b$, and $T(a, b) = 0$ if $a > b$.

3. Return $\operatorname{argmax}_{\mathcal{G}_k} W(\mathcal{G}_k)$.

This procedure identifies the most challenging cases as an ordered list of subgroups $\mathcal{G}_k$ with degraded model performance. They experimentally evaluate this in three application settings: disease forecasting, classifying satellite imagery by country, and distribution shifts in sentiment analysis. Their approach differs from our contribution in the area of identifying challenging subgroups (Part I) in that they do not admit a causal interpretation and that all potential subgroups must be decided in advance.

Part I

Identifying the Most Challenging Cases

4 | Object Detection

4.1 Introduction

Object detection is an important task in computer vision that focuses on detecting instances of visual objects of specified classes from a scene. Usually, that scene is manifested as a 2D digital image. The goal is to predict tight bounding boxes for more exact class predictions at higher confidence scores. Fundamentally, object detection is answering the question 'what in the scene is where?'

Object detection is driven by benchmark datasets where each datum $D = (I, L)$ with I an image and L a set of ground truth bounding boxes b_k. Each $b_k = (c_k, x_k, y_k, w_k, h_k)$, where c_k is the class of the object in this box, (x_k, y_k) is the position of the top left corner of the box in the image, and (w_k, h_k) is the width and height respectively of the box in the image. A per-prediction attribute that we care about is intersection over union (IOU). For a predicted box b_k, its intersection is the highest overlap it has with a ground truth box g_j that has not been claimed. Its union is the total combined area of b_k and g_j, accounting for their overlap. Its IOU is then the intersection divided by union, which is bounded between 0 and 1. After a g_j has been used by some b_k, it is considered claimed and unavailable to other b_k.

What metric do we use? To evaluate improvements, we need a metric that correlates well with whether the bounding boxes are getting tighter and whether the class accuracy is improving. The aforementioned IOU metric works well for single boxes, however we care most about understanding performance over an entire test dataset. IOU is poor for that objective because there is too much variability in images based on the number of ground truth boxes they have. Instead, the field has settled on mean Average Precision (mAP). It ranges from 0 to 100, the higher the better. It works by first taking all detections over all test images and drawing class-specific precision-recall curves. For each class, the average precision is the area under the curve. The mAP is the mean of these scores. However, we usually are evaluating mAP@x, where 'x' is threshold, typically 0.5. This threshold is used to gate true positive detections by their IOU. Consequently, mAP@0.5 would be mean Average Precision but where the only positives are those with IOU $\geqslant 0.5$.

Modern deep learning approaches For an extensive background on the subject, please see Zou et al. [220]. In this section, we present background understanding on modern deep learning methods for object detection in order to better position our contribution. The first major modern milestone was Girshick et al. [69], or RCNN.

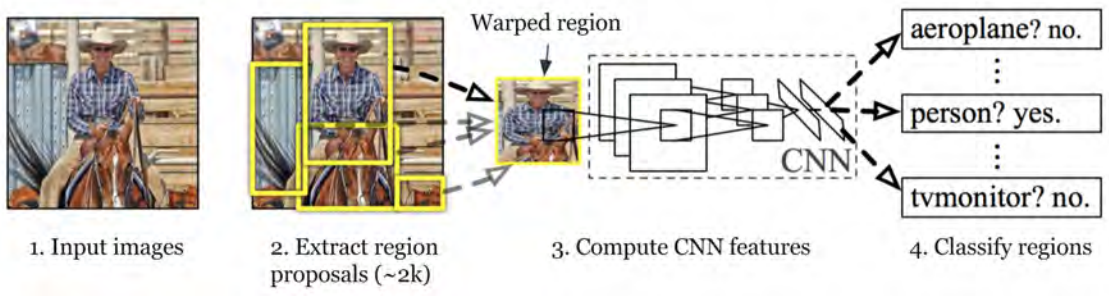


Figure 4.1: Visual description of how RCNN [69] works. The second step is performed by selective search and the fourth step by a Support Vector Machine.

In RCNN, we first assume access to a convolutional neural network (CNN) pre-trained for classification on ImageNet. For each of our images, we then use selective search [189] to find a set of potential object bounding boxes. In order to be fed into the CNN, we then warp these bounding boxes to be a fixed size. That requires fine-tuning the CNN to work with these warped images. After fine-tuning, each proposal is fed into the CNN to get a feature vector. These vectors are then used to train a binary Support Vector Machine [46] (SVM) for each class independently. Additionally, a regression model is trained to adjust the predicted bounding boxes, also using the CNN features as input. This paper jumped the field forward immensely, advancing mAP from 33% to 58%.

RCNN was slow because the number of proposals offered by selective search was very high, we needed to generate a CNN vector for every image region, and the fine-tuning was a manual process. It was also challenging to extend because a new SVM would need to be trained for each class of interest. From the same team, Fast RCNN [68] was 200x faster and improved the mAP score to 70%. It works by unifying the three models into one joint framework. The CNN features were now aggregated in one forward pass on the image, and the same trunk is used for the object classifier and the bounding-box regressor. To do this last step, they debuted ROI Pooling.

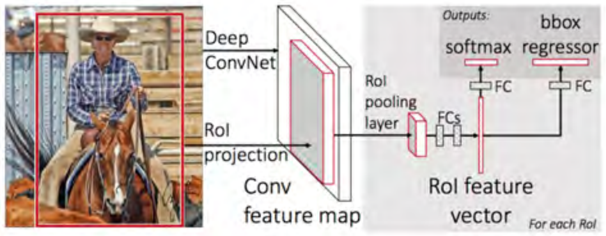


Figure 4.2: Visual description of Fast RCNN [68]. It unifies the models from RCNN into one joint framework that, assuming selective search for the proposals, is able to do both classification and bounding box regression in one step.

While this was a great improvement, it meant that we still had not solved a core speed problem with RCNN – the use of selective search. Faster RCNN [157] addresses this issue by replacing it with a Region Proposal Network (RPN) to generate the proposals. The RPN is a CNN by design and so the result was the first real-time object detector trained end-to-end. The succession from RCNN to Fast RCNN to Faster RCNN was by no means the only object detectors at the time, but they represent well the progression of classical object detection being subsumed by modern deep learning techniques. We use Faster RCNN in all of our experiments.

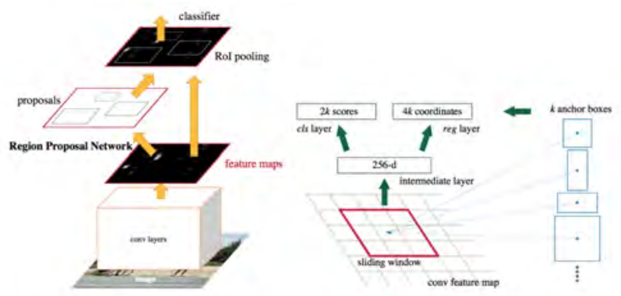


Figure 4.3: Faster RCNN [157] improved upon Fast RCNN by proposing the Region Proposal Network to generate the proposals, thereby replacing the prior bottleneck - the selective search module. With this in place, we could train deep object detection models end-to-end.

Further advances While we use Faster RCNN, we include an amendment that is similarly integrated across the board in object detection modules - Feature Pyramid Networks (FPN, [122]). The insight in FPN is that the deeper levels of a CNN are not conducive to localizing objects, only to classifying them. This is because much of that localizing information is lost as we progress in depth. Before FPN, most approaches would only use the last layer's features for both localization and classification. FPN added lateral connections to take advantage of high-level semantics learned by intermediate layers.

We would be remiss not to mention Mask RCNN [74]. While we do not use it, it represents how the field advanced to do image segmentation at the pixel level and not just bounding boxes. It did this by adding a third branch that predicted the object mask in parallel with the bounding box and object classification predictions. It had to improve the ROI Pooling layer because it requires finer alignment in the pixels than is required when predicting only bounding boxes.

4.2 The most challenging cases

Consider an example scene where a model failed to detect a truck. Many explanations exist, ranging from the scene composition to the weather conditions to the way the light reflects off of a puddle and into the camera. The actual cause is unclear. This may be true even if the model is strong in general at detecting trucks, at detecting given puddles, and detecting when there are

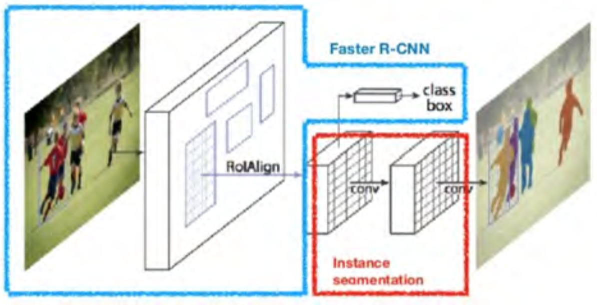


Figure 4.4: Mask RCNN [74] extends Faster RCNN to additionally perform instance level image segmentation by outputting the pixels where an object can be found.

light issues with the camera. It is the group defined by their intersection that is causing this error and it is imperative that we identify this group in order to both a) gather like training data and b) know if we have actually fixed the issue at test time. Real world examples of this abound for all perceptual object detection systems in the wild.

Formalizing within the framework from Section 2.1, our challenging cases are contained within the union of the family of groups $\mathcal{G}_k$ and membership functions $\mathcal{W}_k$:

$$\bar{\mathcal{G}} = \{\mathcal{G}_k : \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i)) | u_i \sim \mathcal{G}_k\right] \gg \mathbb{E}\left[\mathcal{L}(\mathcal{Y}(u_i), f(u_i)) | u_i \sim \text{IID}\right]\} \tag{4.1}$$

In the above example, the $\mathcal{W}$ that decides a challenging group is whether the scene has puddles, trucks, and light of a certain type. A familiar pattern for scenes in the wild are that they prove difficult for our neural network models for a long time until they are addressed at the group-level by gathering a dataset and retraining and/or fine-tuning. A notable reason for this is because current data-driven approaches trained to minimize expected error are sensitive to imbalanced data distributions. As a result, models with low expected error can still exhibit large errors on atypical groups of data that are nonetheless important for safe driving.

4.3 Overview of our contribution

The status quo approach to finding $\mathcal{G}_k$ in the AV stack operates by first running it in real-world conditions and recording everything. When there is a disengagement, we save that recording with enough metadata so that humans can parse the metadata and videos to then categorize what went wrong. It is at this point that we start to gain understanding on which groups $\mathcal{G}_k$ does the model perform poorly. We attain this understanding because the humans in the loop can pattern-match to understand the $\mathcal{W}_k$ associated with the failures. With the constituent $\mathcal{G}_k$ in hand, the data collection team then gathers a new large quantity of data representing the challenging cases contained in $\bar{\mathcal{G}}$. We feed this new data into the model's training, either from scratch or fine-tuning, and retest the model on all categories gathered thus far. This approach operates in *hindsight* by analyzing real-world scenes requiring driver intervention. It is also very expensive and time-consuming, because identifying the causal factors in the failures requires human experts to comb through the data and group commonalities.

We propose an alternative method to identify challenging groups in foresight. As shown in Figure 4.5, instead of finding failures from previously collected data, we perform interventions on existing data to find those interventions that are detrimental to the performance of an AV stack. We focus on perception, and object detection specifically, and identify interventions that consistently cause performance drops as challenging groups.

If, in the prior example with the truck, we had arrived at the scene counterfactually by performing a single intervention on another scene, e.g. changing a car to the truck, we would have some clue that the underlying causal error is related to the truck itself. We can duplicate this intervention across many scenes and see if it consistently remains a problem. While the *exact* cause is still opaque, this method provides automatic insight into what interventions cause consistent errors without collecting new data to analyze or manual scrubbing of failures.

Performing such interventions requires the ability to manipulate scenes and re-render images.

We demonstrate this in simulation, although recent advances [142] show promise in migrating our approach to real-world scenes. We assume access to a scene graph representation of the underlying scene on which we perform interventions. These interventions include changing agent properties like position, rotation, or asset type, as well as global weather conditions. While many interventions can potentially fail the detector, not all are useful. A scene with a flying truck could drop perception performance, but it is unlikely to occur in the real world. Ideally, interventions should be from the data distribution. We achieve this by training a density model of scenes (represented as flattened scene graphs) using a masked language model (MLM), a keystone in modern natural language processing pipelines. Taking interventions using the MLM amounts to masking a part of the scene graph and re-sampling from the predicted distribution.

We verify that the prioritized groups we find via intervention are indeed challenging for the base object detector and show that retraining with data collected from these groups helps inordinately compared to adding more IID data. We additionally confirm our hypothesis that interventions on the data distribution are preferred vis a vis data efficiency by comparing against random interventions. The latter are confounded by their propensity to stray from the data distribution. We compare these results against an important baseline we call 'Cause-agnostic Data Collection', which are scenes for which the model performs poorly according to the same custom scoring function used for the interventions. Finally, we examine what happens when we take a second intervention using the MLM and find new veins in which we could mine specific problematic groups, suggesting that there is room to continue this process.

4.4 Background

Notation Our objective is to ascertain the capabilities of a given object detection model ϕ. We represent a scene x as a triplet $x = (G, I, L)$ of a scene graph (includes the camera parameters), scene image, and a set of bounding box labels, respectively. We flatten and discretize G to get

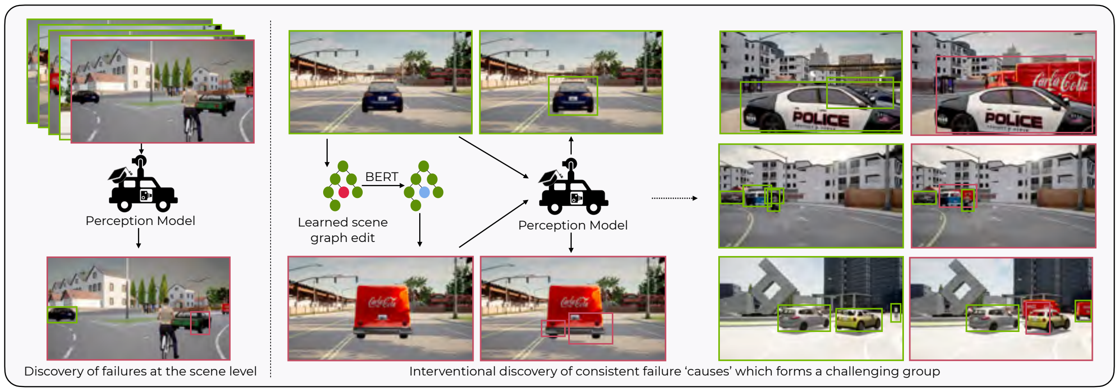

Perception Model
Discovery of failures at the scene level
BERT
Learned scene graph edit
Perception Model
Interventional discovery of consistent failure 'causes' which forms a challenging group

Figure 4.5: Instead of retrospectively discovering individual failure cases for perception, we actively search for causal interventions (edits) to existing scenes that consistently result in perception failures. The middle shows an example of a single intervention causing perception failure, which we attribute to the intervention, as opposed to the left where a combinatorial set of factors could explain the error. Consistent failures through this type of intervention constitute a challenging group for the perception model as seen on the right.

the corresponding sequence $S \in \mathbb{N}^{O(d)}$ where d is the variable number of objects in the scene. The scene image $I \in \mathbb{R}^3$ is the RGB image of the scene as observed by the ego car and is deterministically defined by G. The label L is a set of ground truth bounding boxes $l_k \in \mathbb{R}^4$, where $k < d$ is the number of objects to identify in the scene. Scenes are drawn from a distribution $p_R(x)$ dictated by the scene generation process R. Examples of R include driving scenes from a particular city or simulations from AV simulators. We also define a per-example scoring function $f : (\phi, I, L) \longrightarrow y \in [0, 1]$ as well as a threshold τ with which to gauge whether an intervention was detrimental.

Scene Graphs are 3D world representations, with nodes corresponding to entities and edges to hierarchical relationships between nodes, where the hierarchy is determined by physical presence (e.g. road is a parent of the vehicles on the road). Entities include the vehicles and pedestrians, the weather, the ego agent, and the camera parameters. Each node has associated attributes, exemplified by continuous rotation, continuous position, discrete asset type, etc.

Object Detection in images reached a milestone with Faster RCNN [157]. We use their approach as the representative state of the art detection module via the Detectron2 library [208].

Simulation is crucial to our method. We need a simulator that can initialize from G, have satisfactory traffic policies for autonomous vehicles, and return the current G on command. The chosen CARLA [53] simulator satisfies these constraints and is ubiquitous in the field.

Masked Language Models (MLM) are density models for sequential data. Devlin et al. [52] showed their tremendous efficacy in language generation. They are trained by receiving sequences of discrete tokens, a few of which are masked, and predicting what tokens are in the masked positions. Through this process, they learn the data distribution [137; 200] of those sequences. At inference, they are fed a sequence with a chosen token masked and replace the mask with their prediction. We perform causal intervention on scenes by asking a pre-trained MLM to re-sample a single position from a scene graph - see Section 4.7.1 for details.

4.5 Understanding scenes through intervention

We aim to improve object detection models by utilizing the advantages of AV simulators over collecting real world data, namely that they quickly synthesize scenes in parallel; that we have fine control over the synthesis; and that they grant us supervisory labels automatically. A naive approach is to generate lots of random scenes, test detection on those scenes, and set aside the hard ones for retraining. A more advanced approach is to use adversarial techniques to find hard scenes. Both approaches share two downsides: a) we are much more interested in scenes drawn from a distribution that is similar to the distribution of real-world vehicle scenes and b) there is a combinatorial challenge of understanding what in the scenes was the problem; only if we know why the error is happening can we find test scenes having similar challenges and thus understand if the issue is fixed after retraining.

We propose an efficient procedure that tackles both concerns. We find hard groups of data for a trained model ϕ by taking interventions on scene graphs with an MLM pre-trained on natural scene distributions. The resulting scenes are grouped according to their generating intervention type. We assess the model performance on each group with our surrogate scoring function f. The rationale behind this procedure is that solely identifying challenging *scenes* does not provide insight into how to improve ϕ. However, asserting that a type of *intervention* is consistently challenging narrows greatly where the model's difficulties lay. After finding challenging groups, we utilize hard negative mining [201; 106; 203], a common technique for improving models by first seeking the hardest examples and then emphasizing those examples through retraining or fine-tuning. Our approach notably achieves this without human labelers. See Figure 4.6 for a complete diagram of our approach and Figure 4.9 for qualitative examples. We now explain in detail each of the components of our method.

4.5.1 Per-scene scoring function

The scoring function f should delineate between interventions that were minimal and those that caused a significant change in perception performance, with the assumption being that large negative (positive) changes imply that the intervention (reverse intervention) was detrimental to ϕ.

Our goal in designing f is to replicate the average precision (AP) score's intent, which values having few predictions with high intersection over union (IOU) to ground truth targets. Another goal was to evaluate entire scenes and not just target assets. This is important because even though our interventions can be local to a node (weather is of course global), they may still impact detecting any scene constituent. We choose not to use the popular mAP because it is defined over a dataset and thus is not suitable for identifying individual challenging scenes, which our method requires before aggregating at the intervention level. To compute f, we get the model's predictions and order them by descending confidence. We sequentially align each prediction with the highest IOU ground truth. If IOU $> .05$, an empirically chosen threshold, then we mark this ground truth as claimed. The per prediction score is the product of the prediction's confidence and its IOU. We then take the mean over all predictions to get the model's final score on this example. The result is that predictions with low confidence or poor IOU reduce the model's score, while predictions with high confidence on quality boxes increase the score.

4.5.2 Causal interventions on simulated scenes

We draw from causal inference where interventions allow us to assess the causal links between the scene and the model's score. We change an aspect of a scene sequence S_i, such as a rotation or location of a specific vehicle, render this new scene S'_i as I', and then compute the $\delta = f(\phi, I', L') - f(\phi, I, L) \in [-1, 1]$. We decide sufficiency by whether $|\delta| \geqslant \tau$, the aforementioned threshold parameter. After performing this procedure N times, filtering by sufficiency, and then grouping by the intervention type, we arrive at a prioritized list of challenging groups defined by either rotation, vehicle type, or weather pattern.

Generating interventions Uniformly random interventions produce unlikely scenes under the true data distribution[1]. Even if such an intervention would identify a weakness in the detector, its utility in improving our model is unclear because such a weakness may be very far from a realistic setting. We should favor finding groups that have higher probability under the data distribution. This is especially important for a limited model capacity because learning to detect flying cars and other unrealistic low-priority scenarios might take capacity away from pressing needs.

Formally, with $p_{\mathrm{R}}(x)$ as the generation process, y our surrogate score, and z a confounder that affects both x and y, we need to draw a counterfactual x' that is independent of z with which we can causally probe the model's weaknesses. Sampling from $p_{\mathrm{R}}(x)$ is challenging because

[1]Empirically, so do interventions sampled uniformly over the categories of interest.

retrieving the same scene again with just one change is difficult. We could act directly on the scene graph and model the conditional distributions of a single node change, then select changes via Gibbs sampling, and define interventions as sampling from these conditional distributions. Instead, we choose to discretize the scene [190; 60; 154] and use masked language models [54; 94] because of their resounding recent success modeling distributions of combinatorial sequences relative to other approaches, as demonstrated clearly in language. Specifically, we train an MLM as a denoising autoencoder (DAE) to sample from $p_{\mathrm{R}}(x)$ [19; 127; 195], where the MLM operates on discretized scene graphs, flattened to be sequential. This provides a mechanism to sample counterfactuals from the data distribution DAE [137; 200].

For each scene drawn from the original training distribution, the MLM infers a new scene close to the original distribution by making a singular semantic change over weather, vehicle asset type, rotation, or location. For example, it may choose a vehicle instance and change that vehicle to a different vehicle type. Or it may rotate that vehicle some non-zero amount. For weather, the semantic changes could be over cloudiness, precipitation, precipitation deposits (puddles), wind intensity, or the angle of the sun (light). We never add or delete a node, only semantically change them. Because the MLM was trained to a low perplexity on data drawn from the distribution, it samples scenes that are likely under the original distribution $p_{\mathrm{R}}(x)$. Because it is not the exact distribution and errors will accumulate when applying many interventions sequentially, we intervene for just one step in most of our experiments, equivalent to a single node change in the scene graph. We expand this with an investigation into what happens when we take a second successive intervention step.

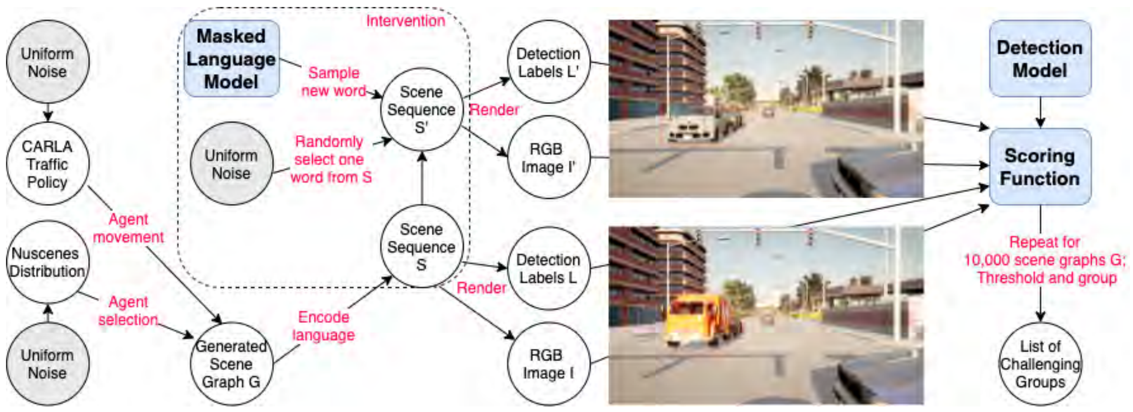


Figure 4.6: A complete diagram of our approach: We intervene on the scene by performing transformations like the pictured yellow truck becoming a white car and then evaluating the delta change in the object detection model's efficacy. The interventions are guided by a trained MLM. Repeat N times and group the scores to attain an ordered list of challenging groups drawn from vehicle type, weather, and rotation.

4.6 Related Work

4.6.1 MLM as a generator

While we believe we are the first to propose using an MLM as a generator in order to take causal interventions, Ng et al. [137] generates from an MLM in order to augment natural language task training with generated examples. Mansimov et al. [127] and Wang and Cho [200] do so in order to generate high quality examples for use in downstream examples, with the former producing molecules closer to the reference conformations than traditional methods and the latter producing quality and diverse sentences. None of these operate on scene graphs.

4.6.2 AV Testing and Debugging

See Corso et al. [45] for a detailed survey on black-box safety validation techniques. We believe that we are the first to take causal interventions in static scenes to test AV detection systems, although multiple approaches [67; 1; 104; 44; 139; 155] test AV systems through adversarial manipulation of actor trajectories and operate on the planning subsystem. Wang et al. [202] generates

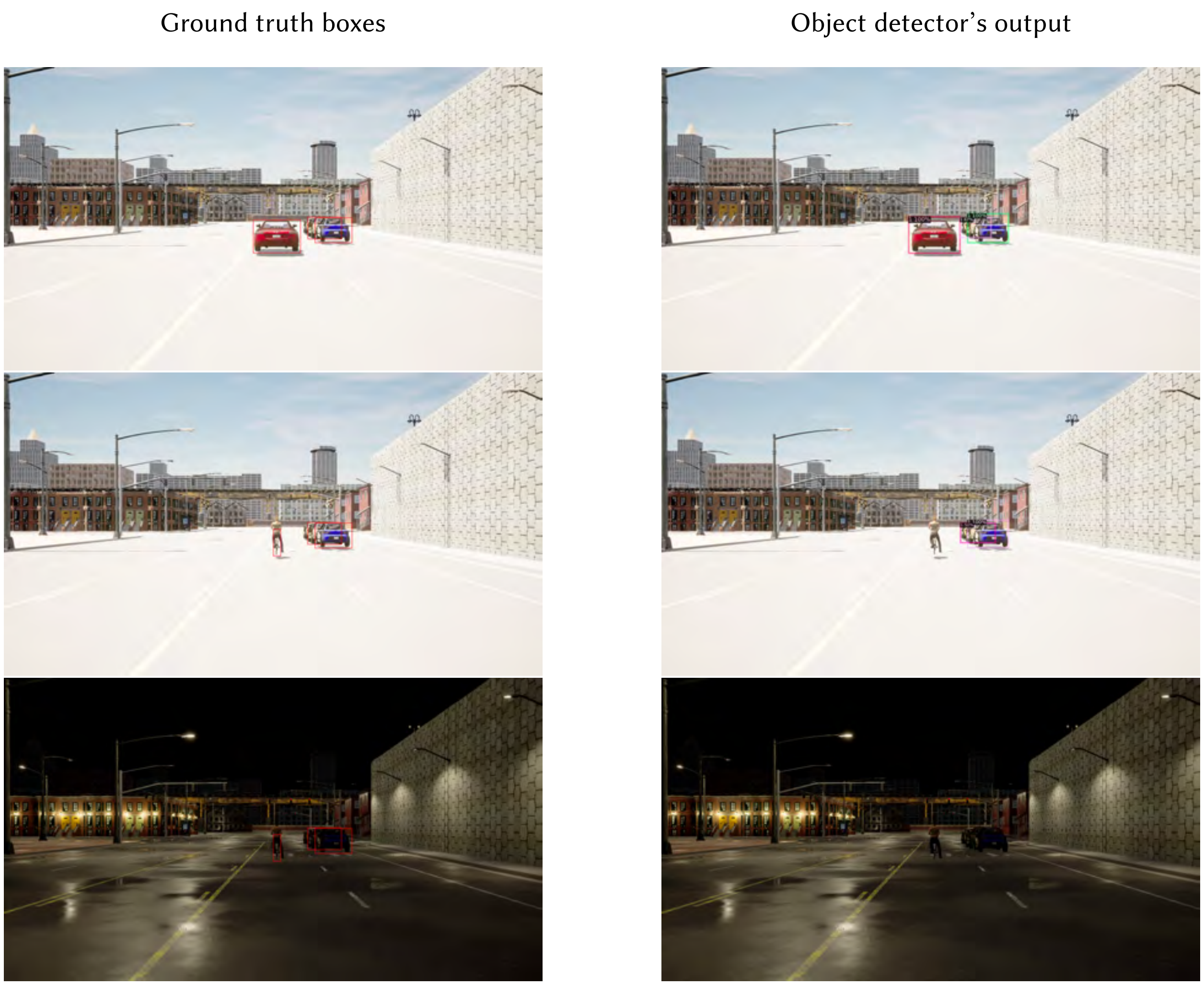


Figure 4.9: Interventions taken by the MLM. The first row is the original scene, the second after an intervention changing the red car to a biker, and the third after an intervention on the weather. The left side shows ground truth and the right shows the detector's predictions. Observe that the model's predictions deteriorate from being essentially perfect to missing the biker to missing every object. See Figure 4.25 in the Supplementary for a rotation example.

adversarial scenarios for AV systems by black-box optimization of actor trajectory perturbations, simulating LiDAR sensors in perturbed real scenes. Prior research has focused on optimization techniques for adversarial scenario generation through the manipulation of trajectories of vehicles and pedestrians. They either test only the planning subsystem in an open-loop manner or the whole AV system in a closed-loop fashion. Unlike our work, they do not allow for causal factor error interpretation. We focus on open-loop evaluation of AV perception and attempt to find causal factors for performance degradation through the generation of in-distribution counterfactuals with a masked language model trained on scene graphs. Concurrently, Leclerc et al. [112] proposed a configurable system to diagnose vulnerabilities in perception systems through synthetic data generation. We show how to generate complex scene manipulations using the MLM and study scenes of significantly higher complexity, although it is possible in theory to implement our method within their framework.

4.6.3 Scene manipulation

Ost et al. [142] learn neural scene graphs from real world videos via a factorized neural radiance field [130], while Kar et al. [91]; Devaranjan et al. [51] generate scene graphs of AV scenes that match the image-level distribution of a real AV dataset as a means to produce realistic synthetic training data. All three can be seen as a precursor to our method for handling real world data. Dwibedi et al. [57] generate synthetic training data for object detectors by learning to cut and paste real object instances on background images, which elicits a confounder because of how artificial the pasted scenes appear.

4.6.4 Adversarial detection

Another way of viewing our work is through the lens of adversarial detection. Xie et al. [209] showed that we should consider the detection task differently from the perspective of adversarial attacks, but did not explore finding root causes. Liu et al. [124] use a differentiable renderer to find adverse lighting and geometry. Consequently, images appear stitched, a confounder to the natural distribution. Athalye et al. [7] synthesizes real 3D objects that are adversarial to 2D detectors. They are limited to single objects, moving in the location, rotation, or pixel space, and do not identify causal factors. Zeng et al. [213]; Tu et al. [188] synthesize 3D objects for fooling AV systems, both camera and LIDAR, with a goal to demonstrate the existence of one-off examples.

4.6.5 Challenging groups

Improving the model to recognize found groups, potentially sourced from the distribution's long tail, is an important goal. Numerous methods [156; 3] do this by re-weighting or re-sampling the training set, with Chang et al. [37] focusing on detection. Sagawa et al. [166] uses regularization and Wang et al. [204] uses dynamic routing and experts. All of these approaches require us to know the problematic groups in advance, which would only happen *after* applying our method. Further, they do not assess why the model is weak, but only seek to fix the problem. This makes it challenging to understand if the core issue has been addressed. Gulrajani and Lopez-Paz [71] suggests that these approaches are not better than ERM, which is how we incorporate our found groups in Section 4.7.

4.7 Experiments

We run experiments to analyze our method and compare it against both taking random interventions and collecting cause-agnostic data. We then go on to assess what happens if we take a second intervention step.

4.7.1 Setup

Model We selected six battle-tested models from Detectron2: 18C4, 18FPN, 34C4, 34FPN, 50C4, and 50FPN. These are common ResNet [75] architectures that include a litany of other attributes such as Feature Pyramid Networks [122]. We created additional configurations that are 2x, 3x, 4x, and 5x wider versions of 50FPN, exemplified by 50FPN2x, for a total of ten tested architectures. The C4 and FPN mix provided variation in model configuration, while the 18, 34, and 50 layer counts and their widths vary in parameters. We made minimal changes to account for training on our dataset and with 4 gpus instead of 8. All models were trained for 90000 steps (8-9 hours) without pre-training; none reached zero training loss.

Datasets We first selected the CARLA preset map – Town03 or Town05. Town03 is the most complex town, with a 5-lane junction, a roundabout, unevenness, a tunnel, and more. Town05 is a squared-grid town with cross junctions, a bridge, and multiple lanes per direction. Both have ample space to drive around in a scene and discover novel views. Then we randomly chose from among the pre-defined weather patterns. We sampled the camera calibration and the number V of vehicle assets according to the Nuscenes [30] distributions, then placed those V vehicles, the ego agent, and $P = 20$ pedestrian assets, at random town waypoints suitable for the asset type. Finally, we attached the calibrated camera to the ego agent and enabled autopilot for all agents. We stabilized the scene for 50 timesteps after spawning, then recorded for 150 steps and saved every 15th frame. We needed the 2D ground truth boxes for each asset, but found the suggested approach[2] lacking because it frequently had trouble with occlusions and other challenging scenarios. See the Supplementary (Section 4.10) for heuristics we developed to help filter the ground truth boxes. For detection results on all charts, we report average precision (AP) over vehicle datasets.

MLM We used the MaskedLMModel architecture[3] from the FairSeq [143] library for our MLM. We train and validate on held out IID datasets of sequences converted from scene graphs, where the dataset was created as described in the prior paragraph. Encoding the scene graph language required us to translate G with continuous node attributes into discrete sequence S. The first 10 tokens corresponded to weather attributes (cloudiness, precipitation, sun altitude angle, etc), the next 5 to camera intrinsics, and the following 15 to the ego agent. After these 30, we had a variable number of agents, each sequentially represented by 17 tokens. The two extra tokens for the non-ego agents were related to vehicle type, which was fixed for the ego agent. Although the

[2]See client_bounding_boxes.py in the CARLA library, commit 4c8f4d5f191246802644a62453327f32972bd536.
[3]See masked_lm.py#L30 in the FairSeq library, commit 1bba712622b8ae4efb3eb793a8a40da386fe11d0

Intervention	Percent > 0.2	Total
Tier 1: Likely Challenging Groups		
DiamondbackBike	24.4	123
Cloudy Dark	19.4	36
GazelleBike	18.9	122
Cloudy Dark Puddles	17.2	29
CrossBike	16.5	121
Rotation - 178	15	20
Rotation - 121	13.0	23
Tier 2: Borderline Groups		
KawasakiBike	6.5	92
Cybertruck	6.4	94
Carla Cola	6.0	198
Sunny Puddles	5.4	56
Tier 3: Easy Groups		
Citroen C3	1.6	188
Mercedes CCC	1.0	206

Table 4.1: Illustrative table of interventions, ordered by percent of times they were involved in a high magnitude δ edit. Section 4.7.3 suggests our cutoff resides between 6.0 and 6.4.

10 weather attributes were each continuous, we selected these vectors from 15 weather choices during training and so, with regards to the encoding, they each corresponded to discrete choices. Because the camera intrinsics were drawn from the (realistic) discrete Nuscenes distribution, their encoding was also discrete.

The agent tokens had a set order: discrete type (blueprint), then continuous (x, y, z) locations, then (roll, yaw) rotations. To discretize the locations, we first subtracted their minimum possible value. The resulting $v \in [0, 600)$ was encoded with $w_0 \in [0, 5]$ for the hundreds place, $w_1 \in [0, 99]$ the ones, and $w_2 \in [0, 9]$ the decimal, so $v = 100w_0 + 10w_1 + 0.1w_2$. This small precision sacrifice marginally impacted scene reproduction. We encoded rotation similarly, albeit in $[0, 360)$.

4.7.2 Interventions

In this section, we investigate the relative ordering of groups by the MLM, where the order is determined by the degree to which that group is involved in a detrimental intervention.

Table 4.1 shows selected ordered results from the intervention procedure described in Section 4.5. We performed the procedure on $N = 10000$ test scenes G_k where our ϕ was an 18C model trained on the base 10000 subset from Town03 and $\tau = 0.2$. We additionally filtered the groups to those that occurred at least 20 times in the procedure.

On the left side we see the intervention taken, for example changing a single agent type to a Cybertruck (a large truck made by Tesla) or changing the weather such that it is now sunny with reflective puddles. The second column shows the percentage of scenes that the intervention produced a $\delta \geqslant 0.2$. We include both when the change was *to* that target and the delta was negative as well as when it was *from* that target and the delta was positive. The last column in the table reports how many times in total this intervention occurred in the 10000 scenes.

Summarizing the table, we find that a handful of asset switches appear to be detrimental for the model according to this metric. Small bikes had an outsized effect, as did cloudy weather and the rotations where a car faced the ego agent or turned to the left. Just after the last bike are two large vehicles, the Cybertruck and the Cola Car. The specificity of the weathers and rotations are because they are translations of our discretization. Practically, there is a range of rotation and weather values around the group that would all suffice. Finally, we do not include location results in the table because the MLM frequently re-positioned the asset outside the camera's view. This said more about the asset than it did about the location and was rife with confounders based on what was behind that asset. We could have localized the location interventions more by masking MLM options, but leave that for future work.

4.7.3 Analysis of single-step interventions

After obtaining candidate groups from the designed interventions, we investigated the effect of modifying the data sampling procedure to increase the prevalence of these groups by building and evaluating datasets sampled from the MLM training set. For asset groups, for each datum, we uniformly sampled $n_v \in [3, 6]$ vehicles selected from the scene. We then randomly chose vehicles $v_0, v_1, \ldots, v_{n_v}$ in that scene, including vehicles that may not be in the camera's purview, and changed them to be the target group. So as to not accidentally introduce a bias through the random process, we selected the same vehicles v_k for all group datasets. For rotation groups, we chose those same vehicles but rotated them to be the target rotation instead of switching their asset. For weather groups, we changed those scenes to have the target weather instead.

Does our method correlate with AP score? Figure 4.10 shows evaluation results on these groups when training 18C4 on four disjoint 10000 sized subsets of the data. The models performed best on the IID data from Town03 and just a little bit worse on the same from Town05. Further, they did exceptionally well on those two datasets, validating that they were trained sufficiently. The group results are mostly in line with our expectations from the interventions - the models did well on Citroen and Mercedes, poorly on the rotations, and terribly on the bikes. There is a large jump from the reasonable results on ColaCar and SunnyPuddles to the mediocre results on Cybertruck, which is directionally correct per Table 4.1. However, the strong results on CloudyDark are surprising.

Summarizing, if the threshold for choosing a group is between 5.5% and 6.5% and we focus

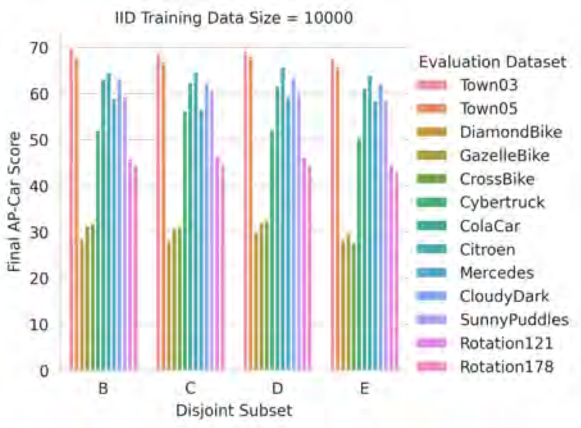


Figure 4.10: Test results with config 18C4 when training on disjoint IID subsets. Results are consistent, suggesting that the harder groups - bikes, rotations, and cybertruck - are ubiquitously hard.

on interventions affecting vehicles directly (rotation and type), then our method correlates well with empirical results. We have likely not found the exact causes plaguing the model, but we have narrowed them greatly. The model's regression when changing a car to a bike may be because it performed poorly on bikes. It may also be because the car was occluding another vehicle or that it itself was not occluded. This is especially true in light of the weather results suggesting that weather is not a conclusive factor. Finding the exact cause is difficult, even in simple settings [6]. We leave such improvements for future work.

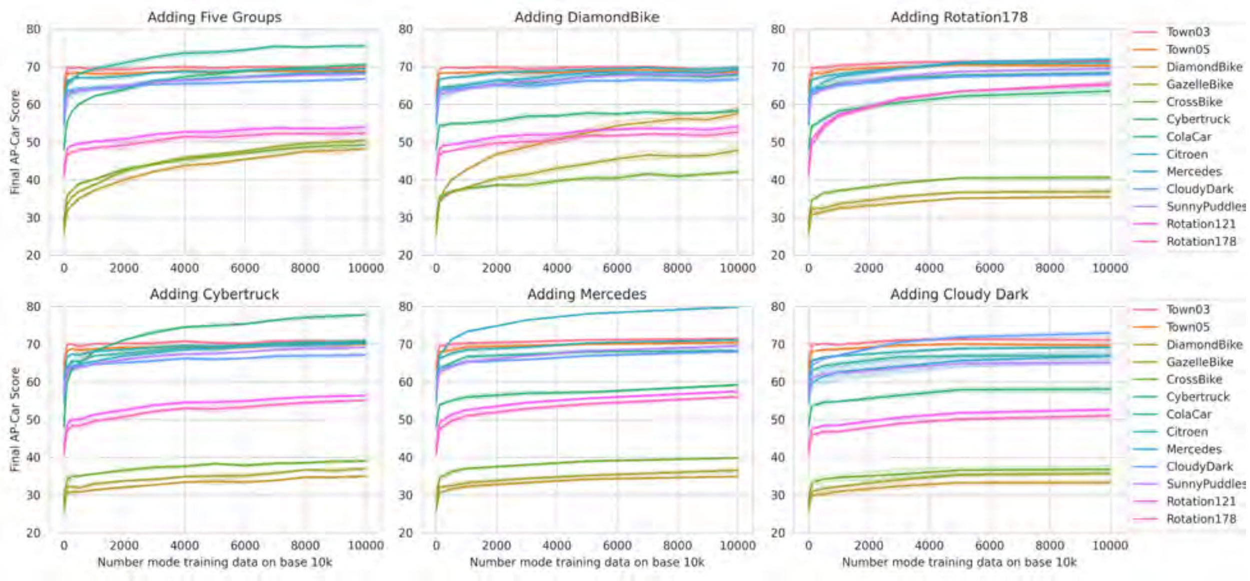


Figure 4.11: Results of training 18C4 on the base IID 10000 training set plus additional group data. The five groups in the top left (Cybertruck, Cola Car, Diamondback, Gazelle, and Crossbike) were added equally. For all charts, adding any one group improved all of the other evaluation scores, and at no point did we lose efficacy on the IID data as a whole. Figure 4.20 (Supplementary) zooms in on the initial jump.

Can we address these issues by increasing capacity? Recent papers [214; 10] suggest that scaling our models will improve results. An affirmative answer would mean we would not need to collect more data. The left side of Figure 4.12 suggests a negative answer when testing over a range of model sizes and types. We see that no model was distinguished with respect to their final values when training on 10000 IID examples.

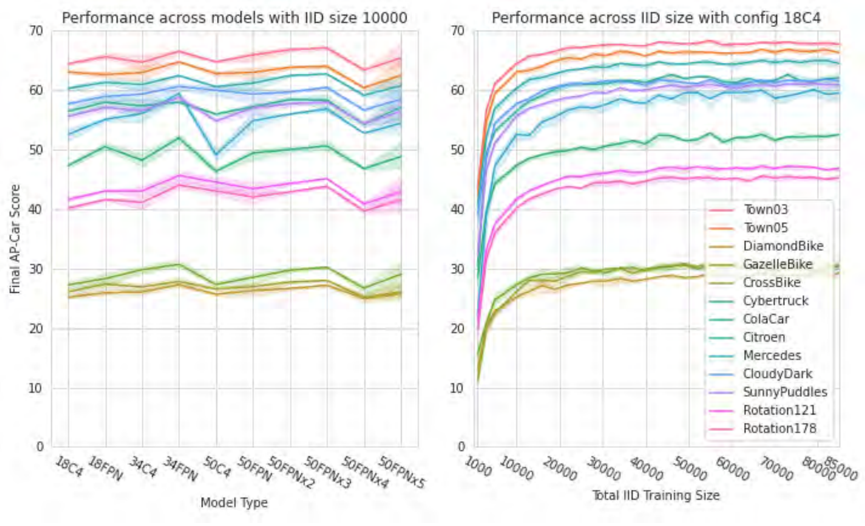


Figure 4.12: Independently increasing the model capacity (left) and increasing the data size (right). No model distinguished themselves and we quickly taper in how effectively the model utilizes the data. We consider the dip in the capacity chart to be an artifact of the training procedure and using the same settings for all models.

What if we increased IID data? This is preferable because IID data is easier to collect than group specific data. The right side of Figure 4.12 suggests this will not be sufficient. Test efficacy on town and group data jumped from 1000 to 10000 IID examples, but then slowed precipitously. Figure 4.19 (Supplementary) affirms that this is unlikely to change by suggesting that the percentage of representation of the group is what matters, rather than absolute count.

What if we increase data and capacity simultaneously? Results remained negative, as seen in Figures 4.13 and 4.21 (Supplementary). The left graphic in Figure 4.13 evaluates all models on 85000 examples and the right one shows results for just the 34C4 model across a range of IID data counts. First, observe that all of the models have similar evaluation scores. Second, they all struggled on the harder groups. And third, as seen more clearly in Figure 4.21, more data yielded a small accretive effect. All else equal, adding data may be better than adding model capacity.

Using group data We expect that adding data from the groups to the training set will improve performance on that group. The top left plot in Figure 4.11 confirms this. We added an even

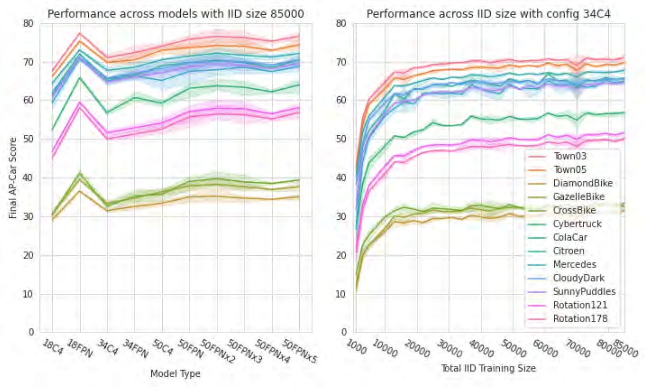


Figure 4.13: Increasing both data and model capacity at the same time. The left side ranges over model capacity with maximum IID data size (85000), while the right side ranges over IID data size with a bigger model - 34C4.

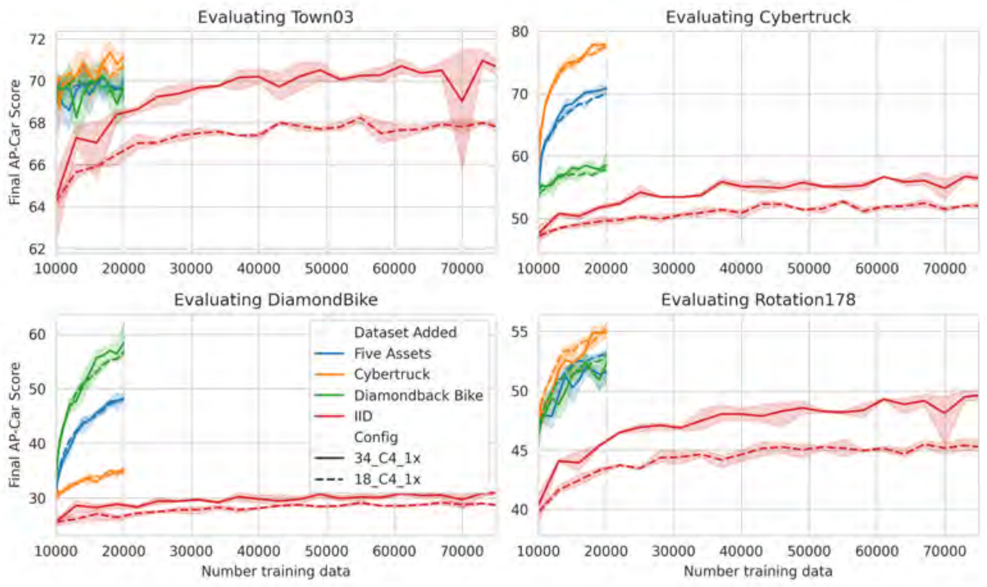


Figure 4.14: How much IID data is required to match a small amount of extra hard group data. Top left shows 20000 more IID data was required to reach par on IID with 250 group data. Bottom left shows that we never reached the same level on Diamondbacks with IID data as with adding Cybertrucks, let alone actual bikes.

amount of each group to the base 10000 IID subset and see that every group improved without impacting the Town03 and Town05 results. The other plots in Figure 4.11 show what happens when we add in training data from any one group M. This predictably improved the model's results on M's validation set. It surprisingly also improved results on *all* of the other M' and the Town data. The improvement to M' is smaller than that to M, but it is notable. The gains for a specific group were more pronounced for like groups - adding data from a biker group (Diamondback, Omafiets, Crossbike) improved the other biker groups more than adding data from the heavy car groups (Cybertruck, Colacar), and vice versa. Adding rotation groups helped ubiquitously albeit not as much as adding a bike group did for the other bikes. The least effective fix was adding the CloudyDark weather mode. Figure 4.19 shows that these trends persisted for a base of 85000 IID data as well.

4.7.4 Comparison with random interventions

As we alluded to in Section 4.5, taking random interventions is problematic because whether the group is reasonable for the distribution will be a confounder. We wish to prioritize the found groups to be those that are more likely seen in the wild. We show here that this is true by taking the 10000 source scenes used for the MLM interventions and applying random manipulations of the same type. For example, if we changed agent a_j's vehicle type in $G_k \rightarrow G_k^{\text{MLM}}$, then we changed a_j to a random vehicle type in $G_k \rightarrow G_k^{\text{Random}}$.

Table 4.2 shows results for random and MLM interventions over the same assets from Table 4.1. Observe that the assets were ordered incorrectly with CarlaCola higher than both Cybertruck and Kawasaki Bike. Random also had a higher percent of high magnitude threshold events; In general, 13.2% of random interventions impacted the model versus 10.2% of MLM interventions. We hypothesize this is because random re-sampling of elements of the scene graphs corresponded to sampling from a data distribution that does not faithfully represent the original training distribution. A 3% difference is large with respect to how much extra work would be required by humans combing through the data for plausibility and whether to include in retraining.

Figure 4.15 shows density plots for rotation and cloudiness interventions, conditioned on the intervention having been detrimental. We use density plots to demonstrate the differences between Random and MLM because these interventions are continuous for Random. For rotation, there was a mostly steady plateau for Random while MLM showed a clear single group aligned with the bi-modal humps in Original. For weather, Original and MLM were almost overlapping and, while Random was similarly bi-modal, its shape was less pronounced and more even as expected. These both reinforce our claim that the advantage of MLM is that it gears us towards higher priority groups to fix that are in line with the actual data distribution.

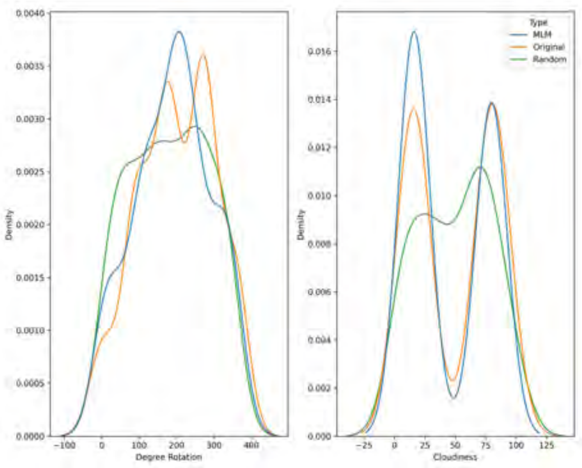


Figure 4.15: Comparing rotation and weather results for MLM and Random intervention strategies. MLM aligns with Original much better than Random does. Further, Random has a much wider berth of possible problematic modes, a concern given practical limits to model capacity and data budgets.

Intervention	MLM	Rand	Rand Total
CrossBike	16.5	19.0	126
GazelleBike	18.9	18.7	171
DiamondbackBike	24.4	15.8	152
Carla Cola	6.0	8.1	210
Cybertruck	6.4	6.8	176
KawasakiBike	6.5	6.6	121
Citroen C3	1.6	3.5	197
Mercedes CCC	1.0	3.8	183

Table 4.2: Results for MLM and Random asset intervention strategies, ordered by the percent of times that they were involved in a high magnitude δ *random* event. While the top three are the same, Random flubbed the dividing line by placing a) Cybertruck above Kawasaki and b) Carla Cola well ahead of both. Its failure rate for the easy cars was much higher and, in general, posited 3% more failures than MLM. All told, its results created more need for human verification and testing and reduced the degree of automation that we could employ to find hard groups.

4.7.5 Comparison with cause-agnostic data collection

We saw in Figures 4.11 and 4.19 (Supplementary) that adding group data into training not only addresses the issue for that group but even improves the performance on other groups. The cost is that we have to perform the entire described procedure to find our interventions and *then* cast a net for data of those types in order to retrain the model. An important baseline comparison would be to find data instances where the model performs poorly on the aforementioned scoring function (Section 4.5) and retrain by including those alongside IID data. This approach, which we christen cause-agnostic data collection, would save us the need to take interventions or gather type-specific data to retrain.

Figures 4.16 and 4.22 (Supplemetary) show grids of results with this approach, respectively for each of our two configurations, covering four threshold values - 0.2, 0.4, 0.6, and 0.8[4]. We test all thresholds because we do not know which will be best a priori. We then randomly draw 150000 IID scenes, test on these scenes, and filter into buckets based on whether the resulting score was less than the given threshold. We randomly choose 10000 scenes from each bucket and add them in tranches to the original 10000 IID data training set.

Observe first that the model's performance increases across the board with this data. For example, on the bikes, which were the most challenging groups, the model increases from below 30 to hover around 40 as more data is added. Next, as expected, the 34C4 model is a bit better than the 18C4 model for all thresholds. Third, as the threshold increases, the results improve. One hypothesis why is because the lower threshold datasets have fewer annotations and consequently emptier scenes than the higher threshold datasets.

Most importantly, how does this compare to our proposed approach? The best results for this baseline are found in threshold 0.8. Compared against the first chart in Figure 4.11 - 'Adding Five Groups' - we see that the IID Town03 and Town05 results are about the same, the easier

[4]We did not include results on thresholds below 0.2 because they showed the same trend.

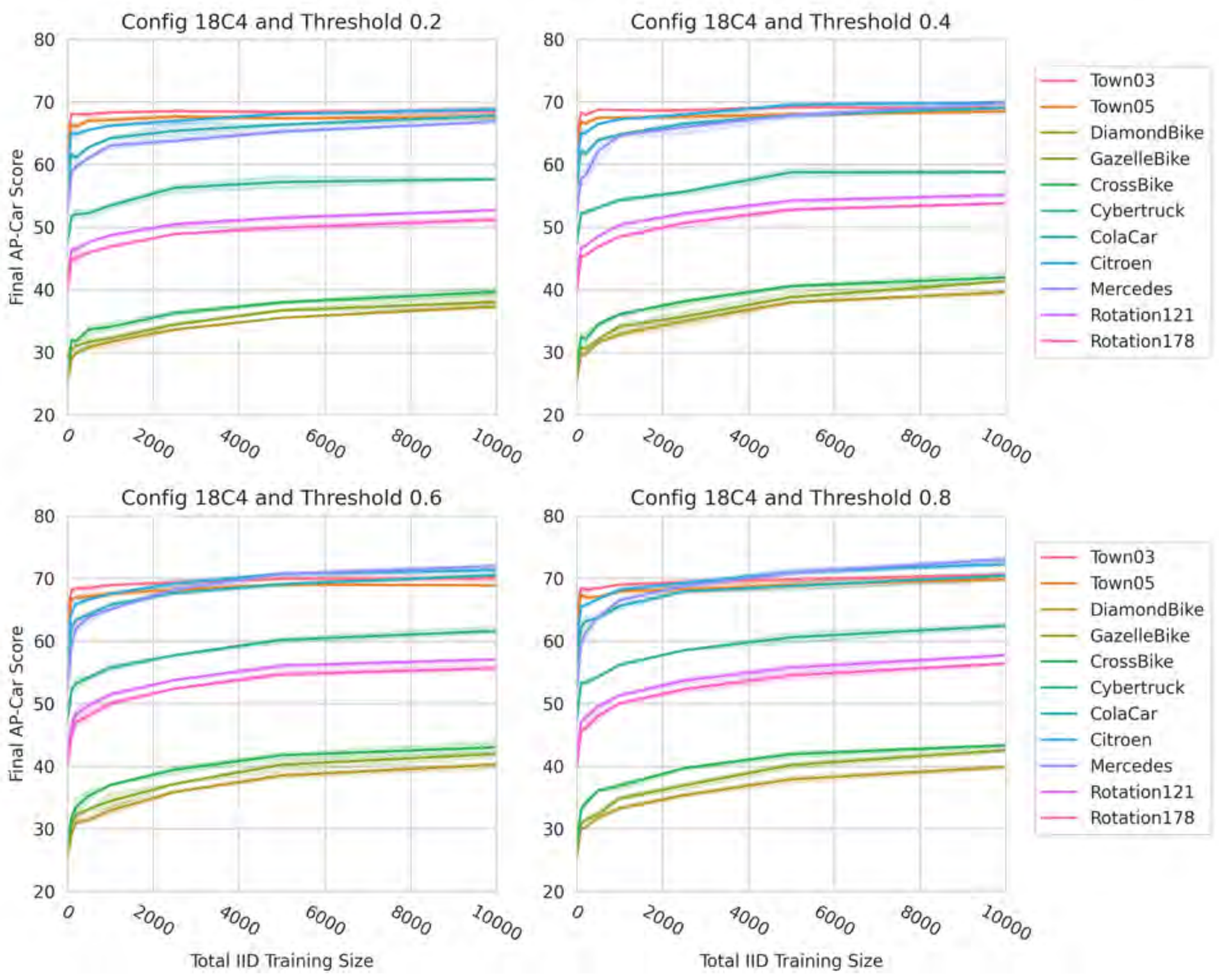


Figure 4.16: Baseline cause-agnostic data collection results. We train 18C4 on the original IID 10000 training set plus additional cause-agnostic data. The latter is chosen by first selecting a threshold from [0.2, 0.4, 0.6, 0.8], then randomly selecting simulated data for which the model gets at most that score using our scoring function from Section 4.5. The graphs suggest a slight AP increase as the threshold increases, likely because lower threshold scores lean disproportionately towards fewer annotations and emptier scenes. Comparing these results with Figure 4.11, we see that this baseline is comparable for arbitrary groups, like the Rotations, but unsurprisingly much worse for data-specific improvements. For example, the first and second charts of Figure 4.11 show that our method achieves much higher gains in the bike classes.

classes (Mercedes and Citroen) slightly surpass our strong results, and the Rotation results are better than ours (high 50s versus low 50s). However, for the classes where we actually add data, our method's results are much better than the cause agnostic results. For example, the most challenging groups - the bikes - reach only an AP score of 43 with cause-agnostic collection but go above 50 with our method. This is not surprising as adding group-specific data *should* boost the performance. In this light, our method's advantages over this baseline are clear. First, we can ascertain which of the groups are actually problematic. This is no small feat; without our method, we would not have actually known which groups to track when performing cause-agnostic data collection. And second, we still produce a large gain over cause-agnostic data collection when we add in group-specific data. That this effect is even more pronounced for the challenging groups suggests that our method is integral for understanding on *which* groups we should spend the additional capital necessary to produce representative datasets.

Why do these groups exist? With causal groups in hand, we can ascertain why our models failed: The bikes are underrepresented in Nuscenes; The model rarely saw turning cars (Rotation 121) due to the town layout; The model rarely saw cars facing it (Rotation 178) due to the traffic policy and car quantity; The large cars cause occlusion labeling issues, Cybertruck more so than Cola car. Without the groups, these issues can only be hypothesized.

4.7.6 Analysis of two-step interventions

We analyze what happens when we take a successive intervention step with the MLM to refine our causal understanding. We consider the following, where $\delta_{kj} = f(\phi, I_j, L_j) - f(\phi, I_k, L_k)$, the change in the model's efficacy from when it evaluates scene k to when it evaluates scene j.

1. Which second steps are detrimental to the one-step edited scene with threshold of $\tau_2 = 0.2$? This assesses which refinements are impactful to first edits that have a minor effect. Here, $\delta_{10} \geq \tau_1 = 0.2$ and $\delta_{21} \geq \tau_2 = 0.2$, which together imply that $0.8 \geq \delta_{10}$ because all $\delta < 1$.

2. Which pairs are detrimental to the original scene with a threshold of $\tau_2 = 0.2$, regardless of the first step's result? This is assessing which pair of refinements are most worth exploring. Here, $\delta_{20} \geq \tau_2 = 0.2$.

3. Conditioned on the one-step scene passing a threshold of $\tau_1 = 0.2$, which two-step scenes are as bad, i.e. they pass a threshold of $\tau_2 = 0.0$[5]? Here, $\delta_{21} \geq 0$ and $\delta_{10} \geq \tau_1 = 0.2$.

So that the search space is not prohibitively large, we limit the possible first step we take to be uniformly randomly chosen from a set J that we previously analyzed and which represent a wide cross section of the challenging interventions - $J = \{$Diamondback Bike, Gazelle Bike, Crossbike, Cybertruck, Carla Cola, Cloudy Dark (CD), Sunny Puddles (SP), Rotation 178, Rotation 121$\}$. We further limit the second step to be from a different category than the first, e.g. if the first choice was an asset change, then the second step must be either a random rotation or weather change. This second step is performed similarly to how we did the original interventions, albeit $N = 60000$ times instead of 10000 . After producing these scenes, we then score them on the same 18C4 model trained on the base 10000 subset from Town03.

Results in Table 4.3 address each question. For Question 1, the small vehicles are again the most problematic interventions, with four bikes, the Isetta (small car), and the two motorcycles (Harley and Yamaha) all in the top eight. After Rotation 10, which is a new addition, there are no second edits for which at least 9% pass the threshold. Because this question requires that the first intervention was not (too) detrimental - otherwise the second intervention would not be able to pass the $\tau_2 = 0.2$ threshold - that these results are similar to the prior results in Table 4.1 is not surprising.

For Question 2, we see very high probability detrimental pairs. Additionally, the first time a non-weather appears as the initial intervention is not until index 113. That the weathers are appearing first is explainable by there being only two weather options possible in the first intervention (by fiat), which makes it easier for them to be selected first than asset changes. There are many more weathers possible in the second intervention, and so any one of them has a hard time distinguishing itself, which makes it challenging for a (rotation, weather) or (asset, weather) pair

[5]We do not report where strictly $\tau_2 > 0.0$ because those results disproportionately tilt towards scenarios where the first edit was only somewhat detrimental. This is because if the first edit brought the score to τ_2 or less, then the second edit would not pass muster regardless of its efficacy.

Intervention	Percent > $\tau_{1,2}$	Total
Question 1		
GazelleBike	21.4	229
CrossBike	20.2	282
Carla Cola	18.9	355
DiamondbackBike	18.3	218
BMW Isetta	16.9	438
KawasakiBike	16.3	282
Harley Davidson	16.2	321
Yamaha YZF	15.6	257
Rotation 10	13.2	120
Question 2		
(CD, KawasakiBike)	62.5	72
(SP, Harley Davidson)	61.5	91
(SP, Rotation 30.0)	60.4	53
(SP, KawasakiBike)	60.3	73
(SP, Isetta)	58.7	121
(SP, Grand Tourer)	56.6	129
(CD, Harley Davidson)	54.3	92
…First non-weather starting edit at 113 …		
(Cybertruck, Rotation 160)	41.3	63
Question 3		
(SP, Rotation 162)	20.9	67
(Carla Cola, Rotation 79)	20.8	48
(CD, Volkswagen)	20.8	125
(CD, CrossBike)	20.3	64
(CD, Rotation 290)	19.8	81
(SP, Jeep)	19.4	103
(CD, DiamondBike)	19.3	57
(CD, Patrol)	17.8	107

Table 4.3: Illustrative table of second-step interventions, ordered by the percent of time that they were involved in a high magnitude δ intervention. See the 'What happens if we take another step?' paragraph in Section 4.7.6 for analysis.

to appear.

However, we are not actually sure why the probabilities are so high. They suggest that it is quite easy for a pair of interventions to confuse the model. Figure 4.17 suggests that the MLM is already off of the data manifold given that the second-step rotations it is choosing have such a different distribution than the selections we see in Figure 4.15. That being said, it is surprising to

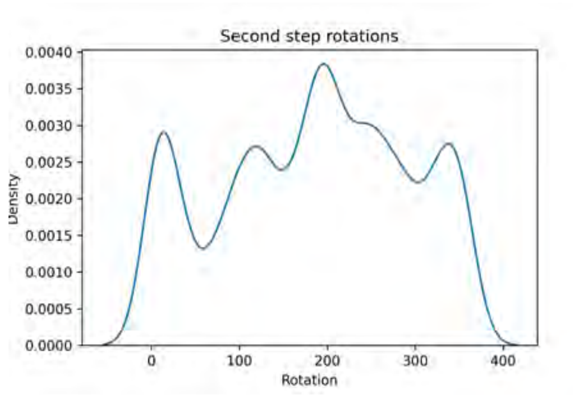


Figure 4.17: Rotation density plot for second interventions, conditioned on the intervention being detrimental. That the shape of this plot is very different from the MLM and Original plots in Figure 4.15 suggests that the MLM is applying a different distribution on the second intervention. In other words, it has already drifted.

us that making the weather sunnier and then changing an asset to a bike for example has such a detrimental effect.

Question 3 is asking which second interventions do not improve the score given that the first intervention was sufficient detrimental. We see a high concentration of first-step weathers in the top, but it is not as ubiquitous as it was in Question 2. While not shown, the results continue to have higher than 10% probabilities up to place 113, with an asset change usually mixed in in at least one intervention.

4.8 Conclusion

Combining causal interventions, MLMs, and simulation, we presented a novel method that finds challenging groups for a detection model in foresight by having the MLM resample scene constituents. These interventions help identify and prioritize groups with poor performance without humans in the loop. We demonstrate our advantage against a baseline using cause-agnostic data upon which the model performs poorly. Our approach is a significant step towards addressing safety-critical concerns in AV. Beyond AV, we think the associated will benefit the causality community because the current state of the art [101] involves static datasets with low complexity tasks.

Our method has limitations. We cannot yet apply it to real world data because we need full control over the scenes for the MLM to properly operate. Ost et al. [142] is a step towards overcoming this concern. Until then, the so-called sim2real gap [165; 86] is ever-present. Another limitation is that while we do show compelling results when taking a second step, these results also suggest that the MLM is already drifting from the data distribution and so its utility is reduced. In light of this, we do not expect our method to continue to work for many steps without further research because the samples will inevitably drift from the data distribution. Intervening multiple times is necessary for understanding complicated causal interactions. Each of these two limitations are of course also potential future directions. A final one is understanding better why many groups improved when adding a single group, which remains a compelling question.

4.9 What did we learn about the most challenging cases?

We learned that we could come up with methods that capably identify what are the challenging cases well before they are illuminated through hindsight analysis. We did this by proposing and verifying a strong approach for causal understanding of what are the challenging cases. We then pushed on our method to deepen our understanding of the causes and found that further interventions are likely no better than random interventions in terms of improving the model. The result is that while our initial approach is sound, more research is needed to extend it to finer causal understanding.

While our approach is applicable to many domains, we focused our study on applications in robotics systems, specifically object detection in the service of autonomous vehicles. Figure 4.18 displays how it fits into the machine learning application framework from Section 2.3. This contribution addresses the data loop and lets us better understand the root causes of the model's problems, which improves our data collection efficiency. It does this by letting us better prioritize what data to collect in order to most improve the model each time we traverse the framework.

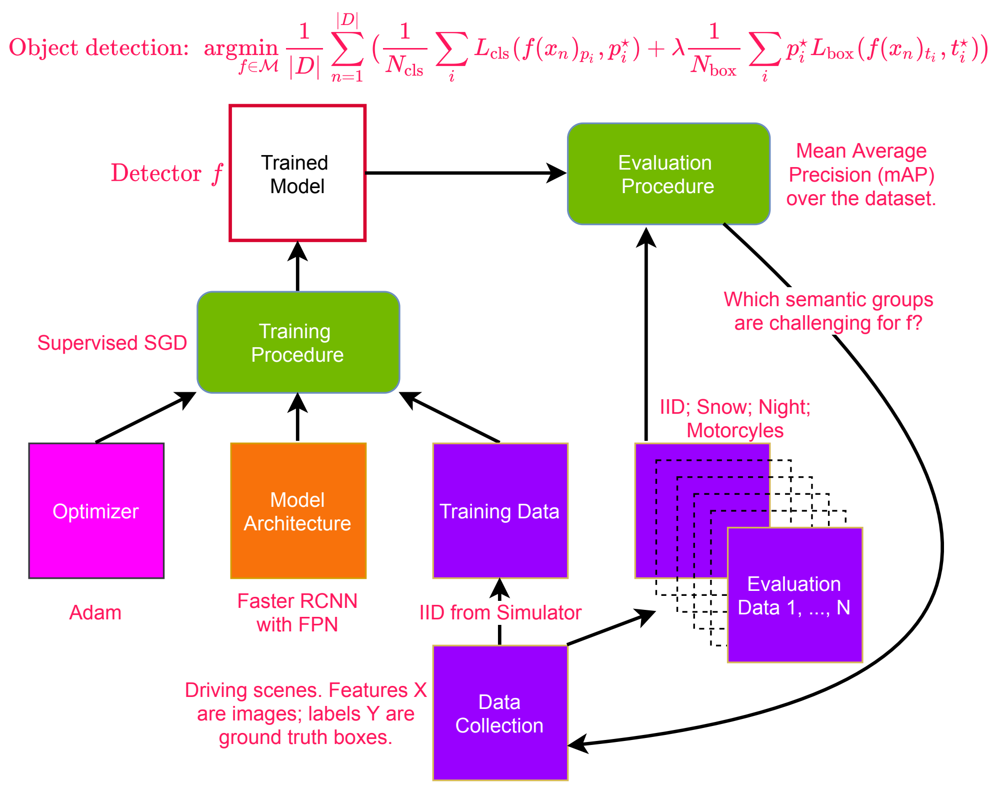

Object detection: $\operatorname{argmin}_{f\in\mathcal{M}} \frac{1}{|D|}\sum_{n=1}^{|D|}\left(\frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}(f(x_n)_{p_i}, p_i^\star) + \lambda\frac{1}{N_{\text{box}}}\sum_i p_i^\star L_{\text{box}}(f(x_n)_{t_i}, t_i^\star)\right)$
Detector f
Trained Model
Evaluation Procedure
Mean Average Precision (mAP) over the dataset.
Which semantic groups are challenging for f?
Supervised SGD
Training Procedure
IID; Snow; Night; Motorcyles
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Adam
Faster RCNN with FPN
IID from Simulator
Driving scenes. Features X are images; labels Y are ground truth boxes.
Data Collection

Figure 4.18: The Machine Learning Application Framework for this contribution. Observe that the mAP scores are computed over each evaluation dataset. This lets us prioritize what data to acquire next, thereby improving the efficiency of the whole process because each loop capably improves the model.

Finally, this contribution is vital towards building a communicative and **visually aware** robot that can capably learn new tasks. This is because the status quo approach relies on hindsight understanding of errors in the wild. That is unsatisfactory given the reasonable demands of having robots in every locale across the globe. In order for them to understand the visual scene, they either will need to have much more capable online learning or will need means to predict where they will likely fail so that we can fix them before they do. The former, online learning, is notably challenging in modern machine learning with trillions of parameters and massive datasets. The latter is the approach that we take and about which we are very excited.

4.10 Supplementary

4.10.1 Table of notation

Symbol	Description
ϕ	Detection model
x	Scene
G	Scene graph
$d \in \mathbb{N}$	Number of objects in scene
$S \in \mathbb{N}^O(d)$	Sequence encoding of scene graph
$I \in \mathbb{R}^3$	Scene image
$L \in \mathbb{R}^{4\times d}$	Scene bounding box labels
$l_k \in \mathbb{R}^4, k < d$	The bounding box of the kth object
R	Scene generation process
$p_R(x)$	Distribution over scenes
$f : (\phi, I, L) \to \mathbb{R}$	Per-example scoring function
$\delta \in \mathbb{R}$	The change in score by intervention: $\delta = f(\phi, I', L') - f(\phi, I, L)$
$\tau \in \mathbb{R}$	Threshold value for classifying interventions as detrimental

Table 4.4: Table of notation

4.10.2 Dataset details

CARLA does not spawn agents that collide with the environment, even the ground. To ensure agents are grounded, for any agent spawn collision, we increase its Z coordinate and try respawning. This allows us to place every agent on the map, albeit some of the conflicting agents have to 'drop' from above, and consequently we wait for 50 timesteps so those agents can settle. In that duration, the autopilot policy guides the agents to satisfactory positions. After those 50 steps, we then record for another 150 steps and save every 15th frame. The resulting episodes each have ten frames with an initial distribution influenced by Nuscenes and CARLA, and a traffic policy influenced by CARLA.

We found the existing suggested approach for getting 2D ground truth boxes lacking because it frequently has trouble with occlusions and other challenging scenarios, so we developed the following heuristics to help filter the boxes. While not airtight, the resulting ground truths were qualitatively more reliable.

- Filter Height: We require that the final $2d$ box is at least 30 pixels. This is in between the easy (40) and medium/hard (25) settings on KITTI [66].
- Max Distance: We require that the ground truth detection not be more than 250 meters away. We enforce this through the use of a depth camera attached to the ego agent.
- Visible Pixel Percent (VPP) and Min Visible Count (MVC): The 2D box is attained by pairing the 3D box with the camera's calibration. With the latter, we get the closest point P to the ego agent. We then get the depth camera's output at the 2D box. VPP asks what percent t of that box is closer than P and filters it if $t \geqslant 80$, ensuring that at least 20% of the object is not occluded. MVC asks how many pixels q are further than P and filters it if $q < 1300$, ensuring that the occluded object is big enough.

4.10.3 Supporting charts

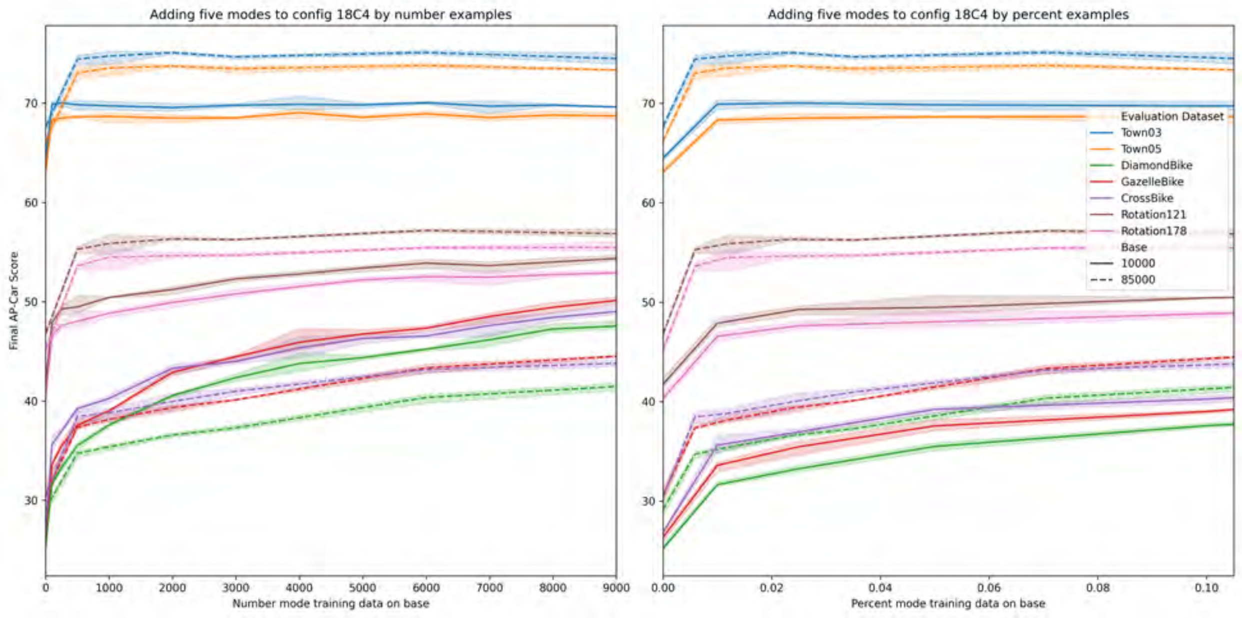


Figure 4.19: Performance of 18C4 on select test sets when adding mode data from the three bikes, the ColaCar, and the Cybertruck on top of either 10000 or 85000 base IID data. Towards improving the results, these two charts show that it is not the absolute count of the mode data that is important but rather the percent of it relative to the IID data. We see that in how the trendlines for the two bases are only consistent in the percent chart. The other modes are not shown for clarity but it holds in general.

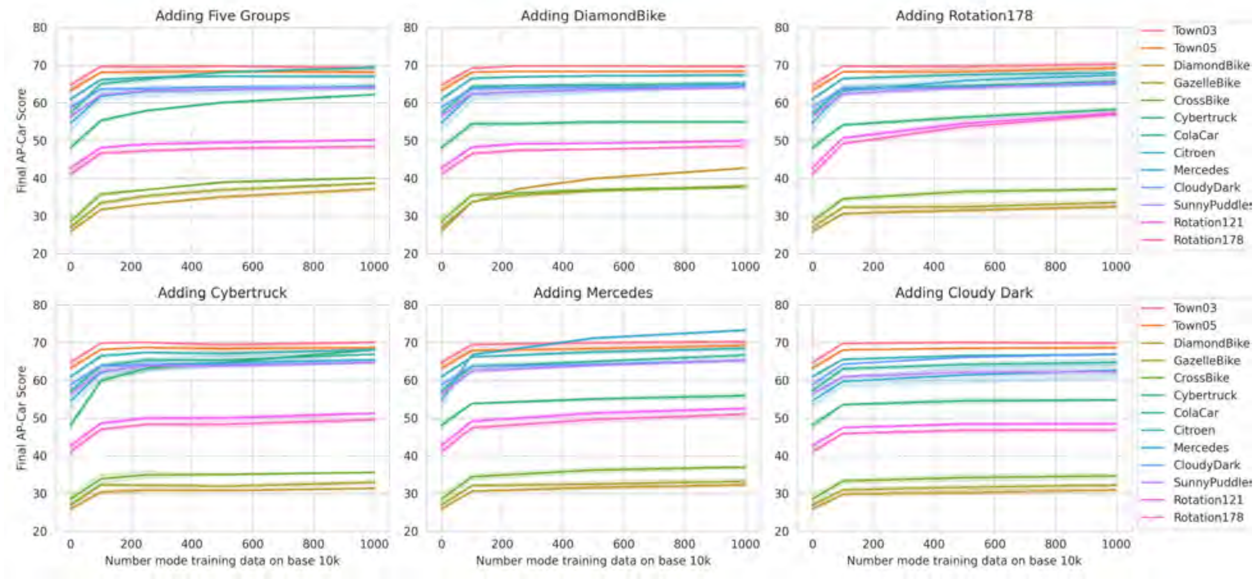


Figure 4.20: Results adding mode data to the base IID 10000 training set. This is the same as Figure 4.11 but zoomed into just [0, 1000]. The five modes in the top left are the Cybertruck, Cola Car, Diamondback, Gazelle, and Crossbike, each added in equal proportion.

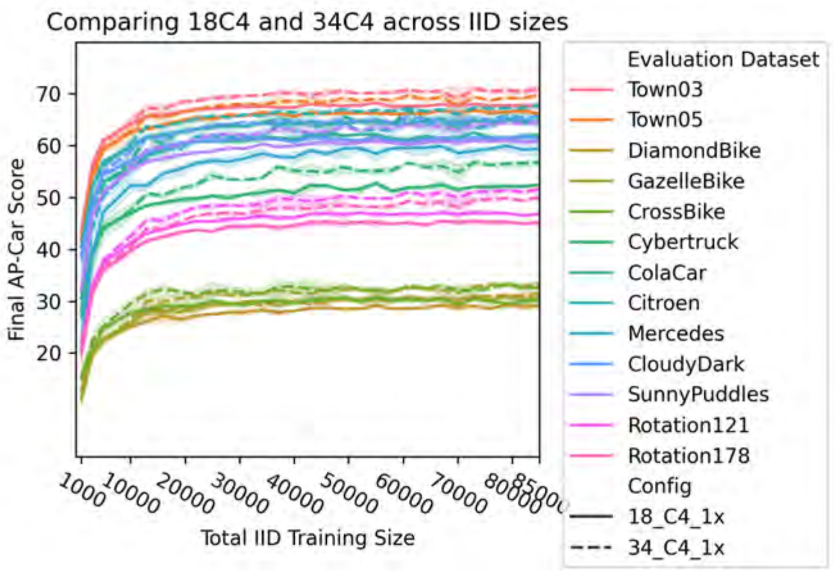


Figure 4.21: We can see that the model size does matter in that for every group the 34C4 model improves over the 18C4 model. However, the increase is quite small and the data quality and quantity appear to matter much more.

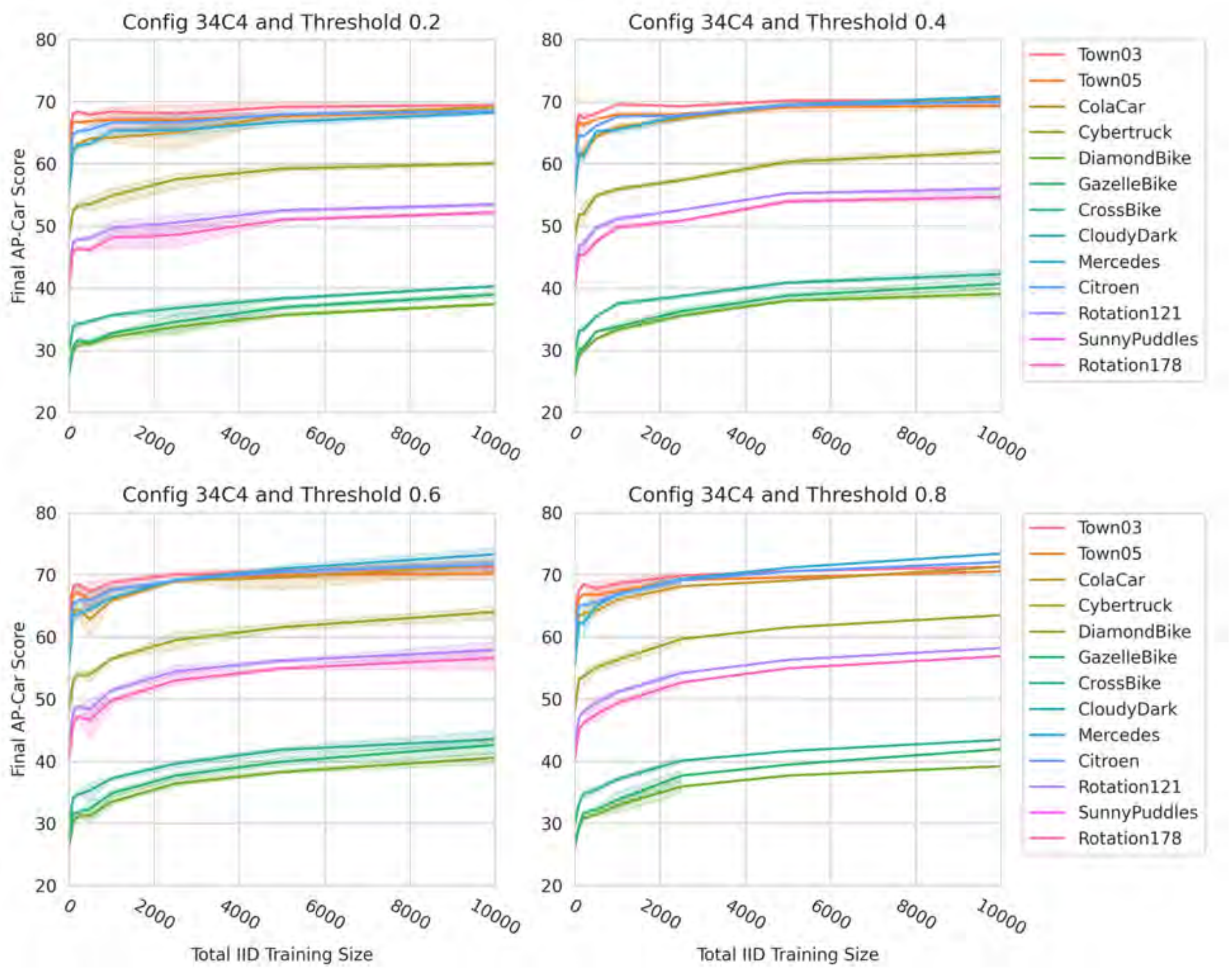


Figure 4.22: Baseline results training 34C4 on the base IID 10000 training set plus additional cause-agnostic data. As specified in Figure 4.16, the additional data is chosen by first selecting a threshold from [0.2, 0.4, 0.6, 0.8], then randomly selecting simulated data for which the model gets at most that score using our scoring function from Section 4.5. This graphic is included for completeness - the results align with what we expect in that they are a little bit better than when using Config 18C4 for the same task and that they are worse than when performing our proposed method.

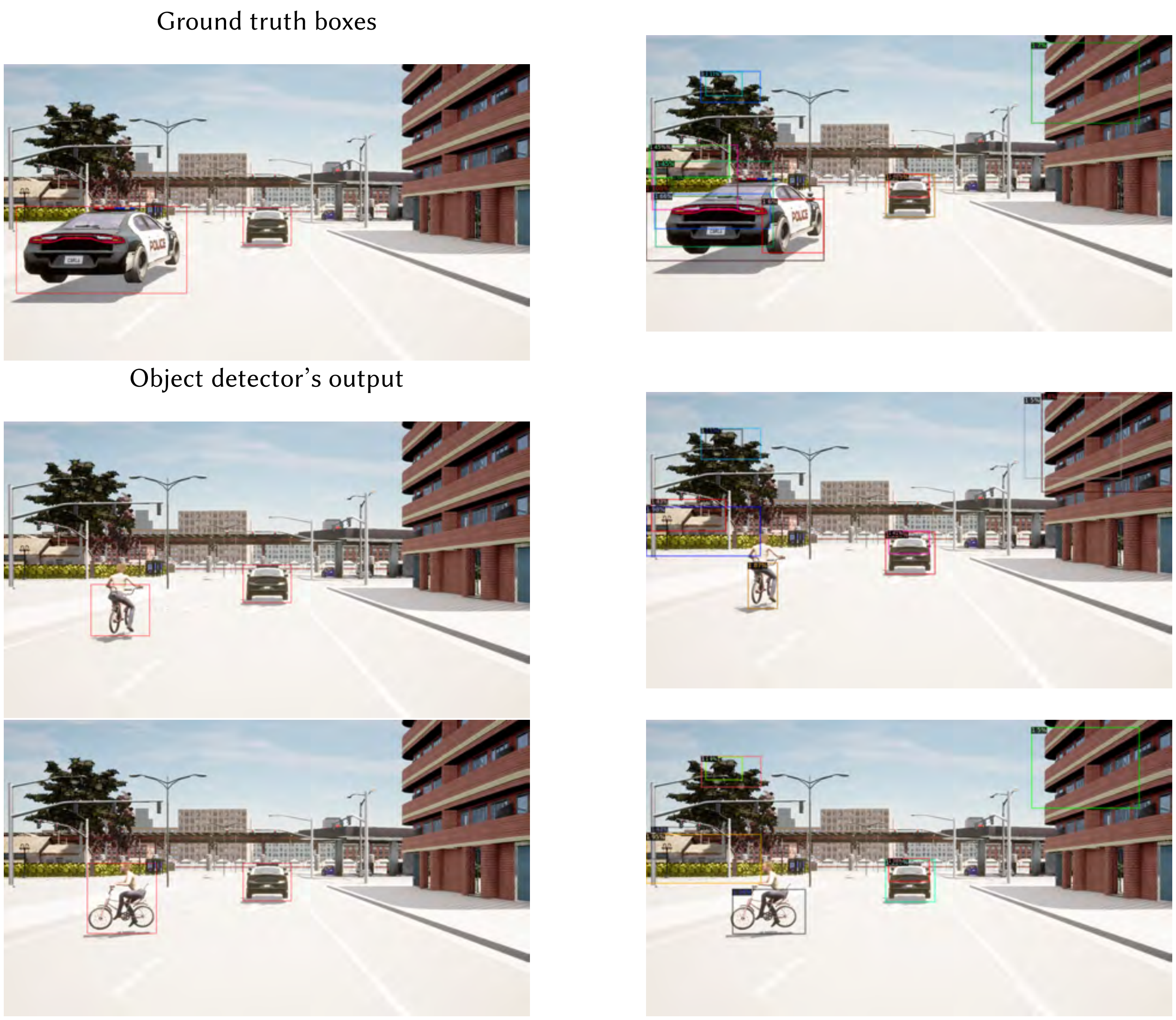


Figure 4.25: Interventions taken by the MLM. The first row is the original scene, the second after an intervention changing the police car to a biker, and the third after an intervention rotating the biker. The left side shows ground truth and the right shows model predictions. The model's predictions were very good for the first scene; in the second scene, it preferred a blank space on the left side to the biker, although the biker did get an 87% confidence. After rotating the biker, that confidence reduces to 34% while the model still hallucinates a vehicle on the left side with 95% confidence.

Part II

Addressing the Most Challenging Cases

5 | Episodic Reinforcement Learning

5.1 Introduction

As discussed in Section 2.1, reinforcement learning traditionally assumes that an agent is learning in an episodic Markov Decision Process (MDP) consisting of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ representing the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $\mathcal{R}$, the discount factor γ, and the transition probability function $\mathcal{P}$. That it is episodic means that there is a limit on the number of steps that we allow the agent to learn before resetting the environment back to an initial state. Most commonly, that state is a fixed special $s_0 \in \mathcal{S}$. To remain Markovian, episodic MDPs must include the step count in their state. In practice, while training agents we do not include this detail and, further, we also ignore the last transition that occurs when we reset the environment from $s_T \longrightarrow s_0$.

The goal is to learn an optimal policy function $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ that expresses the distribution over actions from a state. Frequently, algorithms also learn an accompanying value function $v : \mathcal{S} \longrightarrow \mathbb{R}$[1] that expresses the perceived discounted sum of rewards starting in that state. Formally, the value of a state $s \in \mathcal{S}$ when using policy π is given by:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| S_0 = s, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\right] \tag{5.1}$$

[1]The range of v is often restricted further to $[0, 1]$.

Value functions are widespread even when they have no bearing on inference, such as in popular algorithms like Proximal Policy Optimization [173]. This is because they help stabilize the expected return computation via Generalized Advantage Estimation [172].

While it is not a requirement to use episodic reinforcement learning when training agents to learn tasks, it is a baseline approach for other methods because it directly expresses the concrete motivation of an agent having to decide what action to take in a given state in order to attain the highest task-driven reward.

5.2 The most challenging cases

The nature of this learning means that there is a heavy bias towards the initial start state. Consider the task of opening a tin can. If we always start our agents with the can opener just to their right, then starting it inside of a drawer poses an unexpected challenge. This becomes especially concerning in environments with sparse rewards and long episode lengths. In those conditions, the agent rarely sees a reward and can struggle to update its understanding of the environment. Instead, it learns that almost everything near s_0 gives it zero signal. Without said signal, it becomes inordinately challenging to satisfy the credit assignment problem. As a result, the sample complexity to learn the long range dependencies necessary to the chain of events in a non-trivial task is too high.

A core problem is that the agent typically receives high reward only in states close to the goal state. Reward diminishes as we move farther away and, coupled with the sparse signal, the agent's ability to correct their behavior diminishes. Consequently, the error rate increases, and we commonly see groups of states where the error patterns are similar.

These groups are the challenging cases. With the aforementioned can opener example, one group $\mathcal{G}_k$ are the states defined by $\mathcal{W}_k$ deciding that the tool is in a drawer; the agent has to learn about that possibility. Another $\mathcal{G}_j$ is defined by $\mathcal{W}_j$ deciding that the tool is in the dishwasher. A third could be one that is traditionally found later in an episode - the can opener has been dropped on the floor. Each of these are semantically different states, or groups of data. Agents typically learn on a fixed start state, which is equivalent to having their initial start state $s \in \mathcal{G}_{S_0}$. This leaves them ill prepared when they encounter states that are off the beaten path, like those in $\mathcal{G}_j$ above. This is especially true with sparse rewards and/or long episode lengths.

5.3 Overview of our contribution

There are many solutions that have been posited to address this problem. One solution is to refrain from using reinforcement learning until after the agent has a good starting policy. That policy is often instilled with behavioral cloning [11; 164; 50] from an already existing policy that is satisfactory but not optimal. Human demonstrations are frequently used, such as in AlphaStar [196].

Another solution is to use curiosity-driven learning. In Pathak et al. [146], 'curiosity' is formulated as the error in an agent's ability to predict the consequence of its actions. That error is determined by the visual feature output from a learned self-supervised inverse dynamics model and compared against the actual visual ground truth, a surrogate measure for surprise. These approaches [29; 169] do not produce an optimal policy directly, but instead suggest a different route to exploration by first understanding the environment sufficiently and only then trying to produce a working policy for the task at hand.

We propose an alternative solution to address the most challenging cases. The idea is to create a curriculum for the agent via reversing a single trajectory (i.e. state sequence) of reasonably good, but not necessarily optimal, behavior. That is, we start our agent at the end of a demonstration and let it learn a policy in this easier setup. We then move the starting point backward until the agent is training only on the initial state of the task. We call this technique **Backplay** and display a graphical interpretation in Figure 5.1. As we show in Section 5.7, it vastly decreases the number of samples required to learn a good policy. Notably, the Backplay-trained agent is able to learn just as well in environments with a single sparse reward as in those with a dense shaped reward.

This solution is advantageous over traditional behavioral cloning because it does not require the demonstrator's actions at any step, but instead just the states. This allows for the Backplay-trained agent to design a policy whose efficacy is far greater than that of the demonstrator's policy, which is challenging to do with behavioral cloning.

It is advantageous over curiosity-driven learning because it directly optimizes the policy on the task we care about. Consequently, our resulting policy has higher efficacy when utilized at test time. Curiosity-driven approaches try to make up for this by being more robust to generalized downstream tasks, which is within our scope.

5.4 Backplay Algorithm

5.4.1 Formalization

We use the standard formalism of a single agent Markov Decision Process (MDP) defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$ which gives the probability distribution of the next state given a current state and action. The agent chooses actions by sampling from a stochastic policy $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$, and receives reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ at every time step.

The agent's goal is to construct a policy which minimizes its discounted expected return $R_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$ where r_t is the reward at time t and $\gamma \in [0, 1]$ is the discount factor, and the expectation is taken with respect to both the policy and the environment.

The final component of an MDP is the distribution $\mathcal{B} \in \mathcal{S}$ of initial starting states s_0. This is usually a fixed state $s_0 \in \mathcal{S}$. The key idea in Backplay is that we do not initialize the MDP

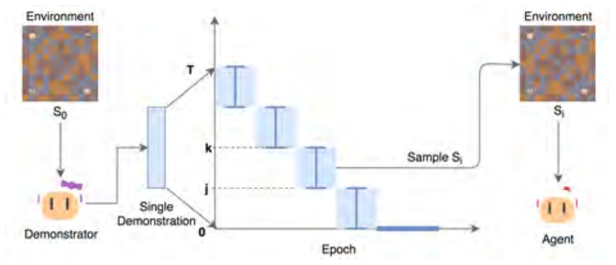


Figure 5.1: Backplay: We first collect a demonstration, from which we build a curriculum over the states. We then sample a state according to that curriculum and initialize our agent accordingly.

in only a fixed s_0. Instead, we assume access to a demonstration which reaches a sequence of states $\{s^d_0, s^d_1, \ldots, s^d_T\}$. For each training episode, we uniformly sample starting states from the subsequence $\{s^d_{T-k}, s^d_{T-k+1}, \ldots, s^d_{T-j}\}$ for some window $[j, k]$. Note that this training regime requires the ability to reset the environment to any state. As training continues, we 'advance' the window according to a curriculum by increasing the values of j and k until we are training on the initial state in every episode ($j = k = T$). In this manner, our hyperparameters for Backplay are the windows and the training epochs at which we advance them.

5.4.2 Quantitative Analysis

Next, we formally explore the conditions under which Backplay can improve the sample-efficiency of RL training. More specifically, we will derive an estimate for the sample complexity of a Q-learning agent trained with Backplay in a sparse reward navigation environment and we will show that it is significantly better than that of standard RL. Note that we simplify to a setup using a single starting state for each training stage instead of sampling from a window. In the notation from Section 5.4, $k = j$. One can verify that the general setup where $k > j$ is a more favorable regime, and therefore our approach provides an upper bound on the sample complexity.

Consider a connected, undirected graph $G = (V, E)$. An agent is placed at a fixed node $v_0 \in V$ and moves to neighboring nodes at a negative reward of -1 for each step. Its goal is to reach a target node $v_* \in V$, terminating the episode. This corresponds to an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$ with state space $\mathcal{S} \sim V$, action space $\mathcal{A} \sim E$, deterministic transition kernel corresponding to

$$P(s_{t+1} = j \mid s_t = i, a_t = (l,k)) = \delta(j = k)\delta(l = i) + \delta(j = i)\delta(l \neq i)$$

and uniform reward $R(s_t, a_t) = -1$ for $s_{t+1} \neq v_*$. Observe that the first term, $\delta(j = k)\delta(l = i)$, occurs in a normal setting where actions proceed from state to state. The second term, $\delta(j = i)\delta(l \neq i)$, accounts for a degenerate case where there are no valid actions and the agent remains in one state. These are disjoint due to the $\delta(l \neq i)$ and $\delta(l = i)$ terms.

Assume that π is a fixed policy on $\mathcal{M}$, such that the underlying Markov chain is irreducible and aperiodic:

$$K_\pi(s', s) = \sum_{a \in \mathcal{A}} P(s' = j \mid s_0 = s, a_0 = a)\pi(a|s_0 = s)$$

Note that $s' = j$ and $s_0 = s$ because we are evaluating K_π as if we are starting state s and trying to reach s'.

K_π has a single absorbing state at $s = s_* := v_*$. Our goal is to estimate the value function of this policy, equivalent to the expected first-passage time of the Markov chain K_π from $s_0 = s$ to s_*:

$$V_\pi(s) = \mathbb{E}\tau(s, s_*) = \mathbb{E}\min\{j \geq 0 \; ; \; s.t. \, s_j = s_*, s_0 = s, s_{i+1} \sim K_\pi(\cdot, s_i)\} \, .$$

We make a simplifying assumption that the function approximation is completely determined by projecting this chain into the process $z_t = \text{dist}_G(s_t, s_*) \in \{0, 1, \ldots, M\}$, with $M = \text{diam}(G)$. That is, we consider a latent variable at each state indicating its distance to the target. Since z_t's transition probabilities are not a function of only z_t, it is in general not Markovian. Its Markov approximation $\bar{z}_t$ is defined as the Markov chain $\overline{K}$ given by the expected transition probabilities

$$\Pr(\bar{z}_{t+1} = l - 1 | \bar{z}_t = l) = \alpha_l := \Pr_\mu(z_{t+1} = l - 1 \mid z_t = l) \, , \tag{5.2}$$

$$\Pr(\bar{z}_{t+1} = l | \bar{z}_t = l) = \beta_l := \Pr_\mu(z_{t+1} = l \mid z_t = l) \, , \; l = 0 \ldots M \tag{5.3}$$

and thus $\Pr(\bar{z}_{t+1} = l + 1 | \bar{z}_t = l) = 1 - \alpha_l - \beta_l = \Pr_\mu(z_{t+1} = l + 1 \mid z_t = l)$ under the stationary distribution μ of K_π. In other words, we create an equivalence class given by the level sets of the optimal value function $V_{opt}(s)$, the shortest-path distance in G.

The analysis of Backplay in the 1D chain $\overline{K}$ is now tractable. Given a demonstration $\mathbf{d} = (d_0 = s_0, \ldots, d_L = s_*)$, $d_l \in \mathcal{S}$, we sample it using a step size $m > 0$ (such that $j = 0 \bmod m$, where j is defined in Section 5.4) to obtain $\bar{d}_l := \text{dist}_G(d_{L-ml}, s_*)$, $l = 0, 1, \ldots, \frac{L}{m}$, which satisfies $\bar{d}_l \leq lm$ for all l. For fixed l, we initialize the chain $\overline{K}$ at $\bar{d}_l$: $\bar{z}_0 = \bar{d}_l$. Since $Pr(\bar{z}_m = 0) \geq \prod_{j=0}^{m-1} \alpha_j := \gamma_{0,m}$, after $O(\gamma_{0,m}^{-1})$ trials of length $\leq M$, we will reach the absorbing state and finally have a signal-carrying update for the Q-function at the originating state.

We can consequently merge that state into the absorbing state and reduce the length of the chain by one. Repeat the argument m times so that after $O(\sum_{j=0}^{m} \gamma_{j,m}^{-1}) = O(m\gamma_{0,m}^{-1})$ trials, the Q-function is updated at $\bar{z}_0$. Repeat at Backplay steps lm, $l = 1, \ldots \frac{M}{m}$, and we reach a sample complexity of

$$T_m = \sum_{k=0}^{\frac{M}{m}-1} O\left(M \sum_{j=0}^{m} \gamma_{km,(k+1)m-j}^{-1} \right) .$$

In the case where $\alpha_l = \alpha$ for all l, we obtain $\gamma_{km,(k+1)m-j}^{-1} = \gamma_{0,m-j} = \alpha^{-m+j}$ and therefore $T_m = O\left(\frac{M^2(1-\alpha^{m+1})}{m(1-\alpha)} \alpha^{-m} \right)$, where **the important term is the rate** $\frac{M^2}{m}\alpha^{-m}$.

On the other hand, [80] shows that the first-passage time $\tau(0, M)$ in a skip-free finite Markov chain of M states with a single absorbing state is a random variable whose moment-generating function $\varphi(s) = \mathbb{E}s^{\tau(s,s_*)}$ is given by

$$\varphi(s) = \prod_{j=1}^{M} \frac{(1 - \lambda_j)s}{1 - \lambda_j s} , \tag{5.4}$$

where $\lambda_1, \ldots, \lambda_M$ are the non-unit eigenvalues of the transition probability matrix. It follows that $\mathbb{E}\tau(0, M) = \varphi'(1) = \sum_{j=1}^{M} \frac{1}{1-\lambda_j} \approx (1 - \lambda_1)^{-1}$, which corresponds to the reciprocal spectral gap. [38] further shows that this reciprocal spectral gap is $\Omega(\alpha^{-M/2})$ in our case, and therefore the model without Backplay will on average take $T_M = \Omega(\alpha^{-M/2})$ trials to reach the absorbing state and receive information.

Hence, in this simple birth-death scenario, the sample complexity gains are exponential in the diameter of the graph - $O(\frac{M^2}{m}\alpha^{-m})$ **vs** $\Omega(M\alpha^{-M/2})$**.**

We can analyze the uniform strategy similarly. The probability that a trajectory initialized at one of the uniform samples will reach the absorbing state is lower bounded by

$$\sum_{j=1}^{M} \alpha^j P(\bar{z}_0 = j) = \frac{1}{M}\sum_{j=1}^{M} \alpha^j = \frac{\alpha - \alpha^{M+1}}{M(1-\alpha)}\,, \tag{5.5}$$

which is approximately $\frac{\alpha}{M}$ when α is small, leading to a sample complexity of $O(M^2\alpha^{-1})$ to update the value function at the originating state, and $O(M^3\alpha^{-1})$ at the starting state. Comparing this rate to Backplay with $m = 1$, observe that the uniform strategy is slower by a factor of M (and one can verify that the same is true for generic step size m by imagining that we first sampled a window of size m and then sub-sampled our state from that window), suggesting that it loses efficiency on environments with large diameter.

This analysis relies on the projection into the 1D skip-free process given by the distance to the target and makes two strong simplifying assumptions. First, we assume that the level sets of our estimated value functions correspond to the level sets of the true value function, ie, for all θ, $V_\theta(s) = V_\theta(s')$ whenever $\text{dist}_G(s', s_*) = \text{dist}_G(s, s_*)$. Second, we assume that the projected process is well described by its Markovian approximation. The first assumption is related to the ability of our function approximator to generalize from visited states s to unvisited states s' such that $\text{dist}_G(s', s_*) = \text{dist}_G(s, s_*)$, and is generally violated unless one leverages prior information. The second assumption has an impact on the derived bounds and could be relaxed by replacing conditional probabilities with infima over the level sets.

As Figure 5.2 suggests, Backplay is not a universal strategy to improve sample complexity. Even in the navigation setting, if the task randomizes the initial state s_0, a single demonstration trajectory does not generally improve the coverage of the state-space outside an exponentially small region around said trajectory. For example, imagine a binary tree and a navigation task that starts at a random leaf and needs to reach the root. A single expert trajectory will be disjoint from half of the state space (because the root is absorbing), thus providing no sample complexity gains on average.

The preceding analysis suggests that a full characterization of Backplay is a fruitful direction for reinforcement and imitation learning theory, albeit beyond the scope of our contribution.

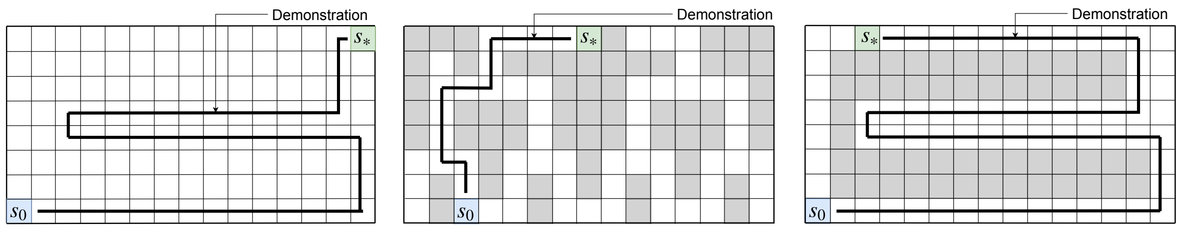
5.4.3 Qualitative Analysis

The previous section gives formal conditions under which Backplay is guaranteed to (exponentially) improve learning speed over standard RL. Here we provide some intuitive examples of environments where Backplay can improve sample-efficiency or lead to a better policy than that of the demonstrator.

In addition, these environments allow us to highlight the differences between Backplay and other methods of reducing sample complexity for deep RL. We hope that this intuition helps guide practitioners on when to choose one or the other.

Figure 5.2 contains three grid worlds. In each, the agent begins at s_0, receives +1 when it reaches s_*, and otherwise incurs a per step cost. They each pose a challenge to model free RL and highlight advantages and disadvantages of Backplay compared to other approaches like behavioral cloning (BC, [11]), generative adversarial imitation learning (GAIL, [78]), and reverse curriculum generation (RCG, [62]). See Table 5.1 for a direct comparison of these algorithms.

The left grid world shows a sparse reward environment in which Backplay can decrease the requisite training time compared to standard RL. Using the sub-optimal demonstration will position the Backplay agent close to high value states. In addition, the agent will likely surpass the expert policy because, unlike in BC approaches, Backplay does not encourage the agent to imitate expert actions. Rather, the curriculum forces the agent to first explore states with large associated



(a) Backplay Helps **(b)** Backplay Helps **(c)** Backplay Hinders

Figure 5.2: Three environments illustrating when Backplay can help or hinder learning an optimal policy. Backplay is expected to learn faster than standard RL on the first and second mazes, but perform worse on the third maze.

value and, consequently, estimating the value function suffers less from the curse of dimensionality. And finally, we expect it to also surpass results from RCG because random movements from the goal state will progress very haphazardly in such an open world.

The middle grid illustrates a maze with bottlenecks. Backplay will vastly decrease the exploration time relative to standard RL, because the agent will be less prone to exploring the errant right side chamber where it can get lost if it traverses to the right instead of going up to the goal. If we used BC, then the agent will sufficiently learn the optimal policy, however it will suffer when placed in nearby spots on the grid as they will be out of distribution; Backplay-trained agents will not have this problem as they also explore nearby states. When an RCG agent reaches the first fork, it has an even chance of exploring the right side of the grid and wasting lots of sample time.

On the rightmost grid world, while Backplay is still likely to surpass its demonstrator, it will have trouble doing better than standard RL because the latter always starts at s_0 and consequently is more likely to stumble upon the optimal solution of going up and right. In contrast, Backplay will spend the dominant amount of its early training starting in states in the sub-optimal demonstration. Note that BC will be worse off than Backplay because by learning the demonstration, it will follow the trajectory into the basin instead of going up the right side. Finally, observe that RCG will likely outperform here given that it has a high chance of discovering the left side shortcut and, if not, it would more likely discover the right side shortcut than be trapped in the basin.

5.5 Related Work

Method	Requirements	Main Idea	Main Weakness
BC	(State, Action) pairs from expert trajectory.	Learn policy that imitates the expert demonstration.	Sub-optimal expert can yield a very poor learned policy.
GAIL	(State, Action) pairs from expert trajectory.	Learn a policy that matches the distribution of expert trajectory pairs.	Difficult to tune; Requires more world interactions; Can perform worse than BC.
RCG	Reversable transition function of environment; Resettable environment.	Take random walks from goal state to build curriculum of initial starting states.	Complexity may increase if random walks reach parts of state space irrelevant to a good policy.
Backplay	State sequence from a 'good enough' trajectory; Resettable environment.	Sample starting state from given trajectory by walking backward along trajectory.	If states in 'good enough' trajectory are not optimal, then can slow learning the *optimal* policy.

Table 5.1: Comparison of Backplay with related work: Behavioral Cloning (BC), Generative Adversarial Imitation Learning (GAIL), and Reverse Curriculum Generation (RCG).

The most related work to ours is a blog post describing a method similar to Backplay used to obtain state-of-the-art performance on the challenging Atari game Montezuma's Revenge that was later published at the NeurIPS Deep RL Workshop [167; 168]. This work was independent of and concurrent to our own. In addition to reporting results on a complex stochastic multi-agent environment, we provide an analytic characterization of the method and an in-depth discussion of what kinds of environments a practitioner can expect Backplay to outperform relative to existing methods.

Hosu and Rebedea [82] is equivalent to our Uniform baseline. They use uniformly random states of an expert demonstration as starting states for a policy. Like Backplay, they show that using a single loss function to learn a policy from both demonstrations and rewards can outperform the demonstrator and is robust to sub-optimal demonstrations. However, they do not impose a curriculum over the demonstration.

5.5.1 Behavioral Cloning

A popular method for improving RL with access to expert demonstrations is behavioral cloning/imitation learning. These methods explicitly encourage the learned policy to mimic an expert policy [11; 164; 50; 215; 109; 135; 77; 78; 8; 117; 147]. Imitation learning requires access to both state and expert actions (whereas Backplay only requires states) and is designed to copy an expert, thus it cannot, without further adjustments (e.g. as proposed by Gao et al. [65]), surpass a sub-optimal expert. See Table 5.1 for a comparison of these approaches.

Other algorithms [153; 120; 48; 49], primarily in dialog, use a Backplay-like curriculum, albeit they utilize behavioral cloning for the first part of the trajectory. This is a major difference as we show that for many classes of problems, we *only* need to change the initial state distribution and do not see any gains from warm-starting with imitation learning. Backplay is more similar to Conservative Policy Iteration [90], a theoretical paper which presents an algorithm designed to operate with an explicit restart distribution.

5.5.2 Automatic Reverse Curriculum Generation

Also related to Backplay are automatic reverse curriculum generation methods [62]. These approaches assume that the final goal state is known and that the environment is both resettable and reversible. The curricula are generated via random walks starting at the goal state [128]. These methods do not require an explicit 'good enough' demonstration as Backplay does. However, they require the environment to be reversible, an assumption that doesn't hold in many realistic tasks such as a robot manipulating breakable objects or complex video games such as Starcraft. In addition, they may fare poorly when the random walks reach parts of the state space that are not actually relevant for learning a good policy. Thus, whether a practitioner wants to generate curricula from a trajectory or a random walk depends on the environment's properties. We discuss this in more detail in Section 5.4.2.

Zhu et al. [218] use a curriculum but manually tune it for each 'stage' of the environment. Within each stage, they use what we call Uniform training, which fails in our most challenging environments.

5.5.3 Using another model

Goyal et al. [70] and Edwards et al. [58] simultaneously proposed the use of a learned backtracking model to generate traces that lead to high value states. Their methods rely on either having the agent visit high reward states or learning a model of the environment capable of generating the states. Both of these are challenging in environments in which the dynamics near starting states are very different from those near goal states.

Finally, Ivanovic et al. [85] use a known (approximate) dynamics model to create a backwards curriculum for continuous control tasks. Their approach requires a physical prior which is not always available and often not applicable in multi-agent scenarios. In contrast, Backplay automatically creates a curriculum fit for any resettable environment with accompanying demonstrations.

Table 5.1 summarizes the discussion above and compares Backplay with other popular methods for speeding up RL. This is also done explicitly in our experiments.

5.6 Pommerman environment

Historically, a majority of multi-agent research has focused on zero-sum two player games. Prominent computer competitions for Poker and Go over the past fifteen years have been vital for developing methods culminating in recent superhuman performance [132; 138; 24; 177]. These benchmarks have also lead to the discovery of new algorithms and approaches like Monte Carlo Tree Search [197; 27; 100; 47] and Counterfactual Regret Minimization [219].

The Pommerman environment was proposed in Resnick et al. [158] as a benchmark for research into environments with more than two players. It is stylistically similar to Bomberman [23], the famous game from Nintendo. At a high level, there are at least four agents all traversing a grid world. Each agent's goal is to have their team be the last remaining. They can plant bombs that, upon expiration, destroy anything (but rigid walls) in their vicinity. It contains both adversarial and cooperative elements. The Free-For-All (FFA) variant has at most one winner and, because there are four players, encourages research directions that can handle situations where the Nash payoffs are not all equivalent. The team variants encourage research with and without explicit communication channels, including scenarios where the agent has to cooperate with previously unseen teammates.

Every battle starts on a randomly drawn symmetric 11x11 grid ('board') with four agents, one in each corner. Teammates start on opposite corners. In team variants, the game ends when both players on one team have been destroyed. In FFA, it ends when at most one agent remains alive. The winning team is the one that has at least one remaining agent. Ties can happen when the game does not end before the max steps or if the last agents are destroyed on the same turn.

Besides the agents, the board consists of wooden and rigid walls. Although it is guaranteed that the agents can reach each other, all paths are initially occluded by wooden walls. See Figure 5.3 for a visual reference.

The benchmark debuted in February 2018 and the first competition was run in June 2018. This yielded eight entrants, of which two were clearly superior to the others. Those two were publicly released for competitors in later competitions to both learn from and benchmark against. The second competition was held at NeurIPS 2018 and is described in detail in Resnick et al. [159]. The agents from that competition were once again released publicly and available to entrants into the third competition, which was held at NeurIPS 2019.

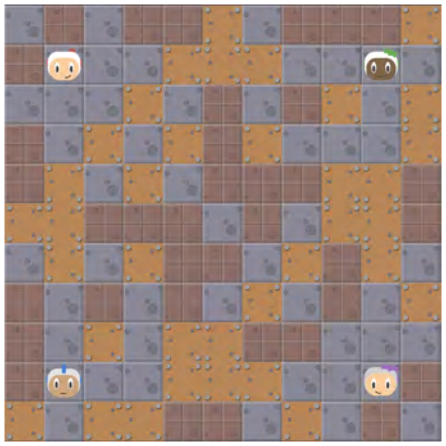


Figure 5.3: Pommerman start state. Each agent begins in one of four positions. Yellow squares are wood, brown are rigid, and gray are passages.

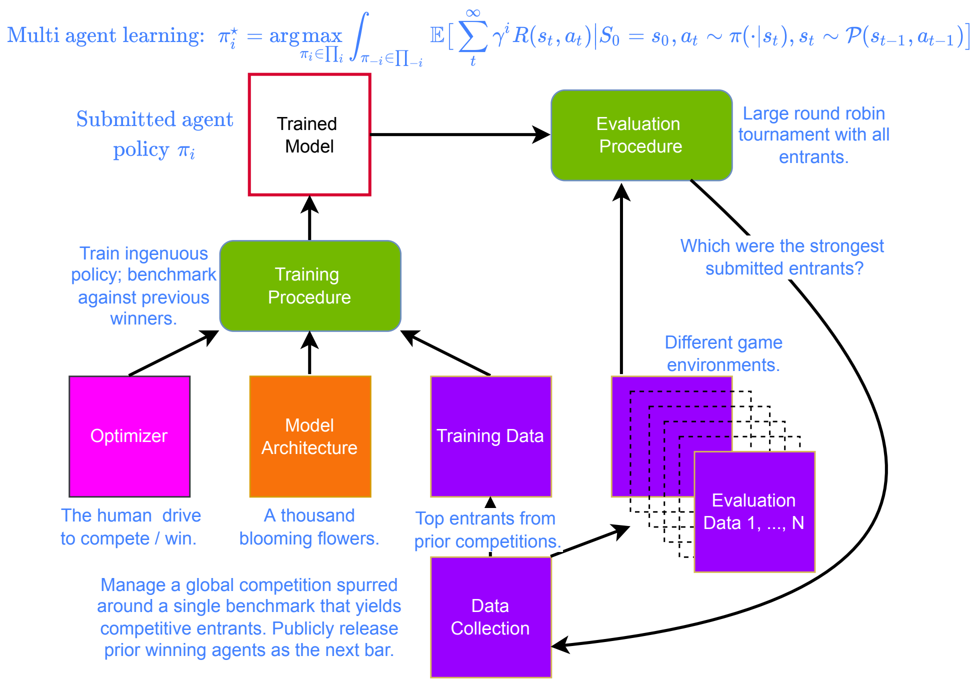

Multi agent learning: $\pi_i^\star = \arg\max_{\pi_i \in \prod_i} \int_{\pi_{-i} \in \prod_{-i}} \mathbb{E}[\sum_t^\infty \gamma^i R(s_t, a_t) | S_0 = s_0, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})]$
Submitted agent policy π_i
Trained Model
Evaluation Procedure
Large round robin tournament with all entrants.
Which were the strongest submitted entrants?
Train ingenuous policy; benchmark against previous winners.
Training Procedure
Different game environments.
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
The human drive to compete / win.
A thousand blooming flowers.
Top entrants from prior competitions.
Manage a global competition spurred around a single benchmark that yields competitive entrants. Publicly release prior winning agents as the next bar.
Data Collection

Figure 5.4: The Machine Learning Application Framework for Pommerman. The world works together to identify the most challenging cases by hosting a large competition and relying on the human drive to compete as the optimizer. Winning agents are then fed back into the system as the top competitors and used as baseline agents in the next competition. The improvement to the framework is found in that Data Collection is now more efficient because we can rely on everyone to create the challenging agents and not just self-play.

Notably, the agents' capabilities grew with each successive competition, as they can be seen as utilizing global manpower to *identify* the most challenging cases. As part of their training algorithms, whether learned or not, the entrants into the next competition then address those challenging cases. For example, while the first two competitions were dominated by tree-search approaches [141], the third competition had competitive learned approaches that were well-matched against the agents from the second competition. Figure 5.4 shows the machine learning application framework for this work. With each global repetition of the loop, the research community solicits better agents (data collection), which then are used in training for the next round. We highlight this because it serves as a great example of how important are the most challenging cases to machine learning research.

This contribution improves the sample complexity of the green Training Procedure block and consequently the efficiency of the entire application framework.

For understanding of how the environment interfaces with our learning process, please see the code repository on Github [134]. In Section 5.7.3, we show Backplay results when training on the purely adversarial FFA environment.

5.7 Experiments

We now move to evaluating Backplay empirically in two environments: a grid world maze and Pommerman. The questions we study across both environments are the following:

- Is Backplay more efficient than training an RL agent from scratch?
- How does the quality of the given demonstration affect the effectiveness of Backplay?
- Can Backplay agents surpass the demonstrator when it is non-optimal?
- Can Backplay agents generalize?

5.7.1 Training Details

We compare several training regimes. The first is **Backplay**, which uses the Backplay algorithm corresponding to a particular sequence of windows and epochs as specified in 5.10. The second, **Standard** is vanilla model-free RL with the agent always starting at the initial state s_0. The last, **Uniform**, is an ablation that considers how important is the curriculum aspect of Backplay by sampling initial states randomly from the entire demonstration. In all these regimes, we use Proximal Policy Optimization (PPO, [174]) to train an agent with policy and value functions parameterized by convolutional neural networks.

On the Maze environment detailed in Section 5.7.2, we also ran comparisons against **Behavioral Cloning** and **Reverse Curriculum Generation**. We chose BC over GAIL [78] for three reasons. First, GAIL requires careful hyperparameter tuning and is thus difficult to train. Second, GAIL requires more environment interactions. And third, GAIL has recently been shown to perform significantly worse than BC [14].

Training details and network architectures for all the environments can be found in 5.10.3 and 5.10.6, while 5.7.4 contains empirical observations for using Backplay in practice.

5.7.2 Maze

We generated mazes of size 24×24 with 120 randomly placed walls, a random start position, and a random goal position. We then used A* to generate trajectories. These included both Optimal demonstrations (true shortest path) and N-Optimal demonstrations (N steps longer than the shortest path). More details on this setup are given in 5.10.2.

Our model receives as input four 24×24 binary maps. They contain ones at the positions of, respectively, the agent, the goal, passages, and walls. It outputs one of five options: Pass, Up, Down, Left, or Right. The game ends when the agent has reached its goal or after a maximum of 200 steps, whereupon the agent receives reward of +1 if it reaches the goal and a per step penalty

Algorithm	Map Set	% Optimal	% 0-5 Optimal	Avg Suboptimality	Std Suboptimality
Standard	All	0	0	N/A	N/A
Uniform	Optimal	27	91	8.26	32.92
Uniform	5-Optimal	51	98	2.04	17.39
Uniform	10-Optimal	49	98	2.04	16.79
Florensa	Optimal	20	51	63.36	78.08
Florensa	5-Optimal	48	77	25.44	56.89
Florensa	10-Optimal	49	69	39.75	70.92
Backplay	Optimal	**31**	**100**	**0.64**	**4.96**
Backplay	5-Optimal	37	94	7.94	33.35
Backplay	10-Optimal	**54**	99	**0.37**	3.49

Table 5.2: Results after 2000 epochs on 100 mazes. Note that for Backplay and Uniform, the Map Set is also the type of demonstrator, where N-optimal has demonstrations N steps longer than the shortest path. From left to right, the table shows: the percentage of mazes on which the agent optimally reaches the goal, percentage on which it reaches in at most five steps more than optimal, and the average and standard deviation of extra steps over optimal. Both Backplay and Uniform succeed on almost all mazes and, importantly, can outperform the experts' demonstrations. On the other hand, Standard does not learn a useful policy and Florensa fails to learn more than $50 - 70\%$ of the maps, which is why its sub-optimality mean and std is so high. Results for Backplay were generally consistent across all seeds. For others, we report their best score. See 5.10.4 for further details.

of -0.03.

Backplay, Uniform, Standard. Table 5.2 shows that Standard has immense trouble learning in this sparse reward environment while both Backplay and Uniform find an optimal path approximately 30-50% of the time and a path within five of the optimal path almost always. Thus, in this environment, demonstrations of even sub-optimal experts are extremely useful, while the curriculum created by Backplay is not necessary. That curriculum does, however, aid convergence speed (5.10.4). We will see in 5.7.3 that the curriculum becomes vital as the environment increases in complexity.

Behavioral Cloning. We trained an agent using behavioral cloning from the same trajectories as the ones used for Backplay. While the agent learns to perfectly imitate those trajectories, we had immense difficult doing better than the expert. All of our attempts to use reward signal to improve the agent's performance (over that of the behaviorally cloned agent) were unsuccessful, even after incorporating tricks in the literature such as those found in [171]. One possible reason is that the agent has no information about states outside of the demonstration trajectory. Because of this, we do not include these results in the tables or graphs as they were not competitive.

Reverse Curriculum Generation. We also compared Backplay to the method proposed by [62]. As illustrated in Table 5.2 and Section 5.10.4, Florensa agents perform significantly worse with higher sample complexity variance compared to Backplay (or Uniform). This suggests that having demonstrations helps inordinately. Further details can be found in 5.10.3.

5.7.3 Pommerman

In this section, we present our results on the Pommerman environment, previously described in detail in Section 5.6. In our experiments, we use the purely adversarial Free-For-All (FFA)

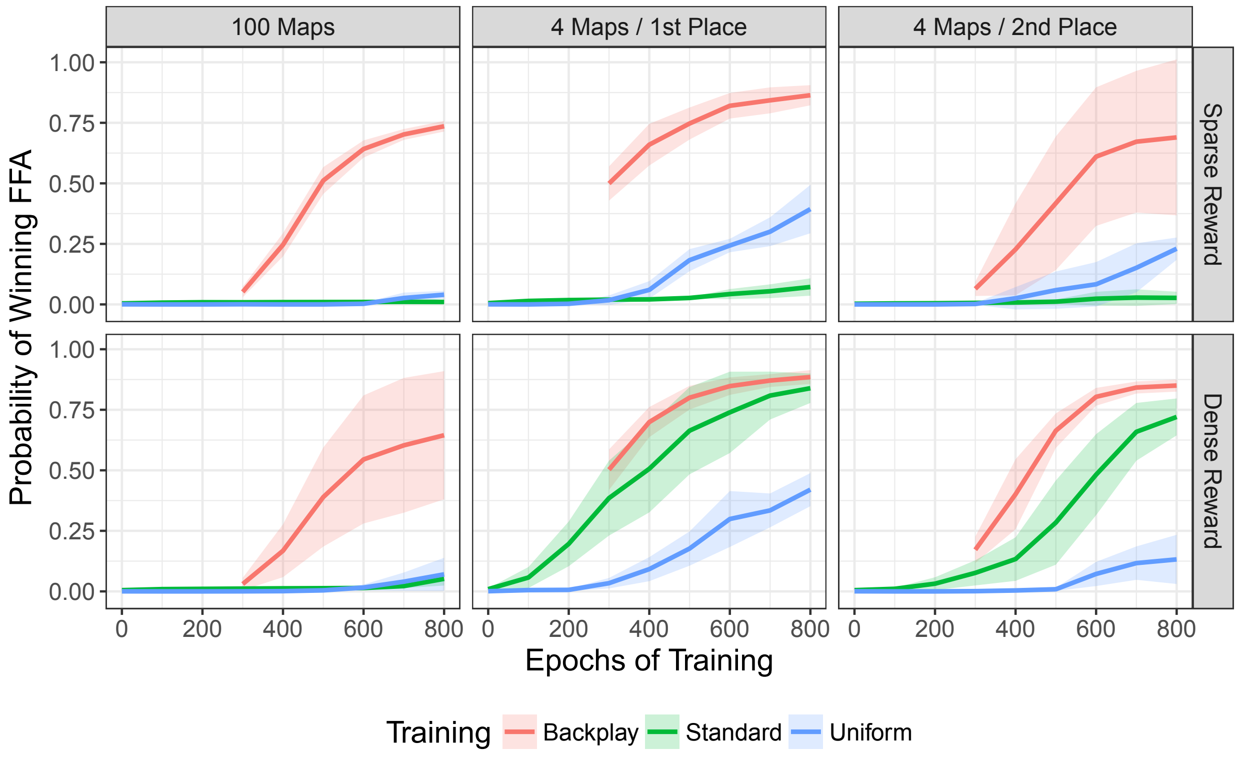


Figure 5.5: Pommerman results (5 seeds). The Backplay curves are shortened to show only those results when starting from the initial state. The 100 Maps column is our starkest result and displays results on games beginning from the initial state regardless of when training occurs. We can see here that Backplay attains strong performance while Uniform and Standard fail to learn anything of note.

environment where the winner of the game is the last agent standing.

The learner is a randomly chosen agent, the three opponents are copies of the winner of the June 3rd 2018 FFA competition, a stochastic agent using a Finite State Machine Tree-Search approach (FSMTS, [217]). To construct a Backplay demonstration we consider games of 4 FSMTS agents and either use the trajectory of the winner or the 2^{nd} place player.

The observation state is represented by 19 11x11 maps, and we feed the concatenated last two states to our agent as input. A detailed description of this mapping is given in 5.10.5. The game ends either when the learning agent wins or dies, or when 800 steps have passed. Upon game end, the agent receives +1 for winning and −1 otherwise (we refer to this as the *Sparse* environment).

Our first three scenarios follow the Sparse setup. We independently consider three Backplay trajectories, one following the winner over 100 maps, one following the winner over four maps, and one following the runner up over four maps, with the latter two set of maps being the same. Figure 5.5 shows that **Backplay can soundly defeat the FSMTS agents in sparse settings when following the winner while standard RL struggles when there are many possible maps**. We see this also works when the expert is not the winner, but the second place agent.

Visit this link for an example gif of our trained Backplay agent, the red agent in the top left. Note that our agent learned to 'throw' bombs, a unique playing style that no prior Pommerman competitor had exhibited (including the FSMTS demonstrator).

Moreover, with enough data Backplay generalizes (to some extent) to novel environments. When we train our agent with data from 100 maps, **Backplay wins 416 / 1000 games on a held out set of ten maps**. We note this was in contrast to our Maze experiments where no approach generalized. Thus, we believe that the lack of generalization was a consequence of not including enough mazes during training.

We also compare Backplay to RL with reward shaping. After hand-tuning, we found that giving the learning agent a reward of +0.1 whenever it collected an item led to better performing agents (*Dense*). Figure 5.5 shows that reward shaping can improve the learning speed of RL agents when using a small set of maps but not when learning over 100 maps. We believe that Backplay's strong results even without the taxing human involvement accompanying reward shaping is a major positive in its favor.

5.7.4 Practical Findings

We trained a large number of models through our research into Backplay. Though these findings are tangential to our main points (and are mainly qualitative), we list some observations here that may be helpful for other researchers working with Backplay.

First, we found that Backplay does not perform well when the curriculum is advanced too quickly, however it does not fail when the curriculum is advanced 'too slowly.' Thus, researchers interested in using Backplay should err on the side of advancing the window too slowly rather than too quickly.

Second, we found that Backplay does not need to hit a high success rate before advancing the starting state window. Initially, we tried using adaptive approaches that advanced the window when the agent reached a certain success threshold. This worked but was too slow. Our hypothesis is that what is more important is that the agent gets sufficiently exposed to enough states to attain a reasonable barometer of their value rather than that the agent learns a perfectly optimal policy for a particular set of starting states.

Third, Backplay can still recover if success goes to zero. This surprising and infrequent result occurred at the juncture where the window moved back to the initial state and even if the policy's entropy over actions became maximal. We are unsure what differentiates these models from the ones that did not recover.

We also explored using DAgger ([164]) for training our agent, but found that it achieved approximately the same win rate ($\sim$ 20%) as what we would expect when four FSMTS agents played each other (given that there are also ties).

5.8 Conclusion

We have introduced and analyzed Backplay, a technique which improves the sample efficiency of model-free RL by constructing a curriculum around a demonstration. We showed that Backplay agents can learn in complex environments where standard model-free RL fails, that they can outperform the 'expert' whose trajectories they use while training, and that they compare very favorably to related methods such as reversible curriculum generation. We also presented a theoretical analysis of its sample complexity gains in a simplified setting.

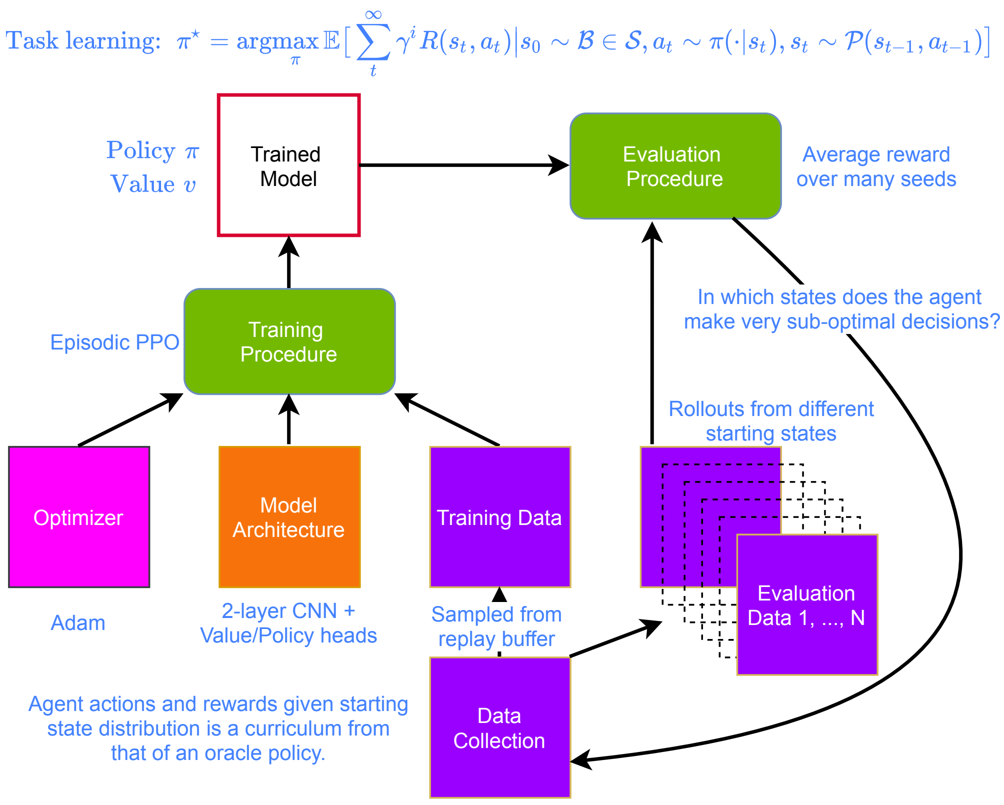
An important future direction is combining Backplay with more complex and complementary methods such as Monte Carlo Tree Search (MCTS, [27; 197]). There are many potential ways to do so, for example by using Backplay to warm-start MCTS.

Another direction is to use Backplay to accelerate self-play learning in zero-sum games. However, special care needs to be taken to avoid policy correlation during training [108] and thus to make sure that learned strategies are safe and not exploitable [25].

A third direction is towards non-zero sum games. It is well known that standard independent multi-agent learning does not produce agents that are able to cooperate in social dilemmas [115; 116; 148; 64] or risky coordination games [211; 149]. In contrast, humans are much better at finding these coordinating and cooperating equilibria [21; 98]. Thus, we conjecture that human demonstrations can be combined with Backplay to construct agents that perform well in such situations.

Other future priorities are to gain further understanding into when Backplay works well, when it fails, and how we can make the procedure more efficient. Could we speed up Backplay by ascertaining confidence estimates of state values? Do the gains in sample complexity come from value estimation like our analysis suggests, from policy iteration, or from both? Is there an ideal rate for advancing the curriculum window and is there a better approach than a hand-tuned schedule?

5.9 What did we learn about the most challenging cases?



Task learning: $\pi^\star = \operatorname{argmax}_{\pi} \mathbb{E}\big[\sum_t^{\infty} \gamma^i R(s_t, a_t) \big| s_0 \sim \mathcal{B} \in \mathcal{S}, a_t \sim \pi(\cdot|s_t), s_t \sim \mathcal{P}(s_{t-1}, a_{t-1})\big]$
Policy π
Value v
Trained Model
Evaluation Procedure
Average reward over many seeds
Episodic PPO
Training Procedure
In which states does the agent make very sub-optimal decisions?
Rollouts from different starting states
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Adam
2-layer CNN + Value/Policy heads
Sampled from replay buffer
Agent actions and rewards given starting state distribution is a curriculum from that of an oracle policy.
Data Collection

Figure 5.6: The Machine Learning Application Framework for this contribution. We change the traditional Reinforcement Learning objective function such that we now draw the starting state from a distribution of states rather than a singular fixed state. The improvement to the framework is found in that the Training Procedure is now made more efficient via the use of a curriculum over the states in the demonstration.

We learned that as long as we can set the starting state of a scene, we can use sub-optimal demonstrations to efficiently address the challenging cases in task learning. The efficiency gain is from the sample complexity that our agent would otherwise spend struggling in those challenging parts of the state space $\bar{\mathcal{G}} \in \mathcal{S}$. This gain is expressed clearly in Figure 5.5.

Observe that there is nothing domain specific about how we have addressed the challenging cases. We just require at least one demonstration and the ability to reset the environment at will. In our case, we focused on maze environments and the game environment Pommerman, but the generalized solution is appropriate for any task learning objective. Figure 5.6 highlights this in the context of the Machine Learning Application Framework. This contribution improves the sample complexity of the green Training Procedure block and consequently the efficiency of the entire application framework.

The status quo learning approaches for creating robots that capably perform tasks in household or factory settings rely on human demonstrations that must be instilled before they reach the end user. This is because of the required sample complexity of their algorithms. This contribution is an important step towards building a robot that can manage that concern and fulfilling the goal of creating a communicative and visually aware robot that **can swiftly learn new tasks**.

5.10 Supplementary

5.10.1 Backplay Hyperparameters

As mentioned in Section 5.4, the Backplay hyperparameters are the window bounds and the frequency with which they are shifted. When we get to a training epoch represented in the sequence of epochs, we advance to the corresponding value in the sequence of windows. For example, consider training an agent with Backplay in the Maze environment (Table 5.3) and assume we are at epoch 1000. We will select a maze at random, an $N \in [16, 32)$, and start the agent in that game N steps from the end. Whenever a pair is chosen such that the game's length is smaller than N, we use the initial state.

There isn't any downside to having the model continue training in a window for too long, albeit the ideal is that this method increases the speed of training. There is however a downside to advancing the window too quickly. A scenario common to effective training is improving success curves punctured by step drops whenever the window advances.

Starting at training epoch	Uniform window
0	[0, 4)
350	[4, 8)
700	[8, 16)
1050	[16, 32)
1400	[32, 64)
1750	[64, 64)

Table 5.3: Backplay hyperparameters for Maze.

Starting at training epoch	Uniform window
0	[0, 32)
50	[24, 64)
100	[56, 128)
150	[120, 256)
200	[248, 512)
250	[504, 800)
300	[800, 800]

Table 5.4: Backplay hyperparameters for Pommerman 4 maps.

Starting at training epoch	Uniform window
0	[0, 32)
85	[24, 64)
170	[56, 128)
255	[120, 256)
340	[248, 512)
425	[504, 800)
510	[800, 800]

Table 5.5: Backplay hyperparameters for Pommerman 100 maps.

5.10.2 Maze: Demonstration Details

For N-Optimal demonstrations, we used a noisy A* where at each step, we follow A* with probability p or choose a random action otherwise. We considered $N \in \{5, 10\}$. In all scenarios, we only selected maps in which there exists at least a path from the the initial state to the goal state, we filtered any path that was less than 35 in length and stopped when we found a hundred valid training games. Note that we held the demonstration length invariant rather than the optimal length (i.e. all N-optimal paths have the same length regardless of N, which means that the length of the optimal path of a N-optimal demonstration decreases with N). This could explain why the results in Table 5.2 (column 1) show that Backplay's performance increases with N (since the larger the N, the smaller the true optimal path, so the easier it is to learn an optimal policy for that maze configuration).

5.10.3 Maze: Network Architecture and Training Parameters

We use a standard deep RL setup for our agents. The agent's policy and value functions are parameterized by a convolutional neural network with 2 layers each of 32 output channels, followed by two linear layers with 128 dimensions. Each of the layers are followed by ReLU activations. This body then feeds two heads, a scalar value function and a softmax policy function over the five actions. All of the CNN kernels are 3x3 with stride and padding of one.

We train our agent using Proximal Policy Optimization (PPO, [174]) with $\gamma = 0.99$, learning rate 1×10^{-3}, batch size 102400, 60 parallel workers, clipping parameter 0.2, generalized advantage estimation with $\tau = 0.95$, entropy coefficient 0.01, value loss coefficient 0.5, mini-batch size 5120, horizon 1707, and 4 PPO updates at each iteration. The number of interactions per epoch is equal to the batch size (102400).

The hyperparameters used for training the agent with Reverse Curriculum Generation [62] are: 10^4 rollout states for nearby sampling, 50 Brownian steps, 200 samples from new starts, 100 samples from old starts, interval for the expected return $[0.1, 0.9]$.

5.10.4 Maze: Learning Curves

Below are our learning curves for the Maze challenge over five seeds. Note that we do not show any results for Standard as it failed to learn much of anything in the time allotted (3500 epochs). Also note that Backplay occasionally sees the actual grid starting position from epoch 1400, but it becomes the default starting state at epoch 1750. To align with this, our Florensa baseline switches to training from the actual start position at epoch 1750, and we show results for Uniform *only* over the initial starting state. Correspondingly, we show the graphs from epoch 1000 as all of the methods have commensurately poor results before that time.

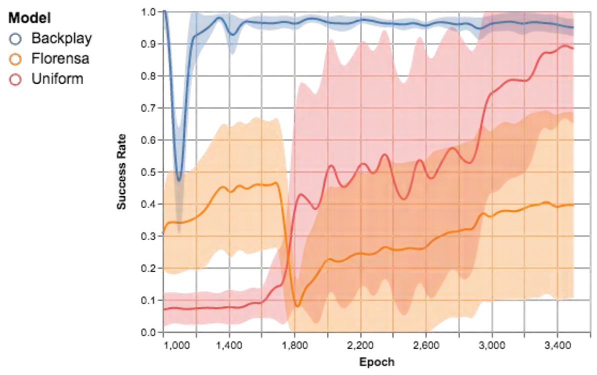


Figure 5.7: Maze results when training with demonstrations of length equal to the optimal path length. Note that Backplay has a very small variance and a very high success rate as early as epoch 1800. On the other hand, Florensa fails to break 70% and has a high variance, and Uniform doesn't achieve a strong success rate until epoch 3000.

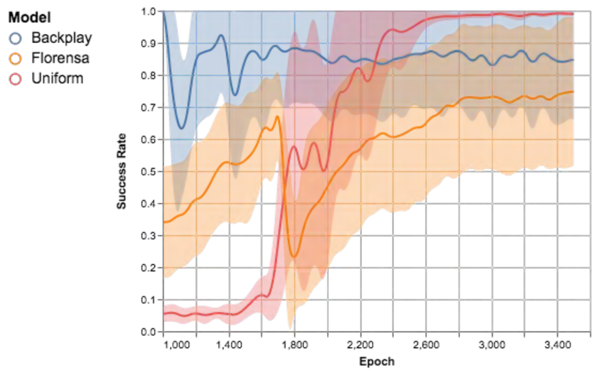


Figure 5.8: Maze results when training with demonstrations that are five steps longer than the optimal path length. Compared to the prior graph, Backplay doesn't do as well, albeit it still performs favorably compared to Florensa, with a consistently higher expected return. Its advantage over Uniform is a reduced amount of necessary samples.

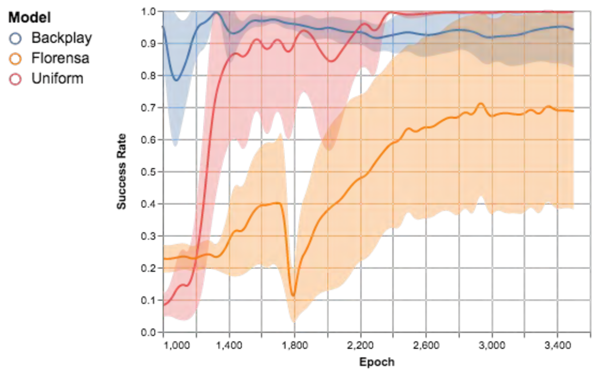


Figure 5.9: Maze results when training with demonstrations that are ten steps longer than the optimal path length. We again see that Backplay does very well compared to the Florensa baseline, with a much stronger expected return and lower variance. We also see, however, that Uniform is an able competitor to both of these as the expert becomes more suboptimal.

5.10.5 Pommerman: Observation State

There are 19 feature maps that encompass each observation. They consist of the following: the agents' identities and locations, the locations of the walls, power-ups, and bombs, the bombs' blast strengths and remaining life counts, and the current time step.

The first map contains the integer values of each bomb's blast strength at the location of that bomb. The second map is similar but the integer value is the bomb's remaining life. At all other positions, the first two maps are zero. The next map is binary and contains a single one at the agent's location. If the agent is dead, this map is zero everywhere. The following two maps are similar. One is full with the agent's integer current bomb count, the other with its blast radius. We then have a full binary map that is one if the agent can kick and zero otherwise.

The next maps deal with the other agents. The first contains only ones if the agent has a teammate and zeros otherwise. This is useful for building agents that can play both team and solo matches. If the agent has a teammate, the next map is binary with a one at the teammate's location (and zero if she is not alive). Otherwise, the agent has three enemies, so the next map contains the position of the enemy that started in the diagonally opposed corner from the agent. The following two maps contain the positions of the other two enemies, which are present in both solo and team games.

We then include eight feature maps representing the respective locations of passages, rigid walls, wooden walls, flames, extra-bomb power-ups, increase-blast-strength power-ups, and kicking-ability power-ups. All are binary with ones at the corresponding locations.

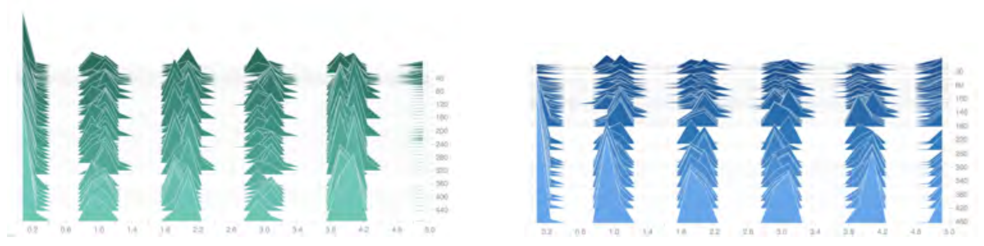
Finally, we include a full map with the float ratio of the current step to the total number of steps. This information is useful for distinguishing among observation states that are seemingly very similar, but in reality are very different because the game has a fixed ending step where the agent receives negative reward for not winning.

5.10.6 Pommerman: Network Architecture and Training Parameters

We use a similar setup to that used in the Maze game. The architecture differences are that we have an additional two convolutional layers at the beginning, use 256 output channels, and have output dimensions of 1024 and 512, respectively, for the linear layers. This architecture was not tuned at all during the course of our experiments. Further hyperparameter differences are that we used a learning rate of 3×10^{-4} and a gamma of 1.0. These models trained for 72 hours, which is ~50M frames. [2]

[2] In our early experiments, we also used a batch-size of 5120, which meant that the sampled transitions were much more correlated. While Backplay would train without any issues in this setup, Standard would only marginally learn. In an effort to improve our baselines, we did not explore this further.

5.10.7 Pommerman: Action Distribution Analysis



(a) Standard Sparse **(b)** Backplay Sparse

Figure 5.10: Typical histograms for how the Pommerman action selections change over time. From left to right are the concatenated counts of the actions (Pass, Up, Down, Left, Right, Bomb), delineated on the y-axis by the epoch. Note how the Standard agent learns to not use bombs.

Pommerman can be difficult for reinforcement learning agents. The agent must learn to effectively wield the bomb action in order to win against competent opponents. However, bombs destroy agents indiscriminately, so placing one without knowing how to retreat often results in negative reward for that agent. Since agents begin in an isolated area, they are prone to converging to policies which do not use the bomb action (as seen in the histograms in Figure 5.10), which leads them to sub-optimal policies in the long-term.

5.10.8 Pommerman: Win Rates

Here we show the per-map win rates obtained by the agent trained with Backlpay on the 100 Pommerman maps.

Win	Maps
90%	24
80%	26
70%	6
60%	5
50%	5

Table 5.6: Aggregate per-map win rates of the model trained with Backplay on 100 Pommerman maps. The model was run over 5000 times in total, with at least 32 times on each of the 100 maps. The *Maps* column shows the number of maps on which the Backplay agent had a success rate of at least the percent in the *Win* column. Note that this model has a win rate of $> 80\%$ on more than half of the maps and a win rate of $> 50\%$ on all maps.

Part III

Understanding the Most Challenging Cases

6 | Multi-Agent Language Learning.

6.1 Introduction

Multi-agent language learning is founded on a premise that agents learn as part of the need to communicate with other agents. This is in contrast to single-agent language learning, exemplified by the recent advances [150; 151; 26] that use unsupervised language modeling to train a model and then speak utterances by generating conditionally from user-defined prompts. These models learn without other agents and are instead trained on a large dataset of utterances to give statistically meaningful responses to queries. On the other hand, the multi-agent approach has the agent learn what to say in conjunction with a task that imparts meaning to the utterances.

Motivation for this is in part from observations on human learning. We clearly do not learn from large utterance datasets, but more likely from interactions with our environment and agents there-in. It is hypothesized that this learning is aided by natural curricula similar to how we manually do the same with trained animals ('shaping') [144]. There is of course a wealth of research into curriculum learning in machine learning, particularly with neural networks [59; 18].

The agent to agent interactions in a normal human environment provide a template for what we need to communicate in order to excel as a social creature. A baby does not need to have any understanding of how to navigate scheduling requirements for a six person meeting and so there is no expectation that it grasps the corresponding terms or how to string them together. On the other hand, it does have need to express why it is upset or what it wants to eat. Consequently, the early interactions that babies have are all designed to elicit from nearby adults how to communicate such primitive desires. When a newborn cries and its father frantically asks it if it's hungry or tired, the fact that the baby cannot respond in kind does not mean the father's utterance was for naught. On the contrary, multi-agent language learning posits that that exchange teaches the baby that certain words are associated with this 'game'. After enough interactions, it will learn how to communicate its desire directly. This base then acts as primitives for the next rung of learned communication, after which an ensuing cascade of learning takes place once the baby, now toddler, has a grasp of the communicative blocks used by the agents in its vicinity.

That process happens on the order of biological time scales and takes roughly nine to fourteen months before any coherent conversation commences [34]. We would of course like to fast-track that learning and so we take advantage of modern machine learning methods like stochastic gradient descent. However, the core of the solution that multi-agent language learning pursues is the same in that there are multiple agents that are together trying to solve a task for which aligned communication would greatly improve the efficiency with which they could solve this task. As long as we can increase the complexity of the task while still training agents to successfully solve the task, the accompanying language should increase in complexity because otherwise the task would be too difficult to solve. Chevalier-Boisvert et al. [41] exemplifies this notion of learning language via a curriculum of multi-agent language tasks.

In the prior example, the task was to figure out what the baby needed. Observe that this experience is much more common than that and occurs well after language has been incepted as well. Online learning of a new word in context of other known words is an example. The game there is the back and forth exchange of asking what that words means and then clarifying the definition received in response. Another example is explaining a concept to a crowd of people, each with unknown expertise. First, the agent explaining the concept has to establish at what level is the audience - the explanation changes greatly along the spectrum of the audience being experts in the field versus them having no understanding at all. After that has been established, the speaker can lay the communication groundwork that has minimal friction for expositing the heart of the matter. Along the way, the listeners ask questions and clarify their own understanding, which additionally serves to refine the how the speaker conveys the material. Through this process, the parties develop their own learned communication amongst the group.

The ReferIt game is a common platform for research in multi-agent language learning, dating back to Lewis [118] and then continuing with works [183; 182] in the 1990s and more modern investigations [63; 111; 73; 61; 110; 125].

The basic variation is that there are two agents, a sender and a receiver. The sender sees some concept, perhaps an image, and must then communicate a message to the receiver. The receiver's job is to determine to which concept the sender's message is referring, usually in the presence of a set of distractor concepts. The agents are rewarded if the receiver is correct. Through this basic game, the agents emerge a language defined by the stream of messages that the sender passes to the receiver.

Advanced variations allow for more steps of communication, a bidirectional channel, a mix of images and text descriptions, human languages in the loop, and so forth. However, at large they are all trying to elicit an emergent language that carries characteristics of human languages.

6.2 The most challenging cases

Regardless of how we teach agents to communicate, it is a requirement that we do so in order to build a robot that can coherently converse with humans. The most advanced language systems today frequently fail because of the challenging cases that they encounter. Examples abound in disparate sub-fields from model compression to fairness & bias.

For example, Hooker et al. [81] found that modern model compression techniques were detrimental to underrepresented groups. Reducing the size of the model was only marginally impacting the overall score, but it was greatly impacting group level scores. After expanding the test suite to account for poorly represented groups, they observed that the model performed inordinately poorly on those groups. In other words, it is the most challenging cases, in this case determined by dataset bias, that are at the core of the difficulties in model compression. Formally, there were semantically important attributes with deciding functions $\mathcal{W}_k$ and corresponding $\mathcal{G}_k$ such that

$$\mathop{\mathbb{E}}_{\mathcal{G}_i \sim \bar{\mathcal{G}}} |\mathcal{G}_i| \gg |\mathcal{G}_k| \tag{6.1}$$

$$\delta > 0 \tag{6.2}$$

$$\mathop{\mathbb{E}}_{(x,y)\sim T^{\text{test}}} f(x,y) - \mathop{\mathbb{E}}_{(x,y)\sim T^{\text{test}}} f_{\text{compressed}}(x,y) < \delta \tag{6.3}$$

$$\mathop{\mathbb{E}}_{(x,y)\sim \mathcal{G}_k} f(x,y) - \mathop{\mathbb{E}}_{(x,y)\sim \mathcal{G}_k} f_{\text{compressed}}(x,y) \gg \delta \tag{6.4}$$

In fairness & bias, Zhao et al. [216] found that datasets for semantic role labeling have significant gender bias, and that models trained on these datasets further amplify the bias at test time. Hutchinson et al. [84] exposes how these issues occur with 'disability' in common English language models and further corroborates research [22] showing that these biases exacerbate problems in downstream application problems due to the prevalence of neural embeddings in modern real-world applications.

Observe that all these concerns in language are frequently related to compositionality. We define this rigorously in Section 6.5, but it is roughly the ability to combine and recombine a relatively small number of elements to create a vast number of complex meaningful expressions [35]. A related view is that the meaning of a complex expression is determined by its structure and the meanings of its constituents [186]. That human communication is compositional helps us inordinately with quickly grasping new concepts because we can grok them as a whole from the word parts that explain them.

Model compression is related to compositionality in that if the agent has truly learned a compositional view of the world, then proper compression will preserve those integral aspects as they are the irreducible components of understanding. We do not see this both because our agents are not learning compositional approaches and because our approaches to compression are arguably not fine enough to distinguish what is or is not a principle element.

Dataset bias is related to compositionality in that the errors due to the inherent biases of our dataset are diminished when there is an underlying compositional understanding of the world. In other words, humans do not make the mistake of assuming all buildings must be three stories high before they have ever seen a skyscraper. They understand that the height of the building is distinct from the notion of a building and its affordances. They have an intuitive (likely learned [180; 179]) grasp of the building blocks underlying their world view and how to compose them together. Our machines do not have this and they run into issues when the dataset from which they learn is not sufficiently balanced.

Compositionality is a particularly challenging aspect of language learning to address and frequently related to the most challenging cases in communication because it is at the core of the ability to transfer to new tasks. Without this capability, agents are limited in how quickly they can adapt to new conversational topics or even amend their prior built-in notions of language. On the other hand, agents trained to exhibit compositionality in their language understanding see new meaning as arising from a composition of concepts in context.

Understanding the root causes of compositionality in language and why they arise helps us build languages that exhibit this property. That then helps us address the challenging cases related to poor systematic generalization in language.

6.3 Overview of our contribution

Our contribution was inspired by an observation of compositional languages documented visually in Figure 6.1. Consider the act of jointly communicating two concepts, color and shape, each having five known entries. An agent that speaks compositionally would require six bits to communicate a 'red circle' and four bits to store the concepts. An agent that speaks non-compositionally requires only five bits to communicate that object but needs 5 bits to store the concepts. This trade-off of storage in the agent for bits in the channel is representative of what we ask of our agents today. Because they are computers with unlimited amounts of storage space *relative* to the concepts we ask them to learn, they do not require a compositional understanding in order to solve the tasks we put before them. However, in a world of infinite concepts, working effectively with other agents (including humans) and learning those concepts quickly requires that their approach with respect to this trade-off become more like the human approach.

In our work [160], published at the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), we try to bridge this gap. We focus on how a neural network, specifically a generative one, can learn a compositional language. Moreover, we ask how this can occur without task-specific constraints, hallmarks of prior works [105; 114]. To accomplish this, we first define in Section 6.5 what is a language and what we mean by compositionality. In tandem, we introduce *precision* and *recall*, two metrics that help us measure how well a generative model at large has learned a grammar from a finite set of training instances. In Section 6.6, we then use a variational autoencoder with a discrete sequence bottleneck to investigate how well the model learns a compositional language, in addition to what affects that learning. This allows us to derive *residual entropy*, a third metric that reliably measures compositionality in our particular environment. We use this metric to cross-validate precision and recall. While there may be a connection between our study and human-level languages, we do not experiment with human data or actual languages in this contribution.

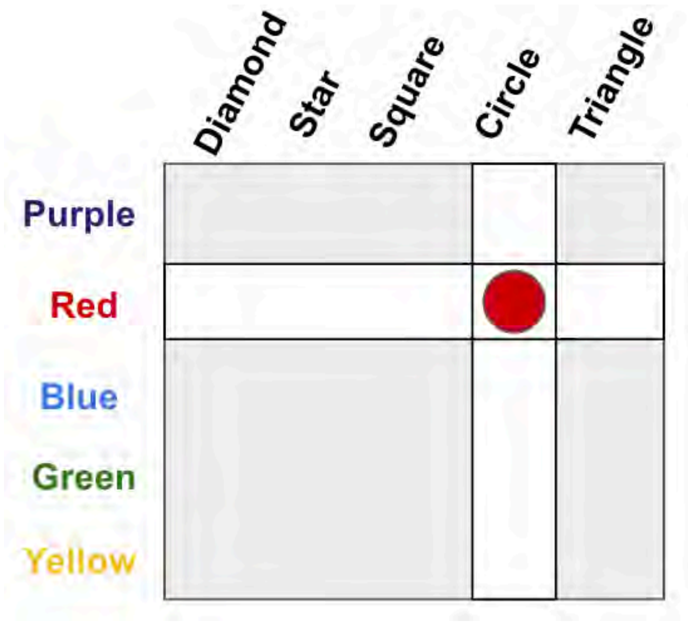


Figure 6.1: The grid above shows five shapes and five colors. Agents with a non-compositional language can use this shared map to communicate "Red Circle" with only $\lceil \log_2 5^2 \rceil = 5$ bits. If they instead used a compositional language, it would require $\lceil \log_2 5 \rceil = 3$ bits for each concept for a total of 6 bits to convey the string. On the other hand, the agent needs 25 memory slots to store the concepts in the former case but only 10 slots in the compositional case. This trade-off exemplifies the motivation for our investigation because it suggests that a key driver of compositionality in language is the capacity of an agent relative to the total number of objects in its environment.

Our environment lets us experiment with a syntactic, compositional language while varying the channel width and the number of parameters, our surrogate for the capacity of a model. Our experiments in Section 6.7 reveal that our smallest models are only able to solve the task when the channel is wide enough to allow for a surface-level compositional representation. In contrast, large models learn a language as long as the channel is large enough. However, large models also have the ability to memorize the training set. We hypothesize that this memorization would lead to non-compositional representations and overfitting, albeit this does not yet manifest empirically. This setup allows us to test our hypothesis that there is a network capacity above which models will tend to produce languages with non-compositional structure.

6.4 Related Work

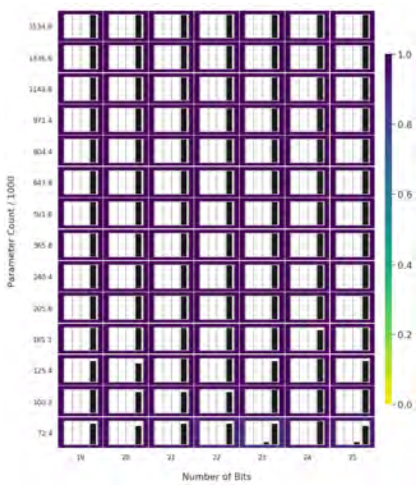
Our work is most similar to Kottur et al. [105], which showed that compositional language arose only when certain constraints on the agents are satisfied. While the constraints they examined were either making their models memoryless or having a minimal vocabulary in the language, we hypothesized about the importance for agents to have small capacity relative to the number of disentangled representations (concepts) to which they are exposed. This is more general because both of the scenarios they described fall under the umbrella of reducing model capacity. To ask this, we built a much bigger dataset to illuminate how capacity and channel width effect the resulting compositionality in the language.

Other works studying compositionality in multi-agent learning include Liska et al. [123]; Spike et al. [181]; Verhoef et al. [194]; Kirby et al. [97]; Zaslavsky et al. [212]. Liska et al. [123] suggests that the average training run for recurrent neural networks does not converge to a compositional solution, but that a large random search will produce compositional solutions. This implies that the optimization approach biases learning, which is also confirmed in our experiments. However, we further analyze other biases. Spike et al. [181] describes three properties that bias models towards successful learned signaling: the creation and transmission of referential information, a bias against ambiguity, and information loss. This lies on a similar spectrum to our work, but pursues a different intent in that they study biases that lead to optimal signaling; we seek compositionality.

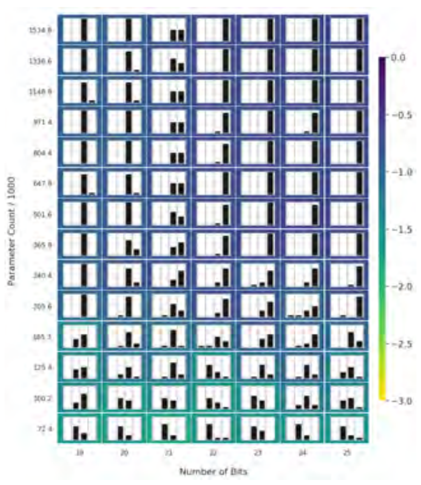
Each of Verhoef et al. [194]; Kirby et al. [97]; Zaslavsky et al. [212] examine the trade-off between expression and compression in both emergent and natural languages, in addition to how that trade-off affects the learners. We differ in that we target a specific aspect of the agent – capacity – and ask how that aspect biases the learning.

Most other works studying compositionality in emergent multi-agent languages [4; 133; 36; 113] have focused on learning interpretable representations. See [83] for a broad survey of the different approaches. By and large, these are orthogonal to our work because none pose the important question we ask – how does an agent's capacity effect the resulting language's compositionality?

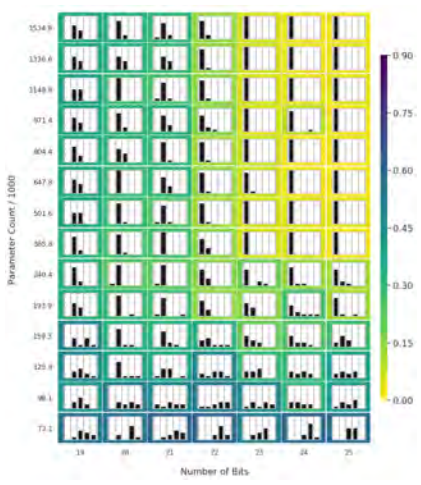
6.5 Compositionality and Language Learning



(a) Precision



(b) Recall



(c) Entropy

Figure 6.2: Histograms for model A showing precision, recall (defined in § 6.5.3), and entropy (defined in § 6.7.1) over the test set. We show results for bits 19 to 25 and parameter range $72k$ to $1534k$ (details in § 6.7). Each bit/parameter combination is trained for 10 seeds over $200k$ steps. Precision and Recall are as described in Eqs. equation 6.9 and equation 6.10 with $M = |D_{\text{test}}|$ and $N = 10000$.

In this section, we provide a mathematical definition for a language and then use that to define what it means for a language to be compositional. We find that it coincides with the above views, which are more abstract. We then discuss what it means for a network to *learn* a compositional language, based on which we derive evaluation metrics.

6.5.1 Defining a language

A *language* L is a subset of Σ^*, where Σ denotes an alphabet and s denotes a string:

$$\Sigma^* = \{s \mid \forall i = 1, \ldots, |s| \quad s_i \in \Sigma \wedge |s| \geq 0\}\,.$$

We constrain a language to contain only finite-length strings, i.e., $|s| < \infty$, implying that L is a finite language. We use $K_{\max}$ to denote the maximum length of any $s \in L$.

We define a *generator* G from which we can sample one valid string $s \in L$ at a time. It never generates an invalid string and generates all the valid strings in L in finite time such that

$$s \sim G \Leftrightarrow s \in L. \tag{6.5}$$

We define the length $|G|$ of the description of G as the sum of the number of non-terminal symbols $\mathcal{N}$ and the number of production rules $\mathcal{P}$, where $|\mathcal{N}| < \infty$ and $|\mathcal{P}| < \infty$. Each production rule $\rho \in \mathcal{P}$ takes as input an intermediate string $s' \in (\Sigma \cup \mathcal{N})^*$ and outputs another string $s'' \in (\Sigma \cup \mathcal{N})^*$. The generator starts from an empty string $\varnothing$ and applies an applicable production rule, selected uniformly at random, until the output string consists only of characters from the alphabet. These are terminal symbols.

6.5.2 Defining compositionality of a language

When the number of such production rules $|\mathcal{P}|$ plus the number of intermediate symbols $|\mathcal{N}|$ is smaller than the size $|L|$ of the language that G generates, we call L a compositional language. In other words, L is compositional if and only if $|L| > |\mathcal{P}| + |\mathcal{N}|$.

One such example is when we have sixty characters in the alphabet, $\Sigma = \{0, 1, 2, \ldots, 59\}$, and six intermediate symbols, $\mathcal{N} = \{C_1, C_2, \ldots, C_6\}$, for a total of $6 \times 10 + 1$ production rules $\mathcal{P}$:

- $\varnothing \longrightarrow C_1C_2C_3C_4C_5C_6$.

- For each $i \in [1, 6]$, $C_i \longrightarrow w$, where $w \in \{10i - 10, \ldots, 10i - 1\}$

From these production rules and intermediate symbols, we obtain a language of size $10^6 \gg 67 = |\mathcal{P}| + |\mathcal{N}|$, and consequently consider it to be compositional. This is also the language that we use in our experiments in Section 6.7.

Returning to the prior abstract definition, we see that the view of compositionality as 'the ability to combine and recombine a relatively small number of elements to create a vast number of complex meaningful expressions' coincides with this analytical one. The referenced elements are the union of the symbols and production rules, and the meaningful expressions are the strings in the language at large. That there are a 'small' number of the former 'relative' to the 'vast' number of the latter is important.

6.5.3 Learning a language

With that setting in mind, we consider the problem of learning an underlying language $L^\star$ from a finite set of training strings randomly drawn from it:

$$D = \left\{ s | s \sim G^\star \right\}$$

where $G^\star$ is the minimal length generator associated with $L^\star$. We assume $|D| \ll |L^\star|$ and our goal is to use D to learn a language L that approximates $L^\star$ as well as possible. We know that there exists an equivalent generator G for L, and so our problem becomes estimating a generator from this finite set rather than reconstructing an entire set of strings belonging to the original language L^*.

We cast the problem of estimating a generator G as density modeling, in which case the goal is to estimate a distribution $p(s)$. Sampling from $p(s)$ is equivalent to generating a string from the generator G. Language learning is then

$$\max_{p \in \mathcal{M}} \frac{1}{|D|} \sum_{n=1}^{|D|} \log p(s_n) + \lambda R(p), \tag{6.6}$$

where R is a regularization term, λ its strength, and $\mathcal{M}$ is a model space.

Evaluation metrics When the language was learned perfectly, any string sampled from the learned distribution $p(s)$ must belong to $L^\star$. Also, any string in $L^\star$ must be assigned a non-zero probability under $p(s)$. Otherwise, the set of strings generated from this generator, implicitly defined via $p(s)$, is not identical to the original language $L^\star$. This observation leads to two metrics for evaluating the quality of the estimated language with the distribution $p(s)$, *precision* and *recall*:

$$\text{Precision}(L^\star, p) = \frac{1}{|L^\star|} \sum_{s \in L} \mathbb{I}(s \in L^\star) \tag{6.7}$$

$$\text{Recall}(L^\star, p) = \sum_{s \in L^\star} \log p(s) \tag{6.8}$$

where $\mathbb{I}(x)$ is the indicator function. These metrics are designed to be fit for any compositional structure rather than one-off evaluation approaches. Because these are often intractable to compute, we approximate them using Monte-Carlo by sampling N samples from $p(s)$ for calculating precision and M uniform samples from $L^\star$ for calculating recall.

$$\text{Precision}(L^\star, p) \approx \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(s_n \in L^\star) \tag{6.9}$$

$$\text{Recall}(L^\star, p) \approx \sum_{m=1}^{M} \log p(s_m), \tag{6.10}$$

where $s_n \sim p(s)$ and s_m is a uniform sample from $L^\star$.

6.5.4 Compositionality, learning, and capacity

When learning an underlying compositional language $L^\star$, there are three possible outcomes:

Overfitting: $p(s)$ could memorize all the strings that were presented when solving Eq. equation 6.6 and assign non-zero probabilities to those strings and zero probabilities to all others. This would maximize precision, but recall will be low as the estimated generator does not cover $L^\star$.

Systematic generalization: $p(s)$ could capture the underlying compositional structures of $L^\star$ characterized by the production rules $\mathcal{P}$ and intermediate symbols $\mathcal{N}$. In this case, $p(s)$ will assign non-zero probabilities to all the strings that are reachable via these production rules (and zero probability to all others) and generalize to strings from $L^\star$ that were unseen during training, leading to high precision and recall. This behavior was characterized in Lake [107].

Failure: $p(s)$ may neither memorize the entire training set nor capture the underlying production rules and intermediate symbols, resulting in both low precision and recall.

We hypothesize that a major factor determining the compositionality of the resulting language is the capacity of the most complicated distribution $p(s)$ within the model space $\mathcal{M}$.[1] When the model capacity is too high, the first case of total memorization is likely. When it is too low, the third case of catastrophic failure will happen. Only when the model capacity is just right will language learning correctly capture the compositional structure underlying the original language $L^\star$ and exhibit systematic generalization [9]. We empirically investigate this hypothesis using a neural network as a language learner, in particular a variational autoencoder with a discrete sequence bottleneck. This admits the interpretation of a two-player ReferIt game [118] in addition to being a density estimator of $p(s)$. Together, these let us use recall and precision to analyze the resulting emergent language.

[1] As the definition of a model's capacity heavily depends on the specific construction, we do not concretely define it here but do later when we introduce a specific family of models with which we run experiments.

6.6 Variational autoencoders and their capacity

A variational autoencoder [96] consists of two neural networks which are often referred to as an encoder f_θ, a decoder g_ϕ, and a prior p_λ. These two networks are jointly updated to maximize the variational lower bound to the marginal log-probability of training instances s:

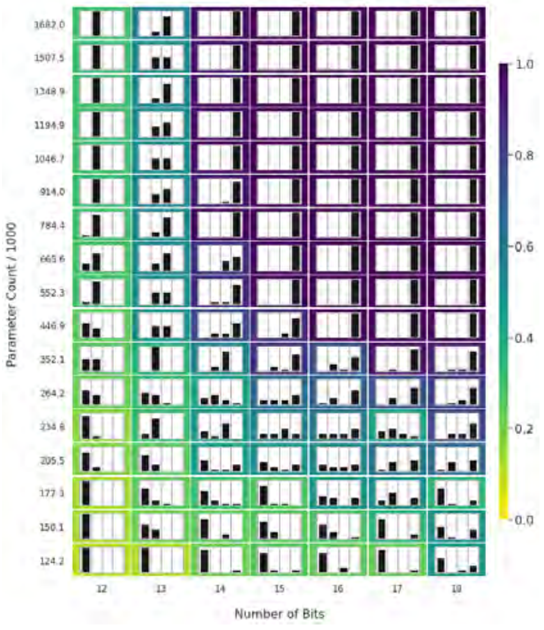
$$\mathcal{L}(\theta, \phi, \lambda; s) = \mathbb{E}_{z \sim f_\theta(z|s)} \left[\log g_\phi(s|z) \right] - \mathrm{KL}(f_\theta(z|s) \| p_\lambda). \tag{6.11}$$

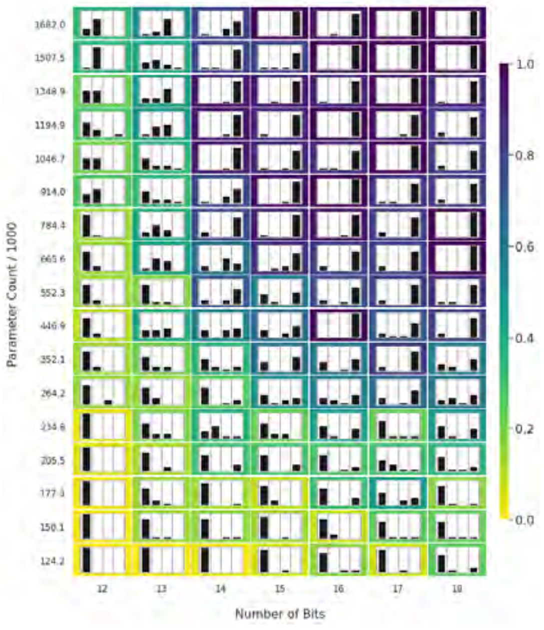
We use $\mathcal{L}$ as a proxy to the true $p(s)$ captured by this model. Once trained, we can efficiently sample $\tilde{z}$ from the prior p_λ and then sample a string from $g_\phi(s|\tilde{z})$.

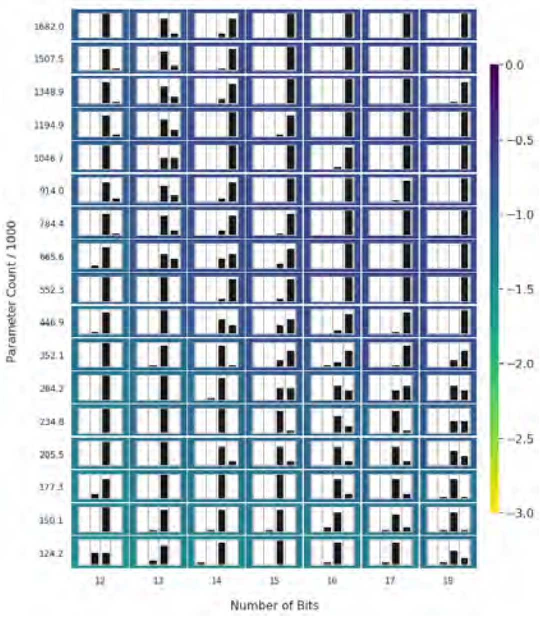
The usual formulation of variational autoencoders uses a continuous latent variable z, which conveniently admits reparametrization that reduces the variance in the gradient estimate. However, this infinitely large latent variable space makes it difficult to understand the resulting capacity of the model. We thus constrain the latent variable to be a binary string of a fixed length l, i.e., $z \in \{0, 1\}^l$. Assuming deterministic decoding, i.e., $\arg\max_s \log g_\phi(s|z)$, this puts a strict upperbound of 2^l on the size of the language $|L|$ captured by the variational autoencoder.

6.6.1 Variational autoencoder as a communication channel

As described above, using variational autoencoders with a discrete sequence bottleneck allows us to analyze the capacity of the model in terms of computation and bandwidth. We can now interpret this variational autoencoder as a communication channel in which a novel protocol must emerge as a by-product of learning. We will refer to the encoder as the speaker and the decoder as the listener when deployed in this communication game. If each string $s \in L^\star$ in the original language is a description of underlying concepts, then the goal of the speaker f_θ is to encode those concepts in a binary string z following an emergent communication protocol. The listener g_ϕ receives this string and must interpret which set of concepts were originally seen by the speaker.


(a) Model A Training


(b) Model A Testing


(c) Model A Recall

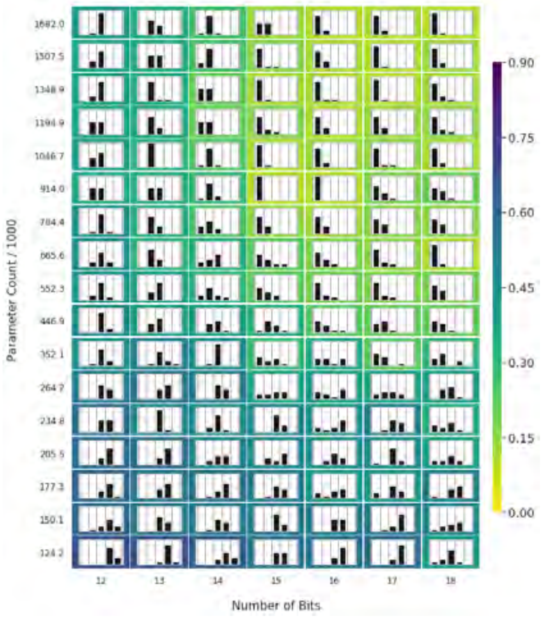

(d) Model A Entropy

Figure 6.3: Results when running Model A with $N = 4$ categories instead of $N = 6$. We show results for bits 12 to 18 and parameter range $124k$ to $1682k$. We need at least $\lceil \log_2 10^4 \rceil = 14$ bits to cover all the input combinations. Observe that there is not much difference to the $N = 6$ scenario show in fig 6.2. See § 6.5.3 and §6.7.1 for the definitions of recall and entropy.

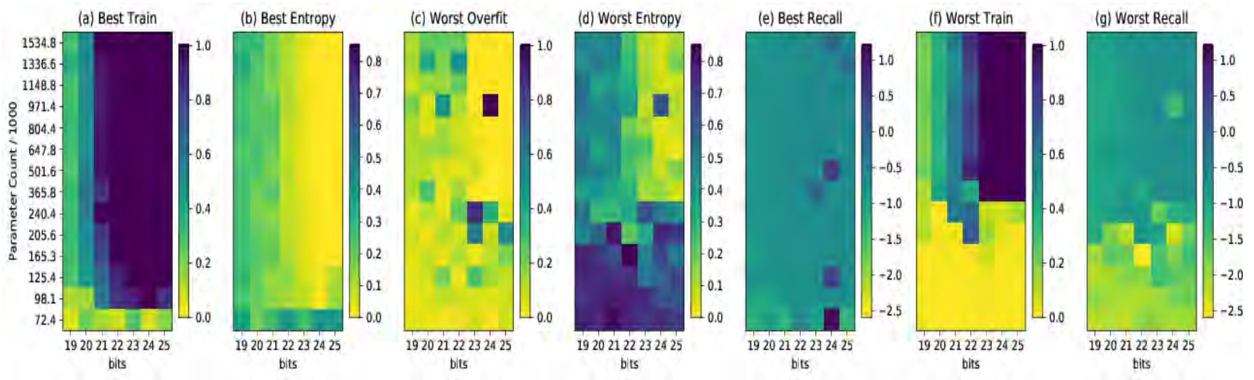


Figure 6.4: Main results for model A showing best and worst performances of the proposed metrics over 10 seeds. See Section 6.7.2 for detailed analysis. Panels (a) and (f) show the accuracy of the training data, (b) and (d) show entropy, (e) and (g) show recall over the test data, and (c) plots the max difference in accuracy between training and test.

Our setup We simplify and assume that each of the characters in the string $s \in L^\star$ correspond to underlying concepts. While the inputs are ordered according to the sequential concepts, our model encodes them using a bag of words (BoW) representation.

The speaker f_θ is parameterized using a recurrent policy which receives the sequence of concatenated one-hot input tokens of s and converts each of them to an embedding. It then runs an LSTM [79] non-autoregressively for l timesteps taking the flattened representation of the input embeddings as its input and linearly projecting each result to a probability distribution over $\{0, 1\}$. This results in a sequential Bernoulli distribution over l latent variables:

$$f_\theta(z|s) = \prod_{t=1}^{l} p(z_t|s;\theta)$$

From this distribution, we can sample a latent string $z = (z_1, \ldots, z_l)$. This interpretation of the probability distribution as a weighted language was previously investigated and supported by Chen et al. [39].

The listener g_ϕ receives z and uses a BoW representation to encode them into its own embedding space. Taking the flattened representation of these embeddings as input, we run an LSTM for $|\mathcal{N}|$ time steps, each time outputting a probability distribution over the full alphabet Σ:

$$g_\phi(s|z) = \prod_{j=1}^{|\mathcal{N}|} p(s_j|z;\phi)$$

To train the whole system end-to-end [185; 133] via backpropogation, we apply a continuous approximation to z_t that depends on a learned temperature parameter τ. We use the 'straight-through' version of Gumbel-Softmax [88; 126] to convert the continuous distribution to a discrete distribution for each z_t. This corresponds to the original discrete distribution in the zero-temperature limit. The final sequence of one hot vectors encoding z is our *message*, which is passed to the listener g_ϕ. If $G_j \sim \text{Gumbel}(0,1)$ and the Bernoulli random variable corresponding to z_t has class probabilities z_{t_0} and z_{t_1}, then $z_t = \texttt{one\_hot}(\arg\max_j[G_j + \log z_{t_j}])$.

The prior p_λ encodes the *message* z using a BoW representation. It gives the probability of z according to the prior (binary) distribution for each z_t and is defined as follows:

$$p_\lambda(z) = \prod_{t=1}^{l} p(z_t|\lambda).$$

This can be used both to compute the prior probability of a latent string and also to efficiently sample from p_λ using ancestral sampling. Penalizing the KL divergence between the speaker's distribution and the prior distribution in Eq. equation 6.11 encourages the emergent protocol to use latent strings that are as diverse as possible.

6.6.2 Capacity of a variational autoencoder with discrete sequence bottleneck

This view of a variational autoencoder with discrete sequence bottleneck presents an opportunity for us to separate the model's capacity into two parts. The first part is the capacity of the communication channel, imposed by the size of the latent variable. As described earlier, the size of the original language $L^\star$ that can be perfectly captured by this model is strictly upper bounded by 2^l, where l is the preset length of the latent string z. If $l < \log_2 |L^\star|$, the model will not be able to learn the language completely, although it may memorize all the training strings $s \in D$ if $l \geq \log_2 |D|$. A resulting question is whether $2^l \geq |L^\star|$ is a sufficient condition for the model to learn $L^\star$ from a finite set of training strings.

The second part involves the capacity of the speaker and listener to map between the latent variable z and a string s in the original language $L^\star$. Taking the parameter count as a proxy to the number of patterns that could be memorized by a neural network,[2] we can argue that the problem can be solved if the speaker and listener each have $\Omega(l|L^\star|)$ parameters, in which case they can implement a hashmap between a string in the original language $L^\star$ and that of the learned latent language defined by the z strings.

However, when the underlying language is compositional as defined in §**??**, we can have a much more compact representation of the entire language than a hashmap. Given the status quo understanding of neural networks, it is impossible to correlate the parameter count with the language specification (production rules and intermediate symbols) and the complexity of using that language. It is however reasonable to assume that there is a monotonic relationship between the number of parameters $|\theta|$, or $|\phi|$, and the capacity of the network to encode the compositional structures underlying the original language [43]. Thus, we use the parameter count as a proxy to measure the capacity.

In summary, there are two axes in determining the capacity of the proposed model: the length of the latent sequence l and the number of parameters $|\theta|$ ($|\phi|$) in the speaker (listener).[3] We vary these two quantities in the experiments and investigate how they affect compositional language learning by the model.

[2]This is true in certain scenarios such as radial-basis function networks.

[3]We design the variational autoencoder to be symmetric so that the parameter counts of the speaker and the listener are roughly the same.

6.6.3 Implications and hypotheses on compositionality

Under this framework for language learning, we can make the following observations:

- If the length of the latent sequence $l < \log_2 |L^\star|$, it is impossible for the model to avoid the failure case because there will be $|L^\star| - 2^l$ strings in $L^\star$ that cannot be generated from the trained model. Consequently, recall cannot be maximized. However, this may be difficult to check using the sample-based estimate as the chance of sampling $s \in L^\star \backslash \int g_\phi(s|z)p_\lambda(z)\mathrm{d}z$ decreases proportionally to the size of $L^\star$. This is especially true when the gap $|L^\star| - 2^l$ is narrow.

- When $l \geq \log_2 |L^\star|$, there are three cases. The first is when there are not enough parameters θ to learn the underlying compositional grammar given by $\mathcal{P}$, $\mathcal{N}$, and Σ, in which case $L^\star$ cannot be learned. The second case is when the number of parameters $|\theta|$ is greater than that required to store all the training strings, i.e., $|\theta| = O(l|D|)$. Here, it is highly likely for the model to overfit as it can map each training string with a unique latent string without having to learn any of $L^\star$'s compositional structure. Lastly, when the number of parameters lies in between these two poles, we hypothesize that the model will capture the underlying compositional structure and exhibit systematic generalization.

In short, we hypothesize that the effectiveness of compositional language learning is maximized when both the length of the latent sequence is large enough ($l \geq \log |L^\star|$), and the number of parameters $|\theta|$ is between $k(|\mathcal{P}|+|\mathcal{N}|+|\Sigma|)$ and $kl|D|$ for some positive integer k. Our experiments test this by varying the length of the latent sequence l and the number of parameters $|\theta|$ while checking the sample-based estimates of precision and recall (Eq. equation 6.9–equation 6.10).

6.7 Experiments

	Model A	Model B
speaker Total	708k	670k
listener Total	825k	690k
speaker Embedding	100	40
speaker LSTM	200	300
speaker Linear	300	60
listener Embedding	300	125
listener LSTM	300	325

Table 6.1: The base hyperparameter counts used in our experiments for each of models A and B.

DATA As described in §??, we run experiments where the size of the language is much larger than the number of production rules. The task is to communicate 6 concepts, each of which have 10 possible values with a total dataset size of 10^6. We build three finite datasets D_{train}, D_{val}, D_{test}:

$$
\begin{aligned}
S_{\text{train}} &= \left\{ s \in L^{\star} \mid s \neq (*C_1^{\text{val}} * C_2^{\text{val}} *) \wedge s \neq (*C_1^{\text{test}} * C_2^{\text{test}} *) \right\} \\
S_{\text{val}} &= \left\{ s = (*C_1^{\text{val}} * C_2^{\text{val}} *) \right\} \\
S_{\text{test}} &= \left\{ s = (*C_1^{\text{test}} * C_2^{\text{test}} *) \wedge s \notin D_{\text{val}} \right\} \\
D_{\text{train}} &= \text{subsample}\left(S_{\text{train}}, N_{\text{train}}\right) \\
D_{\text{val}} &= \text{subsample}\left(S_{\text{val}}, N_{\text{val}}\right) \\
D_{\text{test}} &= \text{subsample}\left(S_{\text{test}}, N_{\text{val}}\right),
\end{aligned}
$$

where subsample(S, N) uniformly selects N random items from S without replacement. The randomly selected concept values $(C_1^{\text{val}}, C_2^{\text{val}})$ and $(C_1^{\text{test}}, C_2^{\text{test}})$ ensure that concept combinations are unique to each set. The $*$ symbol refers to any number of concepts, as in regular expressions.

MODELS AND LEARNING We train the proposed variational autoencoder (described in §6.6.1) on D_{train}, using the Adam optimizer [95] with a learning rate of 3×10^{-3}, weight decay coefficient of 10^{-4} and a batch size of 1000. The Gumbel-Softmax temperature parameter τ in is initialized to 1. Since systematic generalization may only happen in some training runs [206], each model is trained for each of 10 seeds over $200k$ steps. We trained our models using [145].

We train two sets of models. Each set is built from an independent base model, the architectures of which are described in Table 6.1. We gradually decrease the number of LSTM units from the base model by a factor $\alpha \in (0, 1]$. This is how we control the number of parameters ($|\theta|$ and $|\phi|$), a factor we hypothesize to influence the resulting compositionality. We obtain seven models from each of these by varying the length of the latent sequence l from $\{19, 20, 21, 22, 23, 24, 25\}$. These were chosen because we both wanted to show a range of bits and because we need at least 20 bits to cover the 10^6 strings in L^* ($\lceil \log_2 10^6 \rceil = 20$).

Note that results for the two models are similar and so we arbitrarily spotlight one of them - model A. We additionally show the results for model B in Figs. 6.5, 6.6, and 6.8, but do not dive into its results in the main section.

6.7.1 Evaluation: Residual Entropy

Our setup allows us to design a metric by which we can check the compositionality of the learned language L by examining how the underlying concepts are described by a string. For instance, $(2, 11, 24, 31, 44, 56) \in L^\star$ describes that $C_1 = 2$, $C_2 = 11$, $C_3 = 24$, $C_4 = 31$, $C_5 = 44$ and $C_6 = 56$. Furthermore, we know that the value of a concept C_i is independent of the other concepts $C_{j \neq i}$, and so our custom generative setup with a discrete latent sequence allows us to inspect a learned language L by considering z.

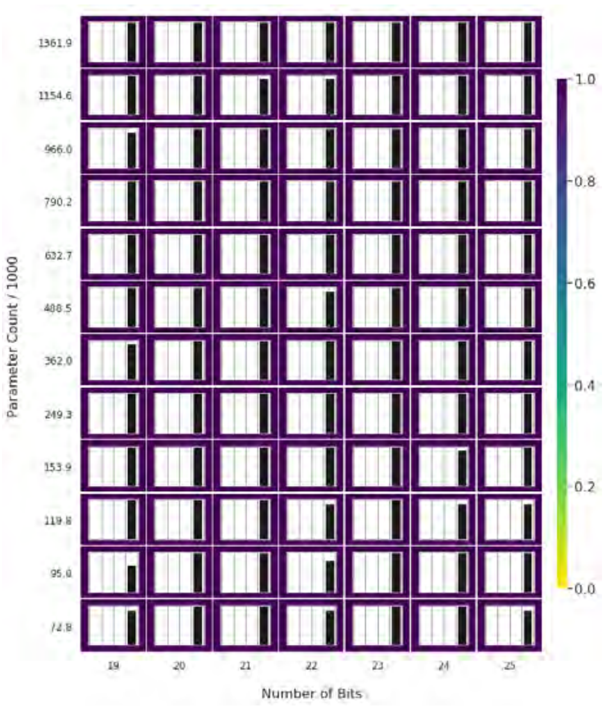
Let p be a sequence of partitions of $\{1, 2, \ldots, l\}$. We define the degree of compositionality as the ratio between the variability of each concept C_i and the variability explained by a latent subsequence $z[p_i]$ indexed by an associated partition p_i. More formally, the degree of compositionality given the partition sequence p is defined as a residual entropy

$$\mathrm{re}(p, L, L^\star) = \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} \mathcal{H}_L(C_i | z[p_i]) / \mathcal{H}_{L^\star}(C_i)$$

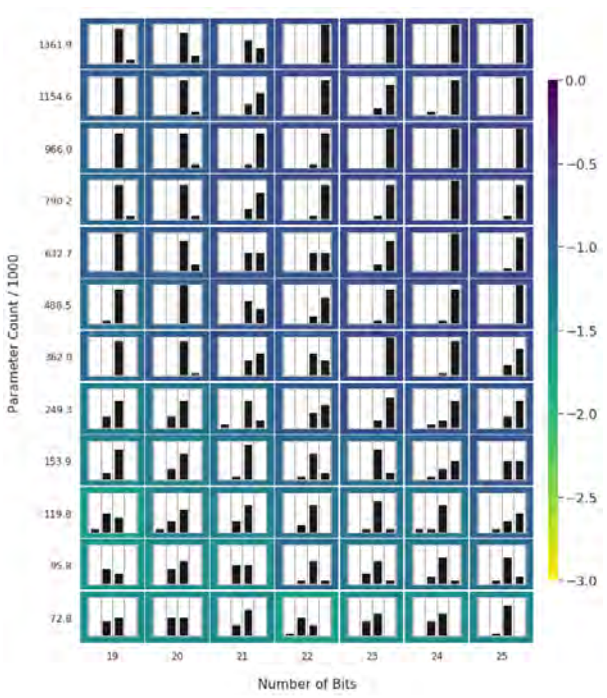
where there are $|\mathcal{N}|$ concepts by the definition of our language. When each term inside the summation is close to zero, it implies that a subsequence $z[p_i]$ explains most of the variability of the specific concept C_i, and we consider this situation compositional. The residual entropy of a trained model is then the smallest $\text{re}(p)$ over all possible sequences of partitions $\mathcal{P}$ and spans from 0 (compositional) to 1 (non-compositional).

$$\text{re}(L, L^\star) = \min_{p \in \mathcal{P}} \text{re}(p, L, L^\star).$$

6.7.2 Analysis



(a) Model B Precision



(b) Model B Recall

Figure 6.5: Precision and recall for model B. Like model A, model B has perfect precision. However, its recall chart shows a different story. The first takeaway is that while there is still a strong region in the top right bounded below by $\sim 360k$ parameters, it does not extend to 22 bits on the left side. This supports our notion of a minimal capacity threshold but adds a wrinkle in that this architecture influences the model's ability to succeed with fewer bits.

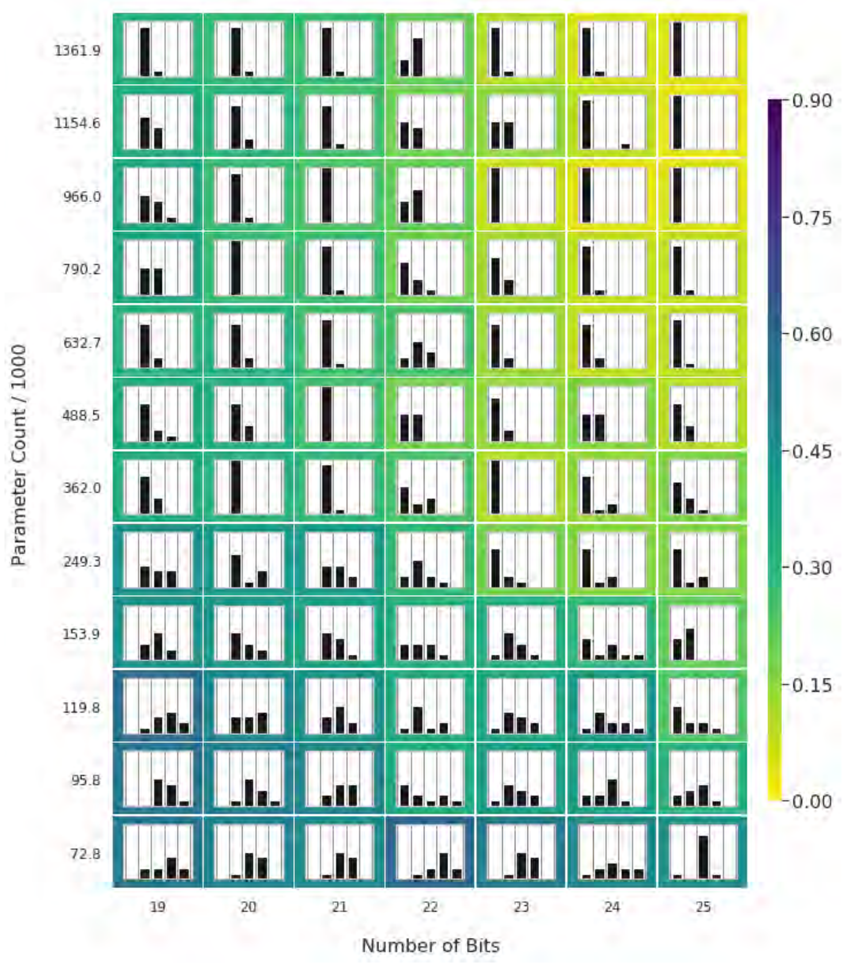


Figure 6.6: Entropy metric for model B as described in §6.7.1. Similar to model A, we see that model B's entropy supports the view we had of its recall in Figure 6.5 - there is a strong region in the top right that supports the notion of a minimal capacity and bit threshold.

Fig. 6.4 shows the main findings of our research. In plot (a), we see the parameter counts at the threshold. Below these values, the model cannot solve the task but above these, it can solve it. Further, observe the curve delineated by the lower left corner of the shift from unsuccessful to successful models. This inverse relationship between bits and parameters shows that the more parameters in the model, the fewer bits it needs to solve the task. Note however that it could only solve the task with fewer bits if it was forming a non-compositional code, suggesting that higher parameter models are able to do so while lower parameter ones cannot.

Observe further that all of our models above the minimum threshold (72,400) have the capacity to learn a compositional code. This is shown by the perfect training accuracy achieved by all of those models in plot (a) for 24 bits and by the perfect compositionality (zero entropy) in plot (b) for 24 bits. Together with the above, this validates that learning compositional codes requires less capacity than learning non-compositional codes.

Plot (c) confirms our hypothesis that large models can memorize the entire dataset. The 24 bit model with 971,400 parameters achieves a train accuracy of 1.0 and a validation accuracy of 0.0. Cross-validating this with plots (d) and (g), we find that a member of the same parameter class is non-compositional and that there is one that achieves unusually low recall. We verified that these are all the same seed, which shows that the agents in this model are memorizing the dataset. This is supplemented by Fig. 6.7 where we see the non-compositional behavior by plotting residual entropy versus overfitting.

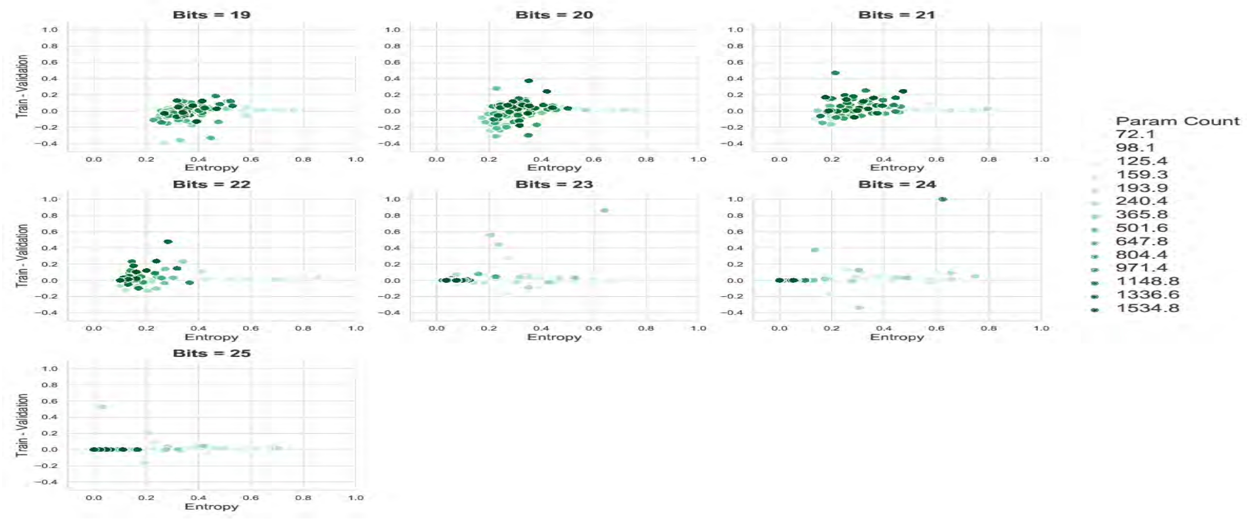


Figure 6.7: Model A Entropy vs Overfitting: Charts showing per-bit results for Entropy vs (Train - Validation) over the parameter range. Observe the two models in bits 23 and 24 which were too successful in producing a non-compositional code and consequently overfit to the data.

Plots (b) and (e) show that our compositionality metrics pass two sanity checks - high recall and perfect entropy can only be achieved with a channel that is sufficiently large (i.e. 24 bits) to allow for a compositional latent representation.

Plot (f) shows that while the capacity does not affect the ability to learn a compositional language across the model range, it does change the *learnability*. Here we find that smaller models can fail to solve the task for any bandwidth, which coincides with literature suggesting a link between overparameterization and learnability [121; 55]. This is supported by the efficacy results in Fig. 6.8.

Finally, as expected, we find that no model learns to solve the task with < 20 bits, validating that the minimum required number of bits for learning a language of size $|L|$ is $\lceil \log(|L|) \rceil$. We also see that no model learns to solve it for 20 bits, which is likely due to optimization difficulties.

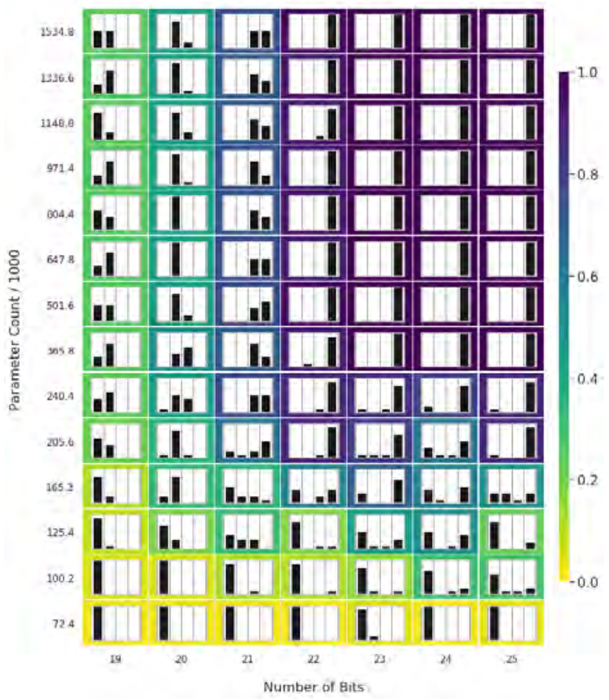
In Fig. 6.2, we present histograms showing precision, recall and residual entropy measured for each bit and parameter combination over the test set. The histograms show the distributions of these metrics, upon which we make a number of observations.

We first confirm the effectiveness of training by observing that almost all the models achieve perfect precision (Fig. 6.2 (a)), implying that $L \subseteq L^{\star}$, where L is the language learned by the model. This occurs even with our learning objective in Eq. equation 6.11 encouraging the model to capture all training strings rather than to focus on only a few training strings.

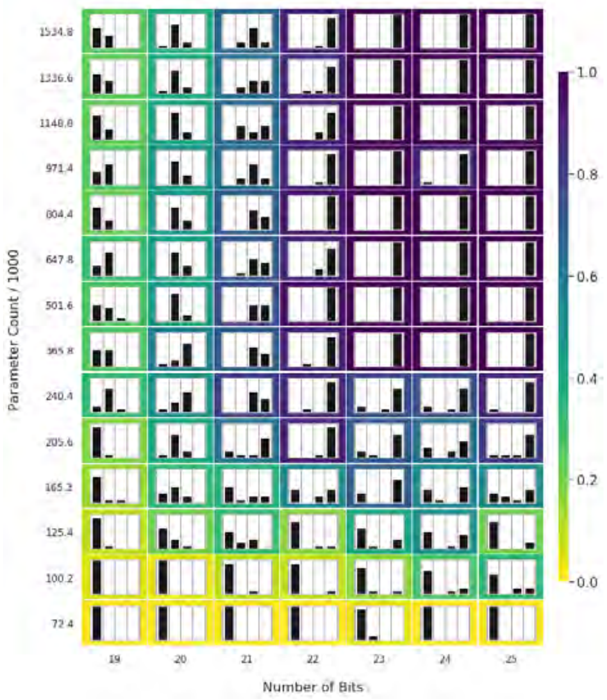
A natural follow-up question is how large is $L^{\star} \backslash L$. We measure this with recall in Fig. 6.2 (b), which shows a clear phase transition according to the model capacity when $l \geq 22$. This agrees with what we saw in Fig. 6.4 and is equivalent to saying $|L^{\star} \backslash L| \gg 0$ at a value that is close to our predicted boundary of $l = \lceil \log_2 10^6 \rceil = 20$. We attribute this gap to the difficulty in learning a perfectly-parameterized neural network.

In Fig. 6.7 we show the empirical relation between entropy and overfitting over the parameter range. While it was hard for the models to overfit, there were some in bits 23 and 24 that did do so. For those models, the entropy was also found to be higher relative to the other models.

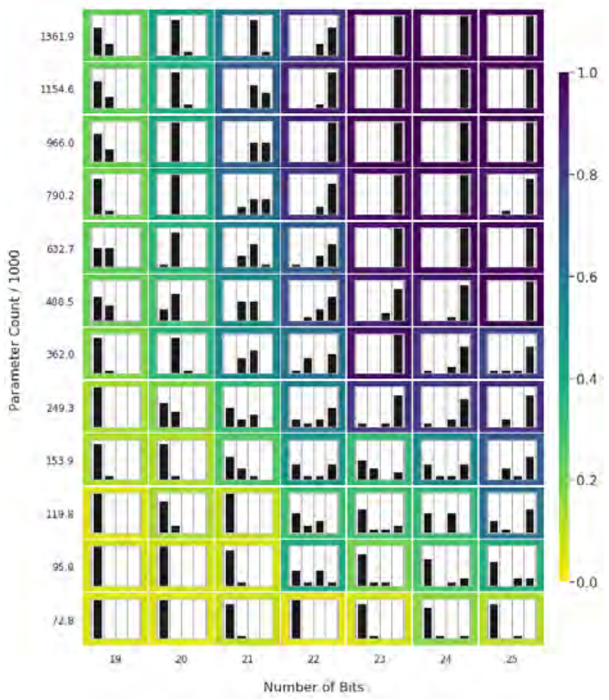
Even when $l \geq 22$, we observe training runs that fail to achieve optimal recall when the number of parameters is ≤ 365800 (Fig. 6.2). Due to insufficient understanding of the relationship between the number of parameters and the capacity in a deep neural network, we cannot make



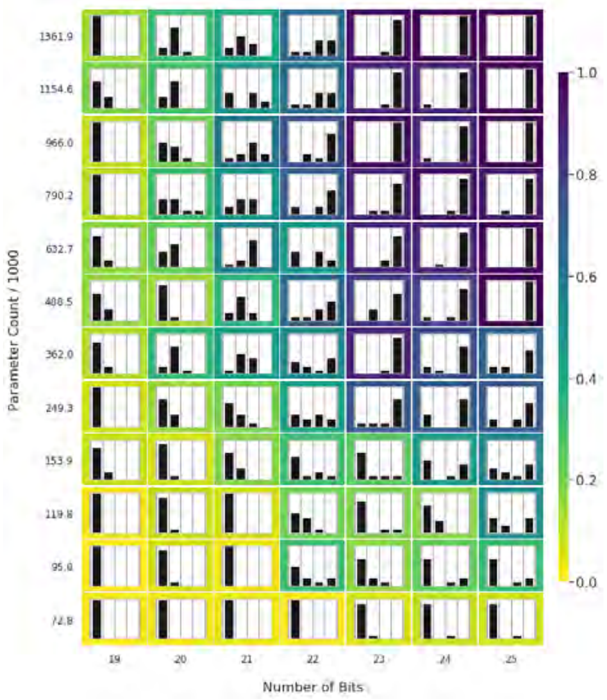
(a) Model A Training



(b) Model A Testing



(c) Model B Training



(d) Model B Testing

Figure 6.8: Efficacy results for models A and B.

a rigorous conclusion. We however conjecture that this is the upperbound to the minimal model capacity necessary to capture the tested compositional language. Above this threshold, the recall is almost always perfect, implying that the model has likely captured the compositional structure underlying $L^\star$ from a finite set of training strings. We run further experiments up to $1.5M$ parameters, but do not observe the expected overfitting.

As shown in Fig. 6.3, we also run experiments with the number of categories reduced from 6 to 4 and similarly do not find the upperbound. It is left for future studies to determine why. Two conjectures that we have are that it is either due to insufficient model capacity to memorize the hash map between all the strings in $L^\star$ and 2^l latent strings or due to an inclination towards compositionality in our variational autoencoder.

These results clearly confirm the first part of our hypothesis - the latent sequence length must be at least as large as $\log |L^\star|$. They also confirm that there is a lowerbound on the number of parameters over which this model can successfully learn the underlying language. We have not been able to verify the upper bound in our experiments, which may require either a more (computationally) extensive set of experiments with even more parameters or a better theoretical understanding of the inherent biases behind learning with this architecture, such as from recent work on overparameterized models [15; 136].

6.8 Conclusion

In this contribution, we hypothesize a thus far ignored connection between learnability, capacity, bandwidth, and compositionality for language learning. We empirically verify that learning the underlying compositional structure requires less capacity than memorizing a dataset. We also introduce a set of metrics to analyze the compositional properties of a learned language. These metrics are not only well motivated by theoretical insights, but are cross-validated by our task-specific metric.

This contribution opens the door for a vast amount of follow-up research. All our models were sufficiently large to represent the compositional structure of the language when given sufficient bandwidth, however there should be an upper bound for representing this compositional structure that we did not reach. We consider answering that to be the foremost question.

Furthermore, while large models did overfit, this was an exception rather than the rule. We hypothesize that this is due to the large number of examples in our language, which almost forces the model to generalize, but note that there are likely additional biases at play that warrant further investigation.

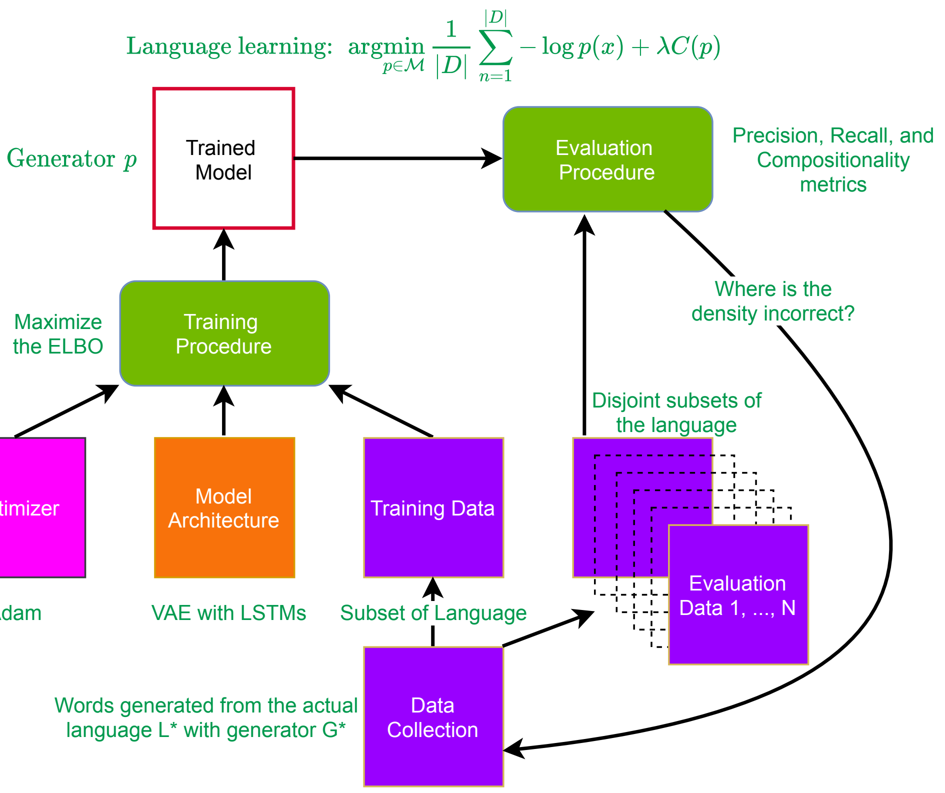

Language learning: $\operatorname{argmin}_{p\in\mathcal{M}} \frac{1}{|D|}\sum_{n=1}^{|D|} -\log p(x) + \lambda C(p)$
Generator p
Trained Model
Evaluation Procedure
Precision, Recall, and Compositionality metrics
Where is the density incorrect?
Maximize the ELBO
Training Procedure
Disjoint subsets of the language
Optimizer
Model Architecture
Training Data
Evaluation Data 1, ..., N
Adam
VAE with LSTMs
Subset of Language
Words generated from the actual language L* with generator G*
Data Collection

Figure 6.9: The Machine Learning Application Framework for this contribution. With a goal of having agents learn languages that are compositional, we investigated an hypothesis that constrains the model architecture to be within certain bounds conditioned on the size of the training data. The resulting understanding lets us rule out classes of model architectures based on their capacity. The improvement to the application framework is found in that our space of possible models to consider is now tighter.

6.9 What did we learn about the most challenging cases?

We have acquired understanding into one of the root causes of compositionality - the agent's capacity - and how that relates to whether the learned language is actually compositional. As described in Section 6.2, this principle is core to downstream learning such that we minimize the difficulties faced with the most challenging cases. Agents with a better understanding of the building blocks of the world and how they compose are agents that can quickly adapt to a new and challenging group.

This work is firmly on the path towards our continued goal of building a **communicative** and visually aware robot that can capably learn new tasks. The communicative part is exactly the application in which we demonstrated our study. Even more important though is the understanding of model architecture and how that affects learning compositional structure because it underlies not just communication but all forms of learning in real-world settings, including vision and task-learning. The Machine Learning Application Framework for this contribution (Figure 6.9) highlights this. We see there how research that constrains the model architecture to be more likely to elicit compositional structure positively effects learning efficiency and consequently the entire application loop.

7 | Conclusion

This dissertation began with a recognition of the long standing societal goal to have household robots. The core of the thesis then focused on the topic of the challenging cases in machine learning. Throughout, I drew a thread between the goal to have robots and the most challenging cases by appealing to a need to identify, address, and understand the latter in order to make headway on deploying such robotic systems in the wild. I specifically recognized three of my contributions that target different parts of the stack with respect to solving for both the most challenging cases and for producing communicative and visually aware robots who can capably learn household tasks.

Before concluding, I am going to synthesize parts of my research into a new direction that I think has exciting promise for the rearrangement task.

7.1 Equivariant Scene Understanding

Assume a house scene with a robot that has to manipulate objects. The objects may be on or in a counter or nearby, and the agent has a specified task such as 'put the can opener on the counter.'

A baseline is to train a model-free agent to do just this task. This is hard because the sample complexity of learning this task from RGB images is very large and simulated approaches have a persistent sim2real gap [165; 86]. We posit that a better approach would be if we could learn this unsupervised and, ideally, with online updates. I argue that we can do this if we had a model that was equipped with a scene graph, and we trained both the visual awareness and the task learning at the same time.

We start with the following observation - scene graph generation from states is an equivariant function with respect to a family of actions taken by the ego agent. Formally, we have $\mathcal{H} : X \to \mathcal{P}$ a function that produces scene graphs $\mathcal{P} = (V, E)$ from state observations where V and E are the vertices and edges of the graph, $x \in X$ a state observation, $\pi_{\text{agent}} : \text{X} \to \text{X}$ a policy function that acts on state observations and produces another state observation via an intermediate action distribution, and $\pi_{\text{graph}} : \text{P} \to \text{P}$ a policy function that acts on scene graphs and produces another scene graph via an intermediate action distribution. The equivariant equation is then:

$$\mathcal{H}(\pi_{\text{agent}}(x)) = \pi_{\text{graph}}(\mathcal{H}(x)) \tag{7.1}$$

To ground the discussion, as well as a nod to our intent for the eventual robot to be visually aware, we restrict our states to the visual domain as RGB images $x \in X = \mathbb{R}^3$. Intuitively, the left-hand side of Equation 7.1 says that if we have an agent act on its image observation and take an action, then we will be in a new state with a new corresponding image. The scene graph of that new image will be equivalent to the right-hand side. The right-hand side formalizes a single action being taken by a graph policy on the scene graph of the original image.

All together, this equivariance suggests that, for a pair of domain-specific action policies, first acting on a scene and then taking the scene graph is equivalent to first taking the scene graph and then acting on that scene graph. With this idea, we train a robot by equipping it with a learned function for scene graph decomposition and two learned action policies, then simultaneously training all three with an equivariance loss. Denoting loss $L_{\text{equivariance}} = ||\mathcal{H}(\pi_{\text{agent}}(x)) - \pi_{\text{graph}}(\mathcal{H}(x))||$, we have a number of ways we can do this:

- Train a model to minimize $L_{\text{equivariance}}$ only. This has degenerate solutions such as when $\mathcal{H}$ produces the empty graph. That concern diminishes when we use a contrastive loss, which we could do on the same scene or on different scenes. Because the policy has not been trained on any task, we expect this to yield a valid but perhaps not complete scene graph due to the policy's random action distribution.

- Train a model to minimize $L_{\text{equivariance}}$, but also add in other losses such as taking more actions before optimizing: $L^k_{\text{equivariance}} = ||\mathcal{H}(\pi^k_{\text{agent}}(x)) - \pi^k_{\text{graph}}(\mathcal{H}(x))||$. This encourages a more thorough understanding of the scene because more random actions will be taken from the starting state that push us into new parts of the state space.
- Along with optimizing $L_{\text{equivariance}}$, optimize the agent on the task directly using RGB input. This adds in a gradient towards task reward and so we expect the agent to improve its policies and not just take random actions in the world.
- Along with optimizing $L_{\text{equivariance}}$, optimize the agent on the task directly using *scene graph* input. It is likely that this will require a curriculum to succeed, but plausibly would be the ideal type of signal to succeed at all of task learning, visual awareness, and scene graph decomposition simultaneously.

Regardless of the specific training setup, the above paints a picture of this research direction as an unsupervised approach to training a robot in the wild to both perform task learning and be visually aware of its surroundings to the extent that it could decompose a scene. We would likely start by doing it in simulation. I expect that we will run into the most challenging cases frequently because most of the gradient will guide the model towards areas of the scene that it does not understand. In any case, this is a research direction where the Backplay curriculum will be very prescient because task demonstrations would be a fantastic way to bootstrap the learning.

7.2 Parting words

To recap, Part I proposed a method for pseudo-automatically identifying the most challenging cases. It focused on object detection in vision and finds which semantic groups of data are most difficult for perception systems in robots, with an emphasis on autonomous vehicles.

Part II presented an algorithm for addressing the most challenging cases in task learning via episodic reinforcement learning. This algorithm uses one potentially sub-optimal demonstration from which we can train optimal policies with much better sample complexities than in vanilla reinforcement learning.

Finally, Part III focused on understanding the root cause of challenging cases, with an emphasis on language learning in multi-agent settings. We presented work illustrating the central role of agent capacity in determining whether the learned language is actually compositional, an important and desired attribute of the learned language.

The work contained in this thesis is but a small step towards overcoming the difficulties associated with the most challenging cases and a tiny hop towards building a task-learning, visually-aware, and communicative robot. This echoes the goals in our field, which all have a long arc and nonlinear progress. In other words, the path to the Emerald City is long but paved with gold and adventure.

Bibliography

[1] Abeysirigoonawardena, Y., Shkurti, F., and Dudek, G. (2019). Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277.

[2] Ali, A., Cauchois, M., and Duchi, J. C. (2022). The lifecycle of a statistical model: Model failure detection, identification, and refitting.

[3] An, J., Ying, L., and Zhu, Y. (2021). Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients.

[4] Andreas, J. (2019). Measuring compositionality in representation learning. In *International Conference on Learning Representations*.

[5] Arjovsky, M. (2021). Out of distribution generalization in machine learning.

[6] Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2020). Invariant risk minimization.

[7] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2017). Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397.

[8] Aytar, Y., Pfaff, T., Budden, D., Paine, T. L., Wang, Z., and de Freitas, N. (2018). Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*.

[9] Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. (2019). Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

[10] Bahri, Y., Dyer, E., Kaplan, J., Lee, J., and Sharma, U. (2021). Explaining neural scaling laws. *CoRR*, abs/2102.06701.

[11] Bain, M. and Sommut, C. (1999). A framework for behavioural claning. *Machine intelligence*, 15(15):103.

[12] Balduzzi, D., Garnelo, M., Bachrach, Y., Czarnecki, W. M., Perolat, J., Jaderberg, M., and Graepel, T. (2019). Open-ended learning in symmetric zero-sum games.

[13] Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., Savva, M., and Su, H. (2020). Rearrangement: A challenge for embodied ai.

[14] Behbahani, F., Shiarlis, K., Chen, X., Kurin, V., Kasewa, S., Stirbu, C., Gomes, J., Paul, S., Oliehoek, F. A., Messias, J., et al. (2018). Learning from demonstration in the wild. *arXiv preprint arXiv:1811.03516*.

[15] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

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

[17] Ben-Tal, A., den Hertog, D., Waegenaere, A. D., Melenberg, B., and Rennen, G. (2013). Robust solutions of optimization problems affected by uncertain probabilities. *Manag. Sci.*, 59:341–357.

[18] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

[19] Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. *CoRR*, abs/1305.6663.

[20] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680.

[21] Bó, P. D. (2005). Cooperation under the shadow of the future: experimental evidence from infinitely repeated games. *American economic review*, 95(5):1591–1604.

[22] Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., and Kalai, A. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

[23] Bomberman, W. (1983). Wikipedia bomberman. https://en.wikipedia.org/wiki/Bomberman.

[24] Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2017). Heads-up limit hold'em poker is solved. *Commun. ACM*, 60(11):81–88.

[25] Brown, N. and Sandholm, T. (2017). Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, pages 689–699.

[26] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin,

M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[27] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

[28] Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *FAT*.

[29] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning.

[30] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2019). nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*.

[31] Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

[32] CAPLAN, L. R. (2001). Evidence based medicine: concerns of a clinical neurologist. *Journal of Neurology, Neurosurgery & Psychiatry*, 71(5):569–574.

[33] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers.

[34] Carpenter, M., Nagell, K., and Tomasello, M. (1998). Social cognition, joint attention, and communicative competence from 9 to 15 months of age. *Monogr Soc Res Child Dev*, 63(4):1–143.

[35] Cavicchio, F., Dachkovsky, S., Leemor, L., Shamay-Tsoory, S., and Sandler, W. (2018). Compositionality in the language of emotion. *PLoS One*, 13(8):e0201970.

[36] Chaabouni, R., Kharitonov, E., Lazaric, A., Dupoux, E., and Baroni, M. (2019). Word-order biases in deep-agent emergent communication. *arXiv:1905.12330 [cs].* arXiv: 1905.12330.

[37] Chang, N., Yu, Z., Wang, Y.-X., Anandkumar, A., Fidler, S., and Alvarez, J. M. (2021). Image-level or object-level? a tale of two resampling strategies for long-tailed detection.

[38] Chen, G.-Y. and Saloff-Coste, L. (2013). On the mixing time and spectral gap for birth and death chains. *arXiv preprint arXiv:1304.4346.*

[39] Chen, Y., Gilroy, S., Maletti, A., May, J., and Knight, K. (2018). Recurrent neural networks as weighted language recognizers. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2261–2271. Association for Computational Linguistics.

[40] Chernozhukov, V., Demirer, M., Duflo, E., and Fernández-Val, I. (2018). Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in india. Working Paper 24678, National Bureau of Economic Research.

[41] Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). Babyai: A platform to study the sample efficiency of grounded language learning.

[42] Chomsky, N. (1957). Logical structures in language. *American Documentation*, 8(4):284–291.

[43] Collins, J., Sohl-Dickstein, J., and Sussillo, D. (2017). Capacity and trainability in recurrent neural networks. In *International Conference on Learning Representations.*

[44] Corso, A., Du, P., Driggs-Campbell, K., and Kochenderfer, M. J. (2019). Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE.

[45] Corso, A., Moss, R. J., Koren, M., Lee, R., and Kochenderfer, M. J. (2020). A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*.

[46] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[47] Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In van den Herik, H. J., Ciancarini, P., and Donkers, H. H. L. M., editors, *Computers and Games*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer.

[48] Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2017a). Embodied question answering. *CoRR*, abs/1711.11543.

[49] Das, A., Kottur, S., Moura, J. M. F., Lee, S., and Batra, D. (2017b). Learning cooperative visual dialog agents with deep reinforcement learning. *CoRR*, abs/1703.06585.

[50] Daumé, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine learning*, 75(3):297–325.

[51] Devaranjan, J., Kar, A., and Fidler, S. (2020). Meta-sim2: Learning to generate synthetic datasets. In *ECCV*.

[52] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[53] Dosovitskiy, A. (2019). carla-simulator/carla.

[54] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

[55] Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019). Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*.

[56] Duchi, J., Glynn, P., and Namkoong, H. (2018). Statistics of robust optimization: A generalized empirical likelihood approach.

[57] Dwibedi, D., Misra, I., and Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310.

[58] Edwards, A. D., Downs, L., and Davidson, J. C. (2018). Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*.

[59] Elman, J. L. (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71–99.

[60] Engel, J. H., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. *CoRR*, abs/1704.01279.

[61] Evtimova, K., Drozdov, A., Kiela, D., and Cho, K. (2018). Emergent communication in a multi-modal, multi-step referential game. In *International Conference on Learning Representations*.

[62] Florensa, C., Held, D., Wulfmeier, M., and Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*.

[63] Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc.

[64] Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2017). Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*.

[65] Gao, Y., Lin, J., Yu, F., Levine, S., Darrell, T., et al. (2018). Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*.

[66] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[67] Ghodsi, Z., Hari, S. K. S., Frosio, I., Tsai, T., Troccoli, A., Keckler, S. W., Garg, S., and Anandkumar, A. (2021). Generating and characterizing scenarios for safety testing of autonomous vehicles. *arXiv preprint arXiv:2103.07403*.

[68] Girshick, R. (2015). Fast r-cnn.

[69] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

[70] Goyal, A., Brakel, P., Fedus, W., Lillicrap, T., Levine, S., Larochelle, H., and Bengio, Y. (2018). Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*.

[71] Gulrajani, I. and Lopez-Paz, D. (2020). In search of lost domain generalization.

[72] Gupta, A., Murali, A., Gandhi, D., and Pinto, L. (2018). Robot learning in homes: Improving generalization and reducing dataset bias.

[73] Havrylov, S. and Titov, I. (2017). Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 2149–2159. Curran Associates, Inc.

[74] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

[75] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[76] Hernandez, C., Bharatheesha, M., Ko, W., Gaiser, H., Tan, J., van Deurzen, K., de Vries, M., Mil, B. V., van Egmond, J., Burger, R., Morariu, M., Ju, J., Gerrmann, X., Ensing, R., Frankenhuyzen, J. V., and Wisse, M. (2016). Team delft's robot winner of the amazon picking challenge 2016.

[77] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., et al. (2017). Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*.

[78] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573.

[79] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[80] Hong, W. and Zhou, K. (2017). A note on the passage time of finite-state markov chains. *Communications in Statistics-Theory and Methods*, 46(1):438–445.

[81] Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. (2020). Characterising bias in compressed models.

[82] Hosu, I.-A. and Rebedea, T. (2016). Playing atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077*.

[83] Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). The compositionality of neural networks: integrating symbolism and connectionism. *Journal of Artificial Intelligence Research*.

[84] Hutchinson, B., Prabhakaran, V., Denton, E., Webster, K., Zhong, Y., and Denuyl, S. (2020). Social biases in nlp models as barriers for persons with disabilities.

[85] Ivanovic, B., Harrison, J., Sharma, A., Chen, M., and Pavone, M. (2018). Barc: Backward reachability curriculum for robotic reinforcement learning. *arXiv preprint arXiv:1806.06161*.

[86] Jakobi, N. (1998). Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In Husbands, P. and Meyer, J.-A., editors, *Evolutionary Robotics*, pages 39–58, Berlin, Heidelberg. Springer Berlin Heidelberg.

[87] James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. (2019). Rlbench: The robot learning benchmark learning environment.

[88] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

[89] Julia Angwin, J. L. (2016). Machine bias.

[90] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In Sammut, C. and Hoffmann, A. G., editors, *ICML*, pages 267–274. Morgan Kaufmann.

[91] Kar, A., Prakash, A., Liu, M.-Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., and Fidler, S. (2019). Meta-sim: Learning to generate synthetic datasets. In *ICCV*.

[Karpathy] Karpathy, A. Cvpr 2021 workshop on autonomous vehicles.

[93] Kent, D. M. and Kitsios, G. D. (2009). Against pragmatism: on efficacy, effectiveness and the real world. *Trials*, 10:48 – 48.

[94] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2021). Transformers in vision: A survey.

[95] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[96] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[97] Kirby, S., Tamariz, M., Cornish, H., and Smith, K. (2015). Compression and communication in the cultural evolution of linguistic structure. *Cognition*, 141:87–102.

[98] Kleiman-Weiner, M., Ho, M. K., Austerweil, J. L., Littman, M. L., and Tenenbaum, J. B. (2016). Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *COGSCI*.

[99] Klein, E. (2020). *Why we're polarized*.

[100] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg. Springer-Verlag.

[101] Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2020). WILDS: A benchmark of in-the-wild distribution shifts. *CoRR*, abs/2012.07421.

[102] Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2019). Ai2-thor: An interactive 3d environment for visual ai.

[103] Konda, V. and Tsitsiklis, J. (2000). Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, pages 1008–1014. MIT Press.

[104] Koren, M., Alsaif, S., Lee, R., and Kochenderfer, M. J. (2018). Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE.

[105] Kottur, S., Moura, J., Lee, S., and Batra, D. (2017). Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967. Association for Computational Linguistics.

[106] Kumar, B. G. V., Harwood, B., Carneiro, G., Reid, I. D., and Drummond, T. (2017). Smart mining for deep metric learning. *CoRR*, abs/1704.01285.

[107] Lake, B. M. (2019). Compositional generalization through meta sequence-to-sequence learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 9788–9798. Curran Associates, Inc.

[108] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Perolat, J., Silver, D., Graepel, T., et al. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203.

[109] Laskey, M., Staszak, S., Hsieh, W. Y.-S., Mahler, J., Pokorny, F. T., Dragan, A. D., and Goldberg, K. (2016). Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 462–469. IEEE.

[110] Lazaridou, A., Hermann, K. M., Tuyls, K., and Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations*.

[111] Lazaridou, A., Peysakhovich, A., and Baroni, M. (2017). Multi-Agent Cooperation and the Emergence of (Natural) Language. In *International Conference on Learning Representations*.

[112] Leclerc, G., Salman, H., Ilyas, A., Vemprala, S., Engstrom, L., Vineet, V., Xiao, K., Zhang, P., Santurkar, S., Yang, G., et al. (2021). 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*.

[113] Lee, J., Cho, K., and Kiela, D. (2019). Countering language drift via visual grounding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4376–4386, Hong Kong, China. Association for Computational Linguistics.

[114] Lee, J., Cho, K., Weston, J., and Kiela, D. (2018). Emergent translation in multi-agent communication. In *International Conference on Learning Representations*.

[115] Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems.

[116] Lerer, A. and Peysakhovich, A. (2017). Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*.

[117] Lerer, A. and Peysakhovich, A. (2018). Learning social conventions in markov games. *arXiv preprint arXiv:1806.10071*.

[118] Lewis, D. (1969). *Convention: A philosophical study*. Harvard University Press.

[119] Lewis, S. (2019). The racial bias built into photography. *The New York Times*, 25.

[120] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541.

[121] Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems 31*, pages 8157–8166. Curran Associates, Inc.

[122] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection.

[123] Liska, A., Kruszewski, G., and Baroni, M. (2018). Memorize or generalize? searching for a compositional RNN in a haystack. *CoRR*, abs/1802.06467.

[124] Liu, H. D., Tao, M., Li, C., Nowrouzezahrai, D., and Jacobson, A. (2018). Adversarial geometry and lighting using a differentiable renderer. *CoRR*, abs/1808.02651.

[125] Lowe*, R., Gupta*, A., Foerster, J., Kiela, D., and Pineau, J. (2020). On the interaction between supervision and self-play in emergent communication. In *International Conference on Learning Representations*.

[126] Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

[127] Mansimov, E., Wang, A., and Cho, K. (2019). A generalized framework of sequence generation with application to undirected sequence models. *CoRR*, abs/1905.12790.

[128] McAleer, S., Agostinelli, F., Shmakov, A., and Baldi, P. (2018). Solving the rubik's cube without human knowledge. *arXiv preprint arXiv:1805.07470*.

[129] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2022). A survey on bias and fairness in machine learning.

[130] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.

[131] Montague, R. (1970). Universal grammar. *Theoria*, 36(3):373–398.

[132] Moravcík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. H. (2017). Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR*, abs/1701.01724.

[133] Mordatch, I. and Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence*.

[134] MultiAgentLearning (2018). Pommerman. https://github.com/MultiAgentLearning/playground.

[135] Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*.

[136] Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2020). Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*.

[137] Ng, N., Cho, K., and Ghassemi, M. (2020). SSMBA: self-supervised manifold based data augmentation for improving out-of-domain robustness. *CoRR*, abs/2009.10195.

[138] Noam Brown, T. S. (2017). Libratus: The superhuman ai for no-limit poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 5226–5228.

[139] O'Kelly, M., Sinha, A., Namkoong, H., Duchi, J., and Tedrake, R. (2018). Scalable end-to-end autonomous vehicle testing via rare-event simulation. *arXiv preprint arXiv:1811.00145*.

[140] Oren, Y., Sagawa, S., Hashimoto, T. B., and Liang, P. (2019). Distributionally robust language modeling.

[141] Osogami, T. and Takahashi, T. (2019). Real-time tree search with pessimistic scenarios.

[142] Ost, J., Mannan, F., Thuerey, N., Knodt, J., and Heide, F. (2020). Neural scene graphs for dynamic scenes.

[143] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

[144] Papageorgi, I. (2018). *Shaping*, pages 1–3. Springer International Publishing, Cham.

[145] Paszke, A., Gross, S., Massa, F., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *NeurIPS*, pages 8024–8035. Curran Associates, Inc.

[146] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction.

[147] Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018). Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *arXiv preprint arXiv:1804.02717*.

[148] Peysakhovich, A. and Lerer, A. (2017). Consequentialist conditional cooperation in social dilemmas with imperfect information. *arXiv preprint arXiv:1710.06975*.

[149] Peysakhovich, A. and Lerer, A. (2018). Prosocial learning agents solve generalized stag hunts better than selfish ones. *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*.

[150] Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

[151] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[152] Rahimian, H. and Mehrotra, S. (2019). Distributionally robust optimization: A review.

[153] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

[154] Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876.

[155] Rempe, D., Philion, J., Guibas, L. J., Fidler, S., and Litany, O. (2021). Generating useful accident-prone driving scenarios via a learned traffic prior.

[156] Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2019). Learning to reweight examples for robust deep learning.

[157] Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks.

[158] Resnick, C., Eldridge, W., Ha, D., Britz, D., Foerster, J., Togelius, J., Cho, K., and Bruna, J. (2018a). Pommerman: A multi-agent playground.

[159] Resnick, C., Gao, C., Márton, G., Osogami, T., Pang, L., and Takahashi, T. (2020a). Pommerman & neurips 2018. In Escalera, S. and Herbrich, R., editors, *The NeurIPS '18 Competition*, pages 11–36, Cham. Springer International Publishing.

[160] Resnick, C., Gupta, A., Foerster, J., Dai, A. M., and Cho, K. (2020b). Capacity, bandwidth, and compositionality in emergent language learning.

[161] Resnick, C., Litany, O., Kar, A., Kreis, K., Lucas, J., Cho, K., and Fidler, S. (2022). Causal scene bert: Improving object detection by searching for challenging groups of data.

[162] Resnick, C., Raileanu, R., Kapoor, S., Peysakhovich, A., Cho, K., and Bruna, J. (2018b). Backplay: "man muss immer umkehren".

[163] Roller, S., Boureau, Y., Weston, J., Bordes, A., Dinan, E., Fan, A., Gunning, D., Ju, D., Li, M., Poff, S., Ringshia, P., Shuster, K., Smith, E. M., Szlam, A., Urbanek, J., and Williamson, M. (2020). Open-domain conversational agents: Current progress, open problems, and future directions. *CoRR*, abs/2006.12442.

[164] Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

[165] Sadeghi, F. and Levine, S. (2017). Cad2rl: Real single-image flight without a single real image.

[166] Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2020). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization.

[167] Salimans, T. and Chen, R. (2018a). Learning montezuma's revenge from a single demonstration. https://blog.openai.com/learning-montezumas-revenge-from-a-single-demonstration/. Accessed: 2018-07-12.

[168] Salimans, T. and Chen, R. (2018b). Learning montezuma's revenge from a single demonstration. *CoRR*, abs/1812.03381.

[169] Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., and Gelly, S. (2019). Episodic curiosity through reachability.

[170] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. (2019). Habitat: A platform for embodied AI research. *CoRR*, abs/1904.01201.

[171] Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. (2018). Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835.*

[172] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation.

[173] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017a). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[174] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347.*

[175] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, USA.

[176] Sheehan, T. (2020). Color matters: Rethinking photography and race. In *The Colors of Photography*, pages 55–72. De Gruyter.

[177] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[178] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

[179] Spelke, E., Breinlinger, K., Macomber, J., and Jacobson, K. (1992). Origins of knowledge. *Psychological Review*, 99:605–632.

[180] Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental Science*, 10(1):89–96.

[181] Spike, M., Stadler, K., Kirby, S., and Smith, K. (2017). Minimal requirements for the emergence of learned signaling. In *Cognitive Science*.

[182] Steels, L. (1997). The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34.

[183] Steels, L. and Kaplan, F. (2000). Aibo's first words: The social learning of language and meaning. *Evolution of Communication*, 4.

[184] Stuckler, J., Holz, D., and Behnke, S. (2012). Robocup@home: Demonstrating everyday manipulation skills in robocup@home. *IEEE Robotics Automation Magazine*, 19(2):34–42.

[185] Sukhbaatar, S., Szlam, A., and Fergus, R. (2016). Learning multiagent communication with backpropagation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *NeurIPS*, pages 2244–2252. Curran Associates, Inc.

[186] Szabó, Z. G. (2020). Compositionality. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2020 edition.

[187] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528.

[188] Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., and Urtasun, R. (2020). Physically realizable adversarial examples for lidar object detection.

[189] Uijlings, J. R., Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *Int. J. Comput. Vision*, 104(2):154–171.

[190] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio.

[191] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

[192] Varadhan, R., Segal, J. B., Boyd, C. M., Wu, A. W., and Weiss, C. O. (2013). A framework for the analysis of heterogeneity of treatment effect in patient-centered outcomes research. *Journal of clinical epidemiology*, 66(8):818–825.

[193] Velentgas, P., Dreyer, N. A., Nourjah, P., Smith, S. R., and Torchia, M. (2013). Developing a protocol for observational comparative effectiveness research: A user's guide.

[194] Verhoef, T., Kirby, S., and de Boer, B. (2016). Iconicity and the emergence of combinatorial structure in language. *Cognitive Science*, 40(8):1969–1994.

[195] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA. Association for Computing Machinery.

[196] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, , Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354.

[197] Vodopivec, T., Samothrakis, S., and Šter, B. (2017). On monte carlo tree search and reinforcement learning. *J. Artif. Int. Res.*, 60(1):881–936.

[198] Wager, S. and Athey, S. (2017). Estimation and inference of heterogeneous treatment effects using random forests.

[199] Wald, A. (1945). Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 46:265–280.

[200] Wang, A. and Cho, K. (2019). BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094.

[201] Wang, C., Zhang, X., and Lan, X. (2017). How to train triplet networks with 100k identities? *CoRR*, abs/1709.02940.

[202] Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., and Urtasun, R. (2021a). Advsim: Generating safety-critical scenarios for self-driving vehicles. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[203] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. *CoRR*, abs/1404.4661.

[204] Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. X. (2021b). Long-tailed recognition by routing diverse distribution-aware experts.

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

[206] Weber, N., Shekhar, L., and Balasubramanian, N. (2018). The fine line between linguistic generalization and failure in seq2seq-attention models. In *Proceedings of the Workshop on Generalization in the Age of Deep Learning*, pages 24–27. Association for Computational Linguistics.

[207] Wikipedia contributors (2021). Blind men and an elephant. [Online; accessed 26-Sep-2021].

[208] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. `https://github.com/facebookresearch/detectron2`.

[209] Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., and Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection.

[210] Xu, M., Zhang, Z., Hu, H., Wang, J., Wang, L., Wei, F., Bai, X., and Liu, Z. (2021). End-to-end semi-supervised object detection with soft teacher.

[211] Yoshida, W., Dolan, R. J., and Friston, K. J. (2008). Game theory of mind. *PLoS computational biology*, 4(12):e1000254.

[212] Zaslavsky, N., Kemp, C., Regier, T., and Tishby, N. (2018). Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences*, 115(31):7937–7942.

[213] Zeng, X., Liu, C., Wang, Y.-S., Qiu, W., Xie, L., Tai, Y.-W., Tang, C. K., and Yuille, A. L. (2019). Adversarial attacks beyond the image space.

[214] Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021). Scaling vision transformers.

[215] Zhang, J. and Cho, K. (2016). Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*.

[216] Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints.

[217] Zhou, H., Gong, Y., Mugrai, L., Khalifa, A., Nealen, A., and Togelius, J. (2018). A hybrid search agent in pommerman. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, pages 46:1–46:4, New York, NY, USA. ACM.

[218] Zhu, Y., Wang, Z., Merel, J., Rusu, A. A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., and Heess, N. (2018). Reinforcement and imitation learning for diverse visuomotor skills. *CoRR*, abs/1802.09564.

[219] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. (2008). Regret minimization in games with incomplete information. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 1729–1736. Curran Associates, Inc.

[220] Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey.