UNIVERSITY OF CALIFORNIA

Los Angeles

On robust estimation in causal machine learning

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Jeffrey Jiang

2024

ABSTRACT OF THE DISSERTATION

On robust estimation in causal machine learning

by

Jeffrey Jiang
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Gregory Pottie, Chair

This thesis presents three significant contributions to the field of machine learning, with a focus on Variational Autoencoders (VAEs), energy-based models, and education simulations. Firstly, we demonstrate the ability to impose substantial structure on the latent space of VAEs, enabling out-of-distribution data generation, structural hypothesis testing, and the production of augmentations in the latent space. These findings give us new ways to structure and interpret the latent space, creating robustness and explainability. Secondly, we identify a state-of-the-art defense technique using the unsupervised learning approach of energy-based models. This technique effectively defends against several poisoning techniques without requiring excessive additional training time or significantly reducing test accuracy. Lastly, we have developed a simulation for educational purposes that aims to model and comprehend the interactions between humans and machines. This simulation, built on causal information, provides insights into the design of practical educational experiments and highlights the challenges associated with implementing a dynamic Intelligent Tutoring System (ITS) in an educational context. Interestingly, our simulation reveals that heuristic methods continue to perform on par with deep learning techniques in the presence of unknown subpopulation

distributions and hidden student states. This suggests that despite the rapid advancements in deep learning, heuristic methods retain their effectiveness in certain scenarios. These findings open new avenues for the application of machine learning techniques and provide a solid foundation for future research in these areas.

The dissertation of Jeffrey Jiang is approved.

Lieven Vandenberghe

Suhas Diggavi

Xiang 'Anthony' Chen

Gregory Pottie, Committee Chair

University of California, Los Angeles

2024

To my hardworking parents, Xueping and Weijun,

my amazing sister, Jessica,

and my loving wife, Ramya.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work would not be possible without the help of the people in my lab. I would like to thank my advisor, Professor Greg Pottie, for his guidance and support throughout my time at UCLA. I would also like to thank all of my lab mates for the stimulating discussions and for all the fun we have had in the past few years.

In particular, I would like to thank my co-authors, Sunay Bhat and Omead Pooladzandi, for their help with the research and for many insightful discussions. The works presented in Chapters 4, 5, 6, and 7 are the result of countless discussions and painstaking work. I would also like to thank my summer undergraduate research assistants, Emily Kuczynski and Kevin Hong, for their help with running simulations in Chapter 8, discussing results, and for their enthusiasm.

I would like to thank the educational grants that I have gotten over the years, including the Engineering Bridge program. Finally, I'd like to acknowledge the Engineering and Electrical and Computer Engineering departments at UCLA for their support and funding through my Teaching Assistant opportunities over the years. They offered a great environment for me to grow as a researcher and as a person.

VITA

2016	Semi-Autonomous Aerial Control System Research at UC Berkeley, Berkeley, CA.
2017	B.S. (Electrical Engineering under the Computer Engineering option), UCLA. Magna Cum Laude.
2017–2018	Teaching Assistant, Engineering Department, UCLA. Taught sections of Engineering 183EW (Engineering Ethics) under direction of Professor Donald Browne.
2019	M.S. (Electrical and Computer Engineering), UCLA.
2019	Vision Deep Learning Internship at Nikon Research Corporation of America, Belmont, CA.
2018–2023	Teaching Assistant, Electrical and Computer Engineering, UCLA. Primarily taught sections of ECE180D (systems design laboratory), where senior students were guided to make interactive games with multiple interacting systems.
2022	Robotics Deep Learning Internship at Siemens Corporation, Berkeley, CA.
2019–*present*	Research Assistant, Electrical and Computer Engineering, UCLA.

PUBLICATIONS

De-Biasing Generative Models using Counterfactual Methods.

Bhat S., Jiang J., Pooladzandi O., Pottie G.

Information Theory and Applications Workshop 2022.

Hypothesis Testing using Causal and Causal Variational Generative Models

Jiang J., Pooladzandi O., Bhat S., Pottie G.

NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research.

Towards Composable Distributions of Latent Space Augmentations

Pooladzandi O., Jiang J., Bhat S., Pottie G.

Information Theory and Applications Workshop 2023.

PureEBM: Universal Poison Purification via Mid-run Dynamics of Energy-Based Models

Pooladzandi O., Jiang J., Bhat S., Pottie G.

Submitted to ICML 2024.

CHAPTER 1

Introduction

Recent years have enlightened much of the general public of the current state and the future potential of artificial intelligence (AI) in our society. In light of this new frontier of AI, we've come to realize that an important area of research is to understand how to interface the use of AI with humans. In particular, AI has shown incredible performance in image recognition, natural language processing, and reinforcement learning tasks. Examples can be seen in all sorts of examples, such as self-driving cars, medical diagnosis, and even in education.

However, these advancements still have many subjective weaknesses. Most AI models are heavily over-parameterized, leading to them mostly being unexplainable black-box models. Furthermore, these models often focus on a single objective metric, such as accuracy, which can lead to models that are brittle and overfit to the training data. This brittleness reduces neural networks' ability to generalize to new data, and can possibly be attacked with small perturbations to an input dataset. In addition, the black-box nature of AI reduces human trust in the model, and can lead to dangerous consequences in extreme cases. Thus, we would like to investigate ways to incorporate explainability and human interaction into the use of AI.

Across the thesis, we investigate the improvement of robustness through the use of causal understanding. Causal reasoning is the process of understanding the cause-and-effect relationships between variables. All the projects presented in this thesis use some unsupervised learning with causally-inspired structure or directly manipulate causal structures. The causally-inspired unsupervised learning allows us to understand underlying structures in the

data and provide better explanations for the model's decisions. This all leads to a more robust model that can generalize better to new data and is more trustworthy to humans. These techniques no longer focus on just the single objective metric, but instead focus on a more holistic understanding of the problem. Thus, it may come at a loss of performance, but we believe that the trade-off is worth it.

In this thesis, we explore these causal structures for robustness in several problem settings. In the first, we understand generative models that attempt to disentangle the causal factors of variation in the data, and provide a structural understanding of the latent space. In the second, we propose a defense against adversarial poisoning attacks that uses energy-based models to purify the training data. In the third, we propose a method to simulate human education that uses causal reasoning to generate counterfactual explanations for students and attempts to handle the dynamics of human interactions.

The remainder of the thesis is organized as follows. Chapter 2 gives some background information about the topics covered in the thesis. Chapter 3 describes some initial research investigation on hidden-information games, how they apply to causality, and what makes the problems interesting and difficult. Chapter 4 gives a first experiment on using causal structure on an unsupervised learning to attempt to de-bias data points and generate points unrealistic in the generating dataset. Chapter 5 extends on the idea in Chapter 4 to use causal structure to test for the best causal structure explaining a dataset. Chapter 6 further extends the ideas in the previous two chapters to understand how to structure the latent space of a dataset to specifically correlate certain regions toward certain augmentations, allows for some interesting emergent properties. Chapter 7 describes a state-of-the-art defense mechanism against adversarial poisoning attacks that uses unsupervised energy-based models to purify the training data. Chapter 8 describes a method to simulate human education that uses causal reasoning to generate counterfactual explanations for students and attempts to handle the dynamics of human interactions.

CHAPTER 2

Background

In this chapter, we provide a brief overview of the foundational concepts and techniques that are relevant to the work presented in this thesis. The concepts will be built upon in more depth in later chapters.

2.1 Probabilistic Graphical Models

2.1.1 Conditional Probability and Bayes' Rule

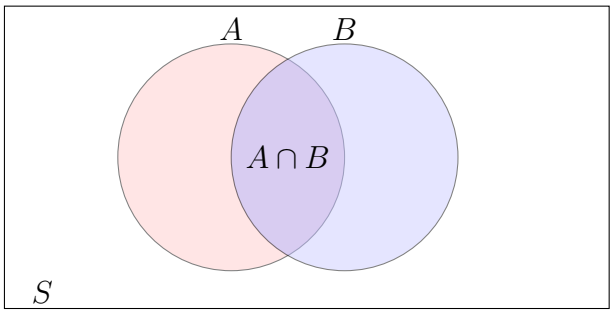


Figure 2.1: A Venn diagram depicting the events A and B in the total state space S.

Conditional probability is an important idea in probability. Given the notation in Figure 2.1, the conditional probability $\mathbb{P}[A|B]$ is the probability of A occurring if B is the possible states remaining. Therefore,

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]} \tag{2.1}$$

In particular, when A and B are independent (*i.e.* $A \perp\!\!\!\perp B$), then $\mathbb{P}[A|B] = \mathbb{P}[A]$, implying $\mathbb{P}[A \cap B] = \mathbb{P}[A]\,\mathbb{P}[B]$.

The basis of probabilistic graphical models comes from Bayes' rule, which states that

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A]\,\mathbb{P}[A]}{\mathbb{P}[B]} \tag{2.2}$$

where $\mathbb{P}[A|B]$ is the probability of event A given event B occurs (sometimes called the posterior), $\mathbb{P}[B|A]$ is the probability of event B given event A occurs (sometimes called the likelihood), $\mathbb{P}[A]$ is the probability of event A (sometimes called the prior), and $\mathbb{P}[B]$ is the probability of event B (sometimes called the marginal distribution).

Bayes' rule establishes a deeper understanding of conditional dependence. Conditional probability connects the probability of A and B in a powerful way that allows for *inference*. Suppose, for instance, your friend says something (A) to you in a loud coffee shop, but you cannot directly hear everything due to the loud noise. You pick out a couple words (B) and use a mixture of prior context ($P(A)$) and the likelihood of hearing those words ($P(B|A)$) to infer what your friend said ($P(A|B)$). Sometimes, you can get a high estimate of $P(A|B)$ and continue the conversation seamlessly. Other times, you might get a low estimate of $P(A|B)$ and ask your friend to repeat themselves.

2.1.2 Conditional Independence and Bayesian Networks

Conditional independence is a key concept in probabilistic graphical models. Two events A and B are conditionally independent given event C if

$$\mathbb{P}[A \cap B|C] = \mathbb{P}[A|C]\,\mathbb{P}[B|C] \tag{2.3}$$

It is important to note that without knowledge of C, A and B may not be independent.

A Bayesian network is a directed acyclic graph (DAG) where each node represents a random variable, and each edge represents a conditional dependency between the random variables. Most importantly, the graph structure of the Bayesian network encodes the conditional independence relationships between the random variables. This comes in several forms. Consider the Bayesian network in Figure 2.2. For any two nodes with no shared

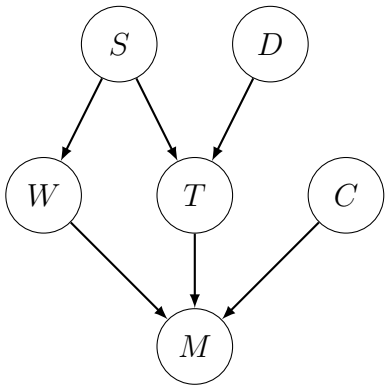


Figure 2.2: A Bayesian network depicting a graph determining where to meet. S is the season, D is the day of the week, W is the weather, T is the time of day, C is the cost, and M is the meeting location.

ancestors, nodes are independent, *e.g.* $C \perp\!\!\!\perp T$. For any two nodes with shared ancestors, nodes are independent given their shared ancestors, *e.g.* $W \perp\!\!\!\perp T \,|\, S$. Finally, the distribution of any node is conditionally independent of all non-descendants given its parents, *e.g.* $M \perp\!\!\!\perp S, D \,|\, W, T, C$.

One benefit of a Bayesian network is the factorization of the joint probability distribution, which heavily reduces the number of parameters. For instance, the joint probability distribution of the Bayesian network in Figure 2.2 can be factorized as follows

$$\mathbb{P}\left[S, D, W, T, C, M\right] = \mathbb{P}\left[S\right] \times \mathbb{P}\left[D\right] \times \mathbb{P}\left[W|S\right] \times \mathbb{P}\left[T|S, D\right] \times \mathbb{P}\left[C\right] \times \mathbb{P}\left[M|W, T, C\right] \quad (2.4)$$

One example of conditional independence is the Markov property, which is a key concept in stochastic processes and time-series probabilistic graphical models. The Markov property states that the future random variables in a stochastic process is conditionally independent of the past states given the current state. This can be represented as a Bayesian network $X_1 \to X_2 \to X_3 \to \cdots \to X_T$, where $X_{t+1} \perp\!\!\!\perp X_{t-1} \,|\, X_t$.

2.1.3 Hidden Markov Models

Hidden Markov Models (HMMs) are a type of probabilistic graphical model that are used to model time series data [Rab89]. An HMM is a directed graphical model with a sequence of hidden states and a sequence of observed variables, as shown in Figure 2.3. Unlike a standard Markov model, the HMM assumes that the state of the system is not directly observable, but instead the system is observed through a set of noisy measurements.

The observed variables are the data that we have access to, while the hidden variables are the latent variables that we do not have access to. The hidden variables are assumed to be Markovian, meaning that the current state of the hidden variables only depends on the previous state of the hidden variables. The observed variables are conditionally independent given the hidden variables. The model is defined by the initial probabilities of the hidden states, transition probabilities between the hidden states, and the emission probabilities of the observed variables given the hidden states.

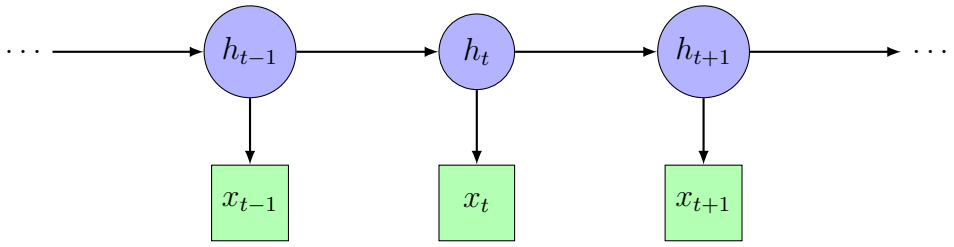


Figure 2.3: A hidden Markov model with three time steps.

There are many ways to use an HMM. First, we can an HMM to generate a sequence of hidden states and observations, given the parameterized distributions. Recursively, this can be done by sampling the distributions:

$$\begin{aligned} h_1 &\sim \pi \\ h_t &\sim \mathbb{P}\left[h_t | h_{t-1}\right] \\ x_t &\sim \mathbb{P}\left[x_t | h_t\right] \end{aligned} \tag{2.5}$$

where π refers to the initial probability distribution. More importantly, we can use Bayesian properties to get an online estimate of the hidden states as we make observations. Recursively, we can calculate the probability of the hidden state h_t being in state k of K total states as follows:

$$\begin{aligned}\mathbb{P}\left[h_1 = k\right] &= \frac{\mathbb{P}\left[x_1|h_1 = k\right]\pi(k)}{\sum_{j=1}^{K}\mathbb{P}\left[x_1|h_1 = j\right]\pi(j)} \\ \mathbb{P}\left[h_t = k\right] &= \frac{\mathbb{P}\left[x_t|h_t = k\right]\mathbb{P}\left[h_t = k|h_{t-1}\right]}{\sum_{j=1}^{K}\mathbb{P}\left[x_t|h_t = j\right]\mathbb{P}\left[h_t = j|h_{t-1}\right]}\end{aligned} \tag{2.6}$$

If we save only the maximum likelihood path to each previous state, this dynamic programming algorithm is known as the Viterbi algorithm. The Viterbi algorithm is used to find the most likely sequence of hidden states given the observed sequence.

Furthermore, HMMs allow for the estimation of the model parameters given the observed data. The Baum-Welch algorithm is used to estimate the model parameters by maximizing the likelihood of the observed data.

HMMs are used in a wide variety of applications, including speech recognition, bioinformatics, and finance. They have the benefit of being able to model complex time series data and are fairly explainable, while being relatively simple to implement and train. However, they still have limitations, such as the requirement of having a fixed number of discrete hidden states and the Markov assumption. Complex, multi-level hidden states are effectively impossible to estimate, given the propagation of errors in the model.

2.2 Causality and Causal Modeling

Causal modeling is the process of constructing models that represent the causal relationships between variables. Causal models are used to understand the underlying mechanisms that generate the observed data, and to make predictions about the effects of interventions on the system. Causal models are typically represented as directed acyclic graphs (DAGs), where the nodes of the graph represent the variables of interest, and the edges of the graph

represent the causal relationships between the variables, which looks similar to a Bayesian network.

One of the consequences of Bayes' rule (Equation 2.2) is that there exists a symmetry between child and parent in a Bayesian network. Causal models, on the other hand, are asymmetric, in that if $A \to B$, then forcing B does not necessarily cause A. A basic example is the sun rising causes the rooster to crow, but if we force a rooster to crow, the sun does not rise. This is a key difference between causal models and Bayesian networks, where traditional Bayesian networks only concern themselves with observing B. This gives rise to the idea of *interventions*, where we can force a variable to a specific value and observe the effects on the rest of the system. [Pea09a] uses the idea of interventions to define the *do-operator*, which is used to represent the effect of interventions in a causal model.

Consider Figure 2.2 as an example causal model. Suppose we know one person can only meet around noon, thereby forcing T to be noon. Suppose in the summer, we are normally less likely to meet at noon because of the heat. In a traditional Bayesian network, if we were to observe $T = \text{noon}$, then we would expect $S = \text{summer}$ to be less likely. This would also suggest that $W = \text{sunny}$ is less likely and implies that $M = \text{hike}$ would be more likely.

According to the causal model, by forcing T to be noon, we effectively break the causal connection between S and T. However, S still causally affects M through W. For example, if S is summer, then $W = \text{sunny}$ is more likely and $M = \text{hike}$ is less likely. As a result, the expected intervened probability distribution $\mathbb{P}[M \mid do(T = \text{noon})]$ is different from the observed probability distribution $\mathbb{P}[M \mid T = \text{noon}]$.

In [Pea09b], Pearl introduces the three rungs of the ladder of causation. The first rung of causal reasoning is the ability to understand the association relationships between variables, which is the lowest level of causal reasoning. These are the more traditional statistical models, which employ observations and correlations to understand the relationships between variables. The second rung of causal reasoning is the ability to understand the interventional relationships between variables. Models on the second rung are able to handle situations

when certain variables are intervened upon. The third rung of causal reasoning is the ability to understand counterfactual relationships between variables. Counterfactual reasoning is the ability to understand the potential outcomes of a system if it had been intervened upon differently. We believe that achieving counterfactual models can allow for true understanding of a system and allows for more human-like reasoning.

2.3 Deep Learning

Deep learning is a subfield of machine learning that is concerned with the development and application of deep neural networks. A single neuron is shown in Figure 2.4 and a single-layer neural network is shown in 2.5. The neuron takes a set of inputs, applies a linear transformation to the inputs, and then applies a non-linear activation function to the result. The neuron is called the neuron because it is analogous to the behavior of a biological neuron. The general neural network, sometimes called a multi-layer perceptron (MLP), is composed of multiple layers of neurons, where the output of one layer is the input to the next layer, hence the term "deep" learning.

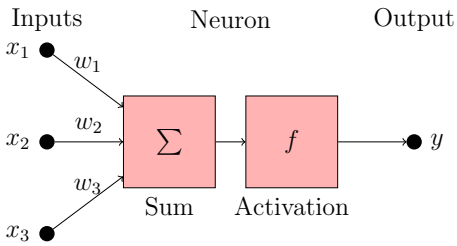


Figure 2.4: A single neuron

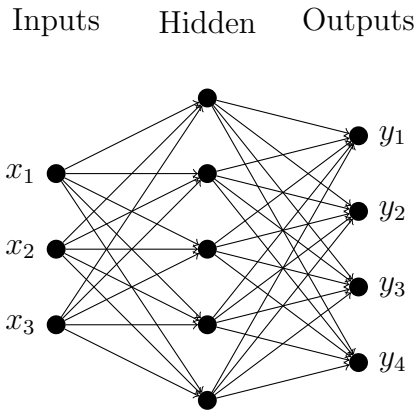


Figure 2.5: A multi-layer perceptron (MLP) with a single hidden layer

Theoretically, a neural network can approximate any linear or non-linear function, given enough neurons and layers. In practice, the difficulty is getting to this functional approximation. Neural networks are trained using a process called back-propagation, which is a method for computing the gradient of the loss function with respect to the parameters of the model. The gradient computed is used to update the parameters of the model using an optimization algorithm, such as stochastic gradient descent. This universal function approximation has powerful implications on a wide variety of applications and can be a possible solution to many problems, to be discussed further in upcoming sections.

However, there are still some limitations to deep learning. Primarily, due to its large number of parameters, deep learning models are often considered to be black-box models, meaning that it is difficult to understand the internal workings of the model. Deep learning models often require a large amount of data to train, which can be difficult to obtain in some domains. Furthermore, deep learning models often require a large amount of computational resources and time to train, which can be a barrier to entry for some. Finally, deep learning models are mostly over-parameterized, which can lead to over-fitting, poor generalization to new data, and poor robustness to data attacks.

2.3.1 Supervised and Unsupervised Learning

Because neural networks are universal function approximators, they can be used for both supervised and unsupervised learning tasks. In supervised learning, the model is trained on a labeled dataset, where the input data x is paired with the correct output y. These labels can be continuous values, which would be regression tasks, or categorical, which would be classification tasks. The goal of the model is to minimize an objective *loss function* $\ell(f(x), y)$ that measures the difference between the predicted output and the true output. For instance, the typical output of a classification neural network is a probability distribution over the categories and the final classification is the category with the highest probability. In particular for classification tasks, categories are arbitrarily human-defined and do not necessarily have

a natural ordering, such as the categories of animals. As a result, classification can often create complex decision boundaries in data, which can both be difficult to interpret but also brittle to small changes in the data.

In unsupervised learning, sometimes known as representation learning, the model is trained on an unlabeled dataset, where the model is tasked with finding the underlying structure of the data. One example is clustering, where the model is tasked with finding the natural groupings of the data. Another example is dimensionality reduction, where the model is tasked with finding a lower-dimensional representation of the data that retains the most important information. In both cases, while the model is able to find its own patterns in the data, it still often needs humans to select the number of clusters or dimensions. Unsupervised learning has the benefit of creating boundaries that are more interpretable and robust to small changes in the data, but it is often more difficult to train and evaluate.

2.3.2 Reinforcement Learning

Reinforcement learning is a category within the domain of machine learning. RL has been used to solve many problems, such as playing games [SHM16, VBC19, SHS17], robotics [KBP13], and even medical applications [SIB22] with varying levels of success. RL is primarily concerned with training autonomous agents to make informed decisions within a given environment, with the overarching objective of maximizing cumulative rewards. This learning paradigm revolves around the iterative process of directly using trial and error to interact with the environment, enabling agents to progressively attain specific goals. Several pivotal components make up the framework of reinforcement learning.

At its core, an agent serves as the central decision-maker, actively engaging with the environment. Informed by its current knowledge and the information it acquires from the environment, the agent makes decisions aimed at optimizing its performance. The environment, on the other hand, represents the external system with which the agent interacts. It provides feedback to the agent in the form of rewards, which the agent strives to maximize

over time.

Integral to the reinforcement learning framework is the concept of a "state," which encapsulates the current situation or configuration within the environment. States encompass all information required for effective decision-making. In addition, the environment will give the agent a reward based on the state of the environment. In parallel, the "action" domain encompasses the set of conceivable choices or decisions at the disposal of the agent during each time step. These actions bear direct influence on state transitions and the resultant rewards received. Figure 2.6 illustrates the reinforcement learning framework, highlighting the interactions between the agent and the environment.

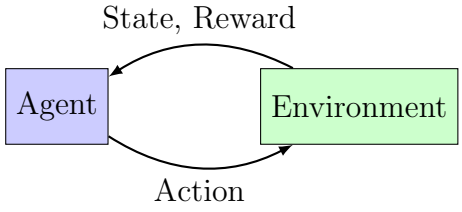


Figure 2.6: Reinforcement Learning Framework

Typically, the goal of a reinforcement learning agent is to maximize its cumulative reward over time. The reward itself is a scalar value that is often human designed to guide the agent towards a specific goal. Different reward functions can lead to drastically different behaviors in the agent, and the design of the reward function is a key part of the reinforcement learning process.

The reinforcement learning agent refines its decision-making process through a continuous cycle of action, observation of resulting states, and acquisition of rewards. This dynamic learning process typically depends on a diverse array of algorithms and techniques, examples are discussed below. In most of our techniques, we focus on finding the next action via the *Q-value*. The Q-value represents our estimate of the expected discounted return for a state-action pair.

However, RL still has many ongoing problems. Unlike supervised learning, RL deals with

a non-stationary non-independent dataset, making sample efficiency an important point of discussion. Therefore, RL has difficulty in low-interaction spaces and can have difficulty adjusting to a changing environment. Finally, RL still suffers from similar problems to supervised learning, in that interpretability and performance often have some level of trade off.

2.3.2.1 Q-Learning

Q-Learning is a simplistic tabular RL method that can be summarized in several key steps [SB18]. We maintain a table where the entries of the table are indexed by (s, a) and maintain the current estimated Q-value, the expected discounted total reward, of taking action a when in state s. While learning, entries of the table are continually updated via the Bellman equation after observing the (s, a, r, s') tuple by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{2.7}$$

where α is the learning rate and γ is the discount factor. Other slight differences in the selection on the estimate of the new state's value can be chosen, but this is the update for Q-learning [SB18]. When using the Q-table, then, we normally want to take the action a such that $a = \underset{\alpha}{\operatorname{argmax}} Q(s, \alpha)$.

Unfortunately, the simplicity of Q-learning does come at a cost. While on the surface, we can view the table as explainable, the explanations are atomic, and there is no systematic way of explaining the difference in action between two similar states. It has low sample efficiency and requires several passes through the entire Q-table to properly get accurate information, which also makes it much less capable of updating to changing populations. Nonetheless, Q-learning in small state and action spaces can provide a basic proof-of-concept and understanding of how RL in education should function.

2.3.2.2 Deep Q-Learning

Deep Q-Learning (DQL) substitutes the Q-table in Q-learning with a neural network. Interestingly, the neural networks can have significantly lower parameters than the Q-table, as the goal of the neural network is to classify the states and group similar states to have similar Q-values. Therefore, while updating the Q-function, it also updates states that it feels are similar. This can produce an increased sample-efficiency and update time. Alternatively, it can also be used to encapsulate more complex state spaces, particularly when there is hidden information in the state space. However, it is still a black-box function and retains difficulty of explanation.

In the DQL approach, we estimate the Q-function (state pair to a list of Q-values per action) as a neural network. However, a major issue with deep learning on RL is the heavy dependence of successive states. Thus, [MKS13] devises the *experience-replay buffer* as a means of storing (s, a, r, s') tuples of a large number of interactions. Then, these experiences are randomly sampled into batches, breaking the heavy dependence of successive actions and restoring some level of independence [MKS13]. From these sampled tuples, we get a stochastic sample of an observed Q-function that we can compute a loss from the squared TD-loss:

$$\delta_t^2 = \Big(r + \gamma \max_{\alpha} Q(s', \alpha) - Q(s, a)\Big)^2 \tag{2.8}$$

to do gradient updates. Again, when trying to decide on the optimal action, it will select a such that $a = \underset{\alpha}{\operatorname{argmax}}\, Q(s, \alpha)$.

2.4 Generative Models

2.4.1 Variational Autoencoders (VAEs)

An autoencoder, shown in Figure 2.7, is a specific type of unsupervised neural network structure that takes high-dimensional input data, such as images, and compresses it into a

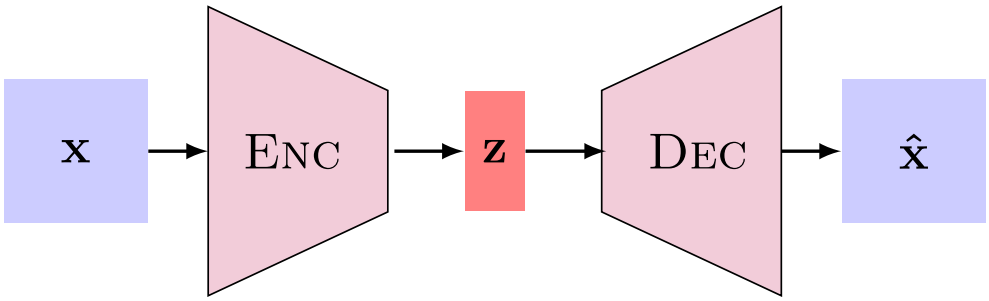


Figure 2.7: (Variational) Autoencoder Architecture

lower-dimensional representation, called the latent space. The idea behind an autoencoder is to have encoder and decoder neural networks of similar structure that work together to compress and restore the input data, thereby making the goal to minimize $\ell(x, \hat{x})$.

A variational autoencoder (VAE), also structurally the same as the one in Figure 2.7, is an extension of the autoencoder such that we also constrain the latent space to be a specific distribution, often a Gaussian distribution [KW13]. The constraint regularizes the problem and reduces overfitting, allowing for the VAE to generate new data by sampling from the latent space and decoding the sampled latent values. The VAE is trained by maximizing the evidence lower bound (ELBO), which, in practice, looks like a joint optimization of a reconstruction term and a KL divergence:

$$\mathcal{L} = \ell(x, \hat{x}) + \lambda \cdot \mathrm{KL}(\mathcal{N}(\mu_z, \sigma_z) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I})) \tag{2.9}$$

where $(\mu_z, \sigma_z) = e(x)$, $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\hat{x} = d(z)$, and λ is a hyperparameter. The KL divergence offers a measure-like distance between the latent space distribution and a standard Gaussian distribution.

2.4.2 Conditional VAE (CVAE)

One of the problems with the VAE is that a randomly sampled image from the latent space can be any image in the dataset. If the VAE could generate both cats and dogs, we would not be able to control the generation of a cat or a dog. If we want to generate a specific type

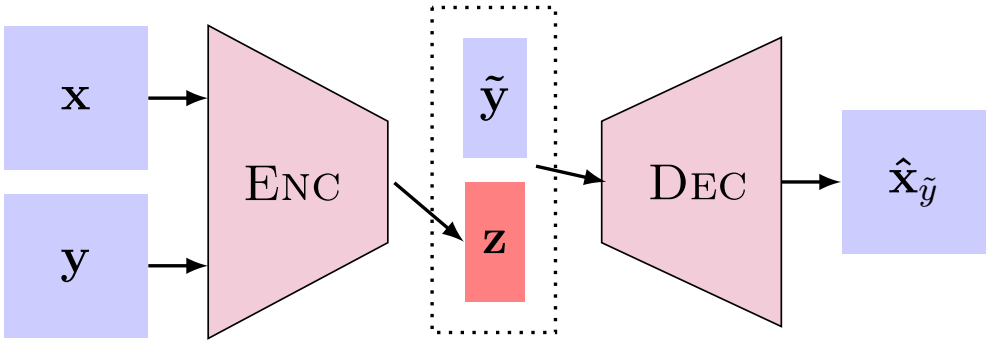


Figure 2.8: Conditional VAE Architecture

of image, such as a cat or a dog, then we would need to condition the latent space on the class.

The conditional variational autoencoder (CVAE), shown in Figure 2.8, is an extension of the VAE that attempts to add some supervised component to the structure [SLY15]. Effectively, the CVAE adds the label y to the input data x and the latent space z. The training of the CVAE is similar to the VAE, but the loss function, which is structurally the same as 2.9, is conditioned on the label y, so $\hat{x} = d(e(x, y), y)$, where e is the encoder and d is the decoder. When generating new data, we simply choose a label y and sample from the latent space $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to generate new data.

CHAPTER 3

Games

3.1 Motivation

Games have been the natural playground for RL, as they are easy to simulate, generally have strong Markovian assumptions, and can be easily adjusted to test different structural hypotheses [SHM16, VBC19, SHS17]. More importantly, games also have natural causal structures involved. First, the rules of the game is a very rigid causal structure. As a result, when playing games, people also try to develop causal methods to understand and play games. Thus, games can be an interesting way to investigate the problem of interacting with humans in a strict, causal way.

Game theory can be a part of the discussion behind games. In considering a competitive multiagent RL problem, the problem will typically be solved by finding the Nash equilibrium of the game. Oftentimes, this deals with the minimax solution, where the environment is playing at its optimal level. However, we are not interested in finding the game-theoretic solution, but rather in understanding the human's strategy. Specifically, the challenge of discovering the human state and winning the game reflects the same balance between exploitation and exploration found in traditional dual control or reinforcement learning problems.

Furthermore, games can be easily be adjusted. These adjustments can result in completely new tasks with similar concepts. Or, we can use new methods to encode implicit causal information through adding interventions. Ultimately, the flexibility of games is a big selling point in their use, as their complexity can start small and increase based on how the

rules have been set. This justifies the use of different methods of causal understanding of both the game space and the individual space that can transfer much easier between different versions of games. Besides just understanding how flexible our causal modeling is, we can also make rule modifications for the explicit purpose of testing interventions.

3.2 Problem Formulation

We define the problem in terms of the vocabulary of an agent-environment interaction in reinforcement learning (RL). Each game has some rigidly defined set of rules that define the possible dynamics of the game and two actors a human and a RL agent. In our case, we interchangeably call the agent the "bot" and the environment actor the "human." In this context, the environment encapsulates both the game, with its proposed structure, and the human, with their playing structure.

The objective of the agent is to handle both aspects of the environment: to achieve an objective in the game (the problem state) and to achieve an understanding of the environment's human actor (the actor state). In particular, the actor state can influence how the problem state dynamics evolve. Because the actor state is human, we assume the environment to be non-stationary, but follows some causal set of human-understandable set of principles. With partial observability, any deviation from the ground truth can quickly degrade prediction performance [SB18].

In general, this non-Markovian "mind-reading" problem is intractable. However, we further assume that the agent is able to perform interventions specifically to get information about the environment. We also assume that actions are non-catastrophic, i.e. the agent can always recover from any action that it takes and that the agent does not always need to take a theoretically optimal action. Thus, at different points in time, we can reduce the uncertainty of the unobserved environment state to better choose future actions. With the combination of gaining information and the constraints on the environment, we can view the problem as

a causal inference problem, where the agent is trying to infer the environment's state based on the interventions that it performs. This primarily reduces the problem to how much information is gained through interaction and what quality of estimation this information can provide. In terms of data, we may or may not assume that there is an abundance of population data, but we always assume that there is a lack of individual data. Instead, based on the population, we form prior knowledge models from which we learn possible dynamics. However, we can only choose to create models small enough that estimation is possible within a small number of observations and know how to adjust these models as more data may come in. Our solution to this is to limit the models to discrete sets through the use of causal models. By applying causal structures, we want to leverage sample efficiency, and also provide explanations based on causal *interventions* and possible *counterfactual* realities should we have performed other actions, plus a transfer between models based on the level of information gained.

3.3 Formal Problem Statement

The RL environment can be separated into a *Game Space* and *Human Space*. Consider the game space, defined for two players, denoted a for agent and e for environment. For now, we will assume that the game has a predefined fixed structure. Each player observes a sequence of observations $\{o_t^a : t \in \{1, \ldots, T\}\}, \{o_t^e : t \in \{1, \ldots, T\}\}$, performs a sequence of actions $\{a_t^a : t \in \{1, \ldots, T\}\}, \{a_t^e : t \in \{1, \ldots, T\}\}$, and receive a sequence of rewards $\{r_t^a : t \in \{1, \ldots, T\}\}, \{r_t^e : t \in \{1, \ldots, T\}\}$. Because the agent and environment do not have to be playing symmetric games, the states and actions do not need to come from the same support. Based purely on the game dynamics, $\mathbb{P}\left[o_{t+1}^e, r_{t+1}^e | o_t^a, a_t^a\right]$ and $\mathbb{P}\left[o_{t+1}^a, r_{t+1}^a | o_t^e, a_t^e\right]$ are some fixed distribution. Of course, because the game itself need not stationary, these distributions can be different at different points in time t.

Now, we discuss the unobserved components from the agent's perspective. For now, we

ignore the reward, as that is defined purely based on the game dynamics. From the agent's perspective, the game can be reduced using Markovian assumptions on the game to

$$\mathbb{P}\left[o_{t+1}^a | o_t^a, a_t^a\right] = \sum_{o_t^e, a_t^e} \mathbb{P}\left[o_t^e | o_t^a, a_t^a\right] \mathbb{P}\left[a_t^e | o_t^e\right] \mathbb{P}\left[o_{t+1}^a | o_t^e, a_t^e\right] \tag{3.1}$$

Here, the t indexing assumes that the t time step for the environment comes after the t time step for the agent.

First, unless the game is fully observed, we only have partial information of o_t^e given o_t^a, *i.e.* we can only have some distribution $\mathbb{P}\left[o_t^e | o_t^a, a_t^a\right]$. The main consideration, though, is that the agent cannot observe the policy $\mathbb{P}\left[a_t^e | o_t^e\right]$, even if we know o_t^e. Furthermore, we assume in our problems that this policy is dynamic. For instance, if we normally assume the self-play minimax solutions, we may assume that $\mathbb{P}\left[a_t^e | o_t^e\right] = \mathbb{P}\left[a_t^a | o_t^a\right]$, but this is not the case in our problem (the state and action spaces may not even be the same). Just to illustrate, if we have a fully observed game, then we know that $\mathbb{P}\left[o_t^e | o_t^a, a_t^a\right]$ is the indicator for the environment state's observation, which the agent can deduce. Then Equation 3.1 simplifies to just understanding the environment's policy.

$$\mathbb{P}\left[o_{t+1}^a | o_t^a, a_t^a\right] = \sum_{a_t^e} \mathbb{P}\left[a_t^e | o_t^e\right] \mathbb{P}\left[o_{t+1}^a | o_t^e, a_t^e\right] \tag{3.2}$$

So, we define a hidden human state $\{h_t : t \in \{1, \ldots, T\}\}$. We want to see if the inclusion of this human hidden state can help us better understand the environment. In the event of infinite multiversal simulation data on the entire population, we can view h as the selector of the specific person in a population. However, in our case, we only have a small amount of data per person, so we can only view h as a selector of a subpopulation. In the worst case, $T = 1$ could imply that there is no further information we can gather from the person about how they play the game, and we can only rely on the population information that we have. Using this, we can potentially estimate the environment's policy as

$$q(o_t^a, a_t^a, h_t) \approx \mathbb{P}\left[o_t^e | o_t^a, a_t^a\right] \mathbb{P}\left[a_t^e | o_t^e\right] \tag{3.3}$$

The inclusion of h does complicate things, though, as it introduces a second level of dynamics and, thus, a second level of estimation uncertainty. If we were to do the estimation, h_t can depend on the entire history of the game, *i.e.* $\mathbb{P}\left[h_{t+1}|h_{1:t}, o^e_{1:t}, r^e_{1:t}\right]$, and perhaps even to previous plays of this game or other games. Even if only a fixed window affects the history, this adds to the difficulty of estimation. Furthermore, even if we were to have all the game data, we would still have to either use unsupervised learning techniques to assign values of h or use some other form of prior knowledge. Thus, we have to choose the possible modeling of h_t conservatively based on the data that we have in order to provide the best possible estimate provided in Equation 3.3. As a step toward this direction, we allow the ability to try and get access to h_t through interventions, analogous to asking a student about how they are studying.

Now we can add the rewards back in to picture. In our case, currently the reward will depend only on the action taken a^a_t and the next observation o^a_{t+1}, but to be more general we can also define

$$r(o^a_t, a^a_t, h_t) \approx \mathbb{P}\left[r^a_{t+1}|o^a_t, a^a_t\right] \tag{3.4}$$

and therefore, our final objective will be the same as most RL problems with maximizing the aggregate expected reward

$$G^a_t = \sum_{k=t+1}^{T} \gamma^{k-(t+1)} r^a_k \tag{3.5}$$

at any point in time, where γ is the RL discount factor to balance short and long-term rewards.

One other differentiating factor we want to consider in our problem is the inclusion of interventions. On one hand, one can view the "exploration" steps in RL to be probing interventions, as we are choosing to step away from the currently found optimal policy to see if there are other solutions. On the other, we can also directly impose extra interventions, actions that are not part of the "basic" set of actions in the game space that would change the state in a way that would normally follow the game rules (usually with a cost in reward

to make the game actions separate). From the perspective of information flow, typically inclusion of probing interventions is an interesting way of changing the environment toward providing better collaboration, even if the probing intervention has a cost.

Another way to view the interventions is that we change the rules into a similar, but different version of the game. The idea is that most of the concepts should still transfer between these versions of the game, but may now have extra tools to attempt to earn the highest reward.

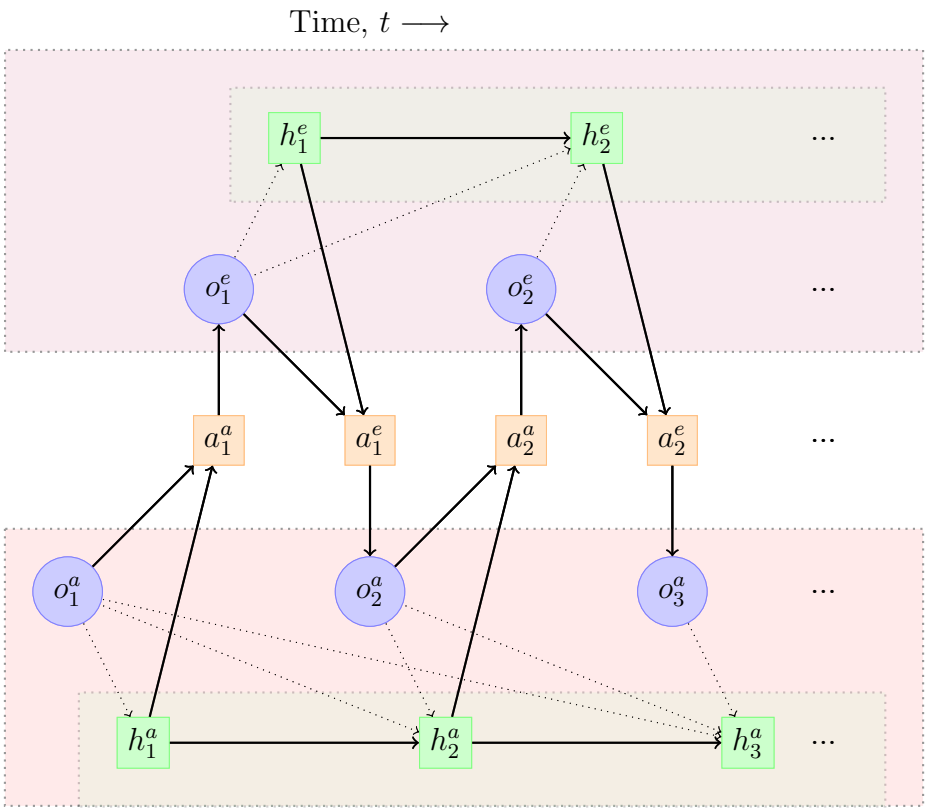


Figure 3.1: A simple hypothesized DAG about how a student may produce results for any single concept in a duration of time. Each of the inputs at the top are time dependent.

3.4 Liar's Poker

Liar's Poker is the first game on which we tested our ideas. A basic overview of the rules:

- All cards are drawn from a deck consisting of N_S suits and N_R ranks, such as the standard 52-card deck (with $N_R = 13, N_S = 4$).

- Each game consists of **rounds**. In each round, players are dealt completely new cards. At the beginning of the game, each player is dealt the same number of cards (in this example 2 cards).

- Each round consists of **turns**. The two players take turns making increasingly large **calls**, which can be some predetermined subset of poker hand designations with some ordering, e.g. calls of the form $N \in \{1, 2, 3, 4\}$ of a kind of one of the 13 ranks, $R \in \{2, 3, 4, \ldots, K, A\}$, characterized by $Call(N_1, R_1)$ with ordering,

  $$Call(N_1, R_1) < Call(N_2, R_2) \iff N_1 < N_2 \text{ or } (N_1 = N_2 \text{ and } R_1 < R_2)$$

  An honest call represents the player's belief that there are at least N cards of rank R in the total pool of cards (cards held by themselves and other players). However, calls do not need to be honest.

- One common variation is setting $R = 2$ cards to be considered wild cards and each can be substituted for any one card in any call to change the call likelihood distributions. Players cannot call $Call(N, 2)$.

- Finally, at any point after the first turn, a player additionally has the option of ending the round by calling the opponent's bluff. All cards are revealed and the last made call is checked to see if it exists *in the total pool of cards*. If it does not, then the bluffing player loses the round, otherwise the player who called the bluff loses. The loser starts the next round with an additional card. If the player loses some number of rounds (has more than some number of cards), then the player loses the game.

Liar's Poker is a competitive game about balancing levels of honesty and illustrates the difficulty of estimating human behaviors in the absence of information. The "greedy" option is to play honestly, as there is no way to lose in the next turn by calling honestly. However, this provides the opponent with maximum information. From there, the opponent can respond accordingly to gain advantage. On the flip side, when the opponent is expecting honesty, playing deceptively can pay off if the opponent uses the information they assumed was honest in their call. This shows a strategic trade-off in playing honestly and deceptively where the structure of the game promotes honesty and information exchange, but selfish interests promote deception. Furthermore, winning comes at a disadvantage in the short-term as the loser gets an additional card, allowing them to have more information about the pool of cards.

Structurally, the game's causal model is described in Figure 3.2. In every action, there is a player state (i.e. their understanding of probability and tendency to play), hidden game information (their cards), and previous calls. The previous calls provide varying amounts of information based on our estimation of the opponent's state. If we are able to understand how the actor likes to play, we can take that information to perform actions that give the agent an advantage. Preliminary results showed that the greedy strategy performed well against many of the other strategies. Even a strategy designed specifically to beat a greedy strategy performed somewhat better than a greedy strategy, assuming the agent does not adapt whatsoever.

However, Liar's Poker ends up being difficult to investigate. One reason is that there is no obvious minimax solution. As a result, there is no guarantee of a "good enough" solution at any point in time. Furthermore, while it can be reduced, there is a lot of hidden information in this game, most of which cannot be observed except post-hoc. While a player can observe the strategy played to inform their actions in the next round, there is a reset in cards at the minimum and so the conditions no longer hold. The only information that a player gets can be untrustworthy. Lastly, there are too many possible strategies and even more possible

strategy dynamics. Even within a single round the players' strategies may vary wildly in response to previous observations.

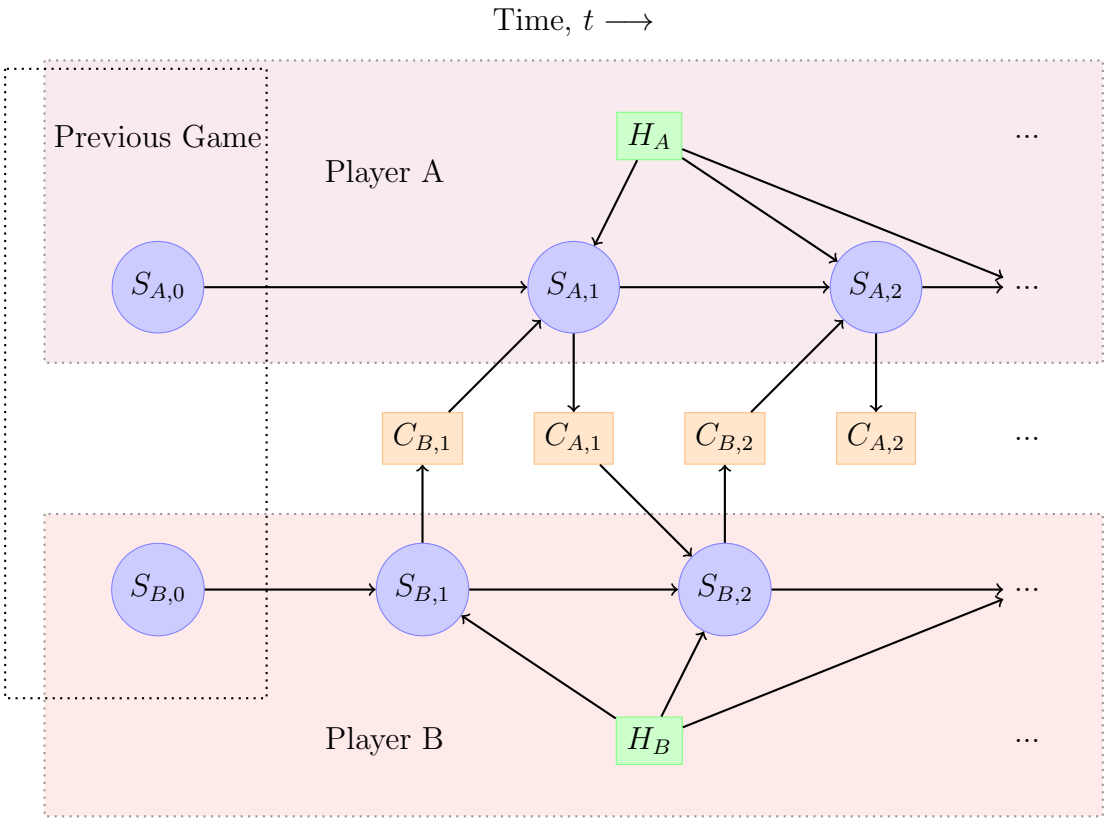


Figure 3.2: A DAG with the causal model that happens within a round, showing what goes on in the first 4 turns of a game. S refers to the strategy of a player, H represents the hand, and C represents the calls. The blue circles (strategies) are always unobservable. The orange rectangles (calls) are always observable. The green rectangles (hands) are sometimes observable (at the end of each round). The hand does not change between the turns in a round.

3.5 Rock Paper Scissors

Rock Paper Scissors (RPS) is a fascinating game focused solely on adversarial "mind-reading."

While it does not translate fully to the other example problems, it does illustrate some of the techniques that we can use. Taking a look from a pure RL perspective, the RPS problem can almost be viewed as a multi-arm bandit problem with arms selected at each of the 3 actions [SB18]. The multi-arm bandit problem is a classic RL problem where the agent has to choose between K different actions, each with an unknown reward distribution. The rewards are given based on how the opponent responds. The main problem with this formulation is that the distributions for the arms are neither constant nor even pseudo-stationary.

When humans play RPS, we implicitly have biases toward or against one of the actions, *e.g.* many say that people are biased to play rock for various reasons. Most of the time, humans will try to rationalize the opponent's decisions, such as "there's no way they play three rocks in a row," which leads to each player having a different flowchart at any given time. We hope to exploit this tendency to gain some advantage over human players.

One of the nice things about playing RPS is that from the game's perspective, there is a very balanced minimax solution — playing in a uniform random way that is completely independent. This play-style will generate an average of 0 reward, regardless of the opponent's strategy. Therefore, this allows for us to balance between exploiting the opponent and gathering information while still maintaining a good reward. Given this, there are some limited techniques we can attempt to use for estimation purposes. These "strategies" are ultimately the causal graph that we can try to use to understand the human's play-style.

Another benefit is that the game states are completely observed. The only thing that is unobserved is the human strategy. Ultimately, the unobserved human strategy poses a large problem in estimation.

Otherwise, the game does match a lot of properties that we want to investigate. Each

game is only a single action and have no game-imposed requirement of any dependencies between games. We can elongate this by having a "best-of" in a sequence of games, but investigating this is not actually different from just observing win percentage over the sequence. It is unrealistic to ever assume we have enough information on a single player to understand their playing style, but we can assume that we have enough information to understand the population distribution of playing styles.

3.5.1 Generation of Strategies

From the most basic (and surprisingly optimal) standpoint, generating RPS is very simple. We just have to choose an underlying distribution and sample from it, and we can make it pseudo-stationary by allowing the underlying distribution to change with probability p every round. Unfortunately, this is not very human-like. The first problem is that humans are known to be bad at being random and will have an implicit bias. Furthermore, over the course of a game, humans almost always fail at generating *independent* random actions. Despite the fact that nothing in the game forces independence, humans will inherently introduce their own dependencies into the game. Humans will instead produce some flowchart of what happens after each action, whether they want to change things up because they won or lost over a certain window, how many times they've played a certain action over a window, etc. Thus, a true model of human players would require some modeling of these possible thought sequences. We can view this as a player-type conditioned longer-window Markov Decision Process, where the state is the last N observations plus the player type, and that state determines the distribution of next action. Note that the observations includes some side information, such as the number of wins and losses over the last N games.

The estimation of longer-window strategies is definitely desirable, as it would allow us to better understand the human player. One basic example is a player that constantly chooses R, P, S, R, P, S, ... in sequence. Only understanding 0-window biases will deem this equivalent to a uniform action player, but obviously the actions are not independent. An

agent that is able to detect this dependency can theoretically achieve a reward of 1 every game.

However, a major problem is that we don't know an optimal window length. While most humans do not have a long window length (probably less than 10), there is no guarantee that the window length is fixed. Thus, we have two options: choose the longest realistically estimable window length as a fixed length or maintain all possible distributions on window lengths less than some maximum, and estimate which window length is most likely. Even ignoring the computational requirements, the first option is not flexible toward shorter term strategies. The second has the problem of having to estimate the window length, which is a difficult problem in itself, especially when the strategies of different window lengths are not necessarily distinct. For example, a strategy centered around number of rocks played in the last 5 games will look similar to a strategy centered around the number of wins, if the player has a specific bias to playing paper.

Finally, humans may change their strategy completely with one of many causal reasons at any point in time. Without strict prior information about when a strategy shifts, the estimation of a change will take some time, meaning the agent may be suboptimal until it recognizes the change.

Furthermore, the strategies themselves can have significant overlap in that multiple strategies can produce the same pattern of actions for a long string of actions, especially if strategies are nominally changing. For instance, the frequent switching between two longer window strategies can appear as a single short window strategy. However, the other struggle with RPS is that we never have ground truth in the estimation problem, so these two regimes can never be accounted for.

3.5.2 Hotspots

One of the main techniques to aid in the estimation problem for RPS is to create a discretized estimation space via the use of strategy hotspots. The idea of hotspots is to cluster population data into H discrete clusters, where we decide H based mostly based on estimation ability. One of the hotspots can always be the minimax solution. However, the minimax solution does not produce any average gain in reward.

Figure 3.3: A clustering of good strategies ($K = 4$) in a 0-window context (bias only). Zero-window strategies are represented by the 2-tuple $(\mathbb{P}[R], \mathbb{P}[P])$.

Using this method, we collected a real dataset using some students in our senior design class ECE 180D. In total, we were able to collect about 500 samples from 11 people. One demonstration on real data that we collected is shown in Figure 3.3. Just as the whispering

says, human players tended to play rock more and so the priors of choosing a good strategy that heavily favors paper makes the most sense.

From these hotspots, we can see that these effectively deal with the digital dynamics. Instead of trying to estimate an exact state, we now only have 4 states (including the "information gathering" state of providing minimax) that we have to estimate. Each state has an associated representative "good enough" action should we be able to estimate. Further, we also have population priors as to how often we expect to be in each state based on past examples. It is much easier to choose among H fixed options (provided by population data) rather than to smoothly learn a state within some error margin.

This method also reveals one of the important asymmetries of causal reasoning. Showing that a hypothesis is true is difficult while showing a hypothesis is false can require only a single data point. A pure gradient descent method would not be able to capture this, and so we would have to move into more Bayesian techniques. Hotspots allow for the quick estimation of player strategy with some understanding of population transition dynamics. Therefore, the use of quantization is a good way to balance estimation quality while preserving our goal of quick estimation.

3.6 Collaborative Game - Sets

After testing Liar's Poker and Rock Paper Scissors, both competitive games, we find *collaborative games* fit the idea of understanding human interactions better. In adversarial games, the environment is the *opponent*. In such an environment, the ideal flow of information between competitors is to have as little as possible. Thus, the onus on forcing information flow is on the structural design of the game. However, it is in both players' best interest to lie and deceive as much as is allowed to by the game, which lowers information quality of actions to the agent. In the collaborative case, both players are now *partners*. The partner does not have an incentive to force dynamics and so the problem results more toward natural

dynamics which can be much better behaved.

Thus, we also investigate *Sets*, a collaborative card game. First, the basic rules are as follows:

1. There is a deck of $D = R \times S$ cards, each with a rank (of which there are R) and suit (of which there are S). Players are each dealt $N \leq \frac{D}{2}$ cards for a 2-person game. When $N = \frac{D}{2}$, then the game state becomes fully observed.

2. There are actually two phases where an action takes place. At the beginning of each round, there is a "betting" round where all players guess the number of tricks they think they can take. In a collaborative game, we assume the sum of tricks guessed is equal to N. In a competitive game, we assume the sum cannot equal N. In the collaborative 2-player game, only one person needs to guess (as the other will naturally be the complement). For now, the guesser will be given to the environment.

3. Every turn, each player plays one card, starting with the winner of the previous round. All following players must follow suit if they have the suit. If they do not, they can play any other card, which normally would concede the trick to the partner. One variation includes a trump suit that would win the trick, thereby ordering it as trump rank ¿ on-suit rank ¿ any other card. The highest valued card of the round wins the trick. The winner takes the trick. There are a total of N tricks in a round.

4. Reward is given $+1$ if the number of tricks obtained is equal to the initial guessed goal, -1 if not.

This game is somewhat different from many of the previous games in that luck plays a small factor in this game because of the betting system. The goal is not to get the most tricks but the correct number of tricks. Therefore, there is a skill to understand what to bet and how to play in order to achieve such a bet.

Consider the collaborative version described. In more complicated versions of the game, a human may not understand what is the "optimal" bet, but could choose one that makes sense. From there, the agent's goal is to guide the flow of the game so that there is probabilistically the best chance of achieving the goal together. This makes this game a simplistic analogy to the education example. Here, we want to assume that the agent, up to their estimation abilities, will always make the optimal move.

Furthermore, the game is highly flexible. We already indicate the possibility of competitive and collaborative versions, which should have similar concepts. There is also a setting of how much is being observed. Unlike the competitive game, we no longer have to assume that someone could be playing completely adversarially, although we can consider it. Instead, the main parameters we care about for the human is their *understanding*, and the dynamics of that understanding through time.

3.6.1 Trivial Examples

In this game, the observation space of the game from either player is the cards that are in their hand, the cards have been played this round, the cards that have already been played in previous rounds, the number of tricks that each player has, and the goal value of each player.

Consider the most trivial game, $D = 2$, $R = 2$, $S = 1$, $N = 1$. In this version of the game, we can denote the cards using one-hot representation both to stay consistent to larger versions of the game and to display one's hand as a single array similar to a card. For brevity, since this game only has one turn, we merge the cards played this turn and previous turns into a single observation. When actually playing the game, the action space is only a single action when the game starts, so the only "action" is the choice of the goal. The tabular state-value function is displayed in Table 3.1.

Immediately looking at this value table, we see that the value is just a shifted XOR

Hand	**Observed**	**Goal**	**Value**
$[1,0]$	$[\cdot,\cdot]$	0	1
$[1,0]$	$[\cdot,\cdot]$	1	-1
$[0,1]$	$[\cdot,\cdot]$	0	-1
$[0,1]$	$[\cdot,\cdot]$	1	1

Table 3.1: Trivial Tabular Sets Value Function

function. For instance, if the hand is represented as $[c_0, c_1]$, one way to represent the value function is simply $2(c_0 \oplus \mathit{goal}) - 1$, where the arithmetic operations just shift the $0 - 1$ properties of the XOR into the 1 and -1 range of this problem. There are many other representations as we have some other guarantees in the problem, such as $c_0 \oplus c_1 = 1$. Thus, the value function for all valid states depends only on 2 of the 5 variables.

Just to give the next most trivial game, we consider just adding a single card $D = 3$, $R = 3$, $S = 1$, $N = 1$. This makes the game partially observed. The values are represented in Table 3.2.

For example, assuming the environment is given the hand $[0, 1, 0]$. In the first stage of the game (goal selection), we don't observe the opponent, resulting in the states of line 7 and 8, which both have a value of 0. The final value depends on whether the opponent has a higher or lower card, which we cannot know, resulting in the 0 value. However, if the agent is given the hand $[0, 1, 0]$, then any rational goal call should result in a win. In fact, the call from the environment should give the agent all the information about the future game without the game even being played.

Notice that for the hand $[0, 1, 0]$, we now have another nested XOR operating as the value function. The XOR-like pattern continues into more complex versions of the game for the same conceptual reason. Having higher cards (from the rank perspective) will generally prefer higher goal values, while having lower cards generally prefer having lower goal values.

Hand	**Observed**	**Goal**	**Value**
$[1,0,0]$	$[\cdot,\cdot,\cdot]$	0	1
$[1,0,0]$	$[\cdot,\cdot,\cdot]$	1	-1
$[0,1,0]$	$[1,0,0]$	0	-1
$[0,1,0]$	$[1,0,0]$	1	1
$[0,1,0]$	$[0,0,1]$	0	1
$[0,1,0]$	$[0,0,1]$	1	-1
$[0,1,0]$	$[0,0,0]$	0	0
$[0,1,0]$	$[0,0,0]$	1	0
$[0,0,1]$	$[\cdot,\cdot,\cdot]$	0	-1
$[0,0,1]$	$[\cdot,\cdot,\cdot]$	1	1

Table 3.2: Most Trivial Partial Information Tabular Sets Value Function

It may not represent exactly the XOR when the value of the goal increases, but the same structure is present.

3.6.2 Dimensionality and Causality

We expand outward to the class of all collaborative, fully-observed 2-player games. There are several properties about these games that are interesting. First, if both agents know the optimal move at any point in time, then as long as the initial goal is set to have a non-zero probability of winning, *i.e.* their value is not -1, then they always can win, which means that the bot's value function only has -1 or 1 if the environment is optimal. When we move to partially-observed games, this property no longer holds, as evidenced in Table 3.2.

Another observation is that the second turn of an N–card game is essentially functionally equivalent to an $(N-1)$–card game. All we have to do is effectively re-index the ranks after removing the cards played in the first turn from the deck and subtract the trick from

the winner's goal (with any goals out of bounds of the $(N-1)$–card game immediately having value -1). Of course the nice thing about this property is that learning can be done recursively. As such, we can actually see that the game state is Markovian, just as most other games.

However, there is something more to it from a causal perspective. The main thing to note is that the $(N-1)$–card game is a lower dimension game than the N–card game. In fact, there can be equivalent state-action pairs in the N–card game that result in the same state in the $(N-1)$–card game, suggesting that there are redundant state-action pairs in the N–card game. Capturing this redundancy highlights a basic form of causal structure in dimensionality reduction. Ideally, we would be able to capture this dimensionality reduction within the game so that the recursive properties can be used. In particular, this means that in two instances of games, the same card could be connected to different nodes in a graphical model.

Overall, there are other possible ways of condensing these variables further via approximate dimensionality reduction, *e.g.* humans may not remember every card that has been played in the round, and so more have a feeling for how strong the remaining cards are in their hand. Using these concepts allows us to both explain the reasoning behind the games but also construct meaningful understanding of how well the human understands the game and the information that they have received. Thus, we expect to be able to causally restrict this game into very few variables which can improve explainability. From a condensed state, we can also attain much better explainability in possible human strategies as a result.

3.6.3 Interventions

One of the main way that Sets is a nice game is that we can directly control some rule changes to create "interventions." Here, we define an intervention as a coded way to break the normal rules of the game to illustrate something. For instance, we can implicitly understand how information is shared in the game by adding an "information" intervention. Here, we only

add one additional rule. The agent is allowed one other action. This action will penalize the reward by some value (e.g. -0.25). The only thing this action does is if the agent was supposed to start, then the environment now starts.

In the last section, we brought up the fact that in the fully observed game, any action with a positive probability of achieving the result should always achieve the goal with optimal play. The other interesting property is that based on the rules, the number of tricks obtained is always known if the environment plays first in sequence that turn and so most of the next observation is guaranteed (it is completely guaranteed if the agent plays a higher card). As such, playing second in a round has more control, hence why this intervention makes sense.

This intervention is then analogous to a teacher providing one-on-one aid to a student who has been estimated to be likely to continue down a trajectory that is not good for the reward. The cost is the amount of time that it takes from the teacher's and student's other tasks. If that cost is warranted in producing a much better chance at a good result, then it should be done. The students over time, if trying to achieve the same goal, should also begin to learn that they should be applying actions that minimize the number of interventions needed, but ultimately this intervention sidesteps the problem of trying to change the student's dynamics in a course and acts more as an implicit understanding of the student's internal state.

In particular, this intervention is only able to help improve inaccurate policies in the *second phase* of the game. There is nothing that the agent can do if the environment ended up choosing a bad starting goal. Table 3.3 shows some example results in a 3-card fully observed game. In this table, these optimal policies may still have some small amount of noise, but almost always deterministically selects the optimal action. Semi-optimal allows for some suboptimal plays but will still be weighted toward optimal actions, and random means that all actions are chosen with equal probability.

Therefore, the work shows that including an intervention helps the win rate except when the play policy was already optimal, in which case it is essentially the same. In the fully-observed case, an optimal partner does not require any interventions. The average reward

for many is comparable, but not always better even with the win-rate increase. The level of improvement depends on the cost that the intervention requires. For example, the difference between a -0.75 and -0.25 reward is that two interventions with the -0.25 reward resulting in a win is still a positive overall reward, but it is negative for the -0.75 reward. Thus, the cost can be seen as a counter of the total amount of interventions that can be made. Ultimately, when the partner needs a lot of help, the intervention cost needs to be much lower for the total reward to go up. We can claim that these require additional attention from external resources in an educational space (tutoring, TA, etc.). However, for many of the cases where the partner has some middling understanding, even with higher costs for interventions, the interventions are worth the improvement of reward.

3.7 Conclusion

We have investigated three games that are simple enough to understand but complex enough to have interesting dynamics. However, we have also found that even these simplistic games pose significant problems in estimation and optimization. In particular, we have found that competitive hidden-information games are difficult to investigate because of the adversarial nature of the game. Collaborative games, though, are much more interesting because information sharing is part of the goal of the game. This provides a good starting point for understanding the problems that we will face in the future, especially when dealing with humans.

Partner Goal Selection	Partner Play Selection	Intervention Cost	Reward	Win Rate
random	random	$-\infty$	-705	32.35%
random	random	-0.75	-914.5	35.8%
random	random	-0.25	-726.75	37.6%
random	random	-0.1	-645.6	37.2%
optimal	random	$-\infty$	1697	92.4%
optimal	random	-0.25	1785.75	98.0%
semi-optimal	random	$-\infty$	583	64.55%
semi-optimal	random	-0.75	245.75	64.0%
semi-optimal	random	-0.25	609.25	70.7%
semi-optimal	random	-0.1	765.6	72.0%
semi-optimal	semi-optimal	$-\infty$	569	64.2%
semi-optimal	semi-optimal	-0.75	354.5	66.05%
semi-optimal	semi-optimal	-0.25	640.5	71.6%
semi-optimal	semi-optimal	-0.1	729.8	71.2%
random	optimal	$-\infty$	-295	42.6%
random	optimal	-0.25	-430	42.95%
optimal	optimal	$-\infty$	1975	99.35%
optimal	optimal	-0.25	1924.25	99.7%

Table 3.3: Results of 3-card fully observed game given interventions. Results are shown for the last 2000 iterations (last 20% of 10000 iterations starting from scratch). The intervention cost of $-\infty$ represents that the intervention action is not legal in this game variation.

CHAPTER 4

De-Biasing Generative Models using Counterfactual Methods

4.1 Introduction

In many fields such as medicine and economics, an explainable model, in particular a causal model, is needed to elicit the effectiveness of interventions. This process makes diligent use of prior knowledge, usually in a structural causal model (SCM) that instantiates unidirectional relationships between the variables using a Directed Acyclic Graph (DAG) [Pea09a]. The confidence needed in a causal model needs to be much higher than in a statistical model as one needs to instantiate beliefs that are invariant and exist outside the domain of the data. Traditionally, this knowledge comes from experimentally derived results, or domain experts with experimental level knowledge. As such, there is a strong interest in the deep learning community to integrate causal methods and information more directly with traditional deep learning architectures. Although recent results show progress in causal deep learning, most methods focus on either causal discovery or the use of prior causal information alone [ZNC19, KSD17, YCG19].

Generative models have been crucial to solving many problems in modern machine learning [KW13]. Since the VAE's inception, many have found that the disentanglement of latent spaces can lead to better performance in generalizability and fine-tuned control over disentangled features. In addition, many techniques have been proposed in recent years as to how to improve disentanglement, largely based on factorization and independence techniques

[HMP17, CLG18].

Recently, an effective approach that blends the space of causal models with generative neural networks was displayed with the CausalVAE, which allowed the decoder to learn a *causally* disentangled representation of latent space variables. One of the key contributions in that paper was the inclusion of a *Causal Layer*. Most impressively, the CausalVAE enforced a causal structure on generating images to noticeably disentangle intentionally dependent latent variables via the use of a causal layer. This causal layer's disentanglement allows the CausalVAE to generate causal interventions. Specifically, when intervening on endogenous variables, the CausalVAE is able to generate images that are outside the normal bounds of the training dataset, as the intervention does not affect the exogenous variables [YLC20].

Here, we combine the ideas of counterfactual causal reasoning and generative modeling by focusing on the causal layer of the CausalVAE. We modify the objective into learning a more refined, isolated causal structure that the latent space must go through, which we call the Causal Counterfactual Generative Model. This allows us to expand the use of the causal layer to more than just single interventions, to also to hypothesize and synthesize datasets of counterfactual causal models in interesting and useful ways.

4.2 Related Work

Causal discovery has increasingly been the focus of deep learning methods which seek to reduce the combinatorial complexity of brute force searches for causal models from observational data. Progress in DAG search using continuously differentiable loss functions and reinforcement learning for score functions has started to integrate deep learning methods with causal discovery and identification [ZAR18, ZNC19].

Building on initial deep causal discovery, causal generative models learn or use causal information for generating data and interventions. CausalGAN is a generative model that learns a prior Structural Causal Model (SCM) for images and label spaces and demonstrates

how interventions in the latent space can generate causally intervened images [KSD17]. DAG-GNN uses graph neural networks with a VAE architecture to extend causal discovery methods to more use-cases [YCG19]. CausalVAE uses a causal layer in the middle of a VAE architecture to learn an implicit causal model that can also generate unseen images with latent space interventions [YLC20]. Causal discovery with generative models capitalize on recent work in disentanglement to ensure the latent space has the necessary variable structure for causal identification [HMP17]. Finally, causal generative models have been used to address the issue of fair or "de-biased" data sets such as DECAF, a causally aware GAN architecture applied explicitly to tabular data [BKB21].

When causal models are known or hypothesized to contain measured confounders, statistical adjustment techniques have long been used to estimate causal effects when the structure is known or identifiable. Inverse Propensity Score Weighting (IPW), or advanced methods like Augmented IPW provide robust or doubly-robust ways to adjust for confounding bias [GQ10].

4.3 Background

4.3.1 Counterfactuals and Interventions

The SCM literature has long explored the benefits of interventions and counterfactual modeling once a causal model is known. Pearl introduces interventions using 'do-calculus' or the explicit setting of a variable to a specific value and calculating the resulting outputs [Pea09a]. In Figure 4.1 below, we introduce a 4-variable DAG with two exogenous and two endogenous variables. An intervention on the right shows how this is effectively breaking the parent nodes into the variable being intervened on, and explicitly setting it to a desired value (x), written using do-calculus notation $do(x)$. This operation allows us to directly fix the value of a latent variable and asymmetrically propagate its value to other variables. Intervened parents should have their adjusted values impact child nodes, but intervened children

should not adjust parent values.

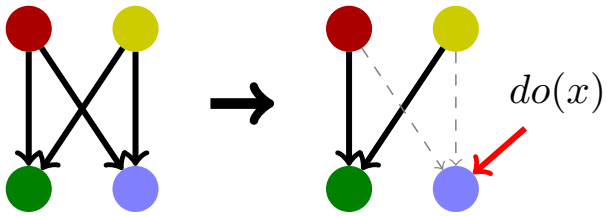


Figure 4.1: Example of a DAG on the left and a mutated counterfactual model on the right with an intervention setting the target variable to an explicit value x.

4.3.2 Counterfactual Models

Extending from the idea of interventions on instances of data, we define counterfactual models as a new model formed by removing a path deemed undesirable or a source of bias as seen in Figure 4.2. This could be a known bias present in the data generating process, or a desire to envision a new data distribution outside the training dataset with a specific graphical modification. Notice, unlike an intervention as in Figure 4.1, the target variable need not be set explicitly but still is a function of the other parent variables. This allows a data distribution to be generated in which the target is still a function of the remaining parent nodes, possibly simulating a "de-biased" or counterfactually constructed dataset, as opposed to explicit instantiations of the intervened variable.

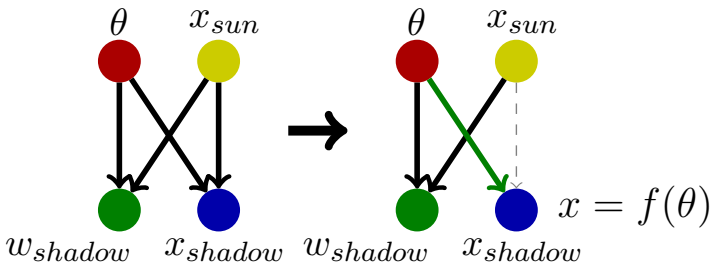


Figure 4.2: Example of a counterfactual model in which a single path is removed to simulate a new distribution of generated data.

4.3.3 Constructing a Causal Generative Model

Following the classic VAE model, given inputs $\mathbf{x}$, we encode into a latent space $\mathbf{z}$ with distribution q_ϕ where we have priors given by $p(\cdot)$ [KW13].

$$\text{ELBO} = \mathbb{E}_{q_\mathcal{X}} \left[\mathbb{E}_{\mathbf{z} \sim q_\phi} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathcal{D}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \right] \tag{4.1}$$

In [YLC20], the causal layer is described as a noisy linear SCM:

$$\mathbf{z} = \mathbf{S}^T \mathbf{z} + \boldsymbol{\epsilon} \tag{4.2}$$

which finds some causal structure of the latent space variables $\mathbf{z}$ with respect to a matrix $\mathbf{S}$. By itself, $\mathbf{S}$ functions as the closest linear approximator for the causal relationships in the latent space of $\mathbf{z}$.

A non-linear mask can be applied to the causal layer so that it can more accurately estimate non-linear situations as well. Suppose $\mathbf{S}$ is composed of column vectors $\mathbf{S}_i$. For each latent space concept i, define a non-linear function $g_i : \mathbb{R}^n \to \mathbb{R}$ and modify equation (4.2) such that

$$\mathbf{z}_i = g_i(\mathbf{S}_i \circ \mathbf{z}) + \boldsymbol{\epsilon} \tag{4.3}$$

where $\circ$ is the Hadamard product. In this formulation, the view of $\mathbf{S}$ changes from one of function estimation to one of adjacency. That is, if $\mathbf{S}$ is viewed as a binary adjacency matrix, the g_i functions take the responsibility of reconstructing $\mathbf{z}$ given only the parents, dictated by $\mathbf{S}_i \circ \mathbf{z}$. In the simplest case, if $g_i(\mathbf{v}) = \sum_j v_j$, the summation of all the values of $\mathbf{v}$, then Equation (4.3) degenerates back to Equation (4.2) [NZF19a].

Including the causal layer introduces many auxiliary loss functions that we mostly adopt [YLC20]. First is a label loss (4.4), where the adjacency matrix $\mathbf{S}$ should also apply to the labels $\mathbf{u}$. This loss is used in pre-training in its linear form to learn a form of $\mathbf{S}$ prior to learning the encoder and decoders. After pre-training, we apply a nonlinear mask f_i that

functions similarly to g_i, but operates on the label space directly, but with the same $\mathbf{S}$.

$$\ell_u = \mathbb{E}_{q_{\mathcal{X}}}\left[\sum_{i=1}^{n}\|u_i - f_i(\mathbf{S}_i \circ \mathbf{u})\|^2\right] \tag{4.4}$$

The latent loss tries to enforce the SCM, described by Equation (4.3).

$$\ell_z = \mathbb{E}_{\mathbf{z}\sim q_\phi}\left[\sum_{i=1}^{n}\|z_i - g_i(\mathbf{S}_i \circ \mathbf{z})\|^2\right] \tag{4.5}$$

Further enforcing the label spaces, we can define a prior $p(\mathbf{z}|\mathbf{u})$. We use the same conventions as in [YLC20] and say that

$$p(\mathbf{z}|\mathbf{u}) \sim \mathcal{N}(\mathbf{u}_n, \mathbf{I})$$

where $\mathbf{u}_n \in [-1, 1]$ are normalized label values. This translates to an additional KL-loss.

Finally, we apply the continuous differentiable loss function (4.6) and apply a scheduling technique to enforce the DAG [ZAR18, YCG19]. The main use is that $\mathbf{A}$ is a DAG if and only if

$$H(\mathbf{A}) := tr\left[(\mathbf{I} + \mathbf{A} \circ \mathbf{A})^n\right] - n = 0 \tag{4.6}$$

The scheduling is done via the augmented Lagrangian

$$\ell_h = \lambda H(\mathbf{A}) + \frac{c}{2}|h(\mathbf{A})|^2 \tag{4.7}$$

where at the end of every epoch, the scheduling update is

$$\lambda_{t+1} = \lambda_t + c_t H(\mathbf{A}_t) \tag{4.8}$$

$$c_{t+1} = \begin{cases} \eta c_t & |H(\mathbf{A}_t)| > \gamma |H(\mathbf{A}_{t-1})| \\ c_t & \text{else} \end{cases}$$

where we set $\eta = 2$ and $\gamma = 0.9$.

4.3.4 Causal Estimation

There are numerous ways to estimate a causal effect once the model has been identified. Perhaps the most common and simplest is the Average Treatment Effect ($\widehat{ATE}$), which is simply the difference of means between a population ($index = i$) treated and untreated group, assuming a binary intervention variable (D), and an outcome variable (Y) as in equation (4.9).

$$\widehat{ATE} = \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i|D_i = 0] \tag{4.9}$$

This naïve method does not consider any confounding variables. One common way to adjust for such confounding bias is to use propensity scores ($\hat{\pi}(X_i)$), which is a model for how likely a sample is to receive the treatment based on the measured covariate factors. The inverse of the propensity score can then be used to weight each sample as in equation (4.10) and thus adjust for the bias of any measured confounders.

$$\widehat{ATE}_{IPW} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{D_i Y_i}{\hat{\pi}(X_i)} - \frac{(1-D_i)Y_i}{1-\hat{\pi}(X_i)}\right] \tag{4.10}$$

Finally, more recent developments in double-robust methods specify both an outcome model and an exposure/propensity score model which can provide accurate estimation if either one of the models is misspecified. Augmented IPW (AIPW) is a specific method that extends IPW below with a set of outcome models estimating the outcome variable as a function of the intervention and all covariates as introduced in [GQ10].

4.4 Problem Setting

4.4.1 Sun Pendulum Image Dataset

A toy pendulum image dataset is introduced in [YLC20]. This dataset is generated by sweeping sun positions (x_{sun}) and pendulum angles (θ) to produce realistic shadow width

(w_{shadow}) and shadow locations (x_{shadow}) from deterministic non-linear functions. Figure 4.3 shows the DAG for this model and an example generated image, in which the sun and pendulum variables are exogenous, and the shadow variables are endogenous. Thus, any causal model will learn to reconstruct the shadow variables from the sun and pendulum variables. Such relationships in observational studies are often invertible as correlation has no directionality. Thus, without causal disentangling, an intervention on shadow position would likely adjust the sun position to match.

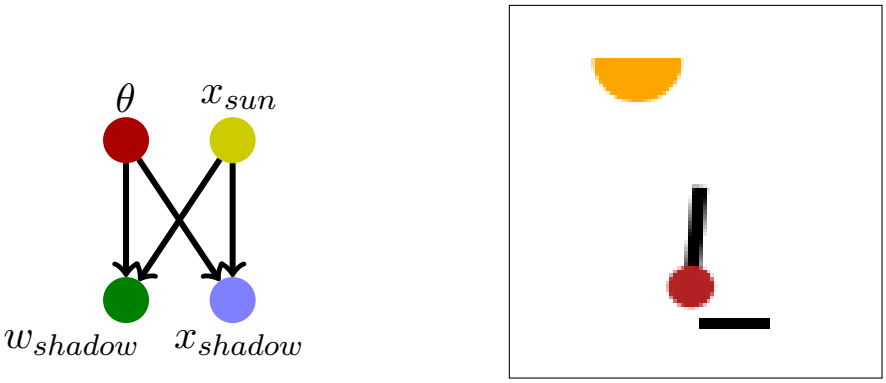


Figure 4.3: Pendulum toy image dataset DAG and example image

Each pendulum entry is determined by the two exogenous variables, pendulum angle (θ) and sun position (x_{sun}). Here, the data samples are generated roughly where the angle of the pendulum and the angle of light from the sun range $\in (-45, 45)$ degrees, generated independently. Then from that, we calculate a physics-based interpretation of the shadow position and width. In the calculation of both of the endogenous variables, we introduce non-linearities by operating on trigonometric functions. In the shadow width case, we also deal with a non-linear maximum function as the width is a positive value. Afterward, in most of the datasets unless otherwise mentioned, we add Gaussian noise to the endogenous variables in the dataset so that the SNR is 10dB.

This dataset is used to demonstrate causal generative model quality through reconstruction fidelity as well as causal learning by intervening on parent and child nodes, showing interventions only propagate forward from parents to children and not vice-versa [YLC20].

4.4.2 Tabular National Study of Learning Mindsets Data

To analyze our methods in a tabular setting, we use a simulated dataset based on The National Study of Learning Mindsets [min21]. This was a randomized study conducted in U.S. public high schools, the purpose of which was to evaluate the impact of a nudge-like intervention designed to instill students with a growth mindset on student achievement. We use a simulated subset of the data based on a model fit to the statistics of the original dataset (the actual dataset was not publicly released). The study includes measured outcomes via an achievement score, a binary treatment of a growth mindset educational intervention (not to be confused with a causal intervention), and 11 other potential confounding factors that could be parents of both the treatment and outcome. We select two of these confounding variables: an average measure of the fixed mindset at each student's school (inversely correlated with achievement and educational intervention) and the students' self-reported expectations of their own success (positively correlated with achievement and educational intervention). The full correlations between all four variables can be seen in Table 4.1. Thus, we maintain a hypothesized DAG structure as in Figure 4.4 identical to the pendulum model. Note that our interest is in regenerating the dataset with the treatment and targets as functions of the confounders, so we do not learn the effect of the intervention on the outcome. We will use our methods to generate datasets in which we can estimate the ATE to estimate our causal effect using a simple difference of means.

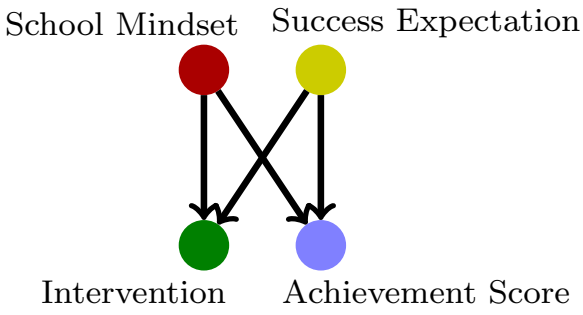


Figure 4.4: School Mindset DAG

The intuitive belief is that a naïve estimate of the ATE, calculated as the difference of means as in equation (4.9), would contain a positive bias due to the confounding variable of a student's own expectation. Students with higher expectations are more likely to participate in the growth mindset course (self-selection bias) but are also likely to have higher achievement anyway. Statistical adjustment techniques, such as Inverse Propensity Weighting (IPW), attempt to control for such confounders by measuring and weighting the effect based on the propensity to be treated. We will use such methods as a baseline for comparison, as we will first generate a dataset approximating the existing data distribution while learning some causal features. We will then employ a counterfactual model removing a confounding link and demonstrate a simulated dataset in which the naïve ATE aligns with the ATE measure using the statistical adjustment methods.

Table 4.1: Correlation of Mindset Variables

	SM	SE	D	Y
School Mindset (SM)	1	−0.054	−0.046	−0.111
Success Expectation (SE)	−0.054	1	0.059	0.439
Intervention (D)	−0.046	0.059	1	0.221
Achievement Score (Y)	−0.111	0.439	0.221	1

4.5 Causal Counterfactual Generative Model

We start from many of the same concepts as the original CausalVAE but begin by changing the enforced structure of the causal layer, allowing us to make direct modifications to the layer after training.

4.5.1 Limits of the CausalVAE for Counterfactuals

The causal layer in [YLC20] has a purpose of passing some causal information about the latent space through from parents to children. However, there is a fundamental difference in how we would like to interpret our problem. Reiterating (4.2),

$$\mathbf{z} = \mathbf{S}^T\mathbf{z} + \boldsymbol{\epsilon}$$

Whatever causal structure is learned by $\mathbf{S}$, there will always be a "leakage" of information via $\boldsymbol{\epsilon}$. This $\boldsymbol{\epsilon}$ can be viewed as the output of a vanilla VAE, meaning that theoretically it can contain contribute everything for image generation. This leakage informs $\mathbf{z}$ without passing through the causal layer, so it weakens the need for $\mathbf{S}$ to learn all the causal structure of the problem. In the image space, this leakage of information improves generation and reconstruction and hence is desirable. However, it does not align with our objective of finding a good underlying causal structure. In the most extreme case, we could, in theory, find $\mathbf{S} = \mathbf{0}$, which is still a valid DAG. In this case, no remaining causal information remains in the layer and the entire CausalVAE reverts to a normal β-VAE.

4.5.2 Envisioning Bias-Free Models with CausalVAE

Here, we introduce CCGM as a modified and extended version of the CausalVAE, allowing for counterfactual models. In particular, we can directly manipulate the causal layer so that undesirable causal links learned from the data can be broken.

In CCGM, the encoder directly generates the output $\mathbf{z}$, which is enforced to be standard-normally distributed. We pass this through our causal layer as one final *mutatable* bottleneck

$$\mathbf{z} = \mathbf{S}^T\mathbf{z} \tag{4.11}$$

That is, it instantiates a linear SCM. One main distinction is that we solidify the structure of $\mathbf{S}$ by having exogenous and endogenous priors. This way, $\mathbf{S}$ can be split into a DAG term

and a diagonal term:

$$\mathbf{S} = \underbrace{\mathbf{A}}_{\text{DAG}} + \underbrace{\mathbf{D}}_{\text{diag.}} \tag{4.12}$$

where $\mathbf{D}$ has 1 on the diagonal for exogenous variables and 0 if endogenous. This ensures that the trivial solution where $\mathbf{S} = \mathbf{I}$ is never learned and enforces a causal relationship from the exogenous variables to the endogenous variables.

Similarly, we add the non-linear mask to the causal layer just as in equation (4.3), but dropping the leakage.

$$\mathbf{z}_i = g_i(\mathbf{S}_i \circ \mathbf{z}) \tag{4.13}$$

When separating adjacency and estimation, we necessarily want to have a pre-training step for 5 epochs, where we train $\mathbf{S}$ to recognize the adjacency of the labels before applying the non-linear mask. After the pre-training, we apply training on both the $\mathbf{S}$ matrix and the non-linear mask, but there should be fewer changes as the mask should take care of the function approximations.

The ultimate goal of our work is to propose counterfactual causal models by directly manipulating this $\mathbf{S}$ matrix. The framework proposed by [YLC20] requires one to retrain the entire model to generate a counterfactual $\mathbf{S}$ while fixing a path in the graph to zero, as their intervention method does not deal with the leakage. This is an expensive task in both time and computing power, making it unscalable for larger $\mathbf{S}$'s. Our method allows us to generate data about a hypothesized counterfactual space directly by breaking links in the causal graph, without the need to retrain the neural network.

4.5.3 General Structure of CCGM

While one could work with image-to-image VAEs, in our examples, we leverage as much tabular data as we can to reduce computational needs. In the pendulum example, we know the labels can be used as a perfect reconstruction of the data and so the labels that are provided act as at least a perfect bottleneck, containing more information than needed, in

the reconstruction of images.

Furthermore, the label-to-label structure can be used as a pre-training step in determining a causal matrix. It then becomes a natural extension to apply the CCGM to tabular data. We no longer require a VAE setup, although we preserve the mild non-linear networks which allow for more complex causal functionality. Our experiments section will show a CCGM capable of generating tabular data with a reasonable representative distribution and a bias removed distribution. Note that noisy tabular data with hypothesized causal models (no known ground truth model or guarantee of endogenous/exogenous priors) present a new set of identification and estimation challenges.

4.6 Experiments

In this section, we evaluate the effectiveness of causal generative models on tabular and image datasets, by answering the following questions: (1) how does the performance of CCGM compare to the state-of-the-art methods in reconstruction and causal logic; (2) how effective is CCGM for eliminating biases in image and tabular datasets; and (3) how CCGM generates counterfactual models without extra training allowing for diverse and flexible data-generation. We compare the performance of CCGM and CausalVAE to generate counterfactual samples from a fixed causal model [YLC20]. We further compare CCGM to advanced statistical adjustment methods for generating "de-biased" datasets vs. controlling biases statistically.

4.6.1 CausalVAE

Our experiments with the standard CausalVAE found that the model could handle interventions on specific latent space data, meaning that its decoder could causally disentangle some concepts. However, non-zero interventions did not appear to be working as intended. Figure 4.6 shows a sweep of interventions on the pendulum and sun position data, respectively, on

the same image. Notice that the first intervention, corresponding to 0, works as intended. However, the pendulum does not change outside the 0 value, while the sun changes somewhat in an expected fashion, but the shadow does not respond.

Furthermore, we noticed little to no change in the results of certain interventions when generating counterfactual models, such as in Figure 4.5 below where a post training removal on the path from sun location to shadow position did not remove the effect propagation. These findings reflect that the CausalVAE was not designed to learn the full causal structure due to the leakage in $\boldsymbol{\epsilon}$.

Figure 4.5: Result which still shows effect propagation (shadow moves) after removing the path from sun location to shadow location in CausalVAE method

4.6.2 Label-to-Label

Our initial experiments pertain to the label-to-label space, where we have four parameters (labels) that provide perfect information for image reconstruction.

We start with a set of labels $\mathbf{u} \in \mathbb{R}^n$, where $n = 4$. These four labels correspond to the pendulum angle θ, the sun position x_{sun}, the shadow width w_{shadow}, and the shadow position x_{shadow}, respectively such that $\mathbf{u} = [\theta, x_{sun}, w_{shadow}, x_{shadow}]^T$. Then, $\mathbf{u}$ passes through an encoder to generate $\mathbf{z} \in \mathbb{R}^n$, where we enforce the prior of $p(z) \sim \mathcal{N}(0, \mathbf{I})$. This step allows us to sample new labels from the latent space drawn from a Gaussian distribution, as we see in the classic VAE [KW13]. Now, we subject $\mathbf{z}$ to the learned causal layer. Based on our designation of $\mathbf{u}$, we set $diag(\mathbf{S}) = [1, 1, 0, 0]$, representing the exogenous and endogenous

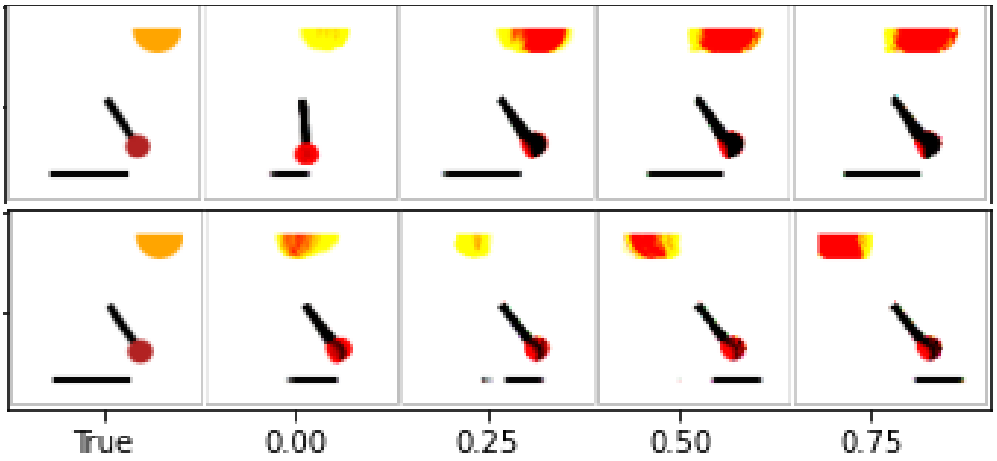


Figure 4.6: A sweep of the pendulum angle (top) and sun position (bottom) in the latent space for CausalVAE. Values are chosen to attempt to see changes. Outside the 0 intervention, other interventions do not seem to make sense, especially with the shadows.

variables of $\mathbf{u}$. Equation 4.13 is applied to $\mathbf{z}$, and the information of $\mathbf{z}$ should be preserved through the causal layer, even though the exact information of the endogenous variables is intentionally dropped.

Finally, this reconstructed latent space vector $\hat{\mathbf{z}}$ is passed into a decoder to reconstruct the original labels $\hat{\mathbf{u}}$. For consistency of visualization and easy of human understanding, we pass these labels into a separate image generator to create all visualized images.

CCGM Generates Clean Label-to-Label Interventions. Since there are few parameters in the label-to-label space we are able to generate clean counterfactual models as well as interventions.

Our primary results for label-to-label is shown in Figures 4.7 and 4.8. The top row of both figures show interventional sweeps on both θ and x_{sun}, respectively. We take a true image and apply a range of interventions sampled from the range of the resultant sampling distribution ($\mathcal{N}(0, 1)$) to generate counterfactual samples. Interventions on exogenous variables shows a response in the shadow variables, but the other exogenous variable should stay constant.

Then, we apply the ideas of a counterfactual model. Instead of doing interventions on

specific values, we break the link of x_{shadow} with θ and x_{sun} in $\mathbf{S}$, respectively. Afterward, if we do the same interventions, the shadow position no longer responds to that intervention. While some results can be subtle, in Figure 4.7, the final image shows a noticeable difference in position before and after the counterfactual model and in Figure 4.8, the first and last interventions both show differences. The connection to shadow width remains, and so the shadow width still responds to the swept variable.

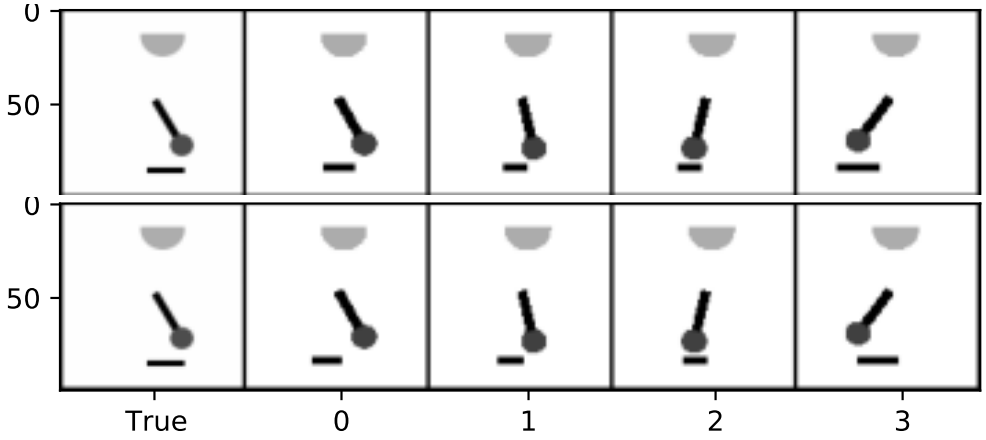


Figure 4.7: Label-to-label (Top) Image response to a sweep of the pendulum angle. Notice that for all interventions, the shadow responds to the pendulum. (Bottom) Image response after shadow position is de-biased from pendulum angle. In particular pay attention to the right-most image. While subtle, the shadow positions between the top and bottom image are very noticeable. A quick scan from left to right on all the intervened images suggests that the midpoint of the shadow remains constant throughout all the swept images. However, it is worth noting that the shadow width still responds as if the shadow had moved to its location.

4.6.3 Label-to-Image

We then consider the more challenging problem of generating an entire image from the label information. Thus, we propose a label-to-image generative model based on the decoder of

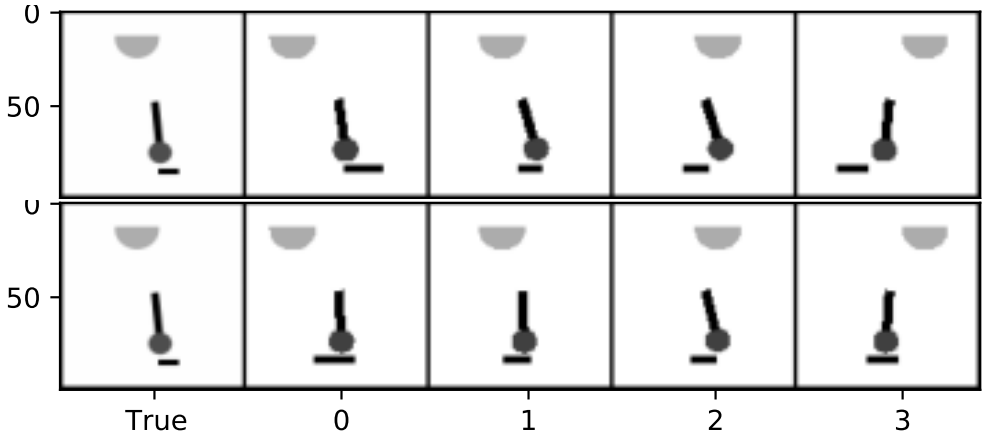


Figure 4.8: Label-to-label (Top) Image response to a sweep of sun positions. Again, notice that the shadow responds to the sun's position. (Bottom) In this case, the changes are more noticeable in that the shadow position remains constant throughout the row and is very different from the expected locations given the sun.

a VAE to use the disentanglement granted by the Causal Layer. We can use a pre-trained version of the **S** matrix coming from the label-to-label VAE to start our training of the causal generative model. For the sake of computational power, we keep the dataset in grayscale to reduce the image size by at least a factor of 3, but the physics aspects are still present.

Other than this additional pre-training step to learn the **S** matrix, the encoding step and the causal layer steps are still operating exactly the same as in the label-to-label VAE. We simply attach an image decoder after the causal layer. As in [YLC20], we see that the images have mostly disentangled the endogenous variables and the encoder is able to create an image where interventions can happen. These images are displayed in Figure 4.9. Notice especially in the shadow position intervention that the sun position and pendulum angle have not changed. With CCGM, we can recreate the results from the label-to-label VAE in the label-to-image generator. These results are shown in Figures 4.10 and 4.11.

As in the label-to-label space, the top row of both of the figures shows the sweeps of θ and x_{sun} in their latent space. With the interventions, the shadow responds accordingly and

the complementary exogenous variable stays relatively consistent. The bottom row shows that the shadow position no longer responds to the interventions that are being enforced. In both Figure 4.10 and 4.11, the first and the last interventional images show noticeable movement from the non-counterfactual interventions.

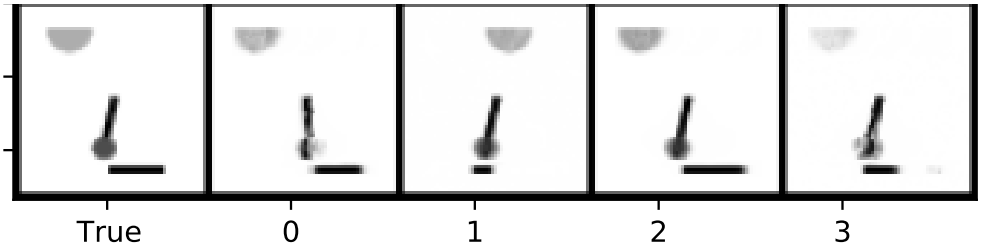


Figure 4.9: Interventions in the label-to-image VAE. Note that shadows respond to pendulum angle and sun position.

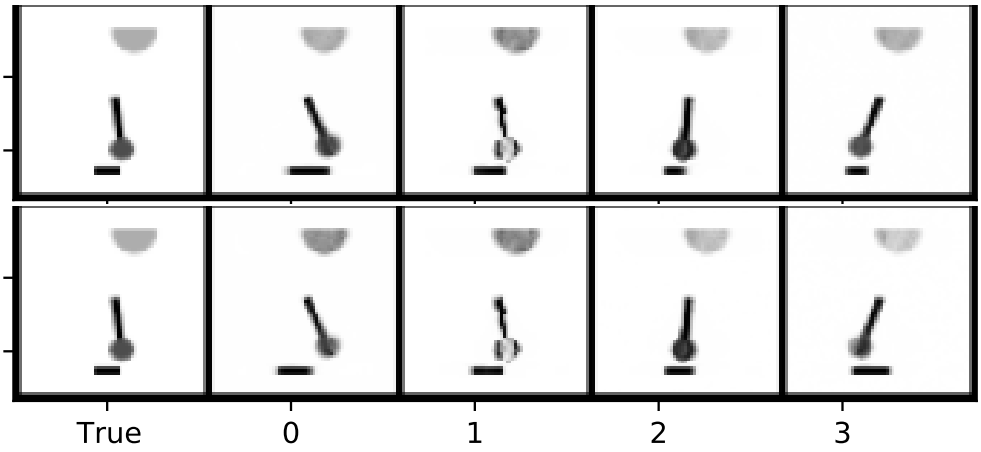


Figure 4.10: Label-to-image (Top) Image response to a sweep of the pendulum angle. Notice that for all interventions, the shadow responds to the pendulum. (Bottom) Image response after shadow position is de-biased from pendulum angle. In this case, both the first and last intervened images show noticeable differences from the pre-counterfactual images and one can observe that the shadow midpoint remains consistent across the row.

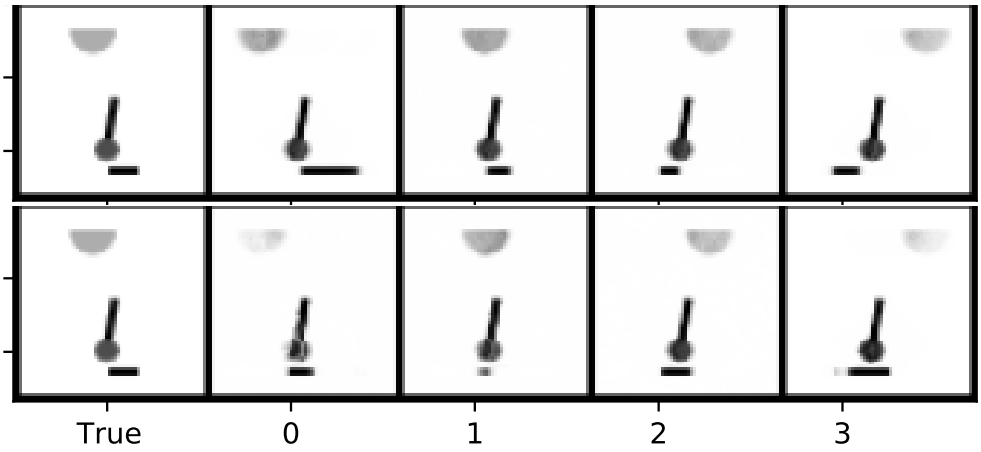


Figure 4.11: Label-to-image (Top) Image response to a sweep of sun positions. Notice that the shadow responds to the sun's position. (Bottom) image response after breaking the sun position to shadow position link.

4.6.4 Mindset Data

We begin by first training our CCGM method on the original Student Mindset data. In Figure 4.12, we see the generated achievement score from our model compared to the original dataset. The model is able to regenerate each of the feature distributions. We take the top 30% as having the intervention (1) and lower 70% as no intervention (0) since this matches the base rates in the original dataset. The current CCGM operates on continuous variables, and we treat the intervention values generated as a probability of treatment.

The results of an ATE on the generated dataset vs the three baseline measurements on the original dataset of ATE, IPW, and AIPW are shown in Figure 4.13. The generated dataset overestimates the ATE bias. We intervene on the path from student expectation to achievement scores, breaking the strongest positive correlation and see the CCGM "De-Bias" ATE estimation clearly drop below the advanced adjustment baseline methods. This makes intuitive sense since we leave the negatively correlated school mindset confounder which likely plays a small role in underestimating the ATE. Note that the advanced methods themselves are not necessarily ground truth, but they reflect an approximate ATE we would

Figure 4.12: Generated distribution of achievement scores closely matches the distribution in the original dataset.

expect a "de-biased" dataset to have.

Further, Table 4.2 shows the full results for all models and multiple de-biasing schemes. It is clear that our generative model is able to produce a "de-biased" dataset accounting for the positive ATE bias for the Student Expectation confounder. Although removing both confounders does marginally increase the 95% bounds for out ATE as we would expect for a negative bias, it does not also do so when we only remove the school mindset. Here it becomes clear our model puts very little weight on this negative bias, a possible limitation of our model with the noisy nature of this dataset and a point of interest for further inquiry. All results are shown with empirical standard deviations and 95% confidence interval bounds using bootstrap sampling methods ($iters = 100$).

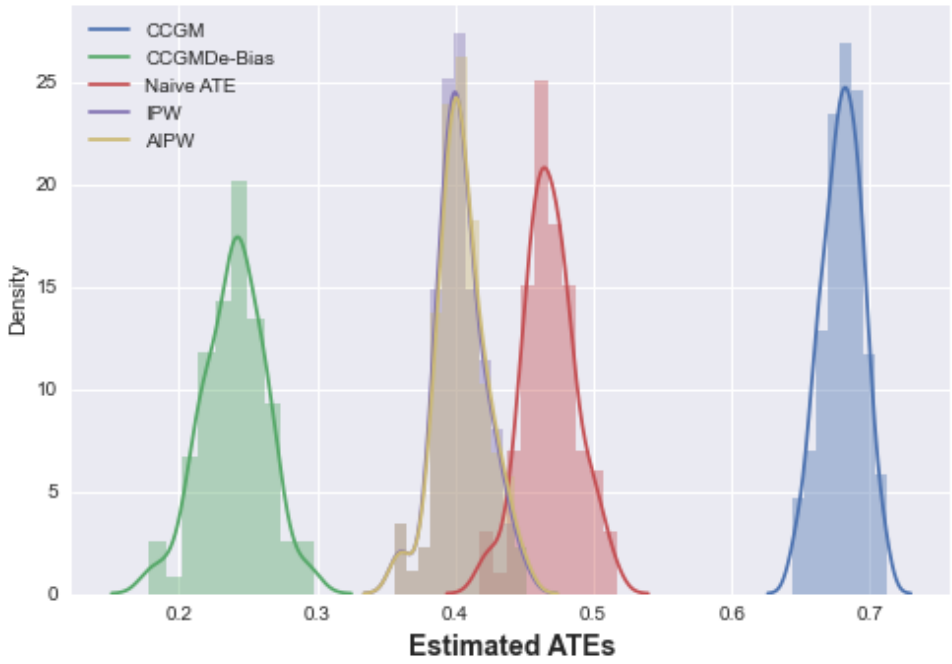


Figure 4.13: Bootstrap distribution of ATEs for various methods and baselines sampling the entire dataset ($n = 10391$) 100 times.

4.7 Conclusion

In this chapter, we demonstrate the value of CCGM, an extension of previous causal generative model work that allows greater flexibility when considering counterfactual models and generating "out-of-distribution" data. We demonstrate the benefits of such a model on a simulated physics image dataset. We show the range of interventions and simulation of images outside the training data, and outside the ground-truth physics, with a simple adjustment after training the model. We then demonstrate results on a tabular dataset where ground-truth is not known. We show that we can learn the original data distributions, and simulate datasets which remove the impact of confounders in ways the make intuitive sense based on advanced statistical adjustment baselines. Much work is needed on refining the precision on noisy datasets, extending the framework to more complex causal models, and exploring the limitations based on the noise present and the target causal structure if known.

Table 4.2: Mindset ATE Results

	Mean ATE	Std Dev	[.025	.975]
CCGM	**0.680**	0.015	0.650	0.707
CCGM De-Bias Student Expectation	**0.240**	0.023	0.192	0.283
CCGM De-Bias Both	**0.242**	0.022	0.204	0.282
CCGM De-Bias School Mindset	**0.677**	0.017	0.647	0.712
AIPW	**0.405**	0.018	0.364	0.441
IPW	**0.404**	0.018	0.363	0.440
Naïve	**0.468**	0.019	0.426	0.507

We believe CCGM is a promising start within a growing field of work in causal generative models.

CHAPTER 5

Causal Structural Hypothesis Testing and Data Generation Models

5.1 Introduction

In most scientific fields, causal information is considered an invaluable prior with strong generalization properties and is the product of experimental intervention or domain expertise. These priors can be in a structural causal model (SCM) form that instantiates unidirectional relationships between variables using a Directed Acyclic Graph (DAG) [Pea09a]. The confidence in causal models needs to be higher than in a statistical model, as its relationships are invariant and preserved outside the data domain. In fields such as medicine or economics, where ground truth is often unavailable, domain experts are relied on to hypothesize and test causal models using experiments or observational data.

Generative models have been crucial to solving many problems in modern machine learning [KW13] and generating useful synthetic datasets. Causal generative models learn or use causal information for generating data, producing more interpretable results, and tackling biased datasets [KSD17, YCG19, BKB21]. Recently, [YLC20] introduces a *Causal Layer*, which allows for direct interventions to generate images outside the distribution of the training dataset in its CausalVAE framework, which we used in CCGM from Chapter 4 and continue to use in this chapter.

CausalVAE and CCGM focus on causal discovery concurrently with simulation (i.e. reconstruction error-based training). But in many real-world applications, a causal model

is available or readily hypothesized. It is often of interest to test various causal model hypotheses not only for in-distribution (ID) test data performance, but for generalization to out-of-distribution (OOD) test data. Thus, we propose CSHTest and CSVHTest, which are causally constrained architectures that forgo structural causal discovery (but not the functional approximation) for causal hypothesis testing. Combined with comprehensive non-random dataset splits to test generalization to non-overlapping distributions, we allow for a systematic way to test structural causal hypotheses and use those models to generate synthetic data outside training distributions.

5.2 Background

5.2.1 Causality and Model Hypothesis Testing

Causality literature has detailed the benefits of interventions, and counterfactual modeling once a causal model is known. Given a structural prior, a causal model can tell us what parameters are identifiable from observational data alone, subject to a no-confounders and conditioning criterion determined by d-separation rules [Pea09a]. Because the structural priors are not known to be ground truth, we assume a more deterministic functional form and can make no assumptions about identifiability [Pea09b]. Instead, we rely on deep neural networks to approximate the functional relationships and use empirical results to demonstrate the reliability of this method to compare structural hypotheses in low-data environments.

Structural causal priors are primarily about the ordering and absence of connections between variables. It is the absence of a certain edge that prevents information flow, reducing the likelihood that spurious connections are learned within the training dataset distribution. Thus, when comparing our architecture to traditional deep learning prediction and generative models, we show how hypothesized causal models might perform worse when testing within the same distribution as the training data, but drastically improve generalization performance when splitting the test and train distributions to have less overlap. This effect is

seen the most in small datasets where traditional deep learning methods, absent causal priors, can "memorize" spurious patterns in the data and vastly overfit the training distribution [AJB17].

Our architectures explore the use of the causal layer, provided with priors, as a hypothesis-testing space. Both CSHTest and CSVHTest accept non-parametric (structural only, no functional-form or parameters) causal priors as a binary Structural Causal Model (SCM) and use deep learning to approximate the functional relationships that minimize a means-squared reconstruction error (MSE). Our empirical results show the benefits of testing structural priors using these architectures to establish a baseline for comparison where stronger causal assumptions cannot be satisfied.

5.3 Causal Hypothesis Gen and Variational Model

5.3.1 Causal Hypothesis Testing with CSHTest

Our model CSHTest, uses a similar causal layer as in both CCGM and CausalVAE [BJP22, YLC20]. The causal layer consists of a structural prior matrix $\mathbf{S}$ followed by non-linear functions defined by multilayer perceptrons (MLPs). We define the structural prior $\mathbf{S} \in \{0,1\}^{d\times d}$ so that $\mathbf{S}$ is the sum of a DAG term and a diagonal term:

$$\mathbf{S} = \underbrace{\mathbf{A}}_{\text{DAG}} + \underbrace{\mathbf{D}}_{\text{diag.}} \tag{5.1}$$

$\mathbf{A}$ represents a DAG adjacency matrix, usually referred to as the causal structural model in literature, and $\mathbf{D}$ has 1 on the diagonal for exogenous variables and 0 if endogenous. Then, given tabular inputs $\boldsymbol{x} \in \mathbb{R}^d$, $\mathbf{S}_{ij}$ is an indicator determining whether variable i is a parent of variable j.

From the structural prior $\mathbf{S}$, each of the input variables is "selected" to be parents of output variables through a Hadamard product with the features $\boldsymbol{x}$. For each output variable, its parents are passed through a non-linear η fully connected neural-network. The η networks

are trained as general function approximators, learning to approximate the relationships between parent & child nodes:

$$\hat{\boldsymbol{x}}_i = \eta_i(\mathbf{S}_i \circ \boldsymbol{x}) \tag{5.2}$$

where $\mathbf{S}_i$ represents the i-th column vector of $\mathbf{A}$, and $\hat{\boldsymbol{x}}_i$ is the i-th reconstructed output [NZF19b]. In the case of exogenous variable $\boldsymbol{x}_i$, a corresponding 1 at $\mathbf{D}_{ii}$, 'leaks' the variable through, encouraging η to learn the identity function while a 0 value forces the network to learn some functional relationship of its parents. The end-to-end structure, as seen in Figure 5.1, is trained on a reconstruction loss, defined by $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}})$. We use the L2 loss (Mean Squared Error):

$$\ell_{\text{CSHTEST}} = ||\boldsymbol{x} - \eta_i(\mathbf{S}_i \circ \boldsymbol{x})||_2^2 \tag{5.3}$$

CSHTEST can be used, then, to operate as a structural hypothesis test mechanism for two structural causal models $\mathbf{S}$ and $\mathbf{T}$. The basic idea is that if $\ell_{\mathbf{S}} < \ell_{\mathbf{T}}$, across the majority of non-random OOD dataset splits for training and testing, then $\mathbf{S}$ is a more suitable hypothesis for the true causal structure of the data than $\mathbf{T}$. In section 5.4.3 we demonstrate the ID, OOD train/test splits to test this generalization capacity, and our experimental results provide baselines for this approach.

5.3.2 Causal Variational Hypothesis Testing with CSVHTest

We extend CSHTEST to a variational model CSVHTEST, that includes sampling functionality like a VAE [KW13]. We do this primarily for a more robust model in low Signal-to-Noise (SNR) regimes and to generate new data points that are not deterministic on the inputs, allowing for more dynamic synthetic data generation. CSVHTEST consists of an encoder, a CSHTEST causal layer and a decoder. Further details are provided in the appendix A.3.1.

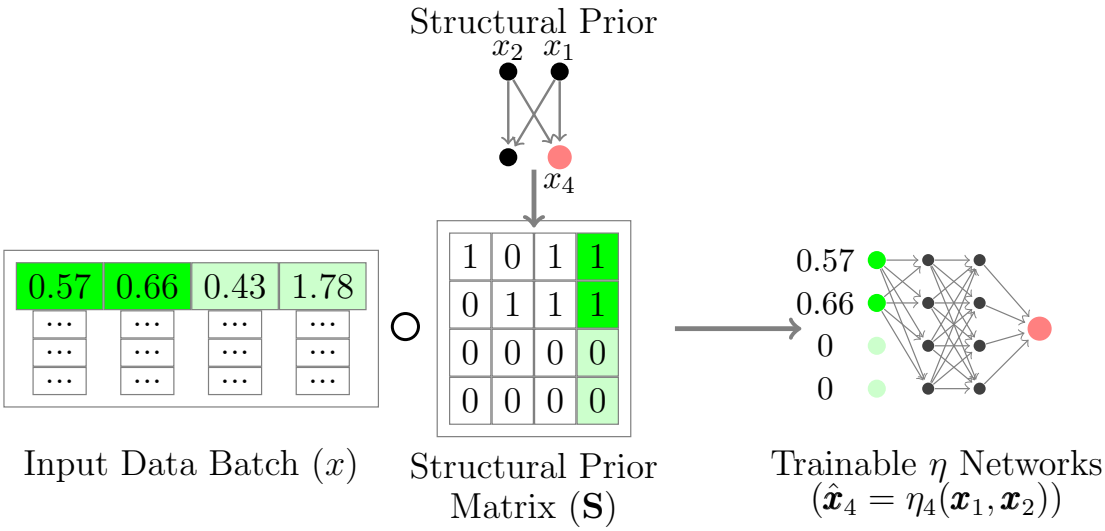


Figure 5.1: Causal Hypothesis Generative Architecture (CSHTest) with an example of how the Structural Prior Matrix selects for the parents of each variable or identity if it is exogenous. The η networks approximate the functional relationships in training.

5.4 Problem Setting

5.4.1 Structural Hamming Distance

In causal and graph discovery literature, the Structural Hamming Distance is a common metric to differentiate causal models by the number of edge modifications (flips in a binary matrix) to transform one graph to another [GKC17, KCW22], often described as the norm of the difference between adjacency matrices:

$$\mathbf{H} = \|\mathbf{A}^i - \mathbf{A}^j\|_1 \tag{5.4}$$

However, Structural Hamming Distance does not account for the "causal asymmetry." The absence of edges is a more profound statement than inclusion, as any edge could have a weight of zero. Hence, we define two types of hypotheses that are incorrect relative to ground truth, which could have the same Structural Hamming Distances:

- *Leaky* hypotheses are causal hypotheses with extra links. In general, having a leaky

hypothesis will produce models that are more prone to overfitting, but with proper weighting, the solution space of a leaky causal hypothesis includes the ground truth causal structure.

- *Lossy* hypotheses are causal hypotheses where we are missing at least one link. Lossy hypotheses are much easier to detect because a lossy hypothesis results in lost information. As such, a lossy hypothesis should never do better than the true hypothesis, within finite sampling and noise errors.

From these definitions, we define the *Positive Structural Hamming Distance* and the *Negative Structural Hamming Distance*. We define these as, for null hypothesis $\mathbf{A}^0$ and alternative $\mathbf{A}^1$,

$$\mathbf{H}^+(\mathbf{A}^1, \mathbf{A}^0) = \|\mathbf{A}^1 > \mathbf{A}^0\|_1 \qquad \mathbf{H}^-(\mathbf{A}^1, \mathbf{A}^0) = \|\mathbf{A}^1 < \mathbf{A}^0\|_1 \tag{5.5}$$

where $\mathbf{H}^+$ counts how *leaky* the alternative hypothesis is and $\mathbf{H}^-$ counts how *lossy* it is. One remark is that $\mathbf{H} = \mathbf{H}^+ + \mathbf{H}^-$, but the "net" Hamming Distance $\Delta\mathbf{H} = \mathbf{H}^+ - \mathbf{H}^-$ can also be a naïve indicator of how much information is passed through the causal layer.

5.4.2 Baseline Models

5.4.2.1 Simulated DAG Baselines

We empirically test our theory that an incorrect hypothesis will result in worse OOD test error using extensive simulations. We use the same methodology as [ZAR18], simulating across multiple DAG nodes sizes, edge counts, OOD variable splits (described further in 5.4.3), and Structural Hamming Distance with iterations at the ground truth and modified DAG levels for robustness. In our experimental results, we calculate the probability an $\mathbf{H}$ of 1 closer to ground truth would have a lower OOD test error as the ratio across our simulations:

$$\mathbb{P}\left[\ell_{\text{CSHT}_{\text{EST}}}(S_j) < \ell_{\text{CSHT}_{\text{EST}}}(S_i)\right]\Big|_1 = \|\mathbf{A}^i - \mathbf{A}_{GT}\|_1 - \|\mathbf{A}^j - \mathbf{A}_{GT}\|_1 \tag{5.6}$$

where GT is ground truth, and so on for differences 2 and 3. In practice, we actually consider the probability conditional on a tuple of the positive and negative Hamming distances $(\mathbf{H}^+, \mathbf{H}^-)$ thus allowing us to distinguish hypotheses that are *leakier*, *lossier*, or the specific mix of the two. Doing so allows us to better consider the fundamental asymmetry in causality. Full hyperparameters and test cases can be found in Appendix A.5.

5.4.2.2 Sun Pendulum Image Dataset

A synthetic pendulum image dataset is introduced in [YLC20], and we use it here to produce a physics-based tabular dataset where we know the ground truth DAG and can test the abilities of CSHTest and CSVHTest. More about the dataset is described in Chapter 4.4.1. We take these values $\mathbf{u} = [\theta, x_{sun}, w_{shadow}, x_{shadow}]^T \in \mathbb{R}^d$, where $d = 4$ and compile a tabular dataset. This methodology provides a physics-based dataset where the causal, ground truth causal model is known to show the abilities of CSHTest and CSVHTest.

5.4.2.3 Medical Trauma Dataset

We also analyze our model on a real-world dataset of brain-trauma ground-level fall patients that includes multiple health factors, with a focus on predicting a decision to proceed with surgery or not. We use an initial SHAP analysis to select three variables of high prediction impact: Glasgow Coma Scale/Score for head trauma severity (GCS), Diastolic Blood Pressure (DBP), the presence of any Co-Morbidities (Co-Morb), one demographic variable Age, along with the Surgery outcome of interest. Without the ground truth, we test two structural models shown in 5.2 based on knowledge of the selected variables and how they may interact to inform the surgery decision.

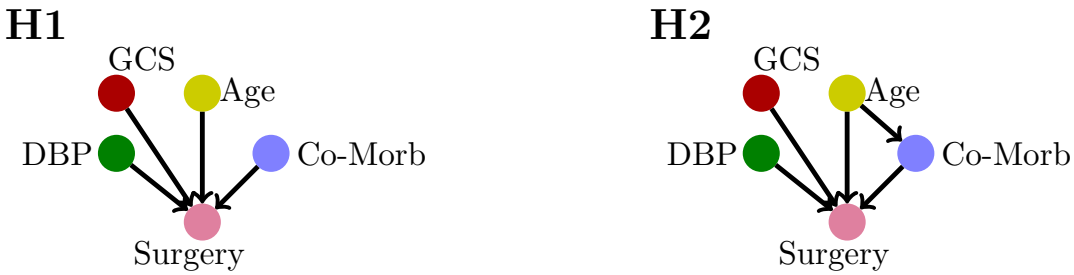


Figure 5.2: Two hypothesized structural causal priors for a medical dataset on trauma patients and the decision to perform surgery, H1 and H2.

5.4.3 Train/Test Data Splits

In order to test generalization error, we use a deliberate non-random split of our datasets (as well as a baseline random split). This is done on a single feature column of the tabular data at a time, splitting the data on that column at either the 25% or 75% quantile, with the larger side (either the upper or lower 75%) becoming the training data. An example of this train test split is visualized for both datasets in 5.3. We recommend viewing OOD test error across as many dimensions and split quantiles as possible given the size of the dataset and the available compute.

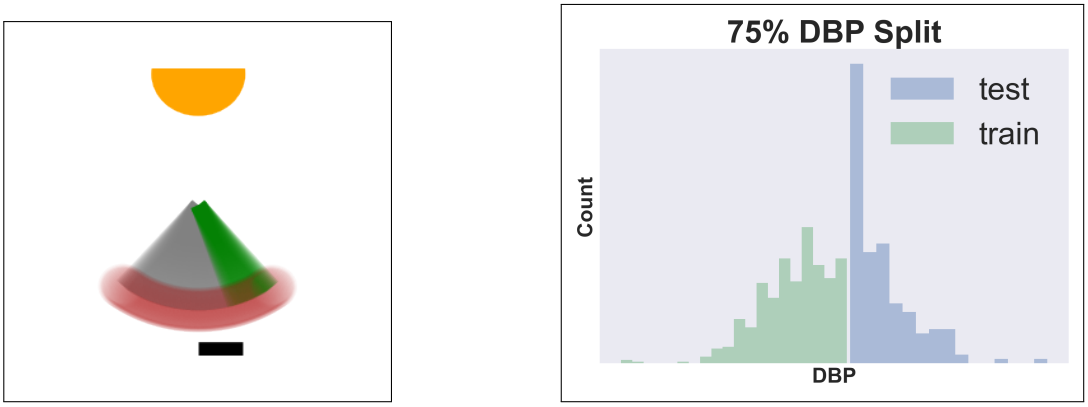


Figure 5.3: a) A 75% data-split on the pendulum angle feature (gray is training angle, green is testing angles b) A 75% data-split on the Diastolic Blood Pressure data.

5.5 Experiments

We test CSHTest and CSVHTest in multiple settings. First, we justify their usage by comparing their performance on both ID and OOD validation to their non-causal counterparts, showing that they operate as normal when trained in ID but perform much better when trained in OOD. We provide a table of relative loss probabilities to help interpret results using extensive simulations. Next, we observe the benefits and limitations of the CSHTest method when we hypothesize several possible causal structures on the pendulum problem. Finally, we hypothesize and compare to structural priors on the medical dataset, and simulate new data.

5.5.1 Generalization Ability of CSHTest

	Pendulum Comparison			
	Random		Split	
Method	Train	Test	Train	Test
NN	0.02	0.02	0.04	10.27
VAE	0.11	0.06	16.97	89.4
CSHTest	0.03	0.03	0.02	0.26
CSVHTest	0.064	0.51	19.81	38.62

Table 5.1: Comparison of Traditional Deep Learning Techniques on a random and deliberate dataset split with CSHTest and CSVHTest when the ground truth causal structural information is known.

We compare the CSHTest with a similarly sized fully-connected NN and CSVHTest with a similarly sized VAE. The CSVHTest also has the same causal layers as the CSHTest so the variational models are larger overall than the CSHTest and NN. Results of the pendulum are shown in Table 5.1. Against ID (random) data, the CSHTest and

CSVHTest effectively perform the same, suggesting that there is no loss in representation by including the Causal Layer.

However, the CSHTest and CSVHTest models generalize much better to OOD data validation than their respective non-causal comparisons. This demonstrates the use of the CSHTest and CSVHTest as causal replacements to the NN and VAE. Conversely, as the out-of-distribution error rate can determine the "closeness" of our model to the true causal model, this would enable the use of the OOD loss as a proxy for causal hypothesis testing.

5.5.2 Simulated DAG Hypothesis Testing

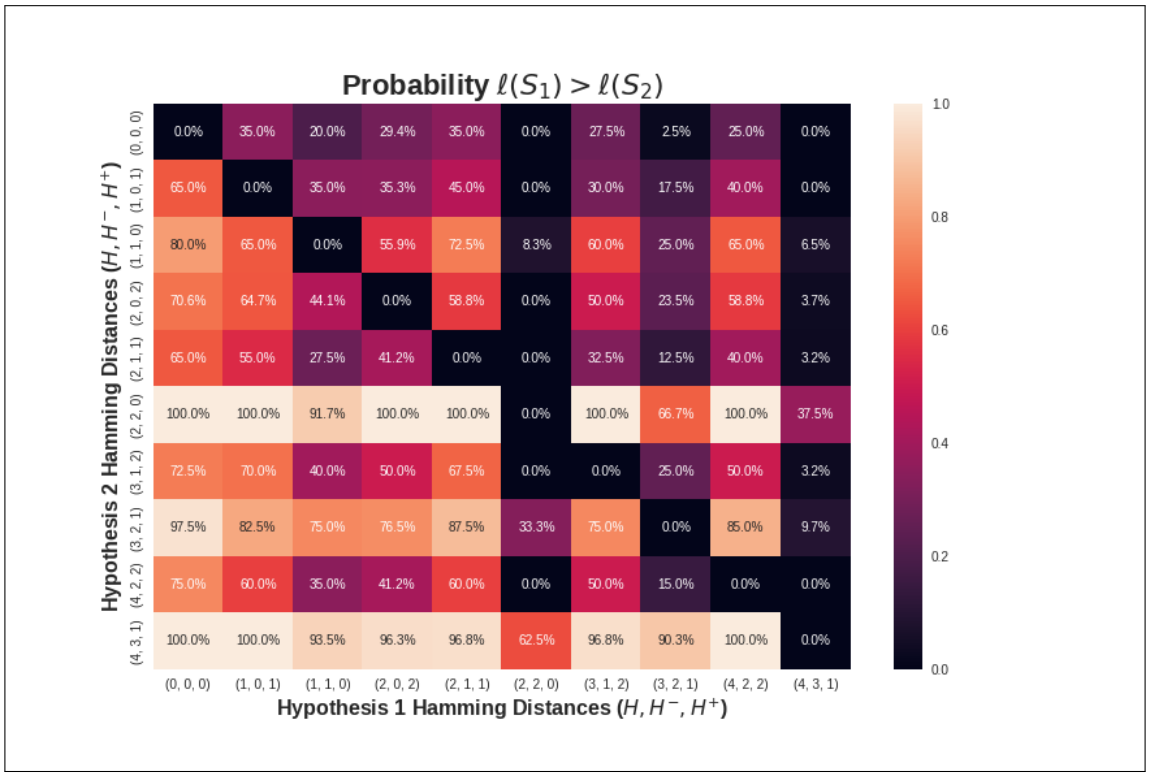


Figure 5.4: Probability table for a 4-node 4-edge DAG size with a linear SEM ground truth model for DAG simulations comparing hypothesis with various Hamming Distance Tuples.

Figure 5.4 shows the type of empirical probability tables we can construct by simulating DAGs of various sizes under numerous conditions detailed in the appendix A.5. We note how

by comparing the Hamming Distance tuples, we do not see a smooth gradient, but jumps as the *leaky*/*lossy* asymmetry is realized. Instead, by also incorporating the net Hamming distance to account for the causal asymmetry, we can explain the jumps. For instance, (2,2,0) shows a marked drop off compared to any lower Hamming Distance model — like ground truth (0,0,0) — because it has two edge losses ($\Delta\mathbf{H} = -2$) which vastly decreases needed information flow. In general, the upper triangle of this matrix should be below 50% and decrease as the Hamming Distances grow and get *lossier*. Within each Hamming distance, the values typically increase as $\Delta\mathbf{H}$ increases. Extensive simulations like this, done with similar assumptions to a comparable real-world problem, can provide baseline probabilities even if ground truth is not known, based on the relative Hamming Distances of the hypotheses. Further results with a DAG of size 5x5 can be found in the appendix A.1.

5.5.3 Pendulum Hypotheses

We introduce 6 enumerated hypotheses for the pendulum dataset, enumerated in Figure 5.5. We have two hypotheses that are 1 Structural Hamming Distance away from ground truth (*leaky* and *lossy*), and 3 that are 2 Structural Hamming Distances away (*2leaky*, *2lossy*, and *leak-loss*). The choice of which edge to add or remove were arbitrary, unless required by design. The individual names of the hypotheses give away their purpose, as they are meant to be leaky or lossy in a specific way to observe the empirical qualities of the different hypothesis tests. Of note, despite including an additional leakage in *2leaky*, we maintain the exogenousity of x_{sun}, meaning that the θ to x_{sun} is purely a leakage term from the SCMs perspective. Secondly, the *leak-loss* hypothesis has a net Hamming distance of 0 and structurally still has the same connectivity as ground truth. However, due to the choice of paths, it likely has some functional limitations.

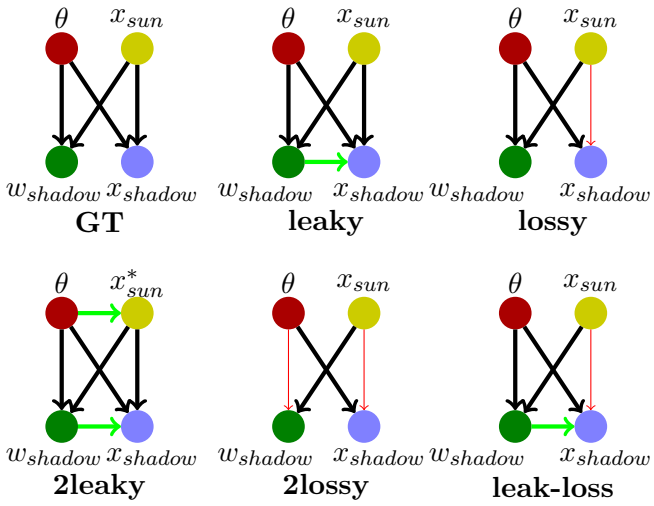


Figure 5.5: The 6 enumerated pendulum hypotheses that we try out. Red and thin arrows are arrows that we remove from the true DAG and green arrows are arrows that we add to the true DAG. In the case of 2leaky, we maintain x_{sun} as an exogenous variable, but allow θ information to also influence its value.

5.5.4 Pendulum Hypothesis Testing

We consider these 6 different hypotheses, shown in Figure 5.5. We arbitrarily do a 75% OOD split of the pendulum dataset on the sun position (as an exogenous example) and the shadow position (as an endogenous example) to test causal hypotheses.

The pendulum results are shown in Figure 5.6. We can clearly distinguish two tiers of results. One tier contains *GT*, *leaky*, and *2leaky*. This tier has a common $\mathbf{H}^- = 0$. All other DAGs, having $\mathbf{H}^- > 0$ show a very clear drop in OOD test performance. Thus, for hypothesis testing, we are able to distinguish causal hypotheses that are missing paths from ground truth. We leave it to further research to explore how to compare many hypotheses that achieve a similar loss, such as a criterion that favors the minimal hypothesized DAG.

Of interest is the *leak-loss* model, which has $\Delta\mathbf{H} = 0$. Its loss is generally lower than the purely lossy hypotheses, but still achieves a higher loss than the ground truth, despite

graphically being the same level of connectedness. This result has the interesting consequence of CSHTest being able to reject causal hypotheses with zero net Hamming distance.

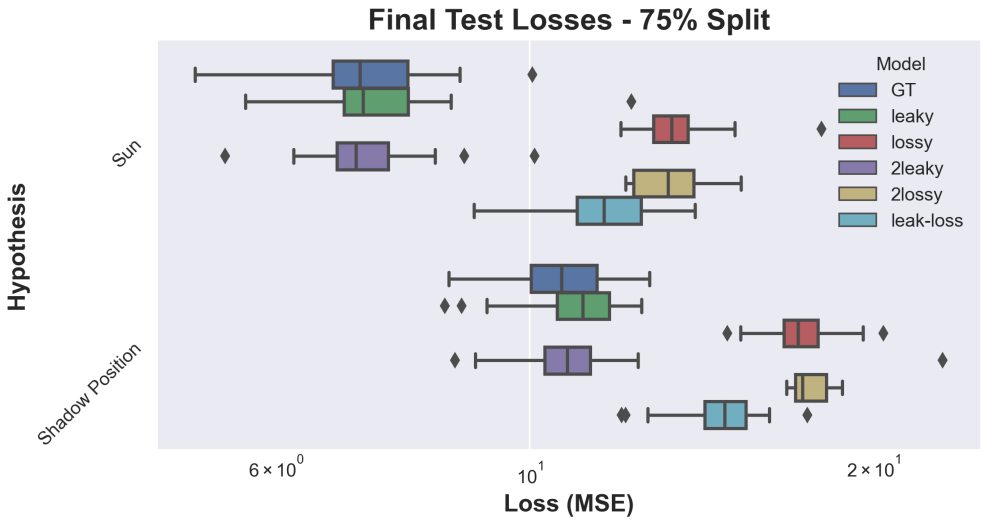


Figure 5.6: Final OOD Test Error Rates of Each Hypothesized DAG structure in the Pendulum Problem over Two Splits. See Appendix A.2.2 for numerical values and training trajectories.

5.5.5 Medical Data Hypothesis Testing

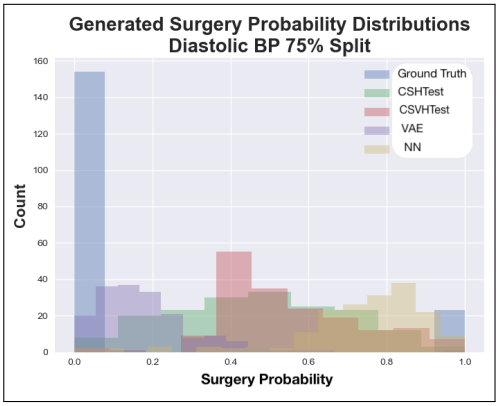
In the medical dataset, the second hypothesis from 5.2, which includes a path from Age to 'No Co-morbidities' generalizes better than without the path, suggesting it is a better causal model. We use both trained architectures to simulate OOD data, with the causal models producing higher fidelity results to what we expect ground truth to be based on a holdout test set over the same distribution 5.7.

5.6 Conclusion

In this chapter, we demonstrate the value of CSHTest and CSVHTest as causal model hypothesis testing spaces and the implications as generative models. We verify the effective-

Medical Results		
Hypothesis	Train	Test
H1	0.024	0.035
H2	0.017	0.025

(a) Medical results of CSHTest for each of the hypothesized causal models.



(b) The test dataset reconstruction distributions for all models using medical H1 on the Diastolic BP 75% data

Figure 5.7: Medical Dataset Results

ness of our methodology on extensive simulated DAGs where ground truth is known, and we further show performance with ground truth and incorrect causal priors on a physics-inspired example. We show how CSHTest can be used to test causal hypotheses using a real-world medical dataset with ground level fall, trauma surgery decisions. CSHTest offers a novel architecture, along with a deliberate data split methodology that can empower practitioners and domain experts to improve causally informed modeling and deep learning. There is extensive further research needed to fully realize the utility of structural causal hypothesis testing in conjugation with deep learning function approximation. We hope to better differentiate leaky causal models, without constraints on losses, using minimum entropy properties such as in [CGK22]. We also hope to extend both CSHTest and CSVHTest to more flexible architecture which can combine recent progress with differential causal inference and binary sampling to better automate full or partial causal discovery. The results are a promising start of much further research integrating deep learning causal models with real-world priors and domain knowledge.

CHAPTER 6

Towards Composable Distributions of Latent Space Augmentations

6.1 Introduction

Data augmentation has become an essential technique in deep learning, allowing models to learn from a diverse set of input images by applying various types of transformations, such as rotation, flipping, cropping, and color shifting. By artificially increasing the size of the training dataset, data augmentation can reduce the risk of overfitting and improve the generalization of the model to new, unseen data.

However, choosing the right set of augmentation techniques for a given task can be challenging. Some types of augmentations may affect the way an image is interpreted, such as flipping or rotating digits in handwritten digit recognition tasks. Practitioners must employ priors on the data to know which augmentations are appropriate.

We observe augmentations as a causal prior on the image space and apply the techniques introduced in Chapters 4 and 5. In this chapter, we introduce a novel approach for latent image augmentation using Variational Autoencoder (VAE) architecture. Our approach allows for the easy combination of multiple augmentation techniques and provides greater control and interpretability of the latent space. Within the latent space of the VAE architecture, we can apply, compose, and invert linear transformations to generate augmented versions of the input images. The key contribution of our work is the use of latent-space linear transforms and a two-step training method to learn mappings between the original and augmented la-

tent spaces, with a surprising emergence of composability. We further demonstrate that our approach can transfer a trained latent space to a new set of augmentations using a multiple decoder architecture, enabling practitioners to transfer certain properties and potential performance improvements dependent on the original augmentations.

Our experiments on the MNIST dataset demonstrate that our proposed approach can improve the performance of VAEs and provide new insights into the underlying structure of the data and the relationship between different augmentations. By viewing augmentations as image-space priors and not data to simply be randomized across, we can constrain the VAE's information bottleneck and improve its generalization ability. In essence, our method learns a low-dimensional, latent-proxy 6.1 for a set of image-space functions, even when the image space model or transformation process is unknown a priori, as long as training samples exist of the augmented images.

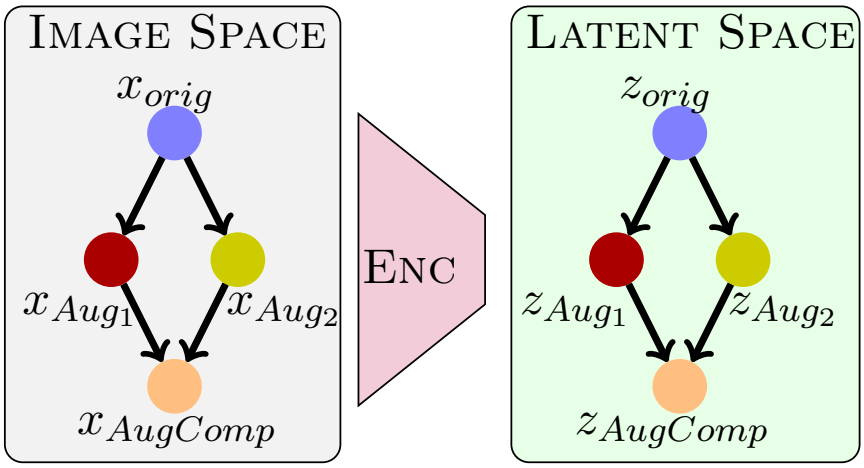


Figure 6.1: Approximate model/DAG *learned* in latent space for *known* image space augmentations

6.2 Related Work

Control and manipulation of a lower-dimensional latent space in generative modeling is an area of ongoing research. The Conditional VAE (CVAE) is an initial extension of a vanilla VAE in which a conditional value or "one-hot" encoding is concatenated to both the Encoder and Decoder inputs [SLY15]. The CVAE does a form of latent space separation by adding dimensions based on a conditional variable, and it presents the most compelling comparison to our own work as it allows unique interpretation of the latent space based on a prior or semantic label. Other extensions of VAEs include the VQ-VAE, which uses a discrete latent space to model discrete data types such as text, and the Flow-based VAE, which uses normalizing flows to model complex posterior distributions [OVK17, SW18]. Latent diffusion models are another approach that iteratively add noise in the training process and can reverse this process in inference to achieve state-of-the-art text to image and image completion and synthesis tasks [RBL22].

The latent space can also be used to apply lower-dimensional modeling or priors. Causal generative models have seen a variety of success with both learning and utilizing causal information and structural models to generate counterfactual images and datasets [YLC20, KSD17, BJP22, JPB22, BKB21]. Our method looks to also extend interpretability of the latent space by approximating image augmentations, a priors-based prep-processing approach in the image-space, in the latent space. This could loosely be thought of as a causal model proxy to the image space causal model, Figure 6.1.

6.3 Background

6.3.1 Variational Autoencoders

The VAE framework is based on the principle of maximum likelihood estimation, where the goal is to maximize the likelihood of the training data under the model. However, in order

to make the optimization tractable, the VAE introduces a variational lower bound on the log likelihood, which can be written as:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] \tag{6.1}$$
$$- \mathrm{KL}\left(e_\phi(z|x^{(i)})||m_\theta(z)\right)$$
$$= \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] \tag{6.2}$$
$$- \int e_\phi(z|x^{(i)}) \log \frac{e_\phi(z|x^{(i)})}{m_\theta(z)} dz$$

where $x^{(i)}$ is a single training example, and θ and ϕ are the parameters of the decoder and encoder, respectively. The first term in the lower bound, $\mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right]$, is known as the reconstruction loss, and it measures the difference between the reconstructed data and the original data. The second term, $\mathrm{KL}\left(e_\phi(z|x^{(i)})||m_\theta(z)\right)$, is known as the KL divergence, and it measures the difference between the approximate posterior distribution and the latent distribution. The first term is the decoding error (the classic rate-distortion theory), and the second term is the extra rate for coding z assuming marginal pdf $m_\theta(z)$.

The Conditional VAE (CVAE) is a natural extension of the VAE framework that adds a conditional input to both the encoder and decoder networks. In the CVAE, the goal is to learn a conditional generative model that can generate new samples from a specific class or condition, given some additional information (additional details in Chapter 2.4.2). By adding the conditional input to the VAE framework, the CVAE can generate samples conditioned on a specific input, which is useful in many applications. For example, in image generation, given a class label as the conditional input, the CVAE can generate images of that class.

6.3.2 Priors in Pre-processing

Data augmentation is not done naively, or without a strong sense of priors. In image datasets, typical augmentations might include crops, rotations, flips, scaling, color modifications, masks, and many more. In order to expand the training domain and learn a more robust model, only augmentations which are invariant to the classification or interpretation

of the resulting image can be applied, and similarly, negative augmentations which impact classification can be used to refine the support of a distribution [SAS21]. As a simple example, a left-right flip is an acceptable transformation for a 0 or 8 digit, but not for a 2 or 9. One can think of augmentation as a causal model or directed-graph in the image space, in which all augmented image distributions are the result of applying a transform or function to a parent node of original images such that the resulting images are still within the same class. Succinctly, given the original dataset D_{orig},

$$D_{aug} = \bigcup_{\substack{i \in \mathcal{A}_c \\ d \in D_{orig} \\ c = class(d)}} f_{aug_i}(d) \tag{6.3}$$

$$s.t. \quad \mathcal{A}_c = \{i : class(d) = class(f_{aug_i}(d)); d \in D_{orig}\} \tag{6.4}$$

for some pre-defined set of augmentations $\{f_{aug_i}\}$.

This model is typically well known and easily applied in pre-processing, but is not made explicit. Although these augmented distributions are classified or interpreted similarly at a high-level, there are functional relationships and structure between them that we look to make explicit.

6.4 Latent Augmentation VAE

6.4.1 Architectural Overview

In the Latent Augmentation VAE, as seen in Figure 6.2, we use trainable linear transforms in the latent space to learn the mappings between original and augmented latent representations resulting in a linear proxy model of the transformations applied in the image space that can be used on test data or to generate new original and augmented images. We also utilize multiple decoder heads such that one can transfer a learned latent space to a new set of augmentations by training an alternative decoder head, which can preserve certain latent space geometries and latent transform properties, improving latent augmentation performance.

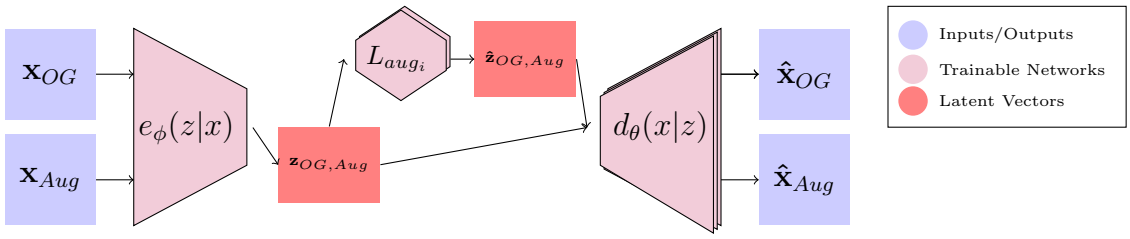


Figure 6.2: Latent Augmentation VAE Architecture

6.4.2 Training LAVAE

For our initial experiments, we focus on pairs of augmentations and their compositions (as in Figure 6.1). Note there is no theoretic reason why more augmentations can not be used, but we use two for illustrating geometric properties. We train the LAVAE in three stages:

1. Train the encoder/decoder and populate the latent space with the original, two types of augmentations, and their composition
2. Learn explicit linear transformations L_{aug_i} between original and augmented latent spaces
3. Transfer trained latent space and transformations by training new decoder on any other set of augmentations

The respective losses for each of the three stages are as follows:

$$\sum_{x \in D_{aug}} \ell_{BCE}(x, \hat{x}) \quad \text{s.t.} \quad \hat{x} = \theta(\phi(x)) \tag{6.5}$$

$$\sum_{x_0 \in D_{orig}} \sum_{k \in \mathcal{A}} (\phi(f_{aug_k}(x_0)) - \phi(x_0) \cdot L_{aug_k})^2 \tag{6.6}$$

$$\sum_{x_0 \in D_{orig}} \sum_{\substack{g(k) \in \mathcal{A}_T \\ k \in \mathcal{A}}} \ell_{BCE}(f_{aug_{g(k)}}(x_0), \theta_T(\phi(x_0) \cdot L_{aug_k})) \tag{6.7}$$

where the trained parameters at each step are highlighted in blue. $\mathcal{A}, \mathcal{A}_\mathcal{T}$ are two equal-length sets of augmentations predefined by $\{f_{aug}\}$ with $g : A \to A_T$ forming a bijective

pairing between the two sets, thereby allowing the latent structure to be preserved by the transformations defined by $\mathcal{A}$. θ_T is an alternative decoder head for each new set of augmentations $\mathcal{A}_T$. In our 2-augmentation case, without loss of generality, we can assign $\mathcal{A} = \{1, 2\}$, $\mathcal{A}_T = \{3, 4\}$ where $g(1) = 3$ and $g(2) = 4$. We can also extend this formulation to any other sets of augmentations given a new mapping. Note that stage 1 also includes the KL Divergence loss to constrain the latent distribution, as described in appendix 2.4.1, with respective weights on KL and reconstruction $\lambda_{KL} = 5$, $\lambda_{recon} = 1$.

We perform experiments with non-linear latent augmentation networks, which show slightly better performance, but lack composability and simple invertibility. We also experiment with combining training stages 1 and 2, but this degraded final performance and reconstruction.

6.4.3 Using LAVAE

Once an LAVAE is trained, there are a wide variety of uses which extend the capabilities over previous VAE methods. There is basic reconstruction of the original, augmented, or composed images:

$$\hat{x} = \theta(\phi(x)) \ \forall x \in D_{aug}$$

We can augment the original images in the latent space:

$$\hat{z}_i = \phi(x_0) \cdot L_{aug_i}$$
$$\hat{x}_i = \theta(\hat{z}_i) \quad | \quad i \in \mathcal{A}$$

where x_i refers to $f_{aug_i}(x_0)$.

We can go from the original images to the composed by multiplying the latent original vector by both the latent augmentation transforms, despite an explicit composition in the

latent space never being trained.

$$\hat{\mathring{z}} = \phi(x_0) \cdot L_{aug_1} \cdot L_{aug_2}$$
$$\hat{\mathring{x}} = \theta(\hat{\mathring{z}})$$

The reverse composition is also effective with some increased reconstruction error, indicating the latent augmentations are somewhat composable.

$$\hat{\mathring{z}}_r = \phi(x_0) \cdot L_{aug_2} \cdot L_{aug_1}$$
$$\hat{\mathring{x}} \approx \theta(\hat{\mathring{z}}_r)$$
$$\hat{\mathring{z}}_r = zL_{aug_2}L_{aug_1} \approx zL_{aug_1}L_{aug_2} = \hat{\mathring{z}}$$

where $z = \phi(x_0)$. Note that this latent space property holds true for our tested augmentations even if the compositions in the image space are not equivalent (such as a 90° rotate and flip will be different depending on the order applied). In this case, we only train the encoder and decoder, in Stage 1, on one of the compositions (f_{aug_1} then f_{aug_2}), so the reverse composition in the latent space is not equivalent to the image space reverse composition with this process.

We can also invert the latent space transforms and go from an augmented input image to an original image, giving us 'any-to-any' functionality:

$$\hat{x}_0 = \theta\left(\phi(x_i) \cdot L_{aug_i}^{-1}\right) \quad | \quad i \in \{1, 2\}$$
$$\hat{x}_0 = \theta\left(\phi(\mathring{x}) \cdot (L_{aug_1} \cdot L_{aug_2})^{-1}\right)$$

Finally, we can run the model recursively, taking our output and running back through the network for the same or different augmentations. We find that even for general augmentations that there is some level of stability in taking the same latent augmentation over and over.

$$\hat{x}_0^{(k)} = \theta(\phi(\hat{x}_i^{(k)}) \cdot L_{aug_i})$$
$$\hat{x}_i^{(k+1)} = \theta(\phi(\hat{x}_0^{(k)}) \cdot L_{aug_i})$$
$$k \in [1, n]$$

We show results on the stability of this use-case as a recursive generator in the next section.

6.5 Experiments

All the displayed results use the test MNIST dataset with a model trained on the training dataset.

6.5.1 LAVAE Reconstruction Results

Figure 6.3 shows the basic and augmented reconstruction results for the "Flips" (flip left / right, flip up / down) augmentation pair.

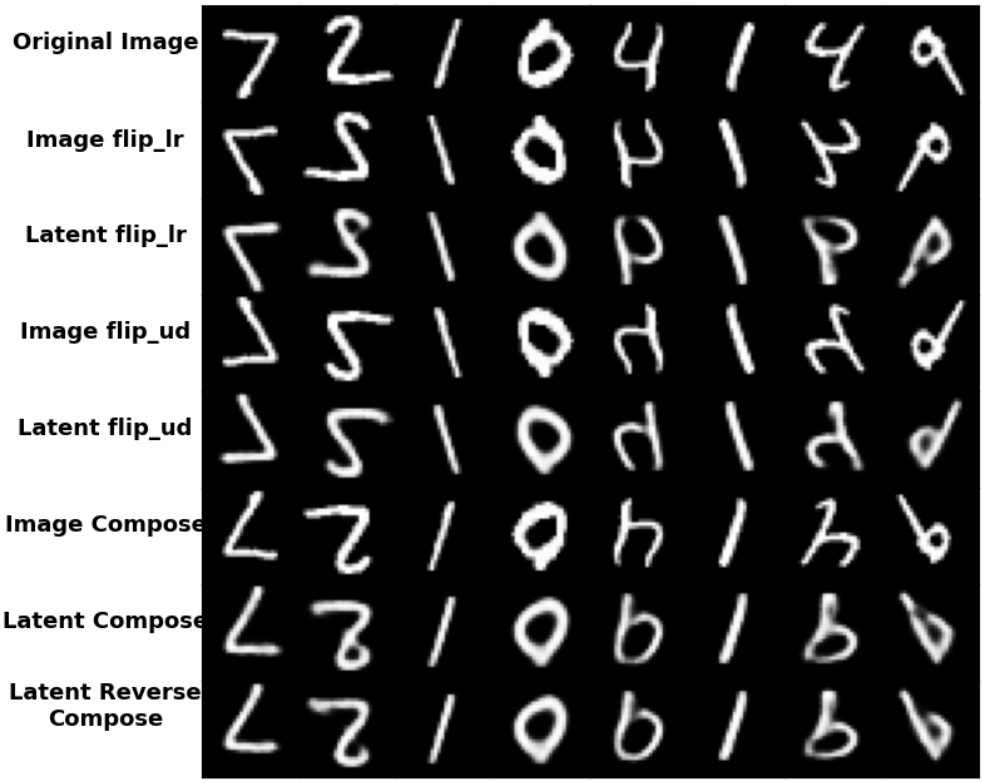


Figure 6.3: Eight samples of "Flips" latent augmentations with baseline image space augmentations for comparison

Figure 6.4 shows the inverse reconstruction results for the Flips augmentation pair.

Figure 6.5 displays two examples of recursive augmentation using flip left/right, where

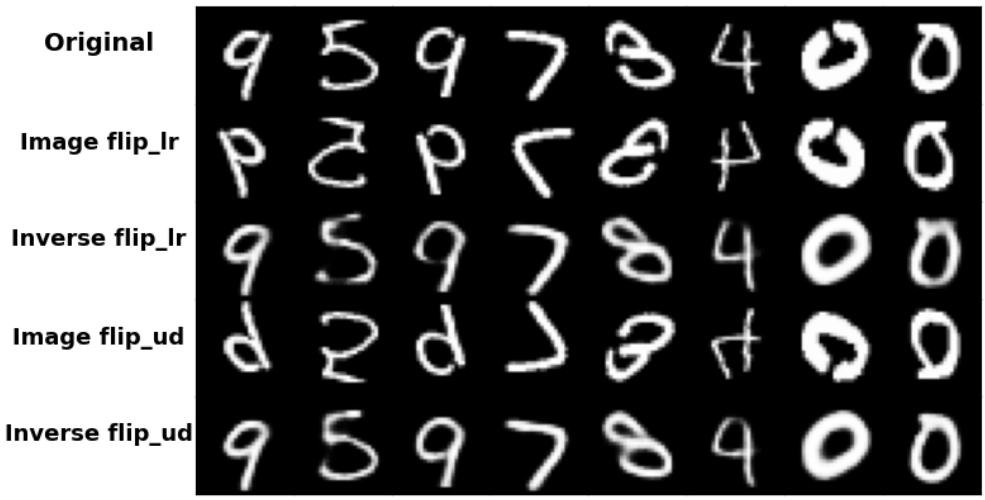


Figure 6.4: Eight samples of "Flips" latent inverse augmentations with original and augmented images (inputs) for comparison

one sample gradually deviates (from a 2 to possibly an 8), while the 7 remains relatively stable. Additionally, we illustrate a lower dimensional projection in a 2D space (using Independent Component Analysis) of the latent vectors and their corresponding "paths" as we repeatedly apply augmentations using LAVAE. This suggests a radius of stability around certain samples with the repeated use of augmentations.

It is worth mentioning that the LAVAE can also be applied to sample the latent space and interpolate between points. To achieve sampling, as the latent space is now divided based on the augmentation, we constructed a simple bounding box using training samples and sampled within that subspace to obtain an original image. Interpolation is simpler, as we only need to provide two test images and sample at regular intervals across all latent dimensions between the two points (16-D in this instance). Additional examples can be found in Appendix B.2.

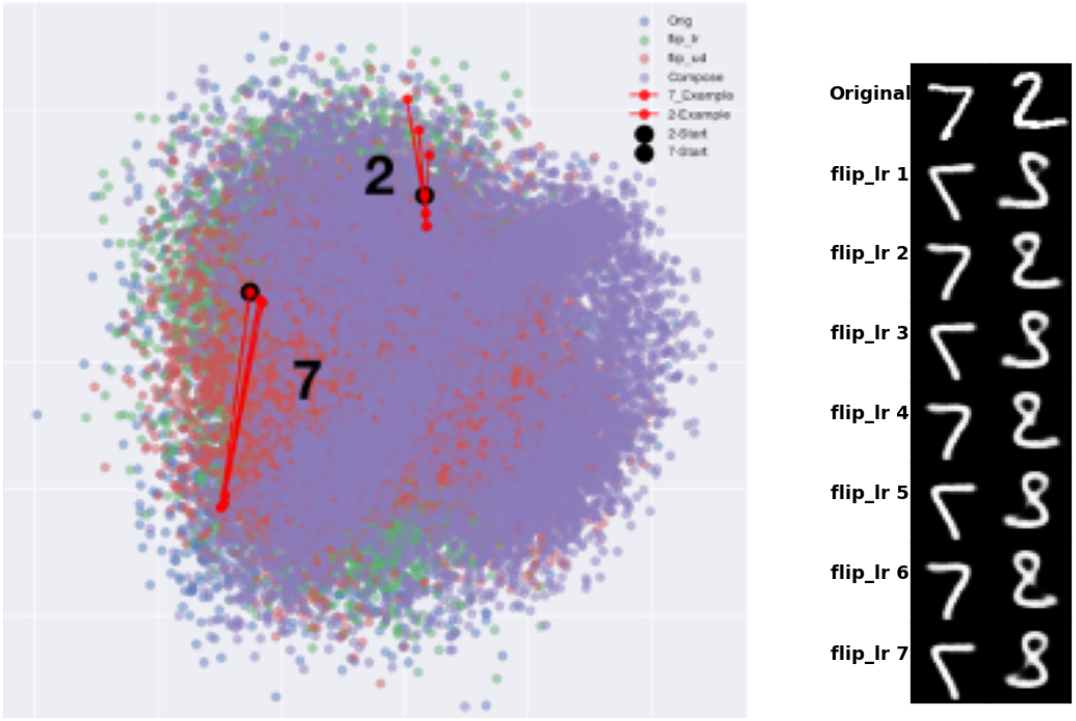


Figure 6.5: Two samples latent trajectory (2-D projection) and reconstructions of recursive flip left/right augmentation. The '7' is stable, but the '2' diverges both in latent space trajectory and reconstructions.

6.5.2 LAVAE Transfer Decoders

LAVAE includes multiple decoder heads to enable the transfer of a trained latent space to any pair of augmentations. Figure 6.6 shows the transfer reconstruction results from "Flips" to "Nested Mini-Image, shear X-direction."

This functionality was included because we saw that transferring to a pair of augmentations could increase the reconstruction performance over training on the augmentation pair originally. This surprising result leads us to believe certain latent space geometries, based on the choice of initial augmentations, better allow for latent augmentation reconstruction and properties such as improved composability. Figure 6.7 shows a heat-map matrix of reconstruction error (Mean-Squared Error in image space) with initial augmentation pair

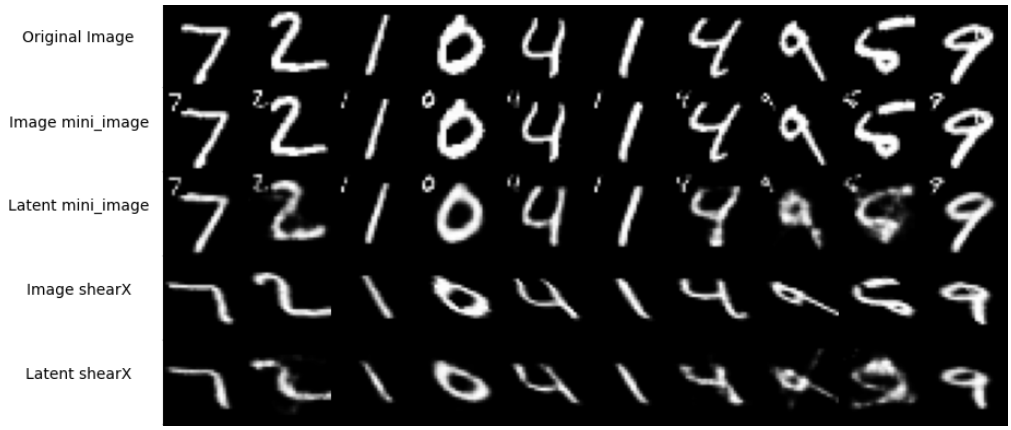


Figure 6.6: Eight samples of "Flips" latent augmentations with baseline image space augmentations for comparison

choice vs. transferred augmentation pair in which transferring from a "Nested Mini-Image, shear X-direction" to "Nested Mini-Image, shear X-direction" performs better than training on "Nested Mini-Image, shear X-direction." Examples of all the augmentation and more results are in appendix B.1.

6.5.3 Conditional VAE Comparisons

As discussed, we realize that the Conditional VAE (CVAE) also uses a naïve form of latent space partitioning, so we wanted to see to what extent it can do the same tasks as the LAVAE. In this case, instead of having the conditional represent the classifications, we want to partition just like with the LAVAE with respect to augmentation.

Thus, we first train the CVAE in the traditional way where the conditional is the augmentation type. Example results are shown in 6.8. The decoder can reconstruct from the latent with high fidelity, but changing the conditional does not augment the image as expected. Instead, it produces a plausible image of that augmentation, but it does not preserve the uniqueness of the original image. Therefore, the conditional variables and the latent variables are "entangled." This suggests that the CVAE cannot naively handle causally-linked

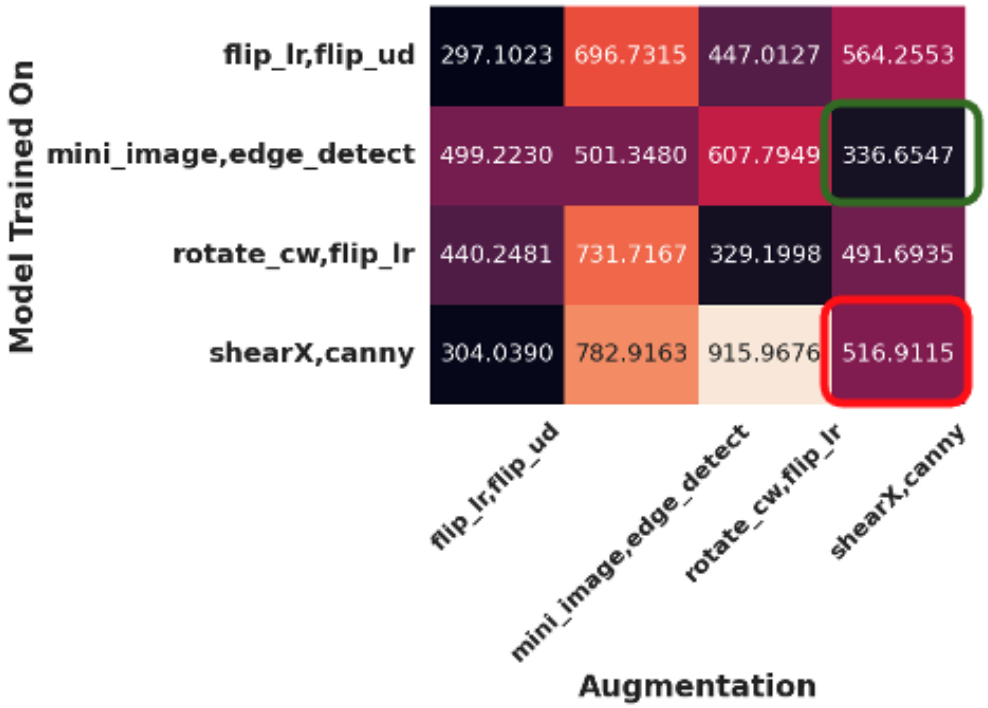


Figure 6.7: Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations)

images across conditionals.

Fundamentally, a causal view of the CVAE would represent the following idea: any $x_i = f_{aug_i}(x_0)$ causally generated from some x_0 should map to the same z. Thus, z should contain the augmentation-invariant information. Then, based on the conditional, the decoder should produce an augmented version of that image. Thus, the conditional contains all the augmentation information, and we say that the augmentation and the image are disentangled. Our second experiment attempts to show that the CVAE encoder and decoder are capable of applying augmentations to an augmentation-invariant latent space, given a similar training method to the LAVAE. However, in doing so, we can see that there is a significant hit in reconstruction loss compared to the LAVAE as can be seen in Figure 6.9. Furthermore, the composition property does not emerge in the same way that it does for the LAVAE.

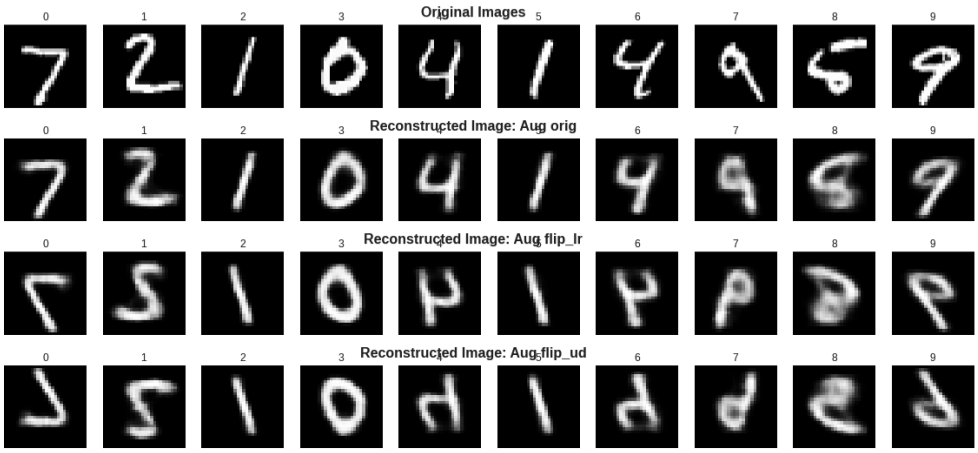


Figure 6.8: Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations)

Table 6.1: "Flips" Reconstruction Errors (MSE)

Model	Orig	Aug_1	Aug_2	$\circ$	**Total**
LAVAE	**68.34**	**75.89**	**71.30**	**81.57**	**297.1**
CVAE_{Trad}	98.11	260.18	351.79	279.82	989.84
CVAE_{AugInv}	99.15	99.16	99.26	299.12	695.68

In Table 6.1, we compare the reconstruction errors of the LAVAE against both methods of training the CVAE on the "Flips" augmentation pair, showing the superior performance of LAVAE.

6.5.4 LAVAE Latent Geometries

In this final section, we present a comparison of 2D projection visualizations using PCA, ICA, and T-SNE algorithms of the latent space geometries for two different pairs of augmentations, "Flips" and "shear X-direction," "canny edge-detect," as shown in Figure 6.10. We leave a more in-depth analysis and interpretation of the latent space geometries for future research. However, we would like to point out that while symmetries seem to exist across augmentation

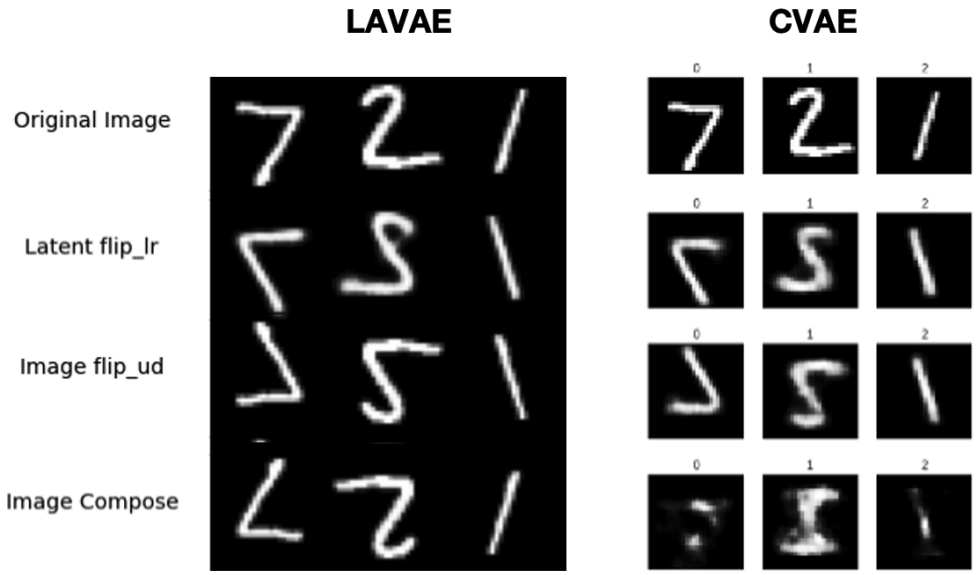


Figure 6.9: LAVAE vs. CVAE reconstructions

pairs, the separation between regions and the relative areas of regions vary significantly across pairs. Additional images can be found in Appendix B.3.

6.6 Conclusion

Data augmentation is a critical technique in deep-learning image models that can enhance generalization. In this chapter, we have introduced a novel approach for *latent* image augmentation using a Variational Autoencoder (VAE) architecture. This method facilitates the combination of multiple augmentation techniques and offers greater control and interpretability of the latent space. Our experiments on the MNIST dataset have shown that our proposed approach outperforms comparable models in both flexibility of usage and performance. Furthermore, our method provides new insights into the underlying structure of the data and the relationship between different augmentations.

By treating augmentations as image-space priors instead of simply randomizing data, we can constrain the VAE's information bottleneck and learn a low-dimensional proxy for the

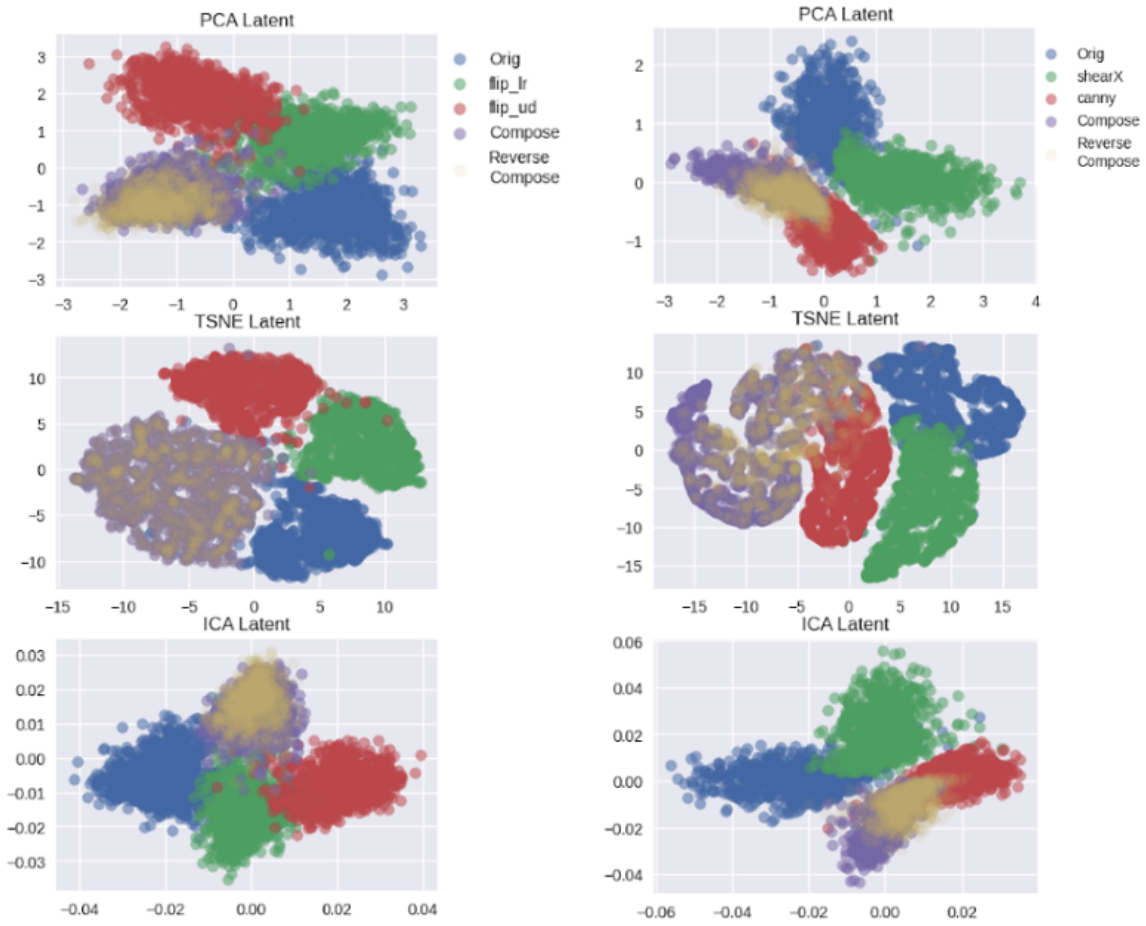


Figure 6.10: "Flips" (left) and "shear X-direction, canny edge-detect" (right) 2-d projections using PCA, ICA, and T-SNE algorithms.

augmentation model. For future work, we plan to explore the combination of the CVAE and the LAVAE, such as producing latent augmentations per class conditional, changing the one-hot conditional to continuous augmentations, and having augmentations operate only on the conditional embedding space. Additionally, we aim to apply our approach to other datasets, including those where the image model might not be known, such as 2D to 3D reconstruction or perspective shift tasks.

CHAPTER 7

PureEBM: Universal Poison Purification via Mid-run Dynamics of Energy-Based Models

7.1 Introduction

Large datasets empower modern, over-parameterized deep learning models. An adversary can easily insert a few powerful, but imperceptible, poisoned images into these datasets, often scraped from the open Internet, and manipulate a Neural Network's (NN) behavior at test time with a high success rate. These poisons can be constructed with or without information on NN architecture or training dynamics. With the increasing capabilities and utilization of large deep learning models, there is growing research in securing model training against such adversarial poison attacks with minimal impact on natural accuracy.

Numerous methods of poisoning deep learning systems have been proposed in recent years. These disruptive techniques typically fall into two distinct categories: backdoor, triggered data poisoning, or triggerless poisoning attacks. Triggered attacks conceal an imperceptible trigger pattern in the samples of the training data leading to the misclassification of test-time samples that contain the hidden trigger [GDG17, TTM18, SGF21, ZPJ22]. In contrast, triggerless poisoning attacks involve introducing slight, bounded perturbations to individual images that align them with target images of another class within the feature or gradient space resulting in the misclassification of specific instances without necessitating further modification during inference [SHN18, ZHL19, HGF20, GFH21, AMW21]. In both scenarios, poisoned examples often appear benign and correctly labeled, making them

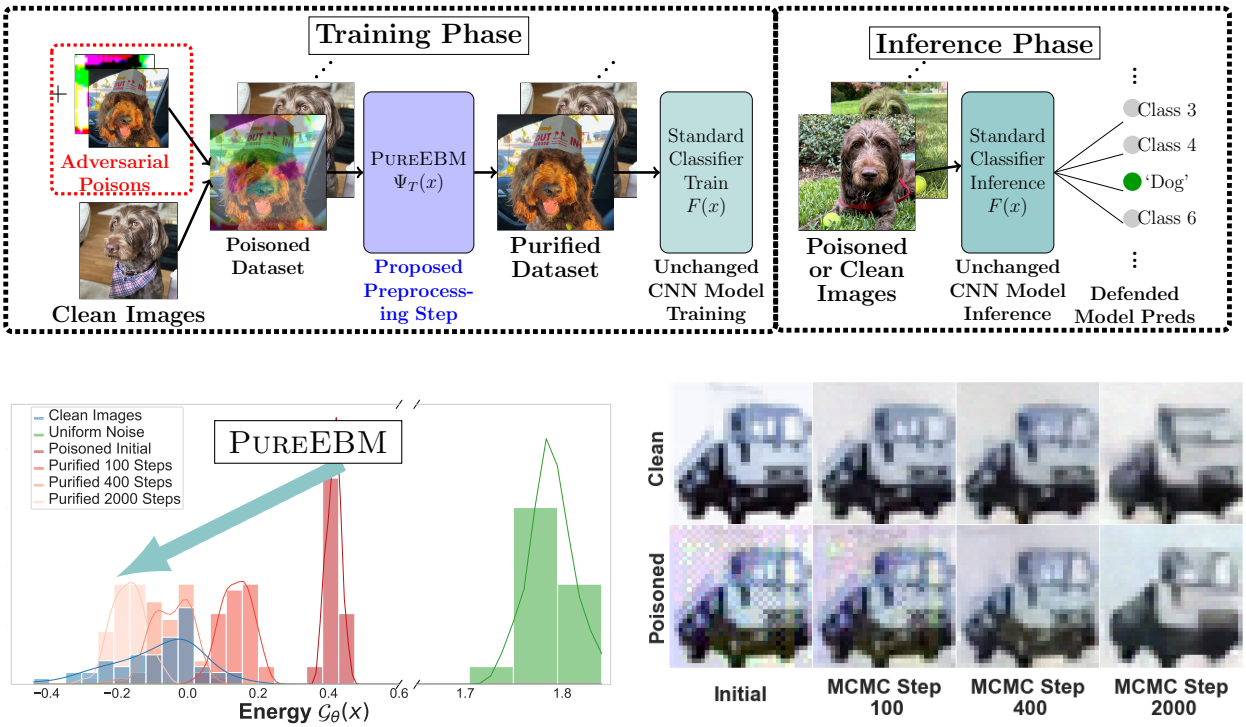


Figure 7.1: **Top** The full PureEBM pipeline is shown where we apply our method as a preprocessing step with no further downstream changes to the classifier training or inference. *Poisoned images are moderately exaggerated to show visually.* **Bottom Left** Energy distributions of clean, poisoned, and purified images. Our method pushes poisoned images via purification into the natural image energy manifold. **Bottom Right** The removal of poisons and similarity of clean and poisoned images with more MCMC purification steps. The purified dataset results in SoTA defense and high classifier natural accuracy.

challenging to detect by observers or algorithms.

Current defense strategies against data poisoning exhibit significant limitations. While some methods rely on anomaly detection through techniques such as nearest neighbor analysis, training loss minimization, singular-value decomposition, feature activation or gradient clustering [CSL08, SKL17, TLM18, CCB19, PGH20, YLM22, PDM22], others resort to robust training strategies including data augmentation, randomized smoothing, ensembling,

adversarial training and maximal noise augmentation [WXK20, LF20, ACG16, MZH19, LLK21, TFY21, LYM23]. However, these approaches either undermine the model's generalization performance [GFS21, YLM22], offer protection only against specific attack types [GFS21, PGH20, TLM18], or prove computationally prohibitive for standard deep learning workflows [ACG16, CCB19, MMS18, YLM22, GFS21, PGH20, LYM23]. There remains a critical need for more effective and practical defense mechanisms in the realm of deep learning security.

In our previous chapters, we have been using VAEs as an unsupervised learning tool that we try to add causal structure to in various ways. In this problem, however, we find that these poisons disrupt the data manifold in a way that attacks a neural network's training ability, even through basic augmentations. Adapting to this problem, we expect that providing augmentations through the use of a supervised learning technique could be a way to tackle these poisoning attacks. In this chapter, we propose a simple but powerful Energy-Based model defense PureEBM, against poisoning attacks. We make the key observation that the energy of poisoned images, found through an unsupervised neural network, is significantly higher than that of baseline images for an EBM trained on a natural dataset of images (even when poisoned samples are present). Using iterative sampling techniques such as Markov Chain Monte Carlo (MCMC) that utilize noisy gradient information from the EBM, we can purify samples of any poison perturbations iteratively. This universal stochastic preprocessing step $\Psi_T(x)$ moves poisoned samples into the lower energy, natural data manifold with minimal loss in natural accuracy. The PureEBM pipeline, energy distributions, and the MCMC purification process on a sample image can be seen in Figure 7.1. This work finds that PureEBM significantly outperforms state-of-the-art defense methods in all tested poison scenarios. Our key contributions in this work are:

- A state-of-the-art stochastic preprocessing defense $\Psi_T(x)$ against adversarial poisons, using Energy-Based models and MCMC sampling

- Experimental results showing the broad application of $\Psi_T(x)$ with minimal tuning and no prior knowledge needed of the poison type and classification model
- Results showing SoTA performance is maintained when the EBM training data includes poisoned samples and/or natural images from a similar out-of-distribution dataset

7.2 Related Work

7.2.1 Targeted Data Poisoning Attack

Poisoning of a dataset occurs when an attacker injects small adversarial perturbations δ (where $\|\delta\|_\infty \leq \xi$ and typically $\xi = 8/255$) into a small fraction, α, of training images. These train-time attacks introduce *local sharp regions* with a considerably higher *training loss* [LYM23]. A successful attack occurs when SGD optimizes the cross-entropy training objective on these poisoned images, maximizing either the inference time impact of a trigger, or modifying a target image classification by aligning poisoned images in the gradient or some feature space. The process of learning these adversarial perturbations creates backdoors in an NN.

In the realm of deep network poison security, we encounter two primary categories of attacks: triggered and triggerless attacks. Triggered attacks, often referred to as backdoor attacks, involve contaminating a limited number of training data samples with a specific trigger (often a patch) ρ (similarly constrained $\|\rho\|_\infty \leq \xi$) that corresponds to a target label, y^{adv}. After training, a successful backdoor attack misclassifies when the perturbation ρ is added:

$$F(x) = \begin{cases} y & x \in \{x : (x, y) \in \mathcal{D}_{test}\} \\ y^{\text{adv}} & x \in \{x + \rho : (x, y) \in \mathcal{D}_{test}, y \neq y^{\text{adv}}\} \end{cases} \tag{7.1}$$

Early backdoor attacks were characterized by their use of non-clean labels [CLL17, GDG17, LMA17, SGF21], but more recent iterations of backdoor attacks have evolved to produce

poisoned examples that lack a visible trigger [TTM18, SSP19, ZPJ22].

On the other hand, triggerless poisoning attacks involve the addition of subtle adversarial perturbations to base images, aiming to align their feature representations or gradients with those of target images of another class, causing target misclassification [SHN18, ZHL19, HGF20, GFH21, AMW21]. These poisoned images are virtually undetectable by external observers. Remarkably, they do not necessitate any alterations to the target images or labels during the inference stage. For a poison targeting a group of target images $\Pi = \{(x^\pi, y^\pi)\}$ to be misclassified as y^{adv}, an ideal triggerless attack would produce a resultant function:

$$F(x) = \begin{cases} y & x \in \{x : (x, y) \in \mathcal{D}_{test} \setminus \Pi\} \\ y^{\text{adv}} & x \in \{x : (x, y) \in \Pi\} \end{cases} \tag{7.2}$$

The current leading poisoning attacks that we assess our defense against are:

- **Bullseye Polytope (BP):** BP crafts poisoned samples that position the target near the center of their convex hull in a feature space [AMW21].
- **Gradient Matching (GM):** GM generates poisoned data by approximating a bi-level objective by aligning the gradients of clean-label poisoned data with those of the adversarially labeled target [GFH21]. This attack has shown effectiveness against data augmentation and differential privacy.
- **Narcissus (NS):** NS is a clean-label backdoor attack that operates with minimal knowledge of the training set, instead using a larger natural dataset, evading state-of-the-art defenses by synthesizing persistent trigger features for a given target class [ZPJ22].

7.2.2 Defense Strategies

Poison defense categories broadly take two primary approaches: filtering and robust training techniques. Filtering methods identify outliers in the feature space through methods

such as thresholding [SKL17], nearest neighbor analysis [PGH20], activation space inspection [CCB19], or by examining the covariance matrix of features [TLM18]. These defenses often assume that only a small subset of the data is poisoned, making them vulnerable to attacks involving a higher concentration of poisoned points. Furthermore, these methods substantially increase training time, as they require training with poisoned data, followed by computationally expensive filtering and model retraining [CCB19, PGH20, SKL17, TLM18].

On the other hand, robust training methods involve techniques like randomized smoothing [WXK20], extensive data augmentation [BCF21], model ensembling [LF20], gradient magnitude and direction constraints [HCK20], poison detection through gradient ascent [LLK21], and adversarial training [GFS21, MMS18, TFY21]. Additionally, differentially private (DP) training methods have been explored as a defense against data poisoning [ACG16, JE19]. Robust training techniques often require a trade-off between generalization and poison success rate [ACG16, HCK20, LLK21, MMS18, TFY21, LYM23] and can be computationally intensive [GFS21, MMS18]. Some methods use optimized noise constructed via Generative Adversarial Networks (GANs) or Stochastic Gradient Descent methods to make noise that defends against attacks [MSH21, LYM23].

Recently [YLM22] proposed EPIc, a coreset selection method that rejects poisoned images that are isolated in the gradient space throughout training, and [LYM23] proposed FrieNDs, a per-image pre-processing transformation that solves a min-max problem to stochastically add l_∞ norm ζ-bound 'friendly noise' (typically 16/255) to combat adversarial perturbations. These two methods are the SoTA and will serve as a benchmark for our PureEBM method in the experimental results.

When compared to augmentation-based and adversarial training methods, our approach stands out for its simplicity, speed, and ability to maintain strong generalization performance. We show that adding gradient noise in the form of iterative Langevin updates can purify poisons and achieve superior generalization performance compared to SoTA defense methods EPIc and FrieNDs. The Langevin noise in our method proves highly effective in removing

the adversarial signals while metastable behaviors preserve features of the original image, due to the dynamics of mid-run chains from our EBM defense method.

7.3 PureEBM: Purifying Langevin Defense against Poisoning Attacks

Given a clean training set $\mathcal{X}_{clean} \subset \mathbb{R}^D$ consisting of i.i.d. sample images $x_i \sim p_{clean}$ for $i = 1, \ldots, n$. Targeted data poisoning attacks modify αn training points, by adding optimized perturbations δ constrained by $\mathcal{C} = \{\delta \in \mathbb{R}^D : \|\delta\|_\infty \leq \xi\}$. Poisons crafted by such attacks look innocuous to human observers and are seemingly labeled correctly. Hence, they are called clean-label attacks. These images define a new distribution $x_i + \delta_i \sim p_{poison}$, so that our training set comes from the mixture of probability distributions:

$$p_{data} = (1 - \alpha)p_{clean} + \alpha p_{poison} \tag{7.3}$$

The goal of adding these poisons is to change the prediction of a set of target examples $\Pi = \{(x^\pi, y^\pi)\} \subset \mathcal{D}_{test}$ or triggered examples $\{(x + \rho, y) : (x, y) \in \mathcal{D}_{test}\}$ to an adversarial label y^{adv}.

Targeted clean-label data poisoning attacks can be formulated as the following bi-level optimization problem:

$$\begin{aligned} &\underset{\substack{\delta_i \in \mathcal{C}_\delta, \rho \in \mathcal{C}_\rho \\ \sum_{i=0}^n \mathbb{1}_{\delta_i \neq \mathbf{0}} \leq \alpha n}}{\operatorname{argmin}} \sum_{(x^\pi, y^\pi) \in \Pi} \mathcal{L}\left(F(x^\pi + \rho; \phi(\delta)), y^{\text{adv}}\right) \\ &\quad s.t. \quad \phi(\delta) = \underset{\phi}{\operatorname{argmin}} \sum_{(x,y) \in \mathcal{D}} \mathcal{L}\left(F(x + \delta_i; \phi), y\right) \end{aligned} \tag{7.4}$$

For a triggerless poison, we solve for the ideal perturbations δ_i to minimize the adversarial loss on the target images, where $\mathcal{C}_\delta = \mathcal{C}$, $\mathcal{C}_\rho = \{\mathbf{0} \in \mathbb{R}^D\}$, and $\mathcal{D} = \mathcal{D}_{train}$. To address the above optimization problem, powerful poisoning attacks such as Meta Poison (MP) [HGF20], Gradient Matching (GM) [GFH21], and Bullseye Polytope (BP) [AMW21] craft the poisons

to mimic the gradient of the adversarially labeled target, i.e.,

$$\nabla\mathcal{L}\left(F_\phi\left(x^\pi\right), y^{\text{adv}}\right) \propto \sum_{i:\delta_i \neq \mathbf{0}} \nabla\mathcal{L}\left(F_\phi(x_i + \delta_i), y_i\right) \tag{7.5}$$

Minimizing the training loss on RHS of Equation 7.5 also minimizes the adversarial loss objective of Equation 7.4.

For the triggered poison, Narcissus (NS), we find the most representative patch ρ for class π given $\mathcal{C}$, defining Equation 7.4 with $\mathcal{C}_\delta = \{\mathbf{0} \in \mathbb{R}^D\}$, $\mathcal{C}_\rho = \mathcal{C}$, $\Pi = \mathcal{D}^\pi_{train}, y^{\text{adv}} = y^\pi$, and $\mathcal{D} = \mathcal{D}_{POOD} \cup \mathcal{D}^\pi_{train}$. In particular, this patch uses a public out-of-distribution dataset $\mathcal{D}_{POOD}$ and only the targeted class $\mathcal{D}^\pi_{train}$. As finding this patch comes from another natural dataset and does not depend on other train classes, NS has been more flexible to model architecture, dataset, and training regime [ZPJ22].

7.3.1 Energy-Based Model

An Energy-Based Model (EBM) is formulated as a Gibbs-Boltzmann density, as introduced in [XLZ16]. This model can be mathematically represented as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(-\mathcal{G}_\theta(x))q(x), \tag{7.6}$$

where $x \in \mathcal{X} \subset \mathbb{R}^D$ denotes an image signal, and $q(x)$ is a reference measure, often a uniform or standard normal distribution. Here, $\mathcal{G}_\theta$ signifies the energy potential, parameterized by a Convolutional Network with parameters θ. The normalizing constant, or the partition function, $Z(\theta) = \int \exp\{-\mathcal{G}_\theta(x)\}q(x)dx = \mathbb{E}_q\left[\exp(-\mathcal{G}_\theta(x))\right]$, while essential, is generally analytically intractable. In practice, $Z(\theta)$ is not computed explicitly, as $\mathcal{G}_\theta(x)$ sufficiently informs the Markov Chain Monte Carlo (MCMC) sampling process.

As which α of the images are poisoned is unknown, we treat them all the same for a universal defense. Considering i.i.d. samples $x_i \sim p_{\text{data}}$ for $i = 1, \ldots, n$, with n sufficiently large, the sample average over x_i converges to the expectation under p_{data} and one can learn a parameter θ^* such that $p_{\theta^*}(x) \approx p_{\text{data}}(x)$. For notational simplicity, we equate the sample

average with the expectation.

The objective is to minimize the expected negative log-likelihood, formulated as:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) \doteq \mathbb{E}_{p_{\text{data}}}\left[\log p_\theta(x)\right]. \tag{7.7}$$

The derivative of this log-likelihood, crucial for parameter updates, is given by:

$$\begin{aligned}\nabla\mathcal{L}(\theta) &= \mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] - \mathbb{E}_{p_\theta}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] \\ &\doteq \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{G}_\theta(x_i^+) - \frac{1}{k}\sum_{i=1}^{k} \nabla_\theta \mathcal{G}_\theta(x_i^-),\end{aligned} \tag{7.8}$$

where x_i^+ are called *positive* samples as their probability is increased and where k samples $x_i^- \sim p_\theta(x)$ are synthesized examples (obtained via MCMC) from the current model, representing the *negative* samples as probability is deceased.

In each iteration t, with current parameters denoted as θ_t, we generate k synthesized examples $x_i^- \sim p_{\theta_t}(x)$. The parameters are then updated as $\theta_{t+1} = \theta_t + \eta_t \nabla\mathcal{L}(\theta_t)$, where η_t is the learning rate.

In this work, to obtain the negative samples x_i^- from the current distribution $p_\theta(x)$ we utilize the iterative application of the Langevin update as the MCMC method:

$$x_{\tau+1} = x_\tau - \Delta\tau \nabla_{x_\tau} \mathcal{G}_\theta(x_\tau) + \sqrt{2\Delta\tau}\epsilon_\tau, \tag{7.9}$$

where $\epsilon_k \sim \mathrm{N}(0, I_D)$, τ indexes the time step of the Langevin dynamics, and $\Delta\tau$ is the discretization of time [XLZ16]. $\nabla_x \mathcal{G}_\theta(x) = \partial\mathcal{G}_\theta(x)/\partial x$ can be obtained by back-propagation. If the gradient term dominates the diffusion noise term, the Langevin dynamics behave like gradient descent. We implement EBM training following [NHH20], see App C.3.1 for details.

In practice, we find that learning the mixture of distributions $p_{data} = (1-\alpha)p_{clean} + \alpha p_{poison}$ yields an EBM with a purifying ability similar to that of training on p_{clean}, suggesting our unsupervised maximum likelihood estimation (MLE) method is unsurprisingly not affected by targeted poisons.

Algorithm 1 Data Preprocessing with PureEBM: $\Psi_T(x)$

Require: Trained ConvNet potential $\mathcal{G}_\theta(x)$, training images $x \in X$, Langevin steps T, Time discretization $\Delta\tau$

for τ in $1 \dots T$ **do**

Langevin Step: draw $\epsilon_\tau \sim \mathrm{N}(0, I_D)$

$$x_{\tau+1} = x_\tau - \Delta\tau \nabla_{x_\tau} \mathcal{G}_\theta(x_\tau) + \sqrt{2\Delta\tau}\epsilon_\tau$$

end for

Return: Purified set $\tilde{X}$ from final Langevin updates

7.3.2 Classification with Stochastic Transformation

Let $\Psi_T : \mathbb{R}^D \to \mathbb{R}^D$ be a stochastic pre-processing transformation. In this work, $\Psi_T(x)$, the random variable of a fixed image x, is realized via T steps of the Langevin update (7.9). One can compose a stochastic transformation $\Psi_T(x)$ with a randomly initialized deterministic classifier $f_{\phi_0}(x) \in \mathbb{R}^J$ (for us, a naturally trained classifier) to define a new deterministic classifier $F_\phi(x) \in \mathbb{R}^J$ as $F_{\phi_0}(x) = E_{\Psi_T(x)}[f_{\phi_0}(\Psi_T(x))]$, which is then trained with cross-entropy loss via SGD to realize $F_\phi(x)$. As it is infeasible to evaluate the above expectation of the stochastic transformations $\Psi_T(x)$ as well as training many randomly initialized classifiers we take $f_\phi(\Psi_T(x))$ as the point estimate of the classifier $F_\phi(x)$. In our case this instantaneous approximation of $F_\phi(x)$ is valid because $\Psi_T(x)$ has a low variance for convergent mid-run MCMC.

7.3.3 Why EBM Langevin Dynamics Purify

The theoretical basis for eliminating adversarial signals using MCMC sampling is rooted in the established steady-state convergence characteristic of Markov chains. The Langevin update, as specified in Equation (7.9), converges to the distribution $p_\theta(x)$ learned from unlabeled data after an infinite number of Langevin steps. The memoryless nature of a steady-

state sampler guarantees that after enough steps, all adversarial signals will be removed from an input sample image. Full mixing between the modes of an EBM will undermine the original natural image class features, making classification impossible [HMZ21]. [NHH20] reveals that without proper tuning, EBM learning heavily gravitates towards *non-convergent ML* where short-run MCMC samples have a realistic appearance and long-run MCMC samples have unrealistic ones. In this work, we use image initialized *convergent learning*. $p_\theta(x)$ is described further by Algorithm 1.

The metastable nature of EBM models exhibits characteristics that permit the removal of adversarial signals while maintaining the natural image's class and appearance [HMZ21]. Metastability guarantees that over a short number of steps, the EBM will sample in a local mode, before mixing between modes. Thus, it will sample from the initial class and not bring class features from other classes in its learned distribution. Consider, for instance, an image of a horse that has been subjected to an adversarial ℓ_∞ perturbation, intended to deceive a classifier into misidentifying it as a dog. The perturbation, constrained by the ℓ_∞-norm ball, is insufficient to shift the EBM's recognition of the image away from the horse category. Consequently, during the brief sampling process, the EBM actively replaces the adversarially induced 'dog' features with characteristics more typical of horses, as per its learned distribution resulting in an output image resembling a horse more closely than a dog. It is important to note, however, that while the output image aligns more closely with the general characteristics of a horse, it does not precisely replicate the specific horse from the original, unperturbed image.

Our experiments show that the mid-run trajectories (100-1000 MCMC steps) we use to preprocess the dataset $\mathcal{X}$ capitalize on these metastable properties by effectively purifying poisons while retaining high natural accuracy on $F_\phi(x)$ with no training modification needed. A chaos theory-based perspective on EBM dynamics can be found in App. C.1.1.

7.3.4 Erasing Poison Signals via Mid-Run MCMC

The stochastic transform $\Psi_T(x)$ is an iterative process, akin to a noisy gradient descent, over the unconditional energy landscape of a learned data distribution. As MCMC is run, the images will move from their initial energy toward p_{data}. As shown in Figure 7.1, the energy distributions of poisoned images are much higher, pushing the poisons away from the likely manifold of natural images. By using mid-run dynamics (150-1000 Langevin steps), we transport poisoned images back toward the center of the energy basin.

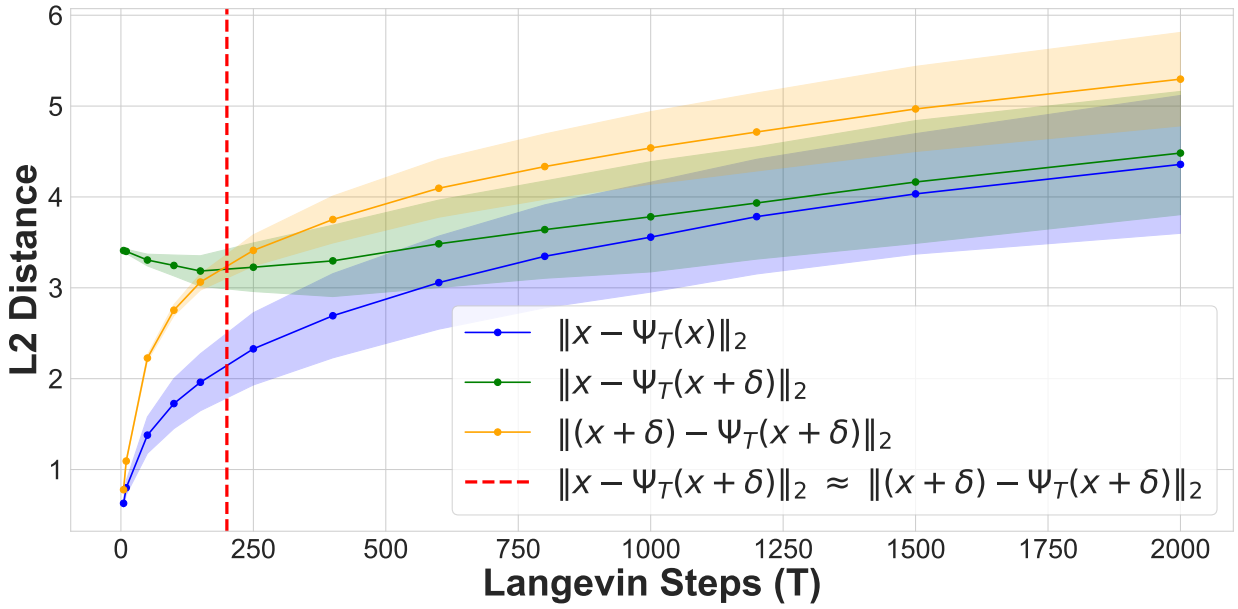


Figure 7.2: Plot of ℓ_2 distances between clean images and clean purified (blue), clean images and poisoned purified (green), and poisoned images and poisoned purified images (orange) at points on the MCMC sampling trajectory. Purifying poisoned images for less than 250 steps moves a poisoned image closer to its clean image with a minimum around 150, preserving the natural image while removing the adversarial features.

In the from-scratch poison scenarios, 150 Langevin steps can fully purify the majority of the dataset with minimal feature loss to the original image. In Figure 7.2 we explore the MCMC trajectory's impacts on ℓ_2 distance of both purified clean and poisoned images

from the initial clean image ($\|x - \Psi_T(x)\|_2$ and $\|x - \Psi_T(x+\delta)\|_2$), and the purified poisoned image's trajectory away from its poisoned starting point ($\|(x+\delta) - \Psi_T(x+\delta)\|_2$). Both poisoned and clean distance trajectories converge to similar distances away from the original clean image ($\lim_{T\to\infty} \|x - \Psi_T(x)\|_2 = \lim_{T\to\infty} \|x - \Psi_T(x+\delta)\|_2$), but the steady increase in image distance of the two trajectories offers an empirical perspective of the metastable, mid-run region. The intersection where $\|(x+\delta) - \Psi_T(x+\delta)\|_2 > \|x - \Psi_T(x+\delta)\|_2$ (indicated by the dotted red line), occurs at ∼150-200 Langevin steps and indicates when purification has moved the poisoned image closer to the original clean image than the poisoned version of the image. This region coincides with the expected start of the mid-run dynamics where our properties are most ideal for purification. Additional purification degrades necessary features for classifier training, as already seen previously in the bottom right of Figure 7.1.

We note that we are not the first to apply EBMs with MCMC sampling for robust classification, but we are, to the best of our knowledge, the first to apply an EBM-based purification method universally as a poison defense and use non-overlapping natural datasets to further extend the generality of EBM purification.

7.4 Experiments

7.4.1 Experimental Details

We compare our method, PureEBM, against previous state-of-the-art defenses EPIc and FrieNDs on the current leading triggered poison, Narcissus (NS) and triggerless poisons, Gradient Matching (GM) and Bullseye Polytope (BP). Triggerless attacks GM and BP have 100 and 50 poison scenarios while NS has 10 (one per class). Primary results use a ResNet18 classifier and the CIFAR-10 dataset. We train a variety of EBMs using the training techniques described in App. 7.3.1 with specific datasets for our experimental results:

1. **PureEBM**: To ensure EBM training is blind to poisoned images, we exclude the

Table 7.1: Poison success and natural accuracy in all poisoned from-scratch training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 100 GM experiments and NS triggers over 10 classes.

	From Scratch				
	200 - Epochs				
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	44.00	$94.84_{0.2}$	$43.95_{33.6}$	$94.89_{0.2}$	93.59
EPIc	10.00	$85.14_{1.2}$	$27.31_{34.0}$	$82.20_{1.1}$	84.71
FrieNDs	**0.00**	$\mathbf{91.15}_{0.4}$	$9.49_{25.9}$	$91.06_{0.2}$	83.03
PureEBM	**0.00**	$\mathbf{92.26}_{0.2}$	$\mathbf{1.27}_{0.6}$	$\mathbf{92.91}_{0.2}$	**2.16**
PureEBM-P	NA	NA	$\mathbf{1.38}_{0.7}$	$\mathbf{92.70}_{0.2}$	**2.78**
PureEBM$_{CN-10}$	**0.00**	$\mathbf{92.99}_{0.2}$	$\mathbf{1.43}_{0.8}$	$\mathbf{92.90}_{0.2}$	**3.06**
PureEBM$_{IN}$	1.00	$\mathbf{92.98}_{0.2}$	$\mathbf{1.39}_{0.8}$	$\mathbf{92.92}_{0.2}$	**2.50**
PureEBM-P$_{CN-10}$	NA	NA	$\mathbf{1.64}_{0.01}$	$\mathbf{92.86}_{0.20}$	**4.34**
	80 - Epochs				
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	47.00	$93.79_{0.2}$	$32.51_{30.3}$	$93.76_{0.2}$	79.43
EPIc	44.00	$92.46_{0.3}$	$21.53_{28.8}$	$88.05_{1.1}$	80.75
FrieNDs	**1.00**	$90.09_{0.4}$	$\mathbf{1.37}_{0.9}$	$90.01_{0.2}$	3.18
PureEBM	**1.00**	$\mathbf{91.36}_{0.3}$	$\mathbf{1.46}_{0.8}$	$\mathbf{91.83}_{0.3}$	**2.49**
PureEBM-P	NA	NA	$\mathbf{1.63}_{1.0}$	$\mathbf{91.49}_{0.3}$	3.47
PureEBM$_{CN-10}$	**1.00**	$\mathbf{92.02}_{0.2}$	$\mathbf{1.50}_{0.9}$	$\mathbf{92.03}_{0.2}$	**2.52**
PureEBM$_{IN}$	**1.00**	$\mathbf{92.02}_{0.2}$	$\mathbf{1.52}_{0.8}$	$\mathbf{92.02}_{0.3}$	**2.81**
PureEBM-P$_{CN-10}$	NA	NA	$\mathbf{1.68}_{1.0}$	$\mathbf{92.07}_{0.2}$	3.34

indices for all potential poison scenarios which resulted in 37k, 45k, and 48k training samples for GM, NS, and BP respectively of the original 50k CIFAR-10 train images.

2. **PureEBM-P**: Trained on the full CIFAR-10 dataset in which 100% of training samples are poisoned using their respective class' NS poison trigger. This model explores the ability to learn robust features even when the EBM is exposed to full adversarial influences during training (even beyond the strongest classifier scenario of 10% poison).

3. **PureEBM$_{\text{CN-10}}$**: Trained on the CINIC-10 dataset, which is a mix of ImageNet (70k) and CIFAR-10 (20k) images where potential poison samples are removed from CIFAR-

Table 7.2: Poison success and natural accuracy in all poisoned transfer training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 50 BP experiments and NS triggers over 10 classes.

	Transfer Learning				
	Fine-Tune				
	Bullseye Polytope-10%		**Narcissus-10%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	46.00	$89.84_{0.9}$	$33.41_{33.9}$	$90.14_{2.4}$	98.27
EPIc	42.00	$81.95_{5.6}$	$20.93_{27.1}$	$88.58_{2.0}$	91.72
FrieNDs	8.00	$87.82_{1.2}$	$3.04_{5.1}$	$89.81_{0.5}$	17.32
PureEBM	**0.00**	$\mathbf{88.95}_{1.1}$	$\mathbf{1.98}_{1.7}$	$\mathbf{91.40}_{0.4}$	**5.98**
PureEBM-p	NA	NA	$16.48_{27.2}$	$88.27_{2.4}$	86.49
PureEBM$_{CN-10}$	**0.00**	$\mathbf{88.67}_{1.2}$	$\mathbf{2.97}_{2.5}$	$\mathbf{90.99}_{0.3}$	**7.95**
PureEBM$_{IN}$	**0.00**	$\mathbf{87.52}_{1.2}$	$\mathbf{2.02}_{1.0}$	$\mathbf{89.78}_{0.6}$	**3.85**

	Linear - Bullseye Polytope			
	BlackBox-10%		**WhiteBox-1%**	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	93.75	$83.59_{2.4}$	98.00	$70.09_{0.2}$
EPIc	66.67	$84.34_{3.8}$	91.00	$64.79_{0.7}$
FrieNDs	33.33	$85.18_{2.3}$	19.00	$60.90_{0.6}$
PureEBM	**0.00**	$\mathbf{92.89}_{0.2}$	**6.00**	$\mathbf{64.51}_{0.6}$
PureEBM-p	NA	NA	NA	NA
PureEBM$_{CN-10}$	**0.00**	$\mathbf{92.82}_{0.1}$	**6.00**	$\mathbf{64.44}_{0.4}$
PureEBM$_{IN}$	**0.00**	$\mathbf{92.38}_{0.3}$	**6.00**	$\mathbf{64.98}_{0.3}$

10 indices [DCA18]. This model investigates the effectiveness of EBM purification when trained on a distributionally similar dataset.

4. **PureEBM$_{\mathbf{IN}}$**: Trained exclusively on the ImageNet (70k) portion of the CINIC-10 dataset. This model tests the generalizability of the EBM purification process on a public out-of-distribution (POOD) dataset that shares no direct overlap with the classifier's training data $\mathcal{X}$.

5. **PureEBM-P$_{\mathbf{CN-10}}$**: Trained on the CINIC-10 dataset where the CIFAR-10 subset is fully poisoned. This variant examines the EBM's ability to learn and purify data where a significant portion of the training dataset is adversarially manipulated, and the clean

images are from a POOD dataset.

A single hyperparameter grid-search for Langevin dynamics was done on the PureEBM model using a single poison scenario per training paradigm (from scratch, transfer linear and transfer fine-tune) as seen in App. C.6. The percentage of classifier training data poisoned is indicated next to each poison scenario. Additional details on poison sources, poison crafting, definitions of poison success, and training hyperparameters can be found in App. C.3.2.

7.4.2 Benchmark Results

Table 7.1 and 7.2 shows our primary results in which **PureEBM achieves state-of-the-art (SoTA) poison defense and natural accuracy in all poison scenarios** and fully poisoned PureEBM-P achieves SoTA performance for Narcissus. Furthermore, **all public out-of-distribution (POOD) EBMs achieve SoTA performance in almost every category** without additional hyperparameter search.

For GM, PureEBM matches SoTA in a nearly complete poison defense and achieves 1.1% less natural accuracy degradation, from no defense, than the previous SoTA. For BP, PureEBM exceeds the previous SoTA with an 8-33% poison defense reduction and 1.1-7.5% less degradation in natural accuracy. For NS, PureEBM matches or exceeds previous SoTA with a 1-8% poison defense reduction and 1.5% less degradation in natural accuracy.

7.4.3 Results on Additional Models and Datasets

Table 7.3 shows results when we apply NS poisons (generated using CIFAR-10) to the CINIC-10 dataset. To ensure no overlap for our EBMs, we train on CINIC-10's validation set, which has the same size and composition as its training set. Table 7.4 shows results for MobileNetV2 and DenseNet121 architectures. **PureEBM is SoTA across all models and in CINIC-10 NS poison scenarios** showing no performance dependence on dataset or model. Full results are in App. C.2.

Finally, the Hyperlight Benchmark CIFAR-10 (HLB) is a drastically different case study from our standard benchmarks with a residual-less network architecture, unique initialization scheme, and super-convergence training method that recently held the world record of achieving 94% test accuracy on CIFAR-10 using a surprising total of 10 epochs [Bal23]. We observe that NS still successfully poisons the HLB model, and does so by the end of the first epoch. Applying EPIc and FrieNDs becomes unclear, as they use model information after a warm-up period, but we choose the most sensible warm-up period of one epoch, even though the poisons have set in. From Table 7.4 subset selection based EPIc is unable to train effectively, and FrieNDs offers some defense. PureEBM still applies with minimal adjustment to the training pipeline and defends effectively against these poisons. Table 7.4 also shows the effect of differing MCMC steps where 25 MCMC steps already offers comparable defense to FrieNDs, and by 50 steps, PureEBM shows SoTA poison defense and natural accuracy. Increasing steps further reduces poison success, but at the cost of natural accuracy and linearly increasing preprocessing time.

The last column of the HLB section shows timing analysis on an NVIDIA A100 GPU. Due to HLB training speeds, timings primarily indicate the processing time of the defenses. PureEBM is faster in total train time and per epoch time than existing SoTA defense methods. We emphasize that, in practice, PureEBM can be applied once to a dataset and used across model architectures, unlike previous SoTA defenses EPIc and FrieNDs, which require train-time information on model outputs. See App. C.4 for further timing.

7.4.4 Further Experiments

Model Interpretability Using the Captum interpretability library, in Figure 7.3, we compare a clean model with clean data to the various defense techniques on a sample image poisoned with the NS Class 5 trigger ρ [KMM20]. Only the clean model and the model that uses PureEBM correctly classify the sample as a horse, and the regions most important to prediction, via occlusion analysis, most resemble the shape of a horse in the clean

Table 7.3: Poison success and natural accuracy when training on CINIC-10 Dataset From Scratch Results with NS Poison

CINIC-10 Narcissus - 1% From-Scratch (200 Epochs)				
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	CIFAR-10 Accuracy (%) ↑
None	$62.06_{0.21}$	$86.32_{0.10}$	90.79	$94.22_{0.16}$
EPIc	$49.50_{0.27}$	$81.91_{0.08}$	91.35	$91.10_{0.21}$
FrieNDs	$11.17_{0.25}$	$77.53_{0.60}$	82.21	$88.27_{0.68}$
PureEBM	$\mathbf{7.73}_{0.08}$	$\mathbf{82.37}_{0.14}$	**29.48**	$\mathbf{91.98}_{0.16}$

and PureEBM images. Integrated Gradient plots show how PureEBM actually enhances interpretability of relevant features in the gradient space for prediction compared to even the clean NN. Additionally, we see that the NN trained with PureEBM is less sensitive to input perturbations compared to all other NNs. See App. C.5 for additional examples.

Flatter solutions are robust to Poisons Recently [LYM23] showed that effective poisons introduce a local sharp region with a high training loss and that an effective defense can smooth the loss landscape of the classifier. We consider the curvature of the loss with respect to our model's weights as a way to evaluate defense success. The PSGD framework [Li15, Li19, Li22, PL24] estimates the Hessian of the loss $\mathbf{H}$ of the model over the full dataset and the poisoned points through training. In information theory, $0.5 \log\det(\mathbf{H})$ is a good proxy for the description length of the model parameters. We find that training with data points pre-processed by the PureEBM stochastic transformation $\Psi_T(x)$ reduces the curvature of the loss of the NN over the full dataset and around poisoned points. In effect, NNs trained with points defended with PureEBM are significantly more robust to perturbation than other defenses. In App. C.5.1, we find that PureEBM and FrieNDs models' parameters diverge from poisoned models more so than EPIc.

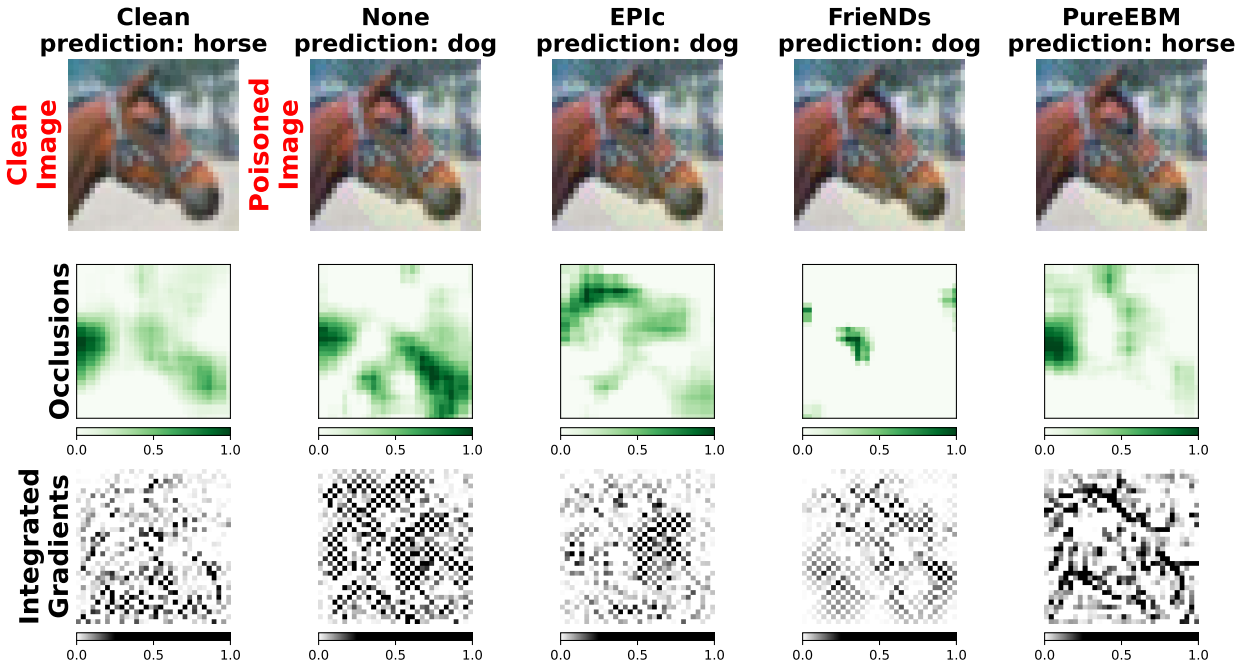


Figure 7.3: Defense Interpretability: Model using PureEBM focuses on the outline of the horse in the occlusions analysis and to a higher degree on the primary features in the gradient space than even the clean model on clean data.

7.5 Conclusion

Poisoning has the potential to become one of the greatest attack vectors to AI models, decreasing model security and eroding public trust. Further discussion of ethics and impact can be found in App. C.8. As a community, we hope to develop robust generalizable ML algorithms. In this work, we present PureEBM, a powerful Energy-Based Model defense against imperceptible train time data poisoning attacks. Our approach significantly advances the field of poison defense and model security by addressing the critical challenge of adversarial poisons in a manner that maintains high natural accuracy and method generality. Through extensive experimentation, PureEBM has demonstrated state-of-the-art performance in defending against a range of poisoning scenarios using the leading Gradient Matching, Narcissus, and Bullseye Polytope attacks. The key to our method's success is a

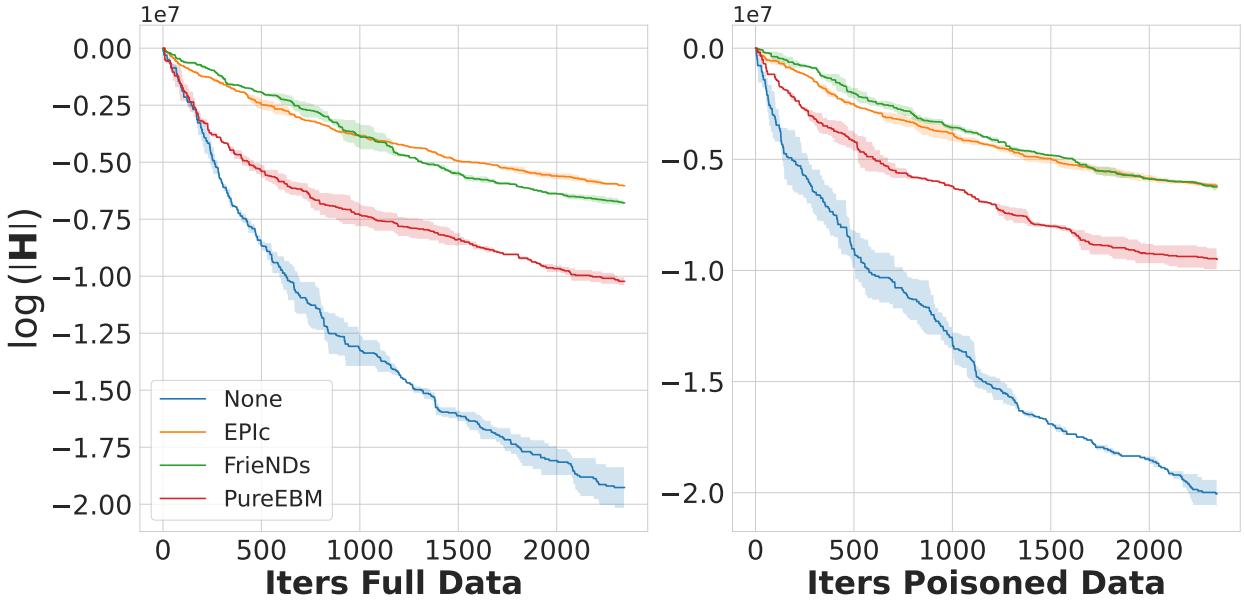


Figure 7.4: Estimate loss curvature - classifier robustness - with $\log(|\mathbf{H}|)$ against both full and poisoned subset of training data. Model trained with PureEBM has the lowest curvature compared to SoTA defense methods.

stochastic preprocessing step that uses MCMC sampling with an EBM to iteratively purify poisoned samples, moving them into a lower energy, natural data manifold. We share similar SoTA results with EBMs trained on out-of-distribution and poisoned datasets, underscoring the method's adaptability and robustness. A versatile, efficient, and robust method for purifying training data, PureEBM sets a new standard in the ongoing effort to fortify machine learning models against the evolving threat of data poisoning attacks. Because PureEBM neutralizes all SoTA data poisoning attacks effectively, we believe our research can have a significant **positive social impact** to inspire trust in widespread machine learning adoption.

Table 7.4: MobileNetV2 and DenseNet121 results and HyperlightBench for a novel training paradigm where PureEBM is still effective.

From Scratch NS-1% (200 epochs)				
	MobileNetV2		**DenseNet121**	
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	$32.70_{0.25}$	$93.92_{0.13}$	$46.52_{32.2}$	$95.33_{0.1}$
EPIc	$22.35_{0.24}$	$78.16_{9.93}$	$32.60_{29.4}$	$85.12_{2.4}$
FrieNDs	$2.00_{0.01}$	$88.82_{0.57}$	$8.60_{21.2}$	$91.55_{0.3}$
PureEBM	$\mathbf{1.64}_{0.01}$	$\mathbf{91.75}_{0.13}$	$\mathbf{1.42}_{0.7}$	$\mathbf{93.48}_{0.1}$
Linear Transfer WhiteBox BP-10%				
	MobileNetV2		**DenseNet121**	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	81.25	$73.27_{0.97}$	73.47	$82.13_{1.62}$
EPIc	56.25	$54.47_{5.57}$	41.67	$70.13_{5.2}$
FrieNDs	41.67	$68.86_{1.50}$	56.25	$80.12_{1.8}$
PureEBM	**0.00**	$\mathbf{78.57}_{1.37}$	**0.00**	$\mathbf{89.29}_{0.94}$
Hyperlight Bench CIFAR-10 NS-1% (10 Epochs)				
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	Train Time (s)
None	$76.39_{16.35}$	$93.95_{0.10}$	95.69	$6.81_{0.62}$
EPIc	$10.58_{18.35}$	$24.88_{6.04}$	50.21	$612.43_{30.16}$
FrieNDs	$11.35_{18.45}$	$87.03_{1.52}$	56.65	$427.50_{0.50}$
PureEBM-25	$10.59_{26.04}$	$92.75_{0.13}$	84.60	$54.70_{0.48}$
PureEBM-50	$\mathbf{2.16}_{1.22}$	$\mathbf{92.38}_{0.17}$	**3.74**	$92.89_{0.48}$
PureEBM-100	$\mathbf{1.89}_{1.06}$	$\mathbf{91.94}_{0.14}$	**3.47**	$168.69_{0.46}$
PureEBM-150	$\mathbf{1.93}_{1.15}$	$\mathbf{91.46}_{0.17}$	**4.14**	$244.72_{0.47}$
PureEBM-300	$\mathbf{1.68}_{0.82}$	$\mathbf{90.55}_{0.21}$	**2.89**	$478.29_{0.47}$

7.6 Acknowledgments

The work reported in this chapter is supported with Cloud TPUs from Google's Tensorflow Research Cloud (TFRC). We would like to acknowledge Jonathan Mitchell, Mitch Hill, Yuan Du and Kathrine Abreu for support on base EBM code. We would like to thank our lab mate Yunzheng Zhu for his help in generating poisons. Thanks to Xi-Lin Li for his insight of collecting curvature information to see if training on samples from PureEBM give a solution that is more robust to input perturbations compared to other defenses.

CHAPTER 8

Simulated Education

8.1 Introduction

Human-machine interactions have become an increasingly important question as AI tools spread. Dealing with humans, however, introduces many challenges. First, people are necessarily dynamic, both in that a person's approach to interacting with tools may change quickly due to unobservable external factors and in that each human's response may have long-term, unobservable, or unknown dependencies. Secondly, there will almost never be enough data to train any model with any individual human, as the amount of experiences an individual human can have is limited. Building upon the work we presented in Chapter 3, we turn to education as a prime example of a human-machine interaction that has long-standing research and interest, but provides a concrete example of the problems that we face.

Artificial intelligence's role in education has become an increasingly important topic lately with the introduction of large language models (LLMs), which provide students with custom responses and artificial discussions about almost any searchable topic on the Internet. However, in this study, we seek to understand how to best create a long-term educational assistant for an individual, targeted toward assisting with classes. Intelligent Tutoring Systems (ITS) are computer-based educational tools that provide adaptive instruction to learners and are considered "intelligent" enough to substitute for human tutors. When considering ITS in the STEM domains, there is an inherent focus on model building and individual progress tracking [FMK21]. As an ITS tracks a student's progress, it will make interventions that

will optimize the student's success in the course.

There are many points that make tracking human learning a difficult task for machines. In particular, education is a time-series problem with considerable hidden information. Hidden information occurs in many forms in the education environment, most clearly in student concept mastery, but also in terms of the possible responses that a student may have to a lecture and the effect of external factors. Each of these kinds of hidden information increases the difficulty of the problem, as we never have enough data from a single student to properly estimate everything, so we have to rely on a mixture of population knowledge and probing. Probing refers to the specific action by which the ITS either asks or quizzes student progress to update its estimate of the student's hidden concept mastery. The time-series nature of the problem adds to its difficulty, as if we save too much past information, the problem becomes intractable, while if we save too little, a past event may become a "hidden external event" from the perspective of the ITS.

To gain better insight of how to best design an ITS, we break down many of the benefits and difficulties of the education problem and create a simulated environment. This simulated environment allows us to explore both realistic and idealistic scenarios with different techniques.

In this chapter, we explore the options for creating an optimal ITS under many simulated environment configurations. Among the questions we would like to answer are:

1. Can we create an ITS that can adapt to a student's individual learning characteristics? Struggling students should be recommended some remedial action, while the same recommendations should not apply to a student making good progress.

2. What is the value of probing? Would an ITS that cannot probe do significantly worse than one that can? And then, if the cost of probing increases, at what point would we no longer find it worthwhile to probe?

3. Can we design a course structure that is more robust to poor probing? That is, if our

probing capabilities are limited, would a quiz structure be significantly better than a finals-only or midterm-final structure?

8.2 Background

8.2.1 Motivation

Recent years have enlightened many more of the public of the current state and the future potential of artificial intelligence (AI) in our society. In light of this new frontier of AI, an important area of research is to understand how to interface the use of AI with humans. Examples can be seen in all sorts of examples, such as self-driving cars, medical diagnosis, and education. AI has the potential to be extremely helpful to humans, but the black-box nature of AI can also be hard to accept for many people and dangerous in extreme cases. Thus, we would like to investigate ways to incorporate explainability and human interaction into the use of AI. Education can also be viewed as a proxy environment, providing some insight into the goals and problems of dealing with dynamic human interactions.

Education is an interesting problem for artificial intelligence. On the positive side, typical educational programs are heavily structured and rules-based. This structure comes from many areas. There is a prerequisite concept structure for many classes, especially in STEM subjects. Furthermore, we have a wealth of expert knowledge in how to teach specific courses, and a general idea for overall good teaching methodology. An example is that we know prerequisite concepts need to be at a certain level before any headway into new concepts can be made.

However, education is challenging for artificial intelligence. Even outside the machine learning space, education is partially observed – no instructor knows everything about a student. Partial observability can hinder the system's ability to adapt appropriately because it might not have access to all relevant contextual information, such as workload in other courses, extracurricular work and events, and family and personal relationships. This can

make it difficult for the system to detect when a student is struggling, engaged, or in need of assistance. Without a complete picture of the learner's interactions, the system will miss opportunities to provide support, which can impact the learning process.

Furthermore, education is a time-series problem. The state of a student is constantly changing with every point in time, especially if going to lecture, actively doing readings or assignments, and discussing with friends. As with many other instances of time-series data, time-series can pose several problems. First, we need to maintain an estimate of the student learning state, which depends, potentially, on points in time far in the past. It also depends on more parameters that we likely have no access to for the individual. Second, the true estimation of a student's state at a given point in time is effectively impossible without an extensive examination, detracting from time spent instructing and learning.

Finally, explainability can be important, as students and teachers may be distrustful of computer recommendations. This can go in many ways, where extremes would be explaining to a struggling student why studying a specific concept is the most effective way to improve, but also in explaining to an excelling student that no further studying is necessary. Furthermore, we would like some method that is at least somewhat flexible in differing population distributions. These two points are closely related when it comes to methodology designs.

8.2.2 Prior Work

Intelligent tutor systems (ITS) have been investigated for a long time and have been shown to outperform other computer aided instruction [KF15, KRW07]. Much work has been contributed via the framework provided by ASSISTments [PSO22]. In particular, many experiments are run in the ASSISTments platform, and they include a long list of features that can quantify the students' skill level at any given point in time. In particular, this work has inspired and advanced the knowledge tracing domain significantly [PSH15, AWN22]. We use these features to create the simulated environment and acknowledge that we would like to do further analysis to see how much we can fit ASSISTments datasets within our

own model. We use basic knowledge tracing in our population model, though we maintain explainability as our primary goal.

The field of reinforcement learning (RL) has been a major driving force in the development of AI. RL has been used to solve many problems, such as playing games [SHM16, MKS13, VBC19, SHS17], robotics [KBP13], and even medical applications [BJS21]. RL deals with an agent interacting with an environment and through the agent's interactions with the environment, learns how to best respond in the future. It combines the ability of online learning with the need to explore possible action consequences in the environment. An adjacent field, *inverse reinforcement learning* (IRL) has also arisen in response to the popularity of RL, where we try to learn an optimal reward function that explains expert knowledge. While we don't directly reference IRL in this work, our use of prior knowledge can be easily extended to a discussion on IRL.

A considerable amount of work has taken place for RL in the field of education [FWH23]. [LXZ20] tests the effectiveness of a Deep Q-Network (DQN), a Deep RL approach, on a simpler education environment, finding that it significantly outperforms heuristic methods. It shares further similarities in that it introduces a population model, using a standard single-step MLP, and a two-concept continuous-valued simulation course. We build upon this simulation by introducing deliberate hidden information, introducing probing and the trade-off of immediate time rewards, probes, and tutors, having a variety of concept mastery dynamics conditioned on student type, and a highly configurable simulation.

8.3 Problem Setting

We view the education problem as a controls problem with unknown system dynamics. The autonomous intelligent tutoring system, a cognitive-dynamic system, is trying to give recommendations to steer the student (the system) in the correct direction. If all the mechanics of the system are deterministically known, course structure, student responses, etc., then we

would have a traditional controls problem.

This education problem is based on the interactions between students, teachers, and the overall structure of the course. Underlying everything, we assume that education, especially in STEM subjects comes with a hierarchical causal structure, *e.g.* calculus builds on algebra, which builds on arithmetic. Whether a student learns a concept depends both on their mastery of prerequisite concepts and whether they have the motivation to spend time.

Students may have the noble goal of achieving knowledge of the course material, but most of the time have the proxy goal of getting a good grade. As a student progresses through a course, she can take many actions and their state necessarily changes over time. However, each student's actions differ based on individual factors and are mostly unobserved. However, we consider that a student will consider a trade-off between extra performance in a course and doing other life activities.

Similarly, professors generally want to educate students in their field, but also have other interests such as research, resulting in limited time. However, professors are typically unaware of the students' states. Therefore, traditionally courses have a hard time providing individualized support to students and require individual tutors.

Given this initial view, we introduce several major hurdles to directly understanding the creation of an ITS.

8.3.1 Hidden Information

Education is a difficult task because of the various levels of hidden information. The control mechanics are not only stochastic, they are unknown and can be changing as a response to actions or due to completely unknown outside sources. At even the basic level, the two major hidden pieces of information at any point in time are the current concept state and the student's responses to any intervention that is taken. We will make restricting assumptions later to make the estimation problem feasible, but *sensing* and *probing* become the main

ways to estimate the knowledge of the dynamics of the system.

8.3.2 Probing, Measurements, and Quantization

All measurements are imperfect and when it comes to human interactions, this is exacerbated by other psychological and social factors. In general, observations in the real world can only be measured up to some level of uncertainty, constrained by physical limits or just in measurement capacity of devices in general. In an ideal scenario, we would be able to directly know each student's mastery for all relevant concepts. However, this direct information cannot actually be obtained. Instead, this can only be measured through examinations, which are expensive in time for both students and instructors and are subject to measurement randomness, or through asking for student self-perception and historical performance, which can help us predict future performance. However, experiments we have run have shown that students often have no comparison metric to determine their own performance. Therefore, if possible, a quantitative measurement is preferred.

On the other hand, we only have a limited number of control actions. There is a many-to-one relationship between possible states and the best possible action. Also, in education, intervention choices are less critical than in some other domains such as drone flight or medicine. These two properties combine to allow for some action ambiguity. As such, having a high precision state space is wasteful. It increases the necessary amount of parameters to draw precise boundaries, data to train such parameters, and difficulty in explanation.

8.3.3 Individual Dynamics

Learning is inherently a dynamic process. Students build upon the information that they have already learned. Weak mastery of a prerequisite can hinder student developments on new dependent concepts. Furthermore, individual dynamics arise from both student differences and external factors. This can both be a reactionary response to something

observable, e.g. an exam or homework, or unobservable, e.g. family trouble or financial issues.

In a traditional Markov process, having full knowledge of the single time-step, $\mathbb{P}[s_t|s_{t-1}]$, fully characterizes the dynamics. Assuming this can be obtained, the main consideration lies in the decision of the state space, which can keep updated latent space information from past points in time. As students will respond to actions chosen by the ITS, we break down the state into the student's direct educational state and the student's observations of the ITS's actions as part of the student's observation space, notated as $\mathbb{P}[s_t|s_{t-1}, a_{t-1}]$. The Markov property is attractive, as it minimizes long-term graph connections, and allows prediction to occur independently of the past. With it, we can start helping a student no matter when they start using the ITS in the course (up to the warm-up period to filling out information about the student state s_t). Theoretically, with sufficiently large state spaces, the entire past can be represented in s_t. However, if this possible state space is too large, solving for the dynamics is intractable.

Creating a relevant state space is thus nontrivial. In the education space, students may change dynamics due to unobserved events, possibly creating issues in the future. These could be due to events within education, such as poor teaching of a concept in a previous institution, or outside of education, such as prioritizing a part-time job. This can create situations where two students with the exact same history in observations in a course could suddenly diverge greatly. For any unobservable or unmodeled event, we require our dynamics to be flexible to sudden changes. This requires some consideration of the *stability-plasticity dilemma* [Hay12].

The other problem is that the dynamics themselves are an individual property. For every individual, $\mathbb{P}[s_t|s_{t-1}, a_{t-1}]$ will be different. Then, should the individual be included in the state space s_t? It is infeasible to do so, as we can never actually get enough data about the individual to accurately estimate the individual's transitions. Not only do individuals just not have the sheer number of interactions for a system to learn from, but also the actual

instance of a transition is usually only relevant once. Therefore, we propose to estimate states of individuals assisted by what we have seen from the population.

8.4 SimEdu

In order to address the difficulties of educational assistance, we use a reinforcement learning (RL) framework. RL allows us to address some issues of dynamically changing environments and population shifting, while still providing a computationally tractable approximation to dynamic programming searches. It also provides us with some basic algorithms to benchmark some ideas behind the education problem, highlight some of its difficulties, and understand what could be optimal approaches. SimEdu introduces a combination of

1. A highly configurable simulated education environment. In particular, we allow the configuration of concept structure, courses, student types, and possible interventions. Based on the simulated aspect, we also allow the exploration of the level of hidden information and how much information interventions can provide to the ITS.

2. A list of RL algorithms that demonstrate the use of the simulated education problem and provide solutions, both in a purely data-driven technique and with fixed heuristic-based models. The idea behind the variety of algorithms is for the demonstration of a point, whether it is for performance or explainability.

3. A knowledge tracing mechanism for population modeling. In the population modeling, there are the considerations of modeling student subtypes, using prior knowledge, and its sample efficiency with regard to individual modeling.

A view of the information flow is provided in Figure 8.1.

In order to create the simulation, we draw from the wealth of teaching experience that is available. This allows us to simulate environments using rules-based structures, giving

causal explanations for certain student responses. These experiences give us a prior understanding of how an educational course should function and inform our design decisions in the simulation process.

Ultimately the goal of this simulated environment is to perform randomized experiments where our RL agent is interacting with a simulated human with predictable but stochastic tendencies. This does not necessarily have to be in the education environment, but education has some nice simplifications that we take advantage of in our problem that makes it easily parallel to a real-world situation. Ideally, up to some level of approximation, the simulated environment can directly reflect data that is gathered in the real world and, then, provide an ITS for future students.

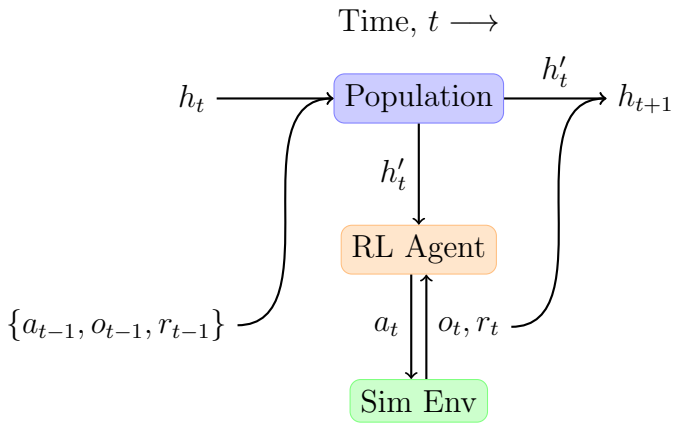


Figure 8.1: A figure showing the time-dynamics of the interactions between the three components of SimEdu: The population model, the RL agent, and the Simulated Environment.

8.4.1 Simulated Environment

First, we introduce SimEdu as a highly-configurable, dynamic time-series environment used for RL probing. The simulated environment gives us a well-defined method of analyzing what conditions are necessary to provide good feedback in the ITS situation.

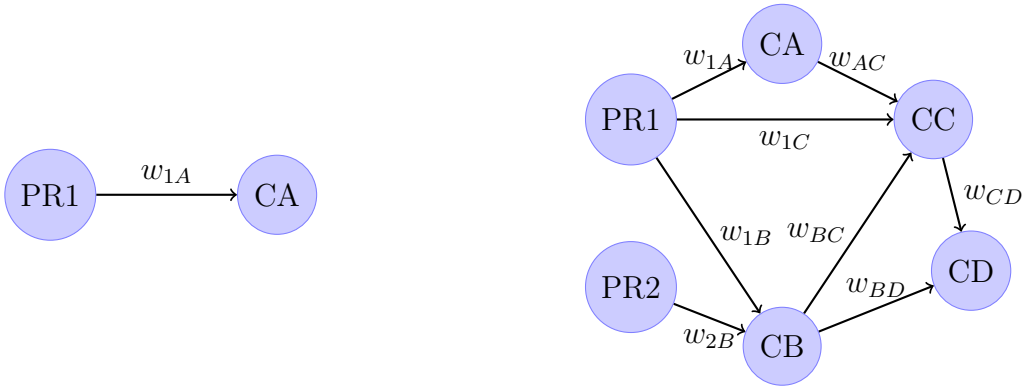


Figure 8.2: Concept DAGs for multi-concept courses (left: one-concept course with prerequisite, right: four-concept course with two prerequisites). Concepts taught in a course are denoted with the precursor C and are denoted by letters (e.g. CA), while prerequisites are denoted with precursor PR and selected via numbers (e.g. PR1).

8.4.1.1 Concept Graphs

Our first assumption about the education problem is that there exists a functional concept graph that guides the educational experience. For example, we normally know that some fundamental understanding of algebra is required before learning calculus, represented as a path in the graph Algebra $\rightarrow$ Calculus. In theory, we can propose a directed acyclic graph (DAG) of the learning concepts of an educational path. In our simulated environment, we primarily deal with linear connections between concepts. This embodies the assumption that there is no way to master a child concept before mastering a parent concept.

In practice, a full educational graph can get highly complex. Instead, we look at potential subgraphs, with the Markovian assumption that knowing the state of the parents of the subgraph, all future interactions are independent of the previous nodes. Two basic educational graphs that we use in our multi-concept courses can be seen in Figure 8.2.

Subgraphs also handle the different levels of granularity concept graphs can have. Within a course, small, individual concepts, *e.g.* factoring, completing the square, and the quadratic formula as methods of solving quadratic equations, are taught one-by-one and may require

specific attention. However, from the perspective of a follow-up course, an overall understanding of the higher-level concept is enough. Following the example, the three methods combine to form an understanding of how to solve a quadratic equation, which is what the next course needs. Even further along, this may even get further abstracted into "algebra understanding."

Any concept DAG will also have some weights $w_{\alpha\beta}$, representing how much the mastery of concept α affects concept β. Suppose we have concept γ. Let $p \in P(\gamma)$ represent the direct parents of γ. Then, the combined concept mastery C'_γ can be represented as:

$$C'_\gamma = \sum_{p \in P(\gamma)} w_{p\gamma} C'_p + \left(1 - \sum_{p \in P(\gamma)} w_{p\gamma}\right) C_\gamma \tag{8.1}$$

where $1 - \sum_{p \in P(\gamma)} w_{p\gamma}$ represents the weight of the concept independent of its parents. This representation, therefore, requires a DAG structure, as it requires the finalized concept mastery for every parent C'_p before C'_γ can be computed. If γ were ever an ancestor of a parent, Equation 8.1 would define a cyclic recursion.

8.4.1.2 Students

Once we have defined a concept graph, we can define students who have some inherent mastery of every relevant concept. The student also contains a couple of additional parameters: the amount of time they are willing to spend in each course time-step and a "motivation" parameter, a valued parameter ranging from 0 to 1 which affects the trajectory of student progress. In this simulation, there are two determining factors of the "student type": the initial state of the concepts (particularly relevant for prerequisite concepts) and their inherent motivation trajectory. This motivation parameter is an abstraction including their actual motivation, inherent study skills, discipline, and other external factors.

As the motivation directly scales the effectiveness of the time spent, we do not also vary the amount of time for the student and include the variation of that abstraction within the

motivation parameter. The time spent directly influences a small amount of immediate time reward regardless of its effectiveness.

For our experiments, we test with several possible trajectories: stable trajectories starting at different levels and trajectories that trend upward and downward to represent students that need a warm-up period and students that burn out throughout a course's time. We also include some small noise to the motivation, based on the possibility of random external factors influencing student motivation.

Students are generated using user-defined distribution specifications. In the current simulation, the specifications include priors about relevant concepts, primarily the prerequisite ones, and prior trajectories for motivation. In our current simulations, we assume that once these priors are set, they are fixed for the course, up to some random noise.

8.4.1.3 Interventions

Once the students are defined, we can define possible interventions. An intervention is any interaction with the student that deals with the concepts or motivation of the student. There are *scheduled interventions*, which are the scheduled lectures and examinations. We mark *time-steps* as the time between scheduled interventions. Then, the *actionable interventions* are the interventions that the ITS has access to that are dynamically chosen based on its observations. These interventions include probing interventions, tutor interventions, and motivation interventions. In our simulation, all actionable interventions have a positive cost associated with them, representing the time that each intervention takes.

Our concept interventions, which constitutes both lectures and tutor interventions, are currently modeled as asymptotic exponential steps, given by

$$\frac{\Delta C}{\Delta t} = k_m(C_{target} - C) \tag{8.2}$$

where C represents the concept mastery, k_m is some constant scaled by both the intervention type and the student's motivation, and C_{target} can be defined per intervention type.

Then, each concept intervention can also give some amount of feedback, where the configured quality of the feedback depends on the type of intervention. In our experiments, lectures provide no feedback, tutor sessions provide a good amount of feedback, and examinations provide a lot of feedback. In our simulation, we define the feedback as some number of i.i.d. Bernoulli samples (Bern(C)).

Our motivation interventions, instead, are discrete steps. We divide the motivation into two main categories, a *study skills* aspect and a *transient motivation* aspect. We have a study-skills intervention that directly improves the study skills of a student permanently, but can only happen once. Then, we have a limited motivation intervention that can improve motivation for a student for a short period of time, in case there is a particular concept that the student needs the extra motivation for. Both interventions currently move the student up one step when active.

Probing interventions provide little to no direct benefit to the student, but provides much more feedback. For our simulations, we offer two types of probes: a realistic probe and an oracle probe. The oracle probe directly elucidates the hidden parameter C that defines the student's concept mastery. One can view this as learning the underlying probability distribution dictating the Bernoulli samples. This type of probe ameliorates the partial observability problem, as it allows the problem to become fully observed, at a cost. The realistic probe, instead, simulates an examination for low cost. This gives the ITS reliable information, but is subject to some random noise. Thus, it cannot give the same quality of information, but it would be more realistic to implement.

8.4.1.4 Course

A course is the list of *scheduled interventions*. We design three types of courses, a *basic one-concept* course, a *prerequisite one-concept* course, and a *four-concept* course. The prerequisite one-concept course and the four-concept course follow a concept graph structurally defined by Figure 8.2.

We design courses to have the following difficulty level:

- The top students (referred to as 'A' students) will pass a course more than 90% of the time with no ITS intervention.
- The bottom students will almost never pass the course without ITS intervention. However, if the student receives significant ITS intervention, they should almost always pass the course.

An ITS can always tutor to help students pass the course, but this would not respect all students' time. Therefore, the goal is to design an ITS that probes for the student type and then determines the amount of aid the student needs to receive.

8.4.2 Population Model

Following the definition of a course, another important aspect to an ITS's success is the population of students coming in. Hidden information is present everywhere in the education problem. We decide that there are primarily two main pieces of student information that will be hidden in the simulation, the student's concept mastery and the student's trajectory, which we abstract as the student's "motivation." These two define the population of students that enter a course.

The student concept mastery refers more to an instantaneous mastery, which can be noisily measured with an evaluation or examination of some kind. On the other hand, the student's motivation cannot be measured instantaneously and could depend on longer-term impacts and events.

8.4.2.1 Simulation

Following the discussion of the simulated environment, the student population is defined with two points: the state of prerequisite concepts at the beginning of the course, and the

overall student type in terms of motivation. Each prerequisite concept is quantized into 4 possible concept masteries, with most students in a natural classroom resting around the passing level of all prerequisite concepts, and 20% students below and above that value.

Everything about the distribution is hidden to the population model and the ITS, except perhaps in how many quantized steps we have split the distribution into.

8.4.2.2 Knowledge Tracing

Knowledge tracing is the task of modeling the time-series nature of student knowledge in order to accurately predict student performance in the future, an inherently difficult problem due to the complexity of humans [PSH15]. We apply Bayesian inference in the form of a Hidden Markov Model (HMM) to provide knowledge tracing. However, we attach an additional component that gives us flexibility in explainability, use of prior knowledge, and exploration noise in reinforcement learning.

When dealing with student concepts, we first discretize the concept understanding into $K = 4$ non-equal buckets. Using these buckets, we can model population information by maintaining initial, transition, and emission probability Dirichlet priors [Str00, OGD22]. The Dirichlet distribution is the conjugate prior of the categorical distribution. This means that sampling from the Dirichlet distribution gives rise to possible categorical probability distributions required by the HMM. The Dirichlet distribution is also flexible, and different parameters can specify distributions with bell-shapes or long tails, depending on necessity. It is also specified with only as many parameters as the number of categories in the probability distributions. Finally, it has an intuitive Bayesian update, where observing a specific interaction can be updated by simply adding a weighted increment to that interaction. Thus, its parameters are always explainable as being the result of past observations and expert priors can be implemented easily.

These give us very explainable ways to both use prior or expert knowledge and interpret

the training as time goes on. We define Dirichlet parameters for each state-action pair. We use the parameters to sample initial, transition, and emission probability distributions in an epoch. However, as the emission probabilities depend only on how we want to define the categories of students, we do not need to learn the distributions. We can set them beforehand as a fixed prior to categorize student progress in each concept. The Dirichlet sampling also immediately provides exploration noise to the reinforcement learning agent, creating some input noise for robustness. From these distributions, we maintain estimated student concept masteries with respect to time, getting both a likelihood of being in each respective bucket, and the maximum likelihood state. As per the traditional HMM step [Rab89],

$$\mathbb{P}\{h_t = k\} = \frac{\sum_{i=1}^{K} \mathbb{P}\{h_{t-1} = i\} P_T(i,k) P_E(k, o_t)}{\sum_{j=1}^{K} \sum_{i=1}^{K} \mathbb{P}\{h_{t-1} = i\} P_T(i,j) P_E(j, o_t)} \tag{8.3}$$

Then, as we update our models, we also update the Dirichlet priors with the state transitions.

Unfortunately, as the Dirichlet parameter method is still fundamentally mostly a tabular approach, this method can be significantly less sample efficient. Therefore, the selection of a relevant state space can heavily affect the training time and effectiveness of our approach in hidden education environments. For example, we have found that the inclusion of the actual value of the time step in the state is not desirable, and logically feels extraneous as doing a specific intervention should not have time-dependent properties.

8.4.2.3 Sub-Population Models

In particular, one of the values in the state to consider is whether to consider subpopulations or student types. Because in our simulation we have defined the population in terms of prerequisite knowledge and motivation, this pair of values can determine a student type. Thus, there is an *initial* student type and a *transitional* student type, which dictates the motivation. In particular, by keeping student type in the model, we can have different Dirichlet parameters for each student type, creating a better understanding of possible student transitions, at the cost of reduced sample efficiency.

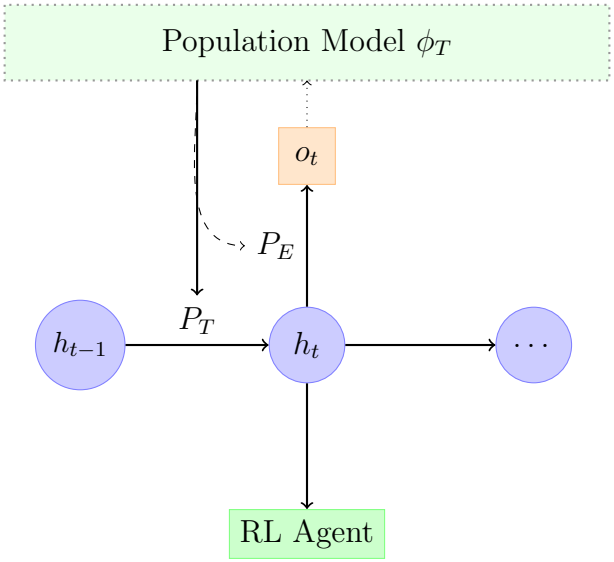


Figure 8.3: Bayesian knowledge tracing via a Dirichlet parameterized sampling technique. The population uses state information (which can possibly come from observations, such as the RL's choice of action) to sample $P_T \sim Dirichlet(\phi_T^{(o_t)})$. After sampling, we proceed with knowledge tracing with standard HMM updates. We can also sample a P_E similarly, but in our case we use P_E as fixed priors.

However, trying to estimate student type is a second-order estimation problem, dependent on accurate estimation in the first order plus having priors in the second order. The estimation of the motivation parameter depends on an accurate estimation of the student concepts, but an accurate estimate of student concepts also depends on the estimation of the motivation parameter, leading to a somewhat cyclical estimation problem. This problem is *intractable*, due to the accumulation of noise. Instead, we assume that students will always take a diagnostic quiz where the student self-assesses their student type. The population model can then use this information to select subpopulations.

While we make the assumption that the population model has some prior information about which student type our student belongs to, this information is not passed to the RL agent, so any information the RL agent can gain can only be through a more accurate

estimate of the student's state. In addition, even if the population model knows which group each student belongs in, it is not given direct information about what the group's expected motivation will be at every time step, or information about the dynamics of how motivation is integrated into the simulation.

8.4.2.4 Experiments

We deal with environments with 3 levels of observability in our experiments. The first is a fully-observed environment, where we have all the oracle information necessary from the environment itself, so that if we allowed an oracle probe intervention for the agent, it would not actually do anything. Next, we deal with environments where only the concept information is hidden. This models an ITS that can ask a student what they believe their motivation is at the start of every session for free. However, the ITS cannot get accurate information about student concepts. Finally, we have an environment where both are hidden from the ITS.

8.4.3 Reinforcement Learning

RL offers a general framework that specifically handles interactions with an unknown environment. As it focuses heavily on interactions, we look to RL as a way to be adaptive to different types of humans while interacting.

However, RL still has many ongoing problems. In particular, unlike supervised learning, RL deals with a non-stationary non-independent dataset, making sample efficiency an important point of discussion. Therefore, RL has difficulty in low-interaction spaces and can have difficulty adjusting to a changing environment. Finally, RL still suffers from similar problems to supervised learning, in that interpretability and performance often have some level of trade-off.

8.4.3.1 Rewards

Overall, the goal that we have set for our ITS is to increase the passing rate of students, while giving students the most amount of free time. Thus, for time step n in an N step course, our reward function is defined as:

$$R_n = \begin{cases} K_{g_n} \times g_n + K_\tau \times \frac{\tau_n}{T_n} & n < N \\ K_{g_n} \times g_n + K_{pass} \times \mathbb{1}_{G \geq G_{pass}} + K_\tau \times \frac{\tau_n}{T_n} & n = N \end{cases} \tag{8.4}$$

where K_{g_n}, K_τ, and K_{pass} refer to the reward weights of the grade reward, time reward, and pass reward respectively. At every time step n, the environment provides a grade reward g_n if a graded intervention takes place, and an immediate time reward, defined by the proportion of time the student has remaining after all the ITS interactions to how much she started with τ_n/T_n. The immediate time reward can be realized as a student having free time to spend on other tasks or enjoyment, or, from the perspective of a tutor, the amount of time available for other students or tasks. At the end of the course we get the final course reward, based on the students' grade $G = \sum_n K_{g_n} g_n$, where we normalize with $\sum_n K_{g_n} + K_{pass} = 1$. For instance, for a one-concept course, we could have $K_{g_N} = 0.4$ (0 for all other values of n), $K_{pass} = 0.6$ and $G_{pass} = 75\%$.

8.4.3.2 Rules-based Strategies

As we use RL to learn some policies, we also want to compare the learned policies to some fixed rules-based policies. These rules-based policies come in several forms. First, we have the policies that do not respond to the student, which are the policy that never tutors and the policy that always tutors if time is available. These policies provide us with baselines for what we expect from non-interactive ITS that we can compare our RL agent and more interactive rules-based strategies with.

Then, we add a layer of "greedy" interactivity. For each possible action of the ITS, we introduce a single interactive conditional for when each action should happen. We introduce

a tutor limit. In this case, we would set the tutor limit to some value higher than G_{pass}, for instance 80% or 85%, such that we tutor until our estimate of the student's concept mastery exceeds this limit. We also introduce study-skills and nudge conditionals for when we perceive that the student's motivation is not maximized. Finally, we probe if the confidence of any concept is below a certain level.

These rules-based strategies provide a stable greedy policy that we have confidence should be fairly successful to students. We call these strategies greedy because oftentimes the goal of the strategy is to simply take an action until it cannot be taken anymore or the condition for the action goes away. However, because of the simplicity of its design, it also provides highly explainable policies. We also note that the rules-based policies only make use of the same information as the model-free DQN and are not directly related to the rules used to design the simulation.

8.5 Experiments

We ran a series of comprehensive experiments involving the manipulation of various adjustable parameters, including motivation dynamics, external factor distributions, and the degree of partial observability.

8.5.1 Baseline Experiments

First, we ran certain baseline experiments so that we could both confirm whether our courses were designed correctly and provide some understanding of the differences among heuristic policies. In these baselines, we typically assume a fully-observed environment. The fully-observed environment gives the ITS full detailed information about the underlying parameters used to compute the noisy observations. Furthermore, we can adjust the student populations to specifically check whether our course performs up to par.

To reiterate, we expect that

- Using a no intervention policy, A students should still pass the class most of the time, but D students should almost never pass the course.
- Using a tutor-only policy, almost every student should pass the course. However, as A students should have passed the course anyway, they should receive a lower total reward.

Results are shown in Table 8.1 and are largely in line with our expectations. "All Students" describes a course with a student population that is within expectations, where most students are B students with smaller percentages on either side. In all cases, without tutoring, A students have a very high pass rate and a high test reward. After applying tutoring interventions, their pass rate goes up marginally, but their test reward always goes down. Similarly, for D students, their pass rate is close to 0 without any interventions, but reaches into 90% when tutored. An interesting point is the definition of "D student" produces few consistently poor students across as courses get more complex, as there are more prerequisite concepts, which can also factor into a student's success. Thus, even though in the four-concept courses, the D students are still performing poorly when tutored every time, the absolute worst students are rare enough to not affect the total number of students much in a real distribution.

8.5.2 Time Reward Experiments

First, we experiment with the reward constants, primarily the time reward constant K_τ. Intuitively, we can understand that as $K_\tau \to 0$, the reward depends only on the grade, and policies will tend toward tutoring as much as possible, as the cost for doing interventions goes to 0. On the flip side, as $K_\tau \to \infty$, it can totally dominate the grade aspect of the reward, and so the policy would no longer attempt to interact with the student whatsoever.

Table 8.2 shows the adaptability of interactive policies in a fully-observed one-concept course. The first two columns serve as baseline comparisons, as they show the expectation

Course Type	Policy	Student Population	Test Reward	Pass Rate
Basic One Concept	No Intervention	All Students	1.0207	84.1%
		A Students	1.1171	97.9%
		D Students	0.4829	3.3%
	Tutor Only	All Students	1.0165	99.8%
		A Students	1.0245	100.0%
		D Students	0.9537	94.7%
Prerequisite One Concept	No Intervention	All Students	1.0228	93.2%
		A Students	1.0832	100.0%
		D Students	0.4208	4.8%
	Tutor Only	All Students	1.0119	100.0%
		A Students	1.0195	100.0%
		D Students	0.9326	94.2%
Four Concept	No Intervention	All Students	0.9632	63.3%
		A Students	1.2057	99.7%
		D Students	0.5267	0.0%
	Tutor Only	All Students	1.0282	99.4%
		A Students	1.0485	100.0%
		D Students	0.6612	46.7%

Table 8.1: A list of design baselines for the three courses that we use.

as we move in either direction for K_τ. As expected, the pass rates of the non-interactive strategies do not change across K_τ and only the test reward changes to reflect the effect of K_τ.

We observe that the DQN is able to adapt to the population differently based on K_τ. For high levels of K_τ, the DQN finds a policy with very high test reward, while finding a pass rate between the no intervention and tutor-only policies. One note is that there is some computational overhead present for extremely low values of K_τ, as the reward does not propagate well into the DQN.

Furthermore, we also include a *tutor limit* rules-based approach, based primarily on our expectation of a good fully-observed policy. The tutor limit policy simply tutors until it reaches some limit, e.g. 85% to be safe, and then stops tutoring. In all but the highest time reward, the rules-based approach actually performs similarly or better than the DQN in both

fronts. This greedy method sacrifices some test reward it can get from the best students for the stability of making sure students always pass, but knows to stop at a certain point to retain a good amount of reward. These results illuminate one big difference between the RL algorithm approaches and the greedy rules-based approaches. The RL algorithms tend to be greedy toward the immediate rewards, procrastinating as much as possible and trying to get as much test reward through immediate rewards. On the other hand, the tutor limit policy does the exact opposite, where it is greedy toward the pass rate and tries for stability in that front.

This implies that having a dynamic understanding of the reward can both be inherently useful and have a visible effect on students. Thus, this allows for the custom selection of how much time a student is willing to spend on a course and gives concrete results as to what the expected results would be for a student. Furthermore, we show that rules-based approaches provide both more stable and explainable solutions with similar results in most cases.

	Policy							
	No Intervention		Tutor Only		DQN		Tutor Limit	
K_τ	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
0.0001	0.800 ± 0.253	80.8%	0.959 ± 0.026	99.9%	0.940 ± 0.010	98.6%	0.941 ± 0.044	99.6%
0.0005	0.811 ± 0.247	81.8%	0.959 ± 0.025	99.9%	0.943 ± 0.015	98.6%	0.944 ± 0.044	99.6%
0.001	0.830 ± 0.236	84.1%	0.961 ± 0.025	99.9%	0.947 ± 0.011	99.0%	0.945 ± 0.056	99.3%
0.002	0.830 ± 0.244	82.6%	0.964 ± 0.026	99.9%	0.949 ± 0.009	98.4%	0.954 ± 0.033	99.8%
0.005	0.855 ± 0.249	81.7%	0.969 ± 0.053	99.4%	0.969 ± 0.006	98.8%	0.972 ± 0.045	99.6%
0.01	0.901 ± 0.252	81.1%	0.984 ± 0.054	99.4%	0.999 ± 0.007	98.4%	1.003 ± 0.054	99.4%
0.02	0.994 ± 0.258	80%	1.017 ± 0.036	99.8%	1.067 ± 0.009	97.7%	1.067 ± 0.044	99.7%
0.05	1.309 ± 0.245	82.5%	1.106 ± 0.061	99.6%	1.318 ± 0.020	95.3%	1.255 ± 0.075	99.4%

Table 8.2: Time Reward Experiments (One Concept)

One thing in particular shows a clear difference between the RL method and the rules-based method is when they tend to tutor. The rules-based approach is primarily greedily improving student concept mastery, up to a limit. On the other hand, the DQN (and many

other RL algorithms) will tend to be greedy for time (gaining the instantaneous rewards).

8.5.3 Partial Observability Experiments

We apply different approaches to showcase the difficulty of partial observability and the effect of probing. Here, we use three different types of observability to make the problem more difficult. Concept-hidden environments hide the status of the concept mastery for the student from the ITS, but allow the ITS to see the motivation (based on some assumption that the student will tell the ITS at the beginning of each section). The unobserved environment hides both parameters. These are all done on the basic one concept setting with $K_\tau = 0.02$.

We introduce the Study Skills (SS) heuristic strategies to indicate that the rules will attempt a study skills and/or nudge intervention when it deems that it would be helpful. The DQN class has access to these interventions, except potentially for the probe and the oracle probe intervention. In addition, when using any partially observed information environment, our policies will attach the relevant population model to track hidden information. Keep in mind that the tutor-only policy in this case does not need any population information here, as there is only a single concept to tutor.

Table 8.3 shows the results of multiple rules-based and DQN policies on these types of environments across differing population types. We include both low-entropy population types (single types of students) and high-entropy populations (an extreme AD student distribution, which just has 50% A students and 50% D students) to get an idea for how much probing helps across the board.

In the first row of Table 8.3, we have the baseline effectiveness of the probe. The probe allows for accurate estimation with the population model, allowing for a limit in tutoring. There are two conclusions to be made. Across the board, the probe allows for better test reward, highlighting its effectiveness. However, the probes still do eat into tutoring time, showing some small pass rate drops for harder student distributions, yet test reward still

		Policy Type					
		No Intervention		Tutor Only		Probe Tutor Limit	
Exp. Type	Population Type	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Fully Observed	Typical	1.0056	81.9%	1.0146	99.5%	1.0503	99.7%
	A Students	1.1170	97.9%	1.0226	100.0%	1.0721	100.0%
	D Students	0.4790	2.9%	0.9518	94.2%	0.9536	94.5%
	AD Students	0.7937	49.8%	0.9925	97.9%	1.0099	96.7%
Concept Hidden	Typical	0.9927	79.8%	1.0166	99.8%	1.0266	99.6%
	A Students	1.1151	97.6%	1.0237	100.0%	1.0451	100.0%
	D Students	0.4805	3.1%	0.9463	93.5%	0.9439	92.6%
	AD Students	0.7861	48.4%	0.9858	96.9%	0.9908	95.8%
Unobserved	Typical	1.0028	81.4%	1.0152	99.7%	1.0263	99.4%
	A Students	1.1174	97.8%	1.0255	100.0%	1.0454	100.0%
	D Students	0.4852	3.8%	0.9483	93.6%	0.9443	92.5%
	AD Students	0.7825	47.9%	0.9901	97.4%	0.9933	96.2%

		Policy Type					
		SS Tutor		Probe SS Tutor Limit		Oracle SS Tutor Limit	
Exp. Type	Population Type	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Fully Observed	Typical	1.0168	99.8%	1.0657	99.7%	1.0535	100.0%
	A Students	1.0247	100.0%	1.0875	100.0%	1.0713	99.9%
	D Students	1.0058	100.0%	1.0312	99.2%	1.0169	99.7%
	AD Students	1.0152	100.0%	1.0557	99.1%	1.0414	99.7%
Concept Hidden	Typical	1.0188	100.0%	1.0614	99.5%	1.0282	99.9%
	A Students	1.0234	100.0%	1.0813	99.9%	1.0455	100.0%
	D Students	1.0062	100.0%	1.0089	95.7%	1.0090	99.6%
	AD Students	1.0144	99.9%	1.0472	98.2%	1.0264	99.8%
Unobserved	Typical	1.0158	99.8%	1.0585	99.6%	1.0052	99.9%
	A Students	1.0201	100.0%	1.0710	100.0%	1.0176	100.0%
	D Students	0.9979	99.5%	1.0016	95.8%	0.9677	97.5%
	AD Students	1.0085	99.5%	1.0355	98.0%	0.9962	99.3%

		Policy Type					
		DQN No Probe		DQN Probe		DQN All	
Exp. Type	Population Type	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Fully Observed	Typical	1.0608	92.1%	1.0676	93.2%	0.9566	99.7%
	AD Students	1.0053	96.8%	1.0106	99.9%	1.0121	99.8%
Concept Hidden	Typical	1.0708	93.8%	1.0471	98.1%	1.0718	93.8%
	AD Students	1.0122	100.0%	0.9999	97.9%	1.0313	98.4%
Unobserved	Typical	1.0644	96.4%	1.0597	92.1%	1.0571	99.7%
	AD Students	1.0196	99.5%	0.9885	97.2%	1.0438	95.3%

Table 8.3: Hidden Information Experiments Across a Variety of Policies.

improves in spite of the pass rate drop.

Comparing the first and the second column shows the effectiveness of having study skills interventions compared to simply tutoring. Again, looking across the second column shows the effectiveness of probing. Interestingly, the oracle probe, even if at the same cost as a regular probe, will result in lower test reward, but higher pass rate, for the unobserved experiments. This is a reflection of its increased confidence in the student parameters, allowing it to more confidently reach the limit threshold of 85%. However, this means that the oracle probe will have possibly used more tutors than necessary to account for the safety window.

Then, we compare the second and third rows of Table 8.3, as they have comparable intervention lists. The DQN policies are capable of finding policies that have similar in test reward to the heuristic methods, but usually with lower pass rates. However, with the same test reward, we typically prefer to have the higher pass rate.

One final interesting result is that the DQN can produce highly variable results, especially with the addition of the probing and oracle interventions. The maximal result is presented in Table 8.3, but a more in-depth analysis is presented in Appendix D.1. Given all this, we see that there is no significant benefit of using RL in the complexities of a partially observed environment compared to our simple heuristic rules-based methods.

8.5.4 Distributional Shift Experiments

Here, we provide some insight into the robustness of the policies. We set up several experiments using the results we obtained in the previous experiment, specifically the population models and policies trained on the typical distribution and the one trained on the *AD* student distribution in the completely unobserved environments. We test these models under both of the two original distributions, and one other distribution containing 25% A students and 75% D students. This 25/75 distribution is a harder class in terms of necessary tutor

interventions, so a lower test reward is expected, but it is slightly easier in terms of probing. Because the *AD* population is closer to the 25/75 distribution, we expect that shift to be easier.

Because we have both the population model and the policy as two independent components, we test situations where only a single one is mismatched. Table 8.4 shows the results of these distributional shifts. Immediately, the DQN shows to be more brittle to distributional change than the heuristic method. However, there is some nuance to the observation.

1. In cases where the population model is different from the test population, but the policy is the same, all policies perform only slightly worse than the in-distribution performance. This simulates a situation where a teacher may have some incorrect preconceived notions of the population of students, but follows a policy left behind by a previous teacher. This teacher performs almost similarly well to those who have trained heavily in that distribution.

2. On the other hand, when a teacher tries to force a policy trained on one distribution of students to a different one, things are not as stable. When going from a more difficult student distribution to an easier one (*e.g.* from the *AD* students to the typical student distribution), the teacher still performs well, but sacrifices some time required. On the other hand, when going from an easier student population to a harder one, the policy performs significantly worse than a trained model. These results are exacerbated with the 25/75 distribution for the policy trained in a typical environment, but not for the one trained in the 50/50 *AD* environment.

These results show both the flexibility of the simulation and the heuristic models to distributional shifts in students. DQN shows some flexibility in these environments, but cannot perform well going from an easy student distribution to a harder one.

SS Probe Tutor Limit		Test Population					
		Typical Students		*AD* Students		25/75 *AD* Students	
Population Model	Policy Distribution	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Typical	Typical	**1.0585**	**99.6%**	1.0113	99.6%	0.9995	98.6%
AD Students	*AD* Students	1.0211	100.0%	**1.0355**	**98.0%**	1.0056	99.4%
Typical	*AD* Students	1.0196	99.8%	1.0097	99.3%	1.0030	99.1%
AD Students	Typical	1.0203	99.9%	1.0106	99.4%	1.0056	99.3%
DQN		Test Population					
		Typical Students		*AD* Students		25/75 *AD* Students	
Population Model	Policy Distribution	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Typical	Typical	**1.0644**	**96.4%**	0.8988	71.4%	0.8039	58.2%
AD Students	*AD* Students	1.0278	99.9%	**1.0196**	**99.5%**	1.0101	99.0%
Typical	*AD* Students	1.0286	99.9%	1.0161	99.0%	1.0092	98.7%
AD Students	Typical	1.0649	96.3%	0.8918	70.4%	0.7940	56.7%

Table 8.4: Distributional Shift Experiments. The table shows test reward results and pass rate of students when changing the population model, the policy, or both. Bolded values are in-distribution results.

8.5.5 Structure Experiments

Finally, based on our observations on the effect of probing interventions, we suspect that we can also encode improved probing directly into the structure of courses. For the purposes of the experiment, we design courses assuming the ITS has limited probing capabilities, as many courses are designed today.

Referencing Figure 8.4, we create 4 different course structures. The first has only finals, testing all 6 concepts in the four-concept course at the end F. The second has a midterm-final structure, testing PR1, PR2 and CA at M and CB, CC, and CD at F. Finally, we have a quiz structure that uses all 4 examinations Q_1, M, Q_2, and F. The final one is an extension of the quiz structure with an additional diagnostic quiz about halfway into the

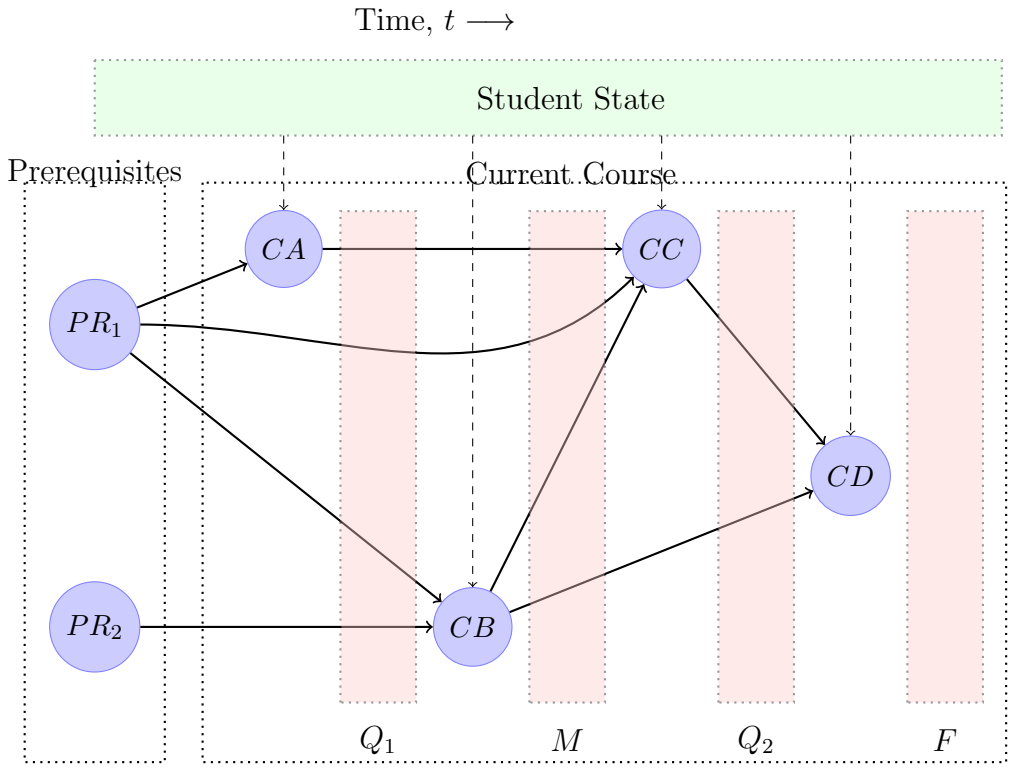


Figure 8.4: The time-dynamic DAG structure of the four-concept course with the concept DAG referenced in 8.2. The structure experiments are defined based on which subset of evaluations (Q_1, M, Q_2, and F) are present.

learning of every concept (and at the beginning of review sections for prerequisite material). The understanding is that with more examinations, the ITS and instructor will have more information about the student, to become capable of adjusting their understanding of the students for more individualized plans. However, the more frequent examinations increase the urgency of tutoring, as the earlier grades can affect the overall total grade in the course. In order to fully align our goals with the balance of probing, we use the AD distribution to maximize the difference between student types.

Table 8.5 shows results with these four different structures, tested on a random policy, a study-skills tutor-limit policy, and a DQN policy, each without the ability to probe. The

random policy effectively shows the difficulty of the class, showing that in our simulation, the difficulty of the class increases with the complexity of the course structure. This is somewhat realistic, in that students who are time-limited lose some flexibility in when they can study for their course. However, this is also a limitation of the design of our system. Students do not forget in this environment, so students are never better off on an exam being tested earlier rather than later.

Moving on to the heuristic and the learned policies, we see that they both achieve much higher test rewards and pass rates compared to the random policy. Overall, with the four concept policy, we see that the DQN's test reward is higher than that of the heuristic policy, but with a lower pass rate. Most importantly, both policies are able to pull the pass rate of the quiz and midterm-final structures up in line with the finals-only structures. While the policies are unable to allow them to surpass the finals-only structure, the amount of improvement showcases the utility of the extra information available in other structures in improving performance. Furthermore, the diagnostics provided in the diagnostic structure does provide a small boost in both methods.

	Random		SS Tutor Limit		DQN	
Course Structure	Test Reward	Pass Rate	Test Reward	Pass Rate	Test Reward	Pass Rate
Finals Only	1.0101	92.4%	1.0235	99.7%	1.0782	97.4%
Midterm-Final	0.9732	86.9%	1.0171	99.1%	1.0575	96.5%
Quizzes	0.9444	82.6%	1.0145	98.8%	1.0453	97.8%
Quiz + Diags	0.9596	85.0%	1.0164	98.8%	1.0659	94.7%

Table 8.5: Four-Concept Structure Experiments.

8.6 Conclusion

In this chapter, we develop SimEdu, a realistic simulation environment, population model, and policy, to explore adaptive assistance of student learning. The system captures the hidden information of learning, the trade-offs in different types of interventions, and the value of probing in the context of a course. We find that for such systems, a deep RL agent performs well, but not significantly better than certain rules-based heuristic systems. It is shown that partial observability directly correlates to problem difficulty and thus, highlights the importance of probing. With the importance of probing, we also show that course structures that encourage more information gain show improvements in student performance. In this simulation, we find that RL does not perform significantly better than well-designed heuristic methods in these hidden information environments, especially in terms of flexibility and explainability.

There are still many ways to extend the work presented in this chapter. There are several extensions just to this simulated environment. Many of the design decisions in the simulation, though complex already, are made to simplify estimation as much as possible. First, one can extend the population model to have different levels of representation, explainability, and trainability. For instance, applying [PSH15] could provide a much improved population model while trading off explainability. We can add some more overarching dynamics, such as forgetting, concept reinforcement via contextual reuse, and concept improvement dependencies across prerequisites.

While the simulation has broad coverage over many hypothetical scenarios, we can use pre-existing observational data to make the simulation more realistic. Further mathematical or data-driven models can be explored that more closely align with students' true concept graphs, trajectories, and conditional dependencies of traits. One exciting extension is unlocked with a continuous stream of education data, either through observing experiments or from direct interactions with students. The SimEdu framework can be adjusted to use

flexibly, where the simulation aspect is also adapting to match the real-world data. This can give rise to a structural hypothesis approach taken in Chapter 5, which allows for an unsupervised method of fitting, simulating, understanding, and intervening. This can be used to understand the dynamics of the real-world education system, and to hypothesize and test interventions for future students.

Plus, education is only one of the many time-series domains that are increasingly aided with computers. Many of the ideas here, though often with different names, can extend to other domains, such as medicine, finance, psychology, personal assistance, and other areas of computer interactions.

CHAPTER 9

Conclusions

In this thesis, we have focused on the use of causal modeling to improve the robustness of machine learning models. Structural causal models attempt to instill a deeper, human understanding of the world to machine learning models.

In this research, we have made three significant findings that contribute to our understanding and application of machine learning techniques, particularly in the areas of Variational Autoencoders (VAEs), energy-based models, and education simulations.

Firstly, we discover that it is possible to impose substantial structure on the latent space of the VAE. This structure enables us to generate data out of distribution, test structural hypotheses, and produce augmentations and their compositions in the latent space. This finding opens up new possibilities for using VAEs in a variety of applications, including causal data augmentation and causal hypothesis testing.

Secondly, we identify a state-of-the-art technique using the unsupervised learning approach of the energy-based model. Remarkably, this technique can defend against several poisoning techniques without requiring excessive additional training time and without significant reduction in test accuracy. This finding has important implications for the security and efficiency of machine learning models.

Lastly, we develop an education simulation that attempts to discretize and understand human interactions with machines. By using causal information to create the simulation, we enable both an understanding of how to design real-world education experiments and an understanding of what are the challenges and difficulties that apply to a dynamic ITS

in an education problem. In our simulation, we find that heuristic methods still perform comparably to deep learning techniques. This finding suggests that heuristic methods can still be effective in certain applications, despite the recent advances in deep learning.

However, as with any research, our study has limitations and raises new questions. Future research could further explore the potential applications of structured VAEs, investigate how poison defenses can interact with other domains, *e.g.* text and additional image domains, and improve the realism of our education simulation so that our conclusions have wider reach or extend the ideas to other similar human interactions. Despite these challenges, we believe that our findings provide a solid foundation for future research in these areas.

The construction of accurate generative simulations of the real world is a difficult task and can have many future implications. With such simulations, a system can begin to hypothesize different outcomes, intervene in order to improve future outcomes, and explain such decisions. Further, in a system that can hypothesize and re-evaluate its own causal models, the system begins to understand the higher levels of causality. This can lead to a more general approach to artificial intelligence, a better understanding of the possibilities of machine learning, and a more cooperative result for human interactions.

REFERENCES

[ACG16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy." In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[AJB17] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. "A Closer Look at Memorization in Deep Networks.", 2017.

[AMW21] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability." In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 159–178. IEEE, 2021.

[AWN22] Ghodai Abdelrahman, Qing Wang, and Bernardo Pereira Nunes. "Knowledge Tracing: A Survey.", 2022.

[Bal23] Tysam& Balsam. "hlb-CIFAR10.", 2023. Released on 2023-02-12.

[BCF21] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. "Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff." In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3855–3859. IEEE, 2021.

[BGS76] Giancarlo Benettin, Luigi Galgani, and Jean-Marie Strelcyn. "Kolmogorov entropy and numerical experiments." *Phys. Rev. A*, **14**:2338–2345, Dec 1976.

[BJP22] Sunay Bhat, Jeffrey Jiang, Omead Pooladzandi, and Gregory Pottie. "De-Biasing Generative Models using Counterfactual Methods.", 2022.

[BJS21] Ioana Bica, Daniel Jarrett, and Mihaela van der Schaar. "Invariant Causal Imitation Learning for Generalizable Policies." In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pp. 3952–3964. Curran Associates, Inc., 2021.

[BKB21] Boris van Breugel, Trent Kyono, Jeroen Berrevoets, and Mihaela van der Schaar. "DECAF: Generating Fair Synthetic Data Using Causally-Aware Generative Networks.", 2021.

[CCB19] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. "Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering." In *SafeAI@ AAAI*, 2019.

[CGK22] Spencer Compton, Kristjan Greenewald, Dmitriy A Katz, and Murat Kocaoglu. "Entropic Causal Inference: Graph Identifiability." In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4311–4343. PMLR, 17–23 Jul 2022.

[CLG18] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. "Isolating Sources of Disentanglement in Variational Autoencoders.", 2018.

[CLL17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. "Targeted backdoor attacks on deep learning systems using data poisoning." *arXiv preprint arXiv:1712.05526*, 2017.

[CSL08] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. "Casting out demons: Sanitizing training data for anomaly sensors." In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 81–95. IEEE, 2008.

[DCA18] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. "CINIC-10 is not ImageNet or CIFAR-10.", 2018.

[FMK21] Shi Feng, Alejandra J. Magana, and Dominic Kao. "A Systematic Review of Literature on the Effectiveness of Intelligent Tutoring Systems in STEM." In *2021 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, 2021.

[FWH23] Bisni Fahad Mon, Asma Wasfi, Mohammad Hayajneh, Ahmad Slim, and Najah Abu Ali. "Reinforcement Learning in Education: A Literature Review." *Informatics*, **10**(3), 2023.

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain." *arXiv preprint arXiv:1708.06733*, 2017.

[GFH21] Jonas Geiping, Liam H Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. "Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching." In *International Conference on Learning Representations*, 2021.

[GFS21] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. "What Doesn't Kill You Makes You Robust (er): Adversarial Training against Poisons and Backdoors." *arXiv preprint arXiv:2102.13624*, 2021.

[GKC17] Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. "Causal Generative Neural Networks.", 2017.

[GQ10] Adam N. Glynn and Kevin M. Quinn. "An Introduction to the Augmented Inverse Propensity Weighted Estimator." *Political Analysis*, **18**(1):36–56, 2010.

[Hay12] Simon Haykin. "Cognitive Dynamic Systems: Perception-action Cycle, Radar and Radio." *Cognitive Dynamic Systems: Perception-Action Cycle, Radar, and Radio*, 06 2012.

[HCK20] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. "On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping." *arXiv preprint arXiv:2002.11497*, 2020.

[HGF20] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. "MetaPoison: Practical General-purpose Clean-label Data Poisoning." *Advances in Neural Information Processing Systems*, **33**, 2020.

[HMP17] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[HMZ21] Mitch Hill, Jonathan Craig Mitchell, and Song-Chun Zhu. "Stochastic Security: Adversarial Defense Using Long-Run Dynamics of Energy-Based Models." In *International Conference on Learning Representations*, 2021.

[JE19] Bargav Jayaraman and David Evans. "Evaluating differentially private machine learning in practice." In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1895–1912, 2019.

[JPB22] Jeffrey Jiang, Omead Pooladzandi, Sunay Bhat, and Gregory Pottie. "Causal Structural Hypothesis Testing and Data Generation Models.", 2022.

[KBP13] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." *The International Journal of Robotics Research*, **32**(11):1238–1274, 2013.

[KCW22] Nan Rosemary Ke, Silvia Chiappa, Jane Wang, Anirudh Goyal, Jorg Bornschein, Melanie Rey, Theophane Weber, Matthew Botvinic, Michael Mozer, and Danilo Jimenez Rezende. "Learning to Induce Causal Structure.", 10 2022. arXiv:2204.04875 [cs, stat].

[KF15] James Kulik and J. D. Fletcher. "Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review." *Review of Educational Research*, **86**, 04 2015.

[KMM20] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. "Captum: A unified and generic model interpretability library for PyTorch.", 2020.

[KRW07] Rohit Kumar, Carolyn Rosé, Yi-Chia Wang, Mahesh Joshi, and Allen Robinson. "Tutorial Dialogue as Adaptive Collaborative Learning Support." pp. 383–390, 01 2007.

[KSD17] Murat Kocaoglu, Christopher Snyder, Alexandros G. Dimakis, and Sriram Vishwanath. "CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training.", 2017.

[KW13] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes.", 2013.

[LF20] Alexander Levine and Soheil Feizi. "Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks." In *International Conference on Learning Representations*, 2020.

[Li15] Xi-Lin Li. "Preconditioned Stochastic Gradient Descent.", 2015.

[Li18a] Xi-Lin Li. "Online Second Order Methods for Non-Convex Stochastic Optimizations.", 2018.

[Li18b] Xi-Lin Li. "Preconditioned Stochastic Gradient Descent." *IEEE Transactions on Neural Networks and Learning Systems*, **29**(5):1454–1466, may 2018.

[Li19] Xi-Lin Li. "Preconditioner on matrix Lie group for SGD." In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Li22] Xilin Li. "Black Box Lie Group Preconditioners for SGD." *arXiv preprint arXiv:2211.04422*, 2022.

[LLB03] Ying-Cheng Lai, Zonghua Liu, Lora Billings, and Ira B. Schwartz. "Noise-induced unstable dimension variability and transition to chaos in random dynamical systems." *Phys. Rev. E*, **67**:026210, Feb 2003.

[LLK21] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. "Anti-backdoor learning: Training clean models on poisoned data." *Advances in Neural Information Processing Systems*, **34**, 2021.

[LMA17] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. "Trojaning attack on neural networks.", 2017.

[LXZ20] Xiao Li, Hanchen Xu, Jinming Zhang, and Hua hua Chang. "Deep Reinforcement Learning for Adaptive Learning Systems.", 2020.

[LYM23] Tian Yu Liu, Yu Yang, and Baharan Mirzasoleiman. "Friendly Noise against Adversarial Noise: A Powerful Defense against Data Poisoning Attacks.", 2023.

[min21] "National Study of Learning Mindsets.", Jul 2021.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957*, 2018.

[MKS13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning.", 2013.

[MMS18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks." In *International Conference on Learning Representations*, 2018.

[MSH21] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. "Learning to Generate Noise for Multi-Attack Robustness.", 2021.

[MZH19] Yuzhe Ma, Xiaojin Zhu Zhu, and Justin Hsu. "Data Poisoning against Differentially-Private Learners: Attacks and Defenses." In *International Joint Conference on Artificial Intelligence*, 2019.

[NHH20] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. "On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

[NZF19a] Ignavier Ng, Shengyu Zhu, Zhuangyan Fang, Haoyang Li, Zhitang Chen, and Jun Wang. "Masked Gradient-Based Causal Structure Learning.", 2019.

[NZF19b] Ignavier Ng, Shengyu Zhu, Zhuangyan Fang, Haoyang Li, Zhitang Chen, and Jun Wang. "Masked Gradient-Based Causal Structure Learning.", 2019.

[OGD22] Kaan Ozkara, Antonious M. Girgis, Deepesh Data, and Suhas Diggavi. "A Generative Framework for Personalized Learning and Estimation: Theory, Algorithms, and Privacy.", 2022.

[OVK17] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning." *CoRR*, **abs/1711.00937**, 2017.

[PDM22] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. "Adaptive Second Order Coresets for Data-efficient Machine Learning.", 2022.

[Pea09a] Judea Pearl. "Causal inference in statistics: An overview." *Statistics Surveys*, **3**(none):96 – 146, 2009.

[Pea09b] Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

[PGH20] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. "Deep k-NN defense against clean-label data poisoning attacks." In *European Conference on Computer Vision*, pp. 55–70. Springer, 2020.

[PL24] Omead Pooladzandi and Xi-Lin Li. "Curvature-Informed SGD via General Purpose Lie-Group Preconditioners.", 2024.

[PSH15] Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. "Deep Knowledge Tracing.", 2015.

[PSO22] Ethan Prihar, Manaal Syed, Korinn Ostrow, Stacy Shaw, Adam Sales, and Neil Heffernan. "Exploring Common Trends in Online Educational Experiments." In Antonija Mitrovic and Nigel Bosch, editors, *Proceedings of the 15th International Conference on Educational Data Mining*, pp. 27–38, Durham, United Kingdom, July 2022. International Educational Data Mining Society.

[Rab89] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE*, **77**(2):257–286, 1989. errata at http://alumni.media.mit.edu/ rahimi/rabiner/rabiner-errata/rabiner-errata.html.

[RBL22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

[SAS21] Abhishek Sinha, Kumar Ayush, Jiaming Song, Burak Uzkent, Hongxia Jin, and Stefano Ermon. "Negative Data Augmentation.", 2021.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2018.

[SGF21] Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch." *arXiv preprint arXiv:2106.08970*, 2021.

[SGG21] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. "Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks.", 2021.

[SHM16] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the game of Go with deep neural networks and tree search." *Nature*, **529**:484–503, 2016.

[SHN18] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks.", 2018.

[SHS17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.", 2017.

[SIB22] Nabeel Seedat, Fergus Imrie, Alexis Bellot, Zhaozhi Qian, and Mihaela van der Schaar. "Continuous-Time Modeling of Counterfactual Outcomes Using Neural Controlled Differential Equations." In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19497–19521. PMLR, 17–23 Jul 2022.

[SKL17] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. "Certified Defenses for Data Poisoning Attacks.", 2017.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models." In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[SSP19] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. "Hidden Trigger Backdoor Attacks.", 2019.

[Str00] Malcolm J. A. Strens. "A Bayesian Framework for Reinforcement Learning." In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, p. 943–950, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[SW18] Jianlin Su and Guang Wu. "f-VAEs: Improve VAEs with Conditional Flows." *CoRR*, **abs/1809.05861**, 2018.

[TFY21] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. "Better safe than sorry: Preventing delusive adversaries with adversarial training." *Advances in Neural Information Processing Systems*, **34**, 2021.

[TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. "Spectral signatures in backdoor attacks." In *Advances in Neural Information Processing Systems*, pp. 8000–8010, 2018.

[TTM18] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Clean-label backdoor attacks.", 2018.

[VBC19] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II." `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[WXK20] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. "Rab: Provable robustness against backdoor attacks." *arXiv preprint arXiv:2003.08904*, 2020.

[XLZ16] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. "A theory of generative convnet." In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 2635–2644, 2016.

[YCG19] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. "DAG-GNN: DAG Structure Learning with Graph Neural Networks.", 2019.

[YLC20] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. "CausalVAE: Structured Causal Disentanglement in Variational Autoencoder.", 2020.

[YLM22] Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. "Not All Poisons are Created Equal: Robust Training against Data Poisoning.", 2022.

[ZAR18] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. "DAGs with NO TEARS: Continuous Optimization for Structure Learning.", 2018.

[ZHL19] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Transferable Clean-Label Poisoning Attacks on Deep Neural Nets." In *International Conference on Machine Learning*, pp. 7614–7623, 2019.

[ZNC19] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. "Causal Discovery with Reinforcement Learning.", 2019.

[ZPJ22] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. "Narcissus: A practical clean-label backdoor attack with limited information." *arXiv preprint arXiv:2204.05255*, 2022.

[ZTD20] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha C. Dvornek, Xenophon Papademetris, and James S. Duncan. "AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients." *CoRR*, **abs/2010.07468**, 2020.

APPENDIX A

CSHTest and CSVHTest: Causal Generative Hypothesis Testing

A.1 DAG Simulation Results

A.1.1 DAG Size 5x5: Linear

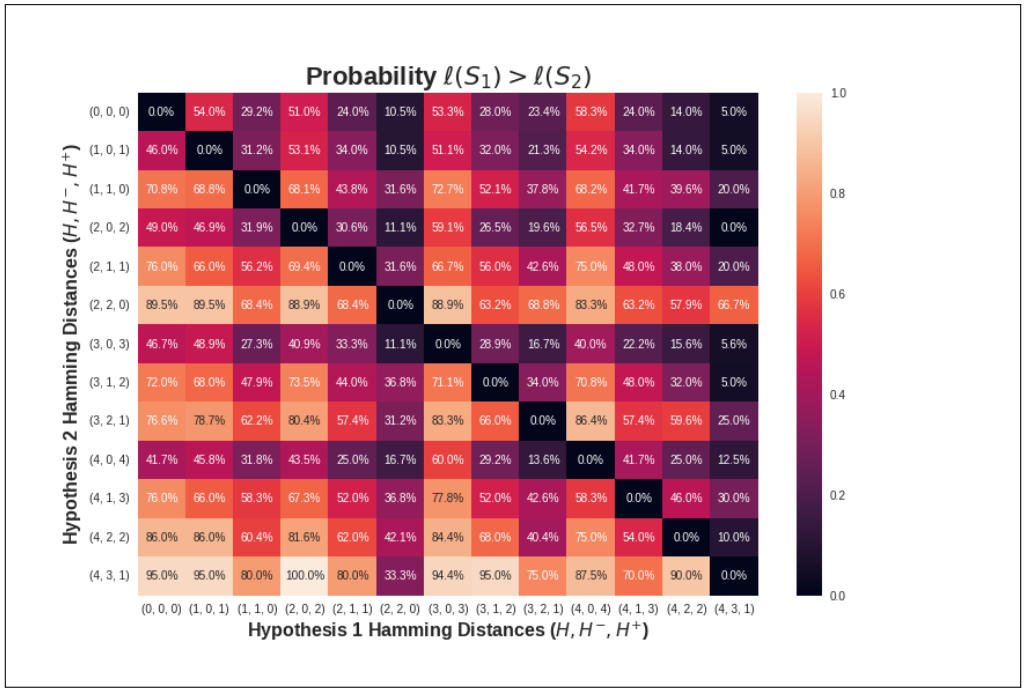


Figure A.1: Probability table for a 5 node 5 edge DAG size with a linear SEM ground truth model for DAG simulations comparing hypothesis with various Hamming Distance Tuples.

A.2 More about the Pendulum

A.2.1 Pendulum Training Architecture and Hyperparameters

Unfortunately, there is a limitation with regard to hyperparameters. Because we are effectively using the average loss over several random initializations onto the generalization set as the primary proxy of SCM "goodness," the hyperparameters that we choose to represent the η neural networks do have possible effects in our overall methods.

- 50 Epochs. 40 Iterations.
- Causal η networks: $[4, 16, 4]$ nodes per output.
- For CSVHTest, Encoder and Decoder networks: $[4, 4]$ node MLP per input.
- Normally-distributed initialization.
- Activation: Soft Leaky ReLU.
- Optimizer: PSGD (discussed further in A.4). Initial Learning Rate of 0.01.
- Cosine Annealing Scheduler. Warm Restarts implemented for non-variational models.

A.2.2 Further Pendulum Results

Full numerical results of Figure A.1. Train and Loss trajectory curves are shown in Figures A.2 and A.3.

A.3 Background Theory

A.3.1 Variational Hypothesis testing and Data Generation with CSVHTest

We extend CSHTest to a variational model CSVHTest, that includes sampling functionality like a VAE [KW13]. Thus, CSVHTest can generate new data points that are not

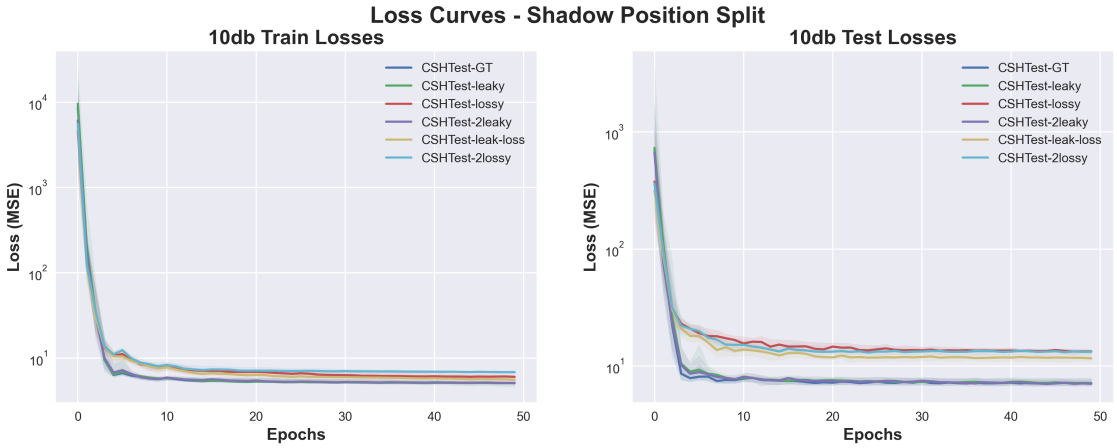


Figure A.2: Loss trajectory of the Sun Split OOD Run

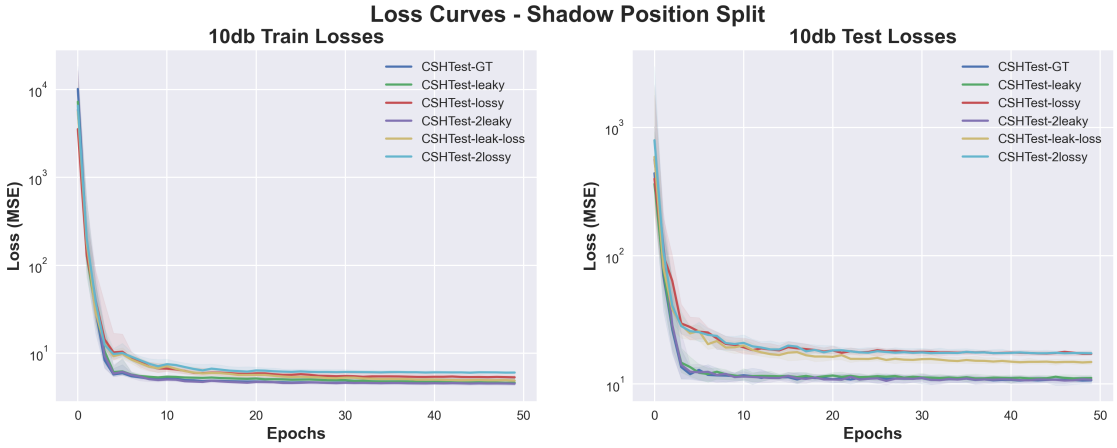


Figure A.3: Loss trajectory of the Shadow Position Split OOD Run

Hypothesis	GT	lossy	leaky	2lossy	2leaky	leak-loss
Sun	7.14 ± 0.92	13.48 ± 1.06	7.41 ± 0.98	13.26 ± 0.92	7.21 ± 0.79	11.66 ± 1.15
Shadow Position	10.63 ± 1.09	17.29 ± 1.16	11.02 ± 0.93	17.56 ± 0.60	10.98 ± 2.16	14.75 ± 1.08

Table A.1: Mean and Standard Deviation of the Final Test Loss in Pendulum Hypothesis Testing Experiments.

deterministic on the inputs, allowing for synthetic data generation. CSVHTest consists of an encoder, a CSHTest causal layer and a decoder.

CSVHTest uses the same encoder and decoder networks as CSHTest defined in Chapter 4.3.3. Further enforcing the label spaces, we can define a prior $p(\mathbf{z}|\mathbf{u})$. We use the same conventions as in [YLC20] and say that

$$p(\mathbf{z}|\mathbf{u}) \sim \mathcal{N}(\mathbf{u}_n, \mathbf{I})$$

where $\mathbf{u}_n \in [-1, 1]$ are normalized label values. This translates to an additional KL-loss.

These encoder and decoder networks do not compress the data, but enable a transformation of the inputs to a normally distributed space, enabling sampling without preventing the relevance of the causal priors given by $\mathbf{S}_i$. Thus, CSVHTest is an extension of CSHTest with

$$\boldsymbol{z}_i = f_{enc}(\boldsymbol{x}_i), \quad \hat{\boldsymbol{z}}_i = \eta_i(\mathbf{S}_i \circ \boldsymbol{z}), \quad \hat{\boldsymbol{x}}_i = f_{dec}(\hat{\boldsymbol{z}}_i) \tag{A.1}$$

The loss function for CSVHTest includes a weighted Kullback–Leibler (KL) divergence loss to normalize the latent space on top of the reconstruction loss as in CSHTest. We also add a weighted latent reconstruction loss for the embedded CSHTest which enforces separation of the encoder and decoders as transformations, and the η networks as the functional

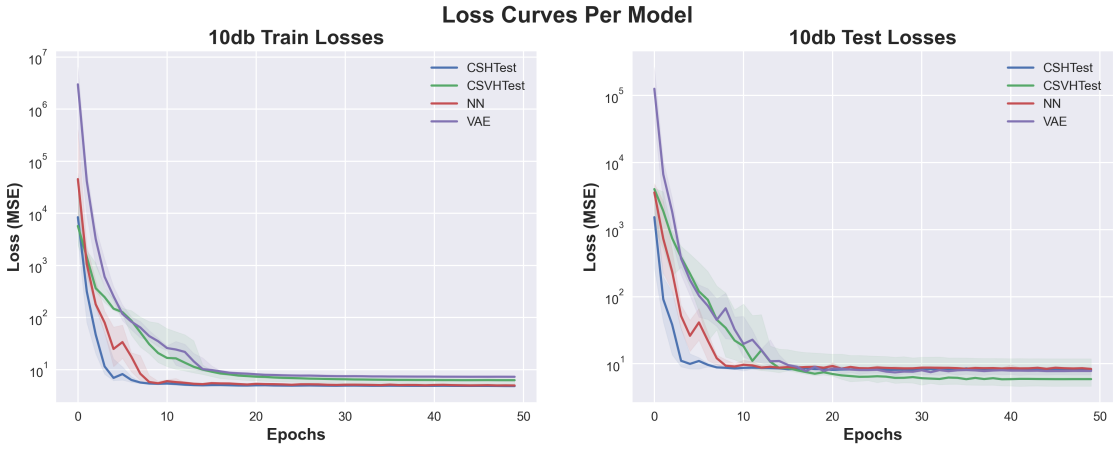


Figure A.4: Comparison of CSHTest, CSVHTest, NN, and VAE in the presence of noise

approximators on these transformations.

$$\ell_{KL} = KL(\boldsymbol{z}_i || \mathcal{N}(0,1)) \tag{A.2}$$

$$\ell_{latent} = \ell(\boldsymbol{z}, \hat{\boldsymbol{z}}) \tag{A.3}$$

$$\ell_{MSE} = ||\boldsymbol{x} - \eta_i(\mathbf{S}_i \circ \boldsymbol{x})||_2 \tag{A.4}$$

$$\ell_{\text{CSHTest}} = \ell_{MSE} + \lambda_{KL} * \ell_{KL} + \lambda_{latent} * \ell_{latent} \tag{A.5}$$

A.3.2 Results of the Effect of Noise on CSHTest and CSVHTest

We notice that the Variational version CSVHTest often works better than CSHTest in the presence of noise. Results are shown in Figure A.4.

A.4 Causal Problems and the Investigation of Optimizers

While working on the causal problems, because of the input of so many zeros in the structural Hadamard product, the loss space is not very well-behaved. As a result, we notice some inconsistency in loss trajectory with different optimizers. We do an initial investigation on the possibilities of different optimizers in our problem space.

Initially optimizers Adam, SGD, Adabelief, and PSGD [Li18b, Li19, Li18a, Li22] are compared primarily for mean OOD MSE test loss across iterations in a limited number of test cases [Li18b, ZTD20]. Due to better performance, Adabelief and PSGD are compared in a more robust set of cases. Figure A.5 shows a scatter plot of the final losses across all test cases, and a subset of the results are listed in Table A.2. Although PSGD routinely outperform AdaBelief both in mean final loss and variance (across 3 iterations per test), there were select conditions and DAGs in which any single optimizer would underperform or not converge. We leave it to further research to investigate optimizer performance and considerations for causally informed deep learning architecture that are constrained in unique ways than traditional deep learning models. PSGD had better loss in 171 cases, and lower variance in 148 of the 176 test cases.

Optimizer Test Cases (176 total, every combination of below):

- DAG Size (number of nodes, number of edges): (4,4), (5,5)
- SEM: linear, nonlinear generative functions
- Model: CSHTest, CSVHTest
- SNR: 0 noise (inf SNR), 7 dB
- Hamm (Structural Hamming Distance): 0 (ground truth), 1
- Split: ID (in-distribution or random), OOD (out-of-dist. or 75% quantile split)
- Split Number: Which node/variable is data being split on (not relevant to ID)

Table A.2 has results for DAG Size (4,4) nonlinear

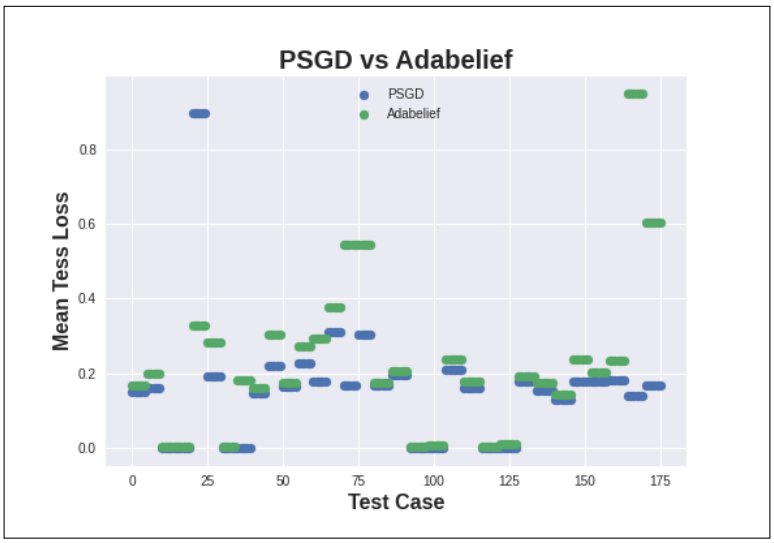


Figure A.5: Comparison of final test loss of optimizers PSGD and AdaBelief across 176 unique tests

A.5 DAG Simulation Settings

A.5.1 Training Hyperparameters

The following fixed setting where used when training models for the simulations. A random seed (1) was used in all experiments.

- 100 Epochs
- Random Weight Matrix $\mathcal{N}(0, 1)$ for linear model weights
- Linear model η nets size: [4,4] MLP
- Non-linear model η nets size: [4,16,8,2] MLP
- activation function: Soft Leaky ReLU
- 10 Iterations of a Ground Truth DAG per DAG size
- 5 Iterations of modified DAGs per Hamming Distance

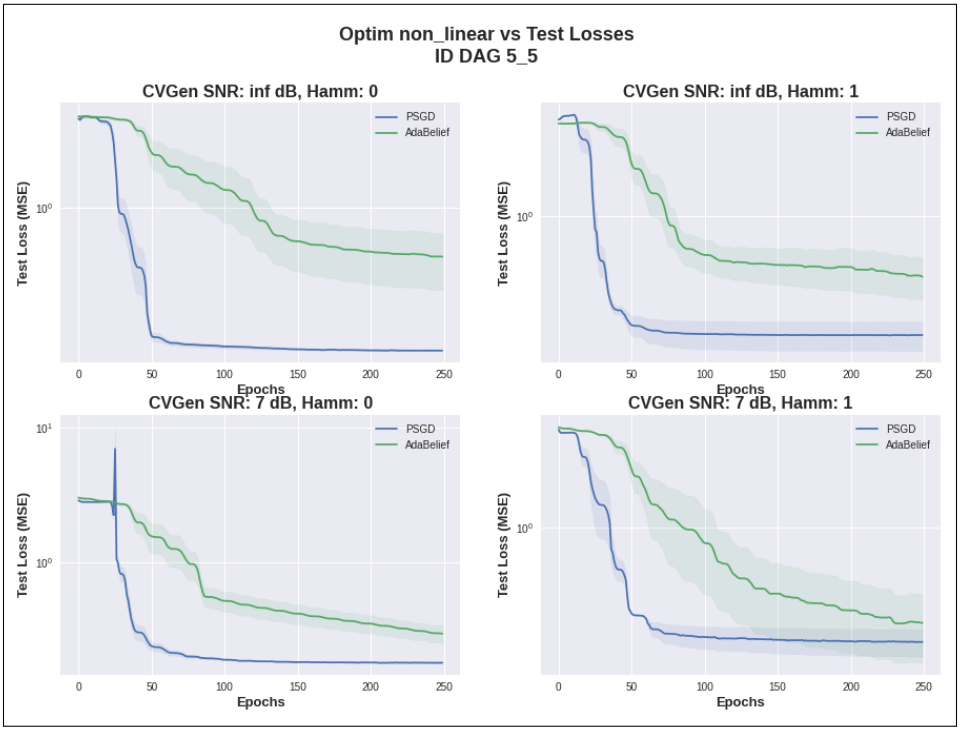


Figure A.6: Example of loss curves for optimizers PSGD and AdaBelief inf,1 dB SNR and 0,1 Hamming Distance.

- 100 data points (N) per DAG of training data
- Optimizer: PSGD
- Noise Gaussian, 0-mean, variance calculated per SNR
- Split Number: Which node/variable is data being split on (not relevant to ID)

A.5.2 Test Cases

- DAG Size (number of nodes, number of edges): (4,4), (5,5)
- SEM: linear generative functions
- Model: CSHTest

- SNR: 0 noise (inf SNR), 5 dB
- Hamm (Structural Hamming Distance): 0 (ground truth), 1, 2, 3, 4
- OOD (out-of-dist. or 75% quantile split)
- Split Number: Which node/variable is data being split, one per each variable (4 for a 4 node graph)

A.6 Final Losses with Sample Variances

The Final Losses with Sample Variances for the DAG simulations are shown in Table A.3.

Model	SNR	Hamm	Split	Split_Num	AdaBelief	PSGD
CSHTest	7.0	0	ID	1	0.17 ± 2.26e-04	0.15 ± 6.74e-06
			OOD	0	0.17 ± 2.26e-04	0.15 ± 6.74e-06
				1	0.17 ± 2.26e-04	0.15 ± 6.74e-06
				2	0.17 ± 2.26e-04	0.15 ± 6.74e-06
				3	0.17 ± 2.26e-04	0.15 ± 6.74e-06
		1	ID	1	0.2 ± 6.69e-03	0.16 ± 1.18e-03
			OOD	0	0.2 ± 6.69e-03	0.16 ± 1.18e-03
				1	0.2 ± 6.69e-03	0.16 ± 1.18e-03
				2	0.2 ± 6.69e-03	0.16 ± 1.18e-03
				3	0.2 ± 6.69e-03	0.16 ± 1.18e-03
	∞	0	ID	1	0.0 ± 4.07e-05	0.0 ± 9.75e-10
			OOD	0	0.0 ± 4.07e-05	0.0 ± 9.75e-10
				1	0.0 ± 4.07e-05	0.0 ± 9.75e-10
				2	0.0 ± 4.07e-05	0.0 ± 9.75e-10
				3	0.0 ± 4.07e-05	0.0 ± 9.75e-10
		1	ID	1	0.0 ± 3.92e-05	0.0 ± 2.54e-10
			OOD	0	0.0 ± 3.92e-05	0.0 ± 2.54e-10
				1	0.0 ± 3.92e-05	0.0 ± 2.54e-10
				2	0.0 ± 3.92e-05	0.0 ± 2.54e-10
				3	0.0 ± 3.92e-05	0.0 ± 2.54e-10
CSVHTest	7.0	0	ID	1	0.33 ± 4.85e-02	0.9 ± 1.54e+00
			OOD	0	0.33 ± 4.85e-02	0.9 ± 1.54e+00
				1	0.33 ± 4.85e-02	0.9 ± 1.54e+00
				2	0.33 ± 4.85e-02	0.9 ± 1.54e+00
				3	0.33 ± 4.85e-02	0.9 ± 1.54e+00
		1	ID	1	0.28 ± 2.72e-02	0.19 ± 9.08e-04
			OOD	0	0.28 ± 2.72e-02	0.19 ± 9.08e-04
				1	0.28 ± 2.72e-02	0.19 ± 9.08e-04
				2	0.28 ± 2.72e-02	0.19 ± 9.08e-04
				3	0.28 ± 2.72e-02	0.19 ± 9.08e-04
	∞	0	ID	1	0.0 ± 3.55e-06	0.0 ± 1.08e-09
			OOD	0	0.0 ± 3.55e-06	0.0 ± 1.08e-09
				1	0.0 ± 3.55e-06	0.0 ± 1.08e-09
				2	0.0 ± 3.55e-06	0.0 ± 1.08e-09
				3	0.0 ± 3.55e-06	0.0 ± 1.08e-09
		1	ID	1	0.18 ± 8.86e-02	0.0 ± 1.62e-08
			OOD	0	0.18 ± 8.86e-02	0.0 ± 1.62e-08
				1	0.18 ± 8.86e-02	0.0 ± 1.62e-08
				2	0.18 ± 8.86e-02	0.0 ± 1.62e-08
				3	0.18 ± 8.86e-02	0.0 ± 1.62e-08

Table A.2: Comparison of final test loss of CSHTest and CSVHTest with optimizers PSGD and AdaBelief

Hypothesis	Split	Model	Train	Test
H1	Random	CSHTest	$0.02 \pm 2.34e-07$	$0.02 \pm 9.42e-08$
		CSVHTest	$7.09 \pm 1.71e+00$	$8.21 \pm 2.44e+00$
	DBP 75%	CSHTest	$0.02 \pm 2.00e-06$	$0.04 \pm 3.82e-06$
		CSVHTest	$0.74 \pm 1.81e-02$	$0.23 \pm 2.32e-02$
	GCS 25%	CSHTest	$0.02 \pm 4.67e-07$	$0.03 \pm 2.67e-06$
		CSVHTest	$0.39 \pm 4.62e+00$	$1.08 \pm 4.28e+00$
H2	Random	CSHTest	$0.02 \pm 6.01e-08$	$0.02 \pm 3.88e-07$
		CSVHTest	$7.1 \pm 3.82e+00$	$8.17 \pm 5.36e+00$
	DBP 75%	CSHTest	$0.02 \pm 5.07e-07$	$0.03 \pm 6.59e-08$
		CSVHTest	$0.74 \pm 3.66e-03$	$0.17 \pm 3.16e+01$
	GCS 25%	CSHTest	$0.02 \pm 1.30e-07$	$0.06 \pm 5.45e-05$
		CSVHTest	$0.36 \pm 2.44e+00$	$3.83 \pm 3.94e+00$

Table A.3: Mean and Standard Deviation of the Final Test Loss in Medical Hypothesis Testing Experiments.

APPENDIX B

Appendix: Latent Augmentation VAE

B.1 Architecture and Training

- Model:
  - Encoder: 2 Conv layers followed by a fully-connected layer. For CVAE, append one-hot representation of the augmentation class to the features before the FC layer.
  - Latent Space: 16 dimensions.
  - Decoder: Fully-connected layer followed by 2 ConvTranspose layers. For the CVAE, again append one-hot representation of the augmentation class to the latent space inputs. For LAVAE, each pair of augmentations gets its own decoder head.
  - For LAVAE, L_{aug_i}: 2 16 $\times$ 16 linear matrices. Will be used regardless of which decoder is being used.
- Optimizer: Adabelief, one for Encoder/Decoder, one for latent augmentation networks, and one for each additional decoder head
  - learning-rate: 0.0001
  - epsilon=1e-16, betas=(0.9,0.999)
- Training Epochs

  - Encoder/Decoder - 100
  - Latent Augmentation Networks - 60
  - Additional Decoders - 100

- Training Parameters
  - batch-size 64

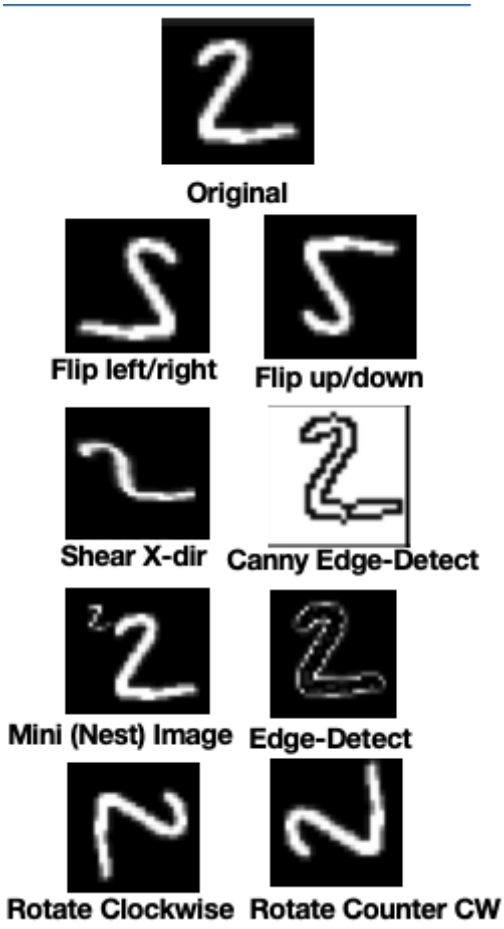


Figure B.1: All Augmentations Visualized

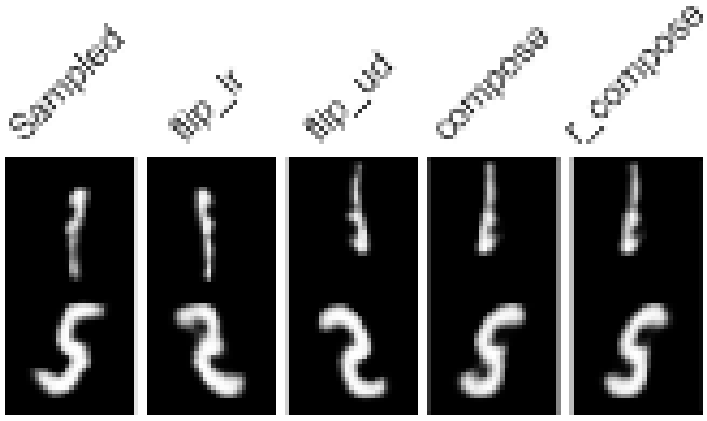


Figure B.2: Sampled digits and augmentations via bounding box method

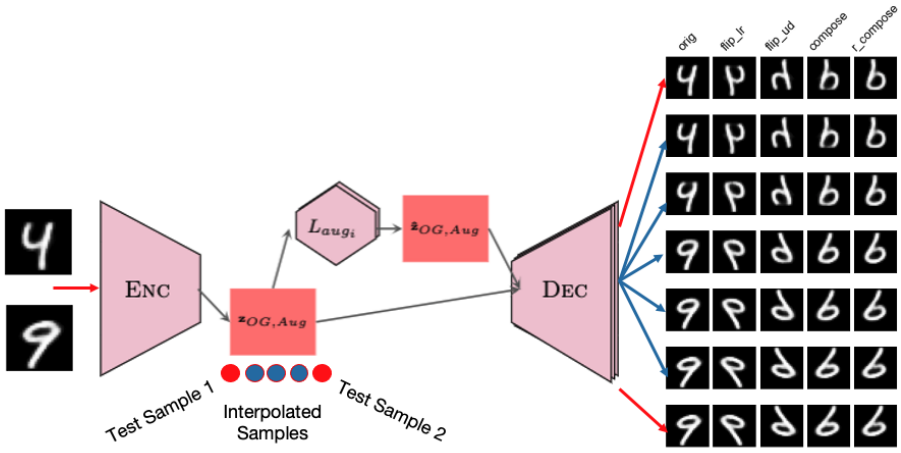


Figure B.3: Interpolating between two test samples (top and bottom row) with augmentations

B.2 Sampling and Interpolation

An example of sampled images showing the structure of the latent space is shown in Figure B.2. Then Figure B.3 shows that we still maintain the interpolation properties of the VAE, as we take equally spaced points in the latent space and decode them to see the progression of the digit and augmentation.

B.3 Latent Space Geometries

The following figures show the 2-D projections of the latent space different choices in augmentations.

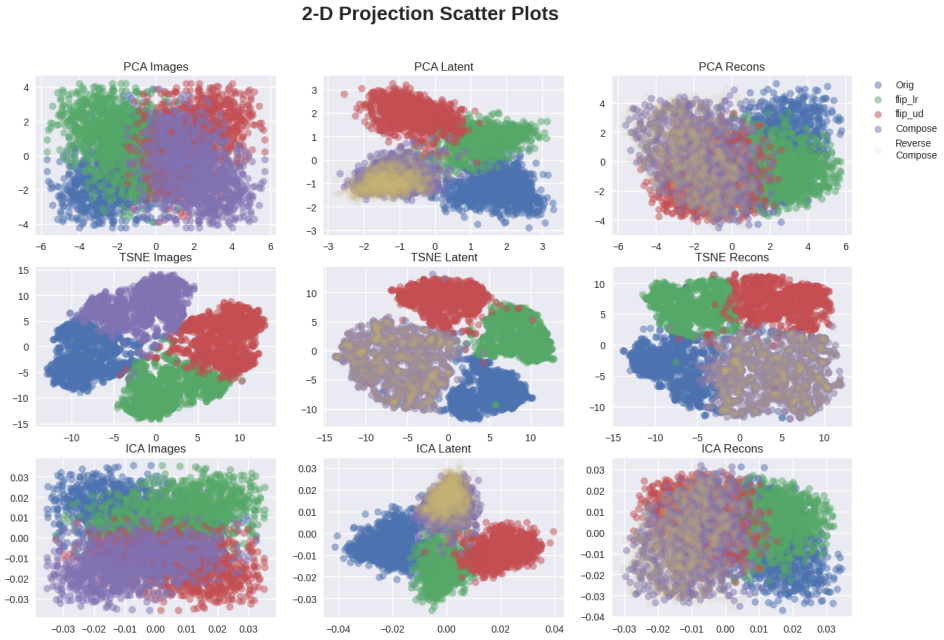


Figure B.4: "Flips" Image, Latent, and Reconstructions Image 2-D Projections

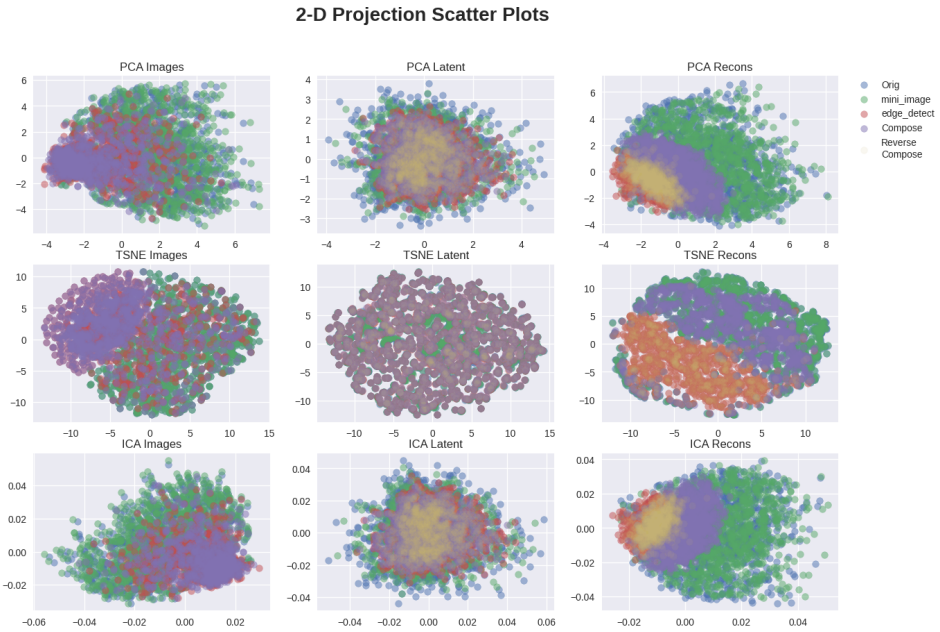


Figure B.5: "Nested Mini-Image, Edge-Detect" Image, Latent, and Reconstructions Image 2-D Projections

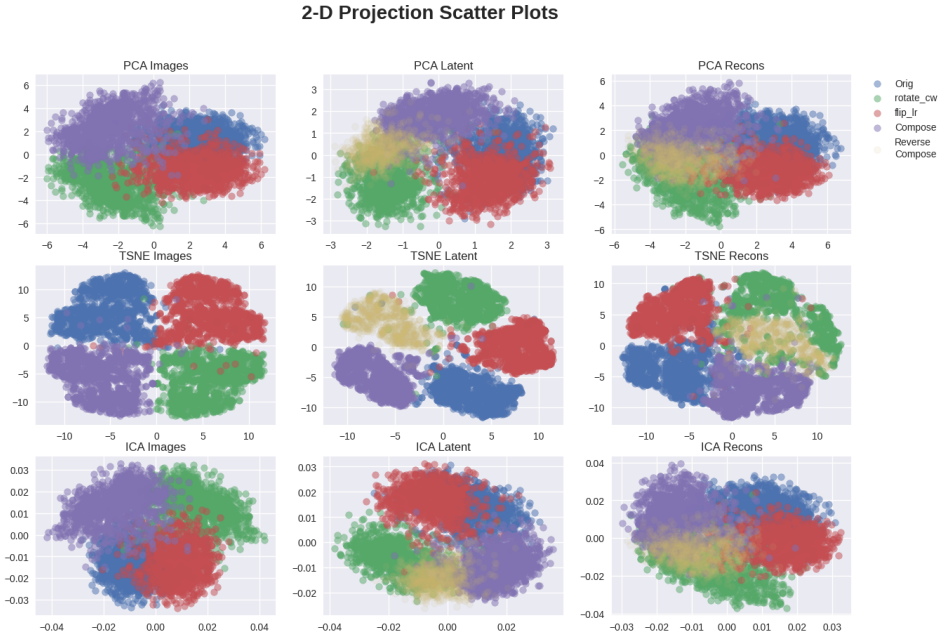


Figure B.6: "90° Clockwise Rotation, Flip left/right" Image, Latent, and Reconstructions Image 2-D Projections

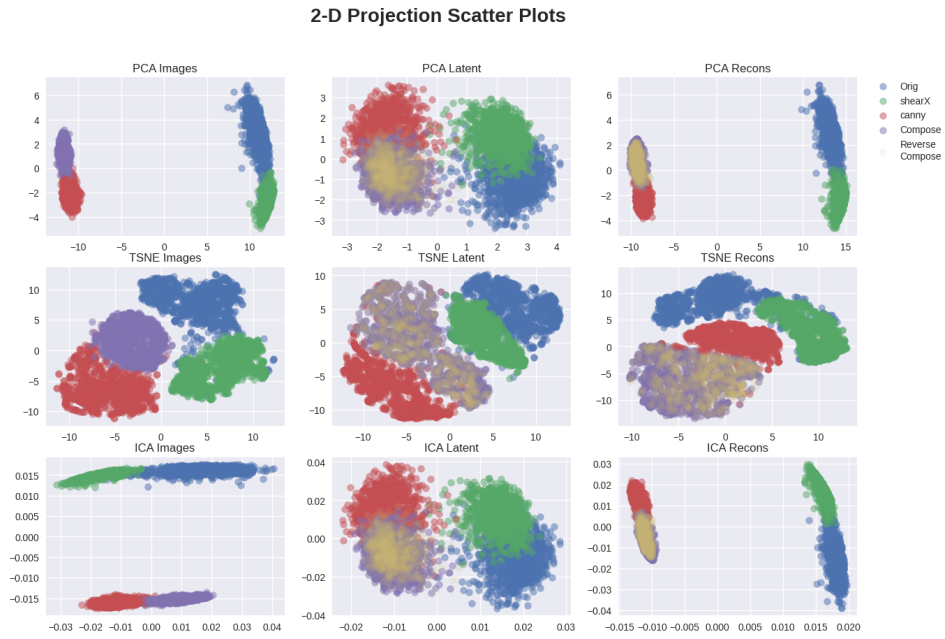


Figure B.7: "X-direction shear, Canny edge-detect" Image, Latent, and Reconstructions Image 2-D Projections

APPENDIX C

PureEBM

C.1 EBM Further Background

C.1.1 Chaotic Dynamics

Chaos theory offers a distinct perspective for justifying the suppression of adversarial signals through extended iterative transformations. In deterministic systems, chaos is characterized by the exponential growth of initial infinitesimal perturbations over time, leading to a divergence in the trajectories of closely situated points — a phenomenon popularly known as the butterfly effect. This concept extends seamlessly to stochastic systems as well. [HMZ21] were the first to show the chaotic nature of EBMs for purification. Here we verify that both poisoned images and clean images have the same chaotic properties.

Stochastic Differential Equations and Chaos

Consider the Stochastic Differential Equation (SDE) given by:

$$\mathrm{d}X_t = V(X)\mathrm{d}t + \eta_{noise}\mathrm{d}B_t, \tag{C.1}$$

where B_t denotes Brownian motion and $\eta_{noise} \geq 0$. This equation, which encompasses the Langevin dynamics, is known to exhibit chaotic behavior in numerous contexts, especially for large values of η_{noise} [LLB03].

Maximal Lyapunov Exponent

The degree of chaos in a dynamical system can be quantified by the maximal Lyapunov exponent λ, defined as:

$$\lambda = \lim_{t\to\infty} \frac{1}{t} \log \frac{|\delta X_{\eta_{noise}}(t)|}{|\delta X_{\eta_{noise}}(0)|}, \tag{C.2}$$

where $\delta X_{\eta_{noise}}(t)$ represents an infinitesimal perturbation in the system state at time t, evolved according to Equation C.1 from an initial perturbation $\delta X_{\eta_{noise}}(0)$. For ergodic dynamics, λ is independent of the initial perturbation $\delta X_{\eta_{noise}}(0)$. An ordered system exhibits a maximal Lyapunov exponent that is non-positive, while chaotic systems are characterized by a positive λ. Thus, by analyzing the maximal Lyapunov exponent of the Langevin equation, one can discern whether the dynamics are ordered or chaotic.

Following the classical approach outlined by [BGS76], we calculate the maximal Lyapunov exponent for the modified Langevin transformation, described by the equation:

$$Z_{\eta_{noise}}(X) = x_\tau - \Delta\tau \nabla_{x_\tau} \mathcal{G}_\theta(x_\tau) + \eta_{noise}\sqrt{2\Delta\tau}\epsilon_\tau, \tag{C.3}$$

This computation is performed across a range of noise strengths η_{noise}. Our findings demonstrate a clear transition from noise-dominated to chaos-dominated behavior. Notably, at $\eta_{noise} = 1$ — the parameter setting for our training and defense algorithms — the system transitions from ordered to chaotic dynamics. This critical interval balances the ordered gradient forces, which encourage pattern formation, against chaotic noise forces that disrupt these patterns. Oversaturation occurs when the gradient forces prevail, leading to noisy images when noise is dominant. These results are illustrated in Figure C.1.

The inherent unpredictability in the paths under $Z_{\eta_{noise}}$ serves as an effective defense mechanism against targeted poison attacks. Due to the chaotic nature of the transformation, generating informative attack gradients that can make it through the defense while causing a backdoor in the network becomes challenging. Exploring other chaotic transformations,

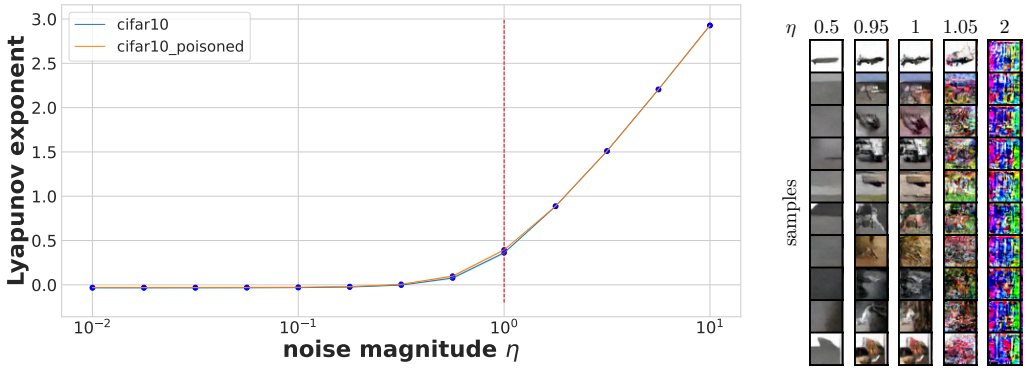


Figure C.1: *Left:* The maximal Lyapunov exponent varies significantly with different values of the noise parameter η_{noise}. Notably, at $\eta_{noise} = 1$, which is the setting used in our training and defense dynamics, there is a critical transition observed. This transition is from an ordered region, where the maximal exponent is zero, to a chaotic region characterized by a positive maximal exponent. This observation is crucial for understanding the underlying dynamics of our model. *Right:* The appearance of steady-state samples exhibits marked differences across the spectrum of η_{noise} values. For lower values of η_{noise}, the generated images tend to be oversaturated. Conversely, higher values of η_{noise} result in noisy images. However, there exists a narrow window around $\eta_{noise} = 1$ where a balance is achieved between gradient and noise forces, leading to realistic synthesis of images.

both stochastic and deterministic, could be a promising direction for developing new defense strategies.

We see that as expected the Lyapunov exponent of the Langevin dynamics on clean and poisoned points are exactly the same.

C.1.2 EBM Purification is a Convergent Process

Energy-based models and Langevin dynamics are both commonly associated with divergent generative models and diffusion processes in the machine learning community, in which samples are generated from a random initialization using a conditional or unconditional

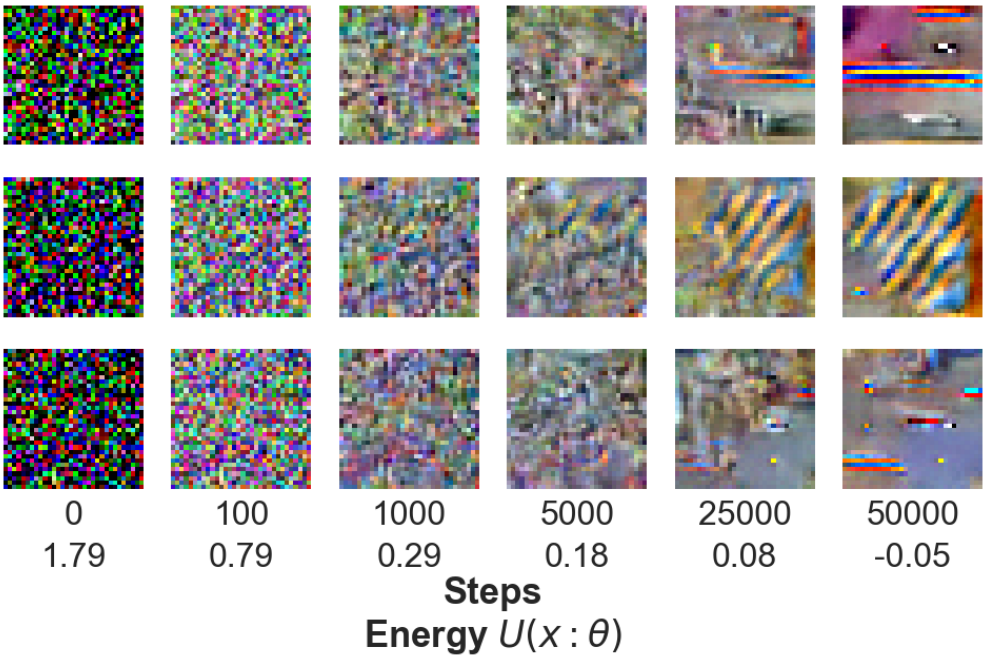


Figure C.2: Random Noise initialization of purification process

probability distribution. In contrast, we emphasize that the EBM and MCMC purification process is a convergent generative chain, initialized with a sample from some data distribution p_{data} with metastable properties that retain features of the original image due to the low energy density around the image [NHH20]. To illustrate this point, Figure C.2 shows the purification process on random noise initialization. Even with long-run dynamics of 50k Langevin steps producing low energy outputs, the resulting 'images' are not meaningful, highlighting the desired reliance on a realistic sample initializing a convergent MCMC chain. Previous analysis demonstrates the mid-run memoryless properties that remove adversarial poisons and enable the EBM purification process once paired with the metastable aspects of the convergent MCMC chain.

C.2 Additional Results

C.2.1 Full Results Primary Experiments

Results on all primary poison scenarios with ResNet18 classifier including all EPIc versions (various subset sizes and selection frequency), FrieNDs versions (with Bernoulli or Gaussian added noise), and all natural PureEBM versions are shown in Tables C.1 and C.2. Asterisk (*) indicates a baseline defense that was selected for the main paper results table due to best poison defense performance.

We note that the implementation made available for EPIc contains discrepancies, occasionally returning random subsets, and drops repeatedly selected points every epoch. We did our best to reproduce results, and choose the best of all version ran to compare to. Further, we note that our results outperform the results reported by [YLM22], listed in the table here as EPIc$_{reported}$.

C.2.2 Extended Poison % Results

We tested applying Narcissus poisons at train-time to higher percentages of the dataset. Results for from-scratch are shown in Table C.3, and for transfer learning in Table C.4. We find that the PureEBM defense is able to effectively defend against poisons at higher percentages, while EPIc and FrieNDs defenses are break down.

C.2.3 Full MobileNetV2 and DenseNet121 Results

In Figures C.5 and C.6, we show the full results for MobileNetV2 and DenseNet121 architectures for all poison scenarios and training paradigms.

Table C.1: Poison success and natural accuracy in all poisoned from-scratch training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 100 GM experiments and NS triggers over 10 classes.

	From Scratch				
	200 - Epochs				
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	44.00	$94.84_{0.2}$	$43.95_{33.6}$	$94.89_{0.2}$	93.59
EPIc-0.1*	34.00	$91.27_{0.4}$	$30.18_{32.2}$	$91.17_{0.2}$	81.50
EPIc-0.2	21.00	$88.04_{0.7}$	$32.50_{33.5}$	$86.89_{0.5}$	84.39
EPIc-0.3*	10.00	$85.14_{1.2}$	$27.31_{34.0}$	$82.20_{1.1}$	84.71
FrieNDs-B	1.00	$\mathbf{91.16}_{0.4}$	$8.32_{22.3}$	$91.01_{0.4}$	71.76
FrieNDs-G*	**0.00**	$\mathbf{91.15}_{0.4}$	$9.49_{25.9}$	$91.06_{0.2}$	83.03
PureEBM	**0.00**	$\mathbf{92.26}_{0.2}$	$\mathbf{1.27}_{0.6}$	$\mathbf{92.91}_{0.2}$	**2.16**
PureEBM-P	NA	NA	$\mathbf{1.38}_{0.7}$	$\mathbf{92.70}_{0.2}$	**2.78**
$\mathbf{PureEBM}_{CN-10}$	**0.00**	$\mathbf{92.99}_{0.2}$	$\mathbf{1.43}_{0.8}$	$\mathbf{92.90}_{0.2}$	**3.06**
$\mathbf{PureEBM}_{IN}$	1.00	$\mathbf{92.98}_{0.2}$	$\mathbf{1.39}_{0.8}$	$\mathbf{92.92}_{0.2}$	**2.50**
$\mathbf{PureEBM\text{-}P}_{CN-10}$	NA	NA	$\mathbf{1.64}_{0.01}$	$\mathbf{92.86}_{0.20}$	**4.34**
	80 - Epochs				
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	47.00	$93.79_{0.2}$	$32.51_{30.3}$	$93.76_{0.2}$	79.43
EPIc-0.1*	27.00	$90.87_{0.4}$	$24.15_{30.1}$	$90.92_{0.4}$	79.42
EPIc-0.2	28.00	$91.02_{0.4}$	$23.75_{29.2}$	$89.72_{0.3}$	74.28
EPIc-0.3*	44.00	$92.46_{0.3}$	$21.53_{28.8}$	$88.05_{1.1}$	80.75
FrieNDs-B	2.00	$90.07_{0.4}$	$\mathbf{1.42}_{0.8}$	$90.06_{0.3}$	2.77
FrieNDs-G*	**1.00**	$90.09_{0.4}$	$\mathbf{1.37}_{0.9}$	$90.01_{0.2}$	3.18
PureEBM	**1.00**	$\mathbf{91.36}_{0.3}$	$\mathbf{1.46}_{0.8}$	$\mathbf{91.83}_{0.3}$	**2.49**
PureEBM-P	NA	NA	$\mathbf{1.63}_{1.0}$	$\mathbf{91.49}_{0.3}$	3.47
$\mathbf{PureEBM}_{CN-10}$	**1.00**	$\mathbf{92.02}_{0.2}$	$\mathbf{1.50}_{0.9}$	$\mathbf{92.03}_{0.2}$	**2.52**
$\mathbf{PureEBM}_{IN}$	**1.00**	$\mathbf{92.02}_{0.2}$	$\mathbf{1.52}_{0.8}$	$\mathbf{92.02}_{0.3}$	**2.81**
$\mathbf{PureEBM\text{-}P}_{CN-10}$	NA	NA	$\mathbf{1.68}_{1.0}$	$\mathbf{92.07}_{0.2}$	3.34

C.2.4 Full CINIC-10 Results

In Table C.7, we show the full results for CINIC-10 for all poison scenarios and training paradigms. We also include the CIFAR-10 accuracy, where we test only on the CIFAR-10 portion of the CINIC-10 test set.

Table C.2: Poison success and natural accuracy in all poisoned transfer training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 50 BP experiments and NS triggers over 10 classes.

	Transfer Learning				
	Fine-Tune				
	Bullseye Polytope-10%		**Narcissus-10%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	46.00	$89.84_{0.9}$	$33.41_{33.9}$	$90.14_{2.4}$	98.27
EPIc-0.1	50.00	$89.00_{1.8}$	$32.40_{33.7}$	$90.02_{2.2}$	98.95
EPIc-0.2*	42.00	$81.95_{5.6}$	$20.93_{27.1}$	$88.58_{2.0}$	91.72
EPIc-0.3	44.00	$86.75_{6.3}$	$28.01_{34.9}$	$84.36_{6.3}$	99.91
FRIENDS-B	8.00	$87.80_{1.1}$	$3.34_{5.7}$	$89.62_{0.5}$	19.48
FRIENDS-G*	8.00	$87.82_{1.2}$	$3.04_{5.1}$	$89.81_{0.5}$	17.32
PureEBM	**0.00**	$\mathbf{88.95}_{1.1}$	$\mathbf{1.98}_{1.7}$	$\mathbf{91.40}_{0.4}$	**5.98**
PureEBM-p	NA	NA	$16.48_{27.2}$	$88.27_{2.4}$	86.49
$\textbf{PureEBM}_{CN-10}$	**0.00**	$\mathbf{88.67}_{1.2}$	$\mathbf{2.97}_{2.5}$	$\mathbf{90.99}_{0.3}$	**7.95**
$\textbf{PureEBM}_{IN}$	**0.00**	$\mathbf{87.52}_{1.2}$	$\mathbf{2.02}_{1.0}$	$\mathbf{89.78}_{0.6}$	**3.85**

	Linear - Bullseye Polytope			
	BlackBox-10%		**WhiteBox-1%**	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	93.75	$83.59_{2.4}$	98.00	$70.09_{0.2}$
EPIc-0.1	91.67	$83.48_{2.9}$	98.00	$69.35_{0.3}$
EPIc-0.2*	66.67	$84.34_{3.8}$	91.00	$64.79_{0.7}$
EPIc-0.3	66.67	$83.23_{3.8}$	63.00	$60.86_{1.5}$
FRIENDS-B	35.42	$84.97_{2.2}$	19.00	$60.85_{0.6}$
FRIENDS-G*	33.33	$85.18_{2.3}$	19.00	$60.90_{0.6}$
PureEBM	**0.00**	$\mathbf{92.89}_{0.2}$	**6.00**	$\mathbf{64.51}_{0.6}$
PureEBM-p	NA	NA	NA	NA
$\textbf{PureEBM}_{CN-10}$	**0.00**	$\mathbf{92.82}_{0.1}$	**6.00**	$\mathbf{64.44}_{0.4}$
$\textbf{PureEBM}_{IN}$	**0.00**	$\mathbf{92.38}_{0.3}$	**6.00**	$\mathbf{64.98}_{0.3}$

C.3 Further Experimental Details

C.3.1 EBM Training

Algorithm 2 is pseudo-code for the training procedure of a data-initialized convergent EBM. We use the generator architecture of the SNGAN [MKK18] for our EBM as our network architecture.

Table C.3: Narcissus transfer fine-tune results at various poison %'s

	Narcissus-1%		
	Avg Poison Success (%) ↓	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**
None	$17.06_{27.0}$	$93.18_{0.1}$	81.97
EPIc-0.1	$15.58_{25.5}$	$92.75_{0.2}$	73.65
EPIc-0.2	$12.33_{23.8}$	$85.86_{2.9}$	74.32
EPIc-0.3	$12.74_{21.2}$	$91.37_{4.0}$	67.45
FrieNDs-B	$\mathbf{1.44}_{0.8}$	$90.61_{0.2}$	**2.49**
FrieNDs-G	$\mathbf{1.34}_{0.7}$	$90.50_{0.2}$	**2.50**
PureEBM	$\mathbf{1.50}_{1.4}$	$\mathbf{91.65}_{0.1}$	5.19
PureEBM-P	$4.50_{7.4}$	$89.61_{0.3}$	24.43
PureEBM$_{CN-10}$	$\mathbf{1.77}_{1.2}$	$\mathbf{91.56}_{0.1}$	4.07
PureEBM$_{IN}$	$\mathbf{1.62}_{0.9}$	$90.91_{0.1}$	3.35
PureEBM-P$_{CN-10}$	$4.33_{6.2}$	$90.99_{0.2}$	21.25
	Narcissus-2.5%		
	Avg Poison Success (%) ↓	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**
None	$22.22_{30.1}$	$93.35_{0.1}$	89.74
EPIc-0.1	$19.77_{27.5}$	$92.72_{0.3}$	87.51
EPIc-0.2	$24.26_{31.2}$	$85.59_{3.3}$	96.07
EPIc-0.3	$12.32_{18.7}$	$92.24_{0.4}$	61.33
FrieNDs-B	$2.25_{3.3}$	$90.44_{0.3}$	11.46
FrieNDs-G	$2.43_{3.6}$	$90.51_{0.2}$	12.61
PureEBM	$\mathbf{1.60}_{1.2}$	$\mathbf{91.27}_{0.1}$	**4.76**
PureEBM-P	$7.93_{12.4}$	$90.26_{0.2}$	39.59
PureEBM$_{CN-10}$	$\mathbf{2.21}_{1.6}$	$\mathbf{91.45}_{0.1}$	**5.02**
PureEBM$_{IN}$	$\mathbf{1.85}_{0.9}$	$90.85_{0.2}$	**3.39**
PureEBM-P$_{CN-10}$	$5.95_{8.5}$	$90.80_{0.2}$	28.88
	Narcissus-10%		
	Avg Poison Success (%) ↓	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**
None	$33.41_{33.9}$	$90.14_{2.4}$	98.27
EPIc-0.1	$32.40_{33.7}$	$90.02_{2.2}$	98.95
EPIc-0.2	$20.93_{27.1}$	$88.58_{2.0}$	91.72
EPIc-0.3	$28.01_{34.9}$	$84.36_{6.3}$	99.91
FrieNDs-B	$3.34_{5.7}$	$89.62_{0.5}$	19.48
FrieNDs-G	$3.04_{5.1}$	$89.81_{0.5}$	17.32
PureEBM	$\mathbf{1.98}_{1.7}$	$\mathbf{91.40}_{0.4}$	**5.98**
PureEBM-P	$16.48_{27.2}$	$88.27_{2.4}$	86.49
PureEBM$_{CN-10}$	$\mathbf{2.97}_{2.5}$	$\mathbf{90.99}_{0.3}$	**7.95**
PureEBM$_{IN}$	$\mathbf{2.02}_{1.0}$	$89.78_{0.6}$	**3.85**
PureEBM-P$_{CN-10}$	$11.84_{19.9}$	$88.77_{1.3}$	66.63

Table C.4: BP transfer linear gray-box results at various poison %'s

	Narcissus-5%		**Narcissus-10%**	
	Poison Success (%) ↓	**Natural Accuracy (%) ↑**	**Poison Success (%) ↓**	**Natural Accuracy (%) ↑**
None	26.00	$93.60_{0.2}$	32.00	$93.60_{0.2}$
EPIc-0.1	12.00	$93.34_{0.4}$	50.00	$92.79_{0.6}$
EPIc-0.2	18.00	$92.53_{1.4}$	34.00	$92.86_{1.4}$
EPIc-0.3	18.00	$92.80_{0.9}$	24.00	$92.89_{1.0}$
FrieNDs-B	4.00	$\mathbf{94.09}_{0.1}$	4.00	$\mathbf{94.11}_{0.1}$
FrieNDs-G	4.00	$\mathbf{94.12}_{0.1}$	4.00	$\mathbf{94.13}_{0.1}$
PureEBM	**0.00**	$93.18_{0.0}$	**0.00**	$92.94_{0.1}$
PureEBM$_{CN-10}$	**0.00**	$93.14_{0.1}$	**0.00**	$92.61_{0.1}$
PureEBM$_{IN}$	**0.00**	$92.09_{0.1}$	**0.00**	$91.51_{0.1}$
	Narcissus-25%		**Narcissus-50%**	
	Poison Success (%) ↓	**Natural Accuracy (%) ↑**	**Poison Success (%) ↓**	**Natural Accuracy (%) ↑**
None	66.00	$92.89_{0.4}$	93.75	$83.59_{2.4}$
EPIc-0.1	70.00	$92.43_{0.8}$	91.67	$83.48_{2.9}$
EPIc-0.2	76.00	$91.72_{2.0}$	66.67	$84.34_{3.8}$
EPIc-0.3	62.00	$90.95_{2.7}$	66.67	$83.23_{3.8}$
FrieNDs-B	26.00	$\mathbf{93.72}_{0.2}$	35.42	$84.97_{2.2}$
FrieNDs-G	22.00	$\mathbf{93.73}_{0.2}$	33.33	$85.18_{2.3}$
PureEBM	**0.00**	$92.92_{0.1}$	**0.00**	$\mathbf{92.89}_{0.2}$
PureEBM$_{CN-10}$	**0.00**	$93.00_{0.1}$	**0.00**	$\mathbf{92.82}_{0.1}$
PureEBM$_{IN}$	**0.00**	$92.75_{0.1}$	**0.00**	$\mathbf{92.38}_{0.3}$

C.3.2 Poison Sourcing and Implementation

Triggerless attacks GM and BP poison success refers to the number of single-image targets successfully flipped to a target class (with 50 or 100 target image scenarios) while the natural accuracy is averaged across all target image training runs. Triggered attack Narcissus poison success is measured as the number of non-class samples from the test dataset shifted to the trigger class when the trigger is applied, averaged across all 10 classes, while the natural accuracy is averaged across the 10 classes on the un-triggered test data. We include the worst-defended class poison success. The Poison Success Rate for a single experiment can be defined for triggerless PSR_{notr} and triggered PSR_{tr} poisons as:

$$PSR_{notr}(F, i) = \mathbb{1}_{F(x_i^{\pi})=y_i^{\text{adv}}} \tag{C.4}$$

Table C.5: MobileNetV2 Full Results

	From Scratch - MobileNetV2				
	200 - Epochs				
	Gradient Matching-1%		Narcissus-1%		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	20.00	$93.86_{0.2}$	$32.70_{24.5}$	$93.92_{0.1}$	73.97
EPIc-0.1	37.50	$91.28_{0.2}$	$40.09_{27.1}$	$91.15_{0.2}$	79.74
EPIc-0.2	19.00	$91.24_{0.2}$	$38.55_{27.5}$	$87.65_{0.5}$	74.72
EPIc-0.3	9.78	$87.80_{1.6}$	$22.35_{23.9}$	$78.16_{9.9}$	69.52
FrieNDs-B	6.00	$84.30_{2.7}$	$\mathbf{2.00}_{1.3}$	$88.82_{0.6}$	4.88
FrieNDs-G	5.00	$88.84_{0.4}$	$\mathbf{2.05}_{1.7}$	$88.93_{0.3}$	6.33
PureEBM	**1.00**	$\mathbf{90.93}_{0.2}$	$\mathbf{1.64}_{0.8}$	$\mathbf{91.75}_{0.1}$	**2.91**
	80 - Epochs				
	Gradient Matching-1%		Narcissus-1%		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	30.00	$92.54_{0.2}$	$27.26_{26.5}$	$92.53_{0.2}$	74.82
EPIc-0.1	16.00	$90.45_{0.3}$	$31.37_{30.9}$	$90.51_{0.3}$	89.36
EPIc-0.2	22.00	$89.90_{0.3}$	$29.22_{27.6}$	$89.91_{0.3}$	76.54
EPIc-0.3	14.00	$90.23_{0.3}$	$30.69_{30.6}$	$90.30_{0.3}$	82.92
FrieNDs-B	**1.00**	$87.89_{0.3}$	$\mathbf{1.98}_{1.1}$	$87.90_{0.4}$	4.00
FrieNDs-G	3.00	$87.90_{0.4}$	$\mathbf{2.00}_{1.4}$	$88.09_{0.3}$	5.07
PureEBM	**1.00**	$\mathbf{89.71}_{0.2}$	$\mathbf{1.79}_{0.8}$	$\mathbf{90.64}_{0.2}$	**2.65**

	Transfer Learning - MobileNetV2				
	Fine-Tune NS-10%			Transfer Linear BP BlackBox-10%	
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	$23.59_{23.2}$	$88.30_{1.2}$	66.54	81.25	$73.27_{1.0}$
EPIc-0.1	$23.25_{22.8}$	$88.35_{1.0}$	65.97	81.25	$69.78_{2.0}$
EPIc-0.2	$19.95_{19.2}$	$87.67_{1.3}$	50.05	56.25	$54.47_{5.6}$
EPIc-0.3	$21.70_{28.1}$	$78.17_{6.0}$	74.96	58.33	$58.74_{9.0}$
FrieNDs-B	$\mathbf{2.21}_{1.5}$	$\mathbf{83.05}_{0.7}$	**5.63**	41.67	$68.86_{1.5}$
FrieNDs-G	$\mathbf{2.20}_{1.4}$	$\mathbf{83.04}_{0.7}$	**5.42**	47.92	$68.94_{1.5}$
PureEBM	$\mathbf{3.66}_{5.4}$	$\mathbf{84.18}_{0.5}$	18.85	**0.00**	$\mathbf{78.57}_{1.4}$

$$PSR_{tr}(F, y^{\pi}) = \frac{\sum_{(x,y)\in\mathcal{D}_{test}\setminus\mathcal{D}^{\pi}_{test}} \mathbb{1}_{F(x+\rho^{\pi})=y^{\pi}}}{|\mathcal{D}_{test} \setminus \mathcal{D}^{\pi}_{test}|} \tag{C.5}$$

C.3.2.1 Bullseye Polytope

The Bullseye Polytope (BP) poisons are sourced from two distinct sets of authors. From the original authors of BP [AMW21], we obtain poisons crafted specifically for a black-

Table C.6: DenseNet121 Full Results

	From Scratch - DenseNet121				
	200 - Epochs				
	Gradient Matching-1%		Narcissus-1%		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	14.00	$95.30_{0.1}$	$46.52_{32.2}$	$95.33_{0.1}$	91.96
EPIc-0.1	14.00	$93.0_{0.3}$	$43.38_{32.0}$	$93.07_{0.2}$	88.97
EPIc-0.2	7.00	$90.67_{0.5}$	$41.97_{33.2}$	$90.23_{0.6}$	86.85
EPIc-0.3	4.00	$88.3_{1.0}$	$32.60_{29.4}$	$85.12_{2.4}$	71.50
FrieNDs-B	1.00	$\mathbf{91.33}_{0.4}$	$8.60_{21.2}$	$91.55_{0.3}$	68.57
FrieNDs-G	1.00	$\mathbf{91.33}_{0.4}$	$10.13_{25.2}$	$91.32_{0.4}$	81.47
PureEBM	**0.00**	$\mathbf{92.85}_{0.2}$	$\mathbf{1.42}_{0.7}$	$\mathbf{93.48}_{0.1}$	**2.60**
	80 - Epochs				
	Gradient Matching-1%		Narcissus-1%		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓
None	19.00	$94.38_{0.2}$	$38.01_{36.3}$	$94.49_{0.1}$	89.11
EPIc-0.1	16.00	$92.78_{0.3}$	$32.85_{33.0}$	$92.87_{0.3}$	79.42
EPIc-0.2	13.00	$92.69_{0.3}$	$30.67_{28.1}$	$92.82_{0.2}$	65.46
EPIc-0.3	15.00	$93.35_{0.2}$	$36.80_{36.0}$	$93.34_{0.2}$	90.41
FrieNDs-B	**1.00**	$89.93_{0.4}$	$5.60_{11.6}$	$90.01_{0.4}$	38.08
FrieNDs-G	**1.00**	$89.97_{0.4}$	$7.59_{18.7}$	$89.89_{0.4}$	60.68
PureEBM	2.00	$\mathbf{91.88}_{0.3}$	$\mathbf{1.59}_{0.9}$	$\mathbf{92.59}_{0.2}$	**3.06**

	Transfer Learning - DenseNet121						
	Fine-Tune					Linear	
	Bullseye Polytope-10%		Narcissus-10%			Bullseye Polytope-10%	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	16.00	$88.91_{0.7}$	$56.52_{38.6}$	$87.03_{2.8}$	99.56	73.47	$82.13_{1.6}$
EPIc-0.1	18.00	$88.09_{1.0}$	$53.97_{39.0}$	$87.04_{2.8}$	99.44	62.50	$78.88_{2.1}$
EPIc-0.2	14.00	$80.44_{3.1}$	$43.66_{36.5}$	$85.97_{2.6}$	97.17	41.67	$70.13_{5.2}$
EPIc-0.3	10.00	$72.84_{11.9}$	$43.24_{43.0}$	$72.76_{10.8}$	100.00	66.67	$70.20_{10.1}$
FrieNDs-B	4.00	$\mathbf{87.06}_{1.0}$	$5.34_{9.9}$	$\mathbf{88.62}_{0.8}$	33.42	60.42	$80.22_{1.9}$
FrieNDs-G	2.00	$\mathbf{87.37}_{0.9}$	$5.55_{10.4}$	$\mathbf{88.75}_{0.6}$	34.91	56.25	$80.12_{1.8}$
PureEBM	**0.00**	$84.39_{1.0}$	$\mathbf{2.48}_{1.9}$	$\mathbf{88.75}_{0.5}$	**7.41**	**0.00**	$\mathbf{89.29}_{0.9}$

box scenario targeting ResNet18 and DenseNet121 architectures, and gray-box scenario for MobileNet (used in poison crafting). These poisons vary in the percentage of data poisoned, spanning 1%, 2%, 5% and 10% for the linear-transfer mode and a single 1% fine-tune mode for all models over a 500 image transfer dataset. Each of these scenarios has 50 datasets that specify a single target sample in the test-data. We also use a benchmark paper that provides a pre-trained white-box scenario on CIFAR-100 [SGG21]. This dataset includes 100 target

Table C.7: CINIC-10 Full Results

CINIC-10 Narcissus - Class 1 From-Scratch				
	200 - Epochs			
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	CIFAR-10 Accuracy (%) ↑
None	$62.06_{0.21}$	$86.32_{0.10}$	90.79	$94.22_{0.16}$
EPIc	$49.50_{0.27}$	$81.91_{0.08}$	91.35	$91.10_{0.21}$
FrieNDs	$11.17_{0.25}$	$77.53_{0.60}$	82.21	$88.27_{0.68}$
PureEBM	$\mathbf{7.73}_{0.08}$	$\mathbf{82.37}_{0.14}$	**29.48**	$\mathbf{91.98}_{0.16}$
	80 - Epochs			
	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	CIFAR-10 Accuracy (%) ↑
None	$43.75_{0.25}$	$85.25_{0.16}$	82.63	$93.36_{0.20}$
EPIc	$37.35_{0.26}$	$81.15_{0.17}$	79.98	$90.50_{0.31}$
FrieNDs	$10.14_{0.22}$	$77.46_{0.54}$	73.16	$87.79_{0.47}$
PureEBM	$\mathbf{4.85}_{0.02}$	$\mathbf{81.65}_{0.15}$	**9.14**	$\mathbf{91.33}_{0.20}$

samples with strong poison success, but the undefended natural accuracy baseline is much lower.

C.3.2.2 Gradient Matching

For GM, we use 100 publicly available datasets provided by [GFH21]. Each dataset specifies a single target image corresponding to 500 poisoned images in a target class. The goal of GM is for the poisons to move the target image into the target class, without changing too much of the remaining test dataset using gradient alignment. Therefore, each individual dataset training gives us a single data point of whether the target was correctly moved into the poisoned target class and the attack success rate is across all 100 datasets provided.

C.3.2.3 Narcissus

For Narcissus triggered attack, we use the same generating process as described in the Narcissus paper, we apply the poison with a slight change to more closely match with the baseline provided by [SGG21]. We learn a patch with $\varepsilon = 8/255$ on the entire 32-by-32 size of the image, per class, using the Narcissus generation method. We keep the number of poisoned samples comparable to GM for from-scratch experiment, where we apply the patch to 500 images (1% of the dataset) and test on the patched dataset without the multiplier. In the fine-tune scenarios, we vary the poison % over 1%, 2.5%, and 10%, by modifying either the number of poisoned images or the transfer dataset size (specifically 20/2000, 50/2000, 50/500 poison/train samples).

C.3.3 Training Parameters

We follow the training hyperparameters given by [YLM22, ZPJ22, AMW21, SGG21] for GM, NS, BP Black/Gray-Box, and BP White-Box respectively as closely as we can, with moderate modifications to align poison scenarios. HyperlightBench training followed the original creators settings, where we substituted in a poisoned dataloader [Bal23].

C.4 Timing Analysis

Table C.9 shows the training times for each poison defense in the from-scratch scenario on a TPU-V3. As PureEBM is a preprocessing step, the purification time ($\sim$400 seconds) is shared across poison scenarios, making it increasingly comparable to no defense as the number of models/scenarios increase. Although EBM training is a compute-intensive process, noted in detail in App. C.3.1, we share results in the section Table 7.1 on how a single EBM on a POOD dataset can obtain SoTA performance in a poison/classifier agnostic way. While subset selection methods like EPIc can reduce training time in longer scenarios, PureEBM

Table C.8: Training Parameters for Poison Defense Experiments

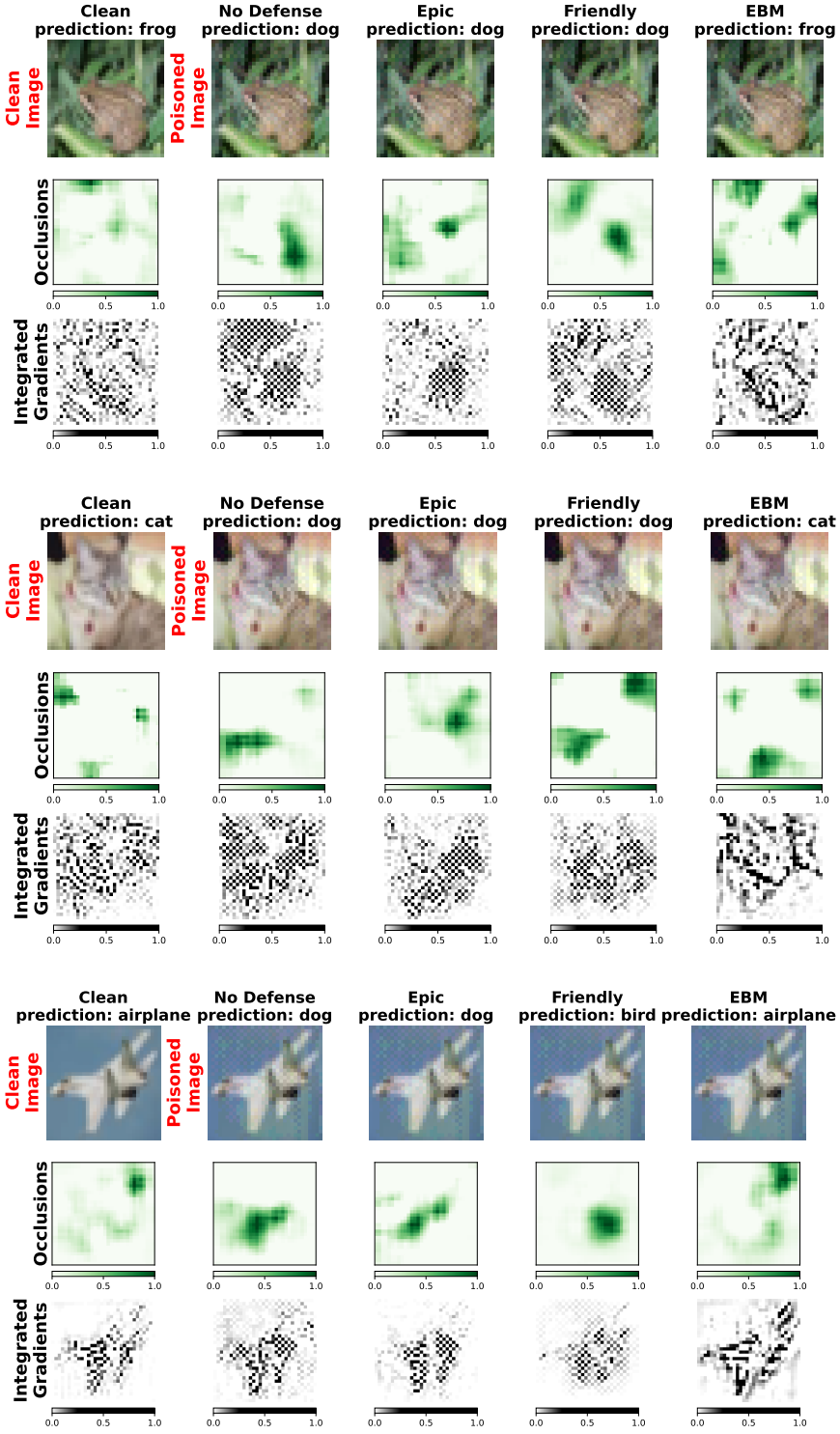
Parameter	From Scratch	Transfer Linear	Transfer Fine-Tune
Device Type	TPU-V3	TPU-V3	TPU-V3
Weight Decay	5e-4	5e-4	5e-4
Batch Size	128	64	128
Augmentations	RandomCrop(32, padding=4)	None	None
Epochs	200 or 80	40	60
Optimizer	SGD(momentum=0.9)	SGD	Adam
Learning Rate	0.1	0.1	0.0001
Learning Rate Schedule (Multi-Step Decay)	100, 150 - 200 epochs 30, 50, 70 - 80 epochs	15, 25, 35	15, 30, 45
Langevin Steps (EBM)	150	500	1000
Langevin Temperature (EBM)	1×10^{-4}	7.5×10^{-5}	1×10^{-4}
Reinitialize Linear Layer	NA	True	True

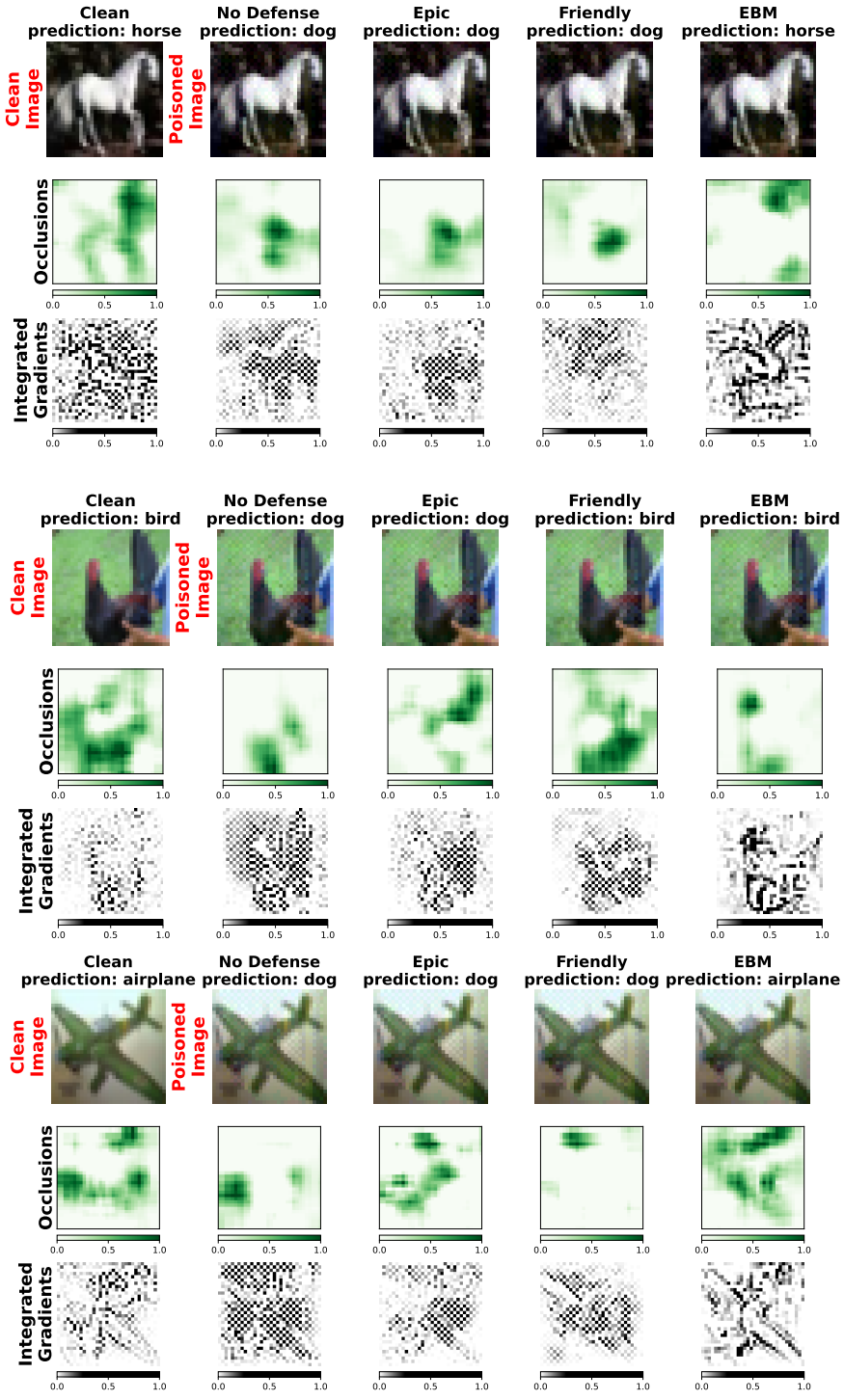
offers superior performance and flexibility to the classifier training pipeline.

Table C.9: Median Wall Clock Train Times From Scratch

Train Time (seconds)				
	Gradient Matching		Narcissus	
epochs	**80**	**200**	**80**	**200**
None	$\mathbf{2202}_{16}$	5482_{49}	$\mathbf{2936}_{94}$	7154_{194}
EPIc	$\mathbf{2256}_{97}$	$\mathbf{5006}_{253}$	3564_{213}	$\mathbf{6359}_{462}$
FrieNDs	7740_{394}	11254_{413}	8728_{660}	12868_{573}
PureEBM	$\mathbf{2213}_{36}$	5520_{47}	$\mathbf{2962}_{92}$	7293_{219}

C.5 Additional Model Interpretability Results




Clean
prediction: horse
No Defense
prediction: dog
Epic
prediction: dog
Friendly
prediction: dog
EBM
prediction: horse
Clean Image
Poisoned Image
Occlusions
Integrated Gradients
0.0
0.5
1.0
Clean
prediction: bird
No Defense
prediction: dog
Epic
prediction: dog
Friendly
prediction: bird
EBM
prediction: bird
Clean Image
Poisoned Image
Occlusions
Integrated Gradients
0.0
0.5
1.0
Clean
prediction: airplane
No Defense
prediction: dog
Epic
prediction: dog
Friendly
prediction: dog
EBM
prediction: airplane
Clean Image
Poisoned Image
Occlusions
Integrated Gradients
0.0
0.5
1.0

C.5.1 Poisoned Parameters Diverge

[YLM22] proposes a subset selection method EPIc which rejects poison points through training. EPIc produces coresets, that under the PL* condition ($\frac{1}{2}\|\nabla_\phi \mathcal{L}(\phi)\|^2 \geq \mu\mathcal{L}(\phi), \forall\phi$), when trained on converges to a solution ϕ^* with similar training dynamics to that of training on the full dataset. While such a property is attractive for convergence guarantees and preserving the overall performance of the NN, converging with dynamics too close to the poisoned parameters may defeat the purpose of a defense. As such we consider the closeness of a defended network's parameters ϕ^* to a poisoned network's parameters ϕ by measuring the L1 distance at the end of training ($\|\phi - \phi^*\|_1$). All distances use the same parameter initialization and are averaged over 8 models from the first 8 classes of the Narcissus poison. In Figure C.5, we specifically consider increasingly higher percentiles of the parameters that moved the furthest away ($\phi_{nth\%}, \phi^*_{nth\%}$). The intuition is that poisons impact only a few key parameters significantly that play an incommensurate role at inference time, and hence we would only need to modify a tail of impacted parameters to defend. As we move to increasingly higher percentiles, both the PureEBM and FriendS defense mechanisms show a greater distance away from the poisoned model weights, indicating significant movement in this long tail of impacted parameters. We find that, as theory predicts, defending with coresets methods yield parameters that are too close to the poisoned parameters ϕ leading to suboptimal defense.

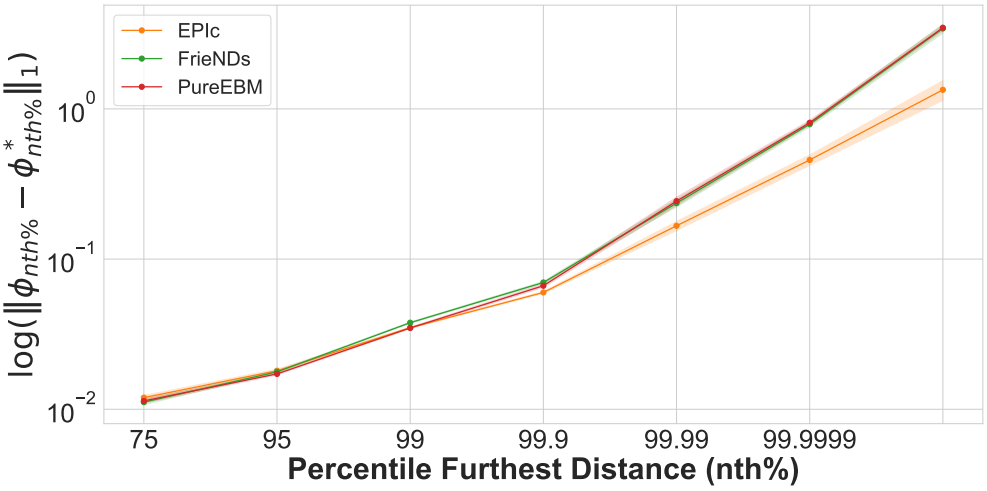


Figure C.5: Comparing parameter distances from defended models to poisoned model (same init) for increasing percentiles of the most moved parameters. PureEBM-trained models show the least movement in the tail of parameter which poisons are theorized to impact most (followed very closely by FrieNDs but well above EPIc).

C.6 EBM Langevin Dynamics Grid Searches

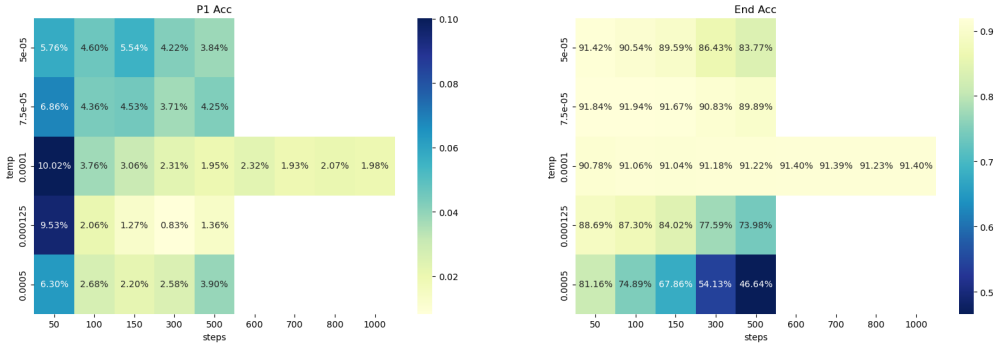


Figure C.6: Grid Search for Langevin steps and temp on Narcissus Fine-Tune Transfer

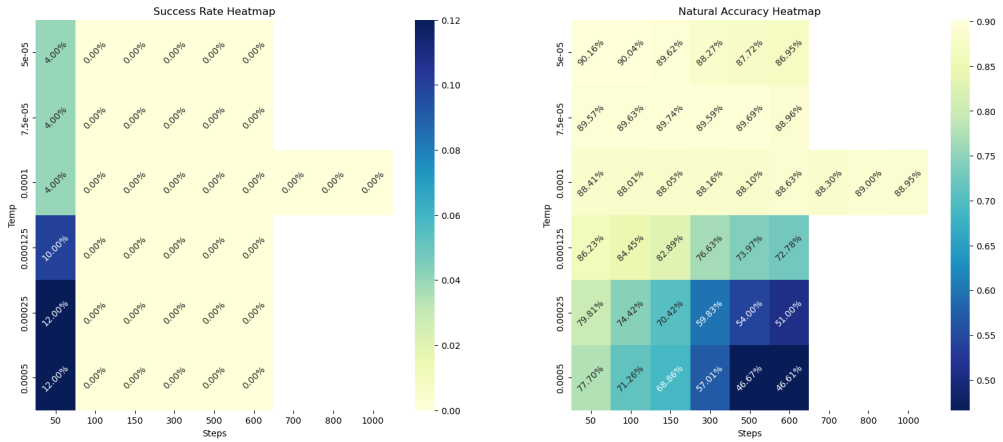


Figure C.7: Grid Search for Langevin steps and temp on Bullseye Polytope Fine-Tune Transfer

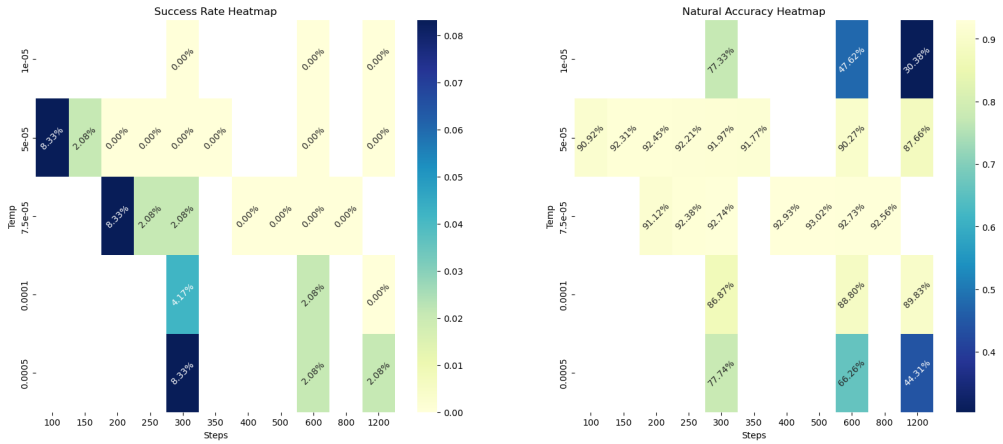


Figure C.8: Grid Search for Langevin steps and temp on Bullseye Polytope Linear Transfer

C.7 Poisoned PureEBM

Given a dataset $x \in \mathcal{X}$ where all samples x have been poisoned, we consider what happens if we train an EBM on $\mathcal{X}$. Specifically, we consider if the fully poisoned PureEBM can 1) purify given poisoned images and 2) how the energies estimated by the poisoned PureEBM compare to that of a clean PureEBM. We see in C.9 that the energies predicted by a poi-

soned PureEBM (left) are significantly closer to clean images compared to estimates from a clean PureEBM (right). This offers us some insight into how the poisoned PureEBM method works so effectively, counter to initial intuition. When we train a PureEBM on clean images we are learning some sampling trajectory towards the maximum likelihood manifold of the clean dataset i.e. when we sample from a clean PureEBM via Langevin Dynamics we move the input image in the direction of an expected clean image. When we train on a fully poisoned dataset it becomes unclear what should happen. Theoretically, if the poison distribution is perfectly learned, one should learn a trajectory toward a poisoned distribution. That is, if one gives a clean image to the poisoned PureEBM, sampling from it should move the clean image towards the poisoned distribution, and the image could become poisoned itself. Another byproduct is that poisoned images, since they have been trained on, should have a low energy. From Figure C.9 left we see that the energies of the poisoned images are much lower than that of Figure 7.1, reproduced here (Fig. C.9 right).

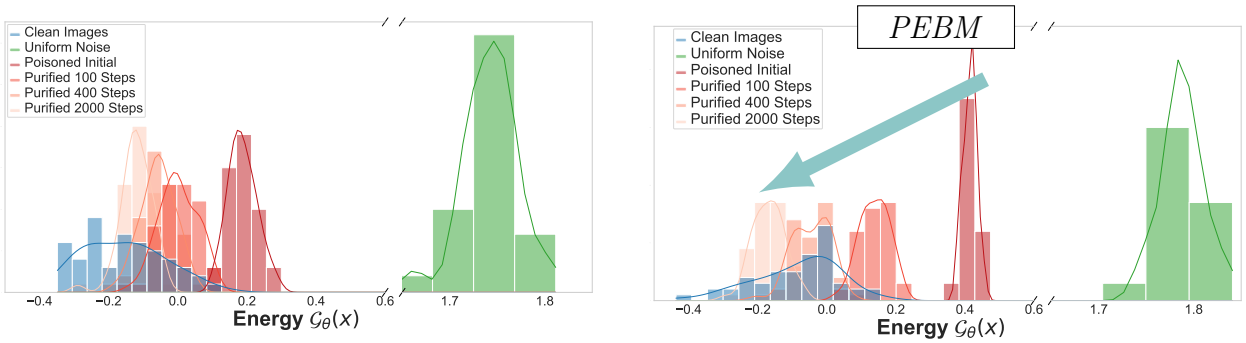


Figure C.9: Energies of poisoned points estimated by a poisoned PureEBM are much closer to clean points than that of poisoned points estimated by a clean PureEBM.

From Tables in C.2 we see that poisoned PureEBMs can perform nearly as well as clean PureEBMs. This means that the reduced energy gap between poisons and clean images in this setting does not hurt the purification process. Thus, the purification process remains universal.

C.8 Potential Social Impacts

Poisoning has the potential to become one of the greatest attack vectors to AI models. As the use of foundation models grows, the community is more reliant on large and diversely sourced datasets, often lacking the means for rigorous quality control against subtle, imperceptible perturbations. In sectors like healthcare, security, finance, and autonomous vehicles, where decision making relies heavily on artificial intelligence, ensuring model integrity is crucial. Many of these applications utilize AI where erroneous outputs could have catastrophic consequences.

As a community, we hope to develop robust generalizable ML algorithms. An ideal defense method can be implemented with minimal impact to existing training infrastructure and can be widely used. We believe that this research takes an important step in that direction, enabling practitioners to purify datasets preemptively before model training with state-of-the-art results to ensure better model reliability. The downstream social impacts of this could be profound, dramatically decreasing the impacts of the poison attack vector and increasing broader public trust in the security and reliability of the AI model.

The poison and defense research space is certainly prone to 'arms-race type' behavior, where increasingly powerful poisons are developed as a result of better defenses. Our approach is novel and universal enough from previous methods that we believe it poses a much harder challenge to additional poison crafting improvements. We acknowledge that this is always a potential negative impact of further research in the poison defense space. Furthermore, poison signals are sometimes posed as a way for individuals to secure themselves against unwanted or even malicious use of their information by bad actors training AI models. Our objective is to ensure better model security where risks of poison attacks have significant consequences. But we also acknowledge that poison attacks are their own form of security against models and have ethical use cases as well.

This goal of secure model training is challenging enough without malicious data poisoners

creating undetectable backdoors in our models. Security is central to being able to trust our models. Because our universal method neutralizes all SoTA data poisoning attacks, we believe our method will have a significant positive social impact to be able to inspire trust in widespread machine learning adoption for increasingly consequential applications.

Algorithm 2 ML with SGD for Convergent Learning of EBM (7.6)

Require: ConvNet potential $\mathcal{G}_\theta(x)$, number of training steps $J = 150000$, initial weight θ_1, training images $\{x_i^+\}_{i=1}^{N_{\text{data}}}$, data perturbation $\tau_{\text{data}} = 0.02$, step size $\tau = 0.01$, Langevin steps $T = 100$, SGD learning rate $\gamma_{\text{SGD}} = 0.00005$.

Ensure: Weights θ_{J+1} for energy $\mathcal{G}_\theta(x)$.

Set optimizer $g \leftarrow \text{SGD}(\gamma_{\text{SGD}})$. Initialize persistent image bank as N_{data} uniform noise images.

for j=1:(J+1) **do**

1. Draw batch images $\{x_{(i)}^+\}_{i=1}^m$ from training set, where (i) indicates a randomly selected index for sample i, and get samples $X_i^+ = x_{(i)} + \tau_{\text{data}}\epsilon_i$, where i.i.d. $\epsilon_i \sim \text{N}(0, I_D)$.

2. Draw initial negative samples $\{Y_i^{(0)}\}_{i=1}^m$ from persistent image bank. Update $\{Y_i^{(0)}\}_{i=1}^m$ with the Langevin equation

$$Y_i^{(k)} = Y_i^{(k-1)} - \Delta\tau \nabla_{Y_\tau} f_{\theta_j}(Y_i^{\tau-1}) + \sqrt{2\Delta\tau}\epsilon_{i,k},$$

where $\epsilon_{i,k} \sim \text{N}(0, I_D)$ i.i.d., for K steps to obtain samples $\{X_i^-\}_{i=1}^m = \{Y_i^{(K)}\}_{i=1}^m$. Update persistent image bank with images $\{Y_i^{(K)}\}_{i=1}^m$.

3. Update the weights by $\theta_{j+1} = \theta_j - g(\Delta\theta_j)$, where g is the optimizer and

$$\Delta\theta_j = \frac{\partial}{\partial\theta}\left(\frac{1}{n}\sum_{i=1}^n f_{\theta_j}(X_i^+) - \frac{1}{m}\sum_{i=1}^m f_{\theta_j}(X_i^-)\right)$$

is the ML gradient approximation.

end for

APPENDIX D

SimEdu

D.1 DQN and SimEdu

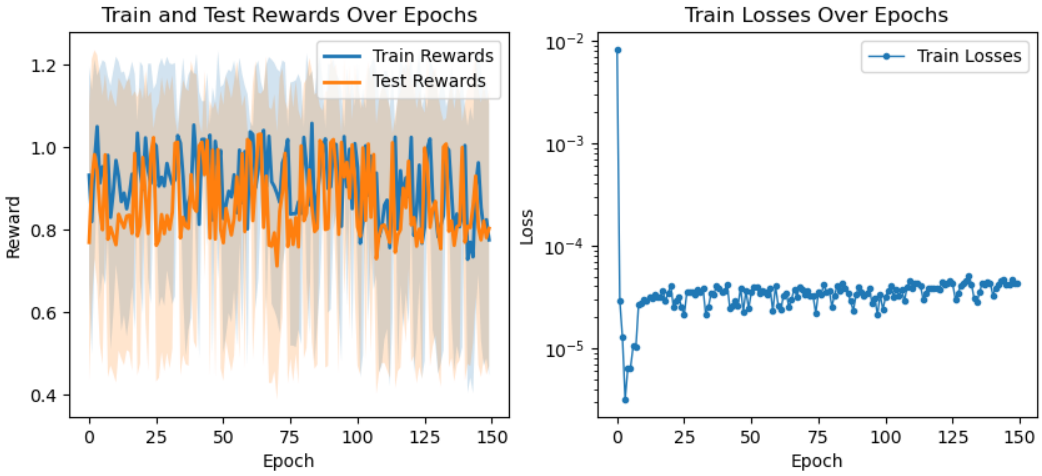


Figure D.1: Example DQN Training Trajectory for DQN without probing capabilities on an unobserved course. Error bars represent the one standard deviation away from the mean across 1000 simulated students.

We observed that DQN, when used with SimEdu, produces a very unstable and inconsistent training process. An example training trajectory is shown in Figure D.1. Notice that within 25 epochs, the training loss has stabilized for the most part. However, the train and test rewards are highly inconsistent. Based on these results, we devise an evaluation metric to choose the best RL agent across the epochs. This evaluation metric is a weighted average between the average test reward, the median test reward, and the pass rate. Part

of the difficulty comes from the design of the reward function, Equation 8.4. Because of the $\mathbb{1}_{G \geq G_{pass}}$, there is a steep drop in reward when the total grade is low. Therefore, the variance of rewards can be very large in a specific epoch.

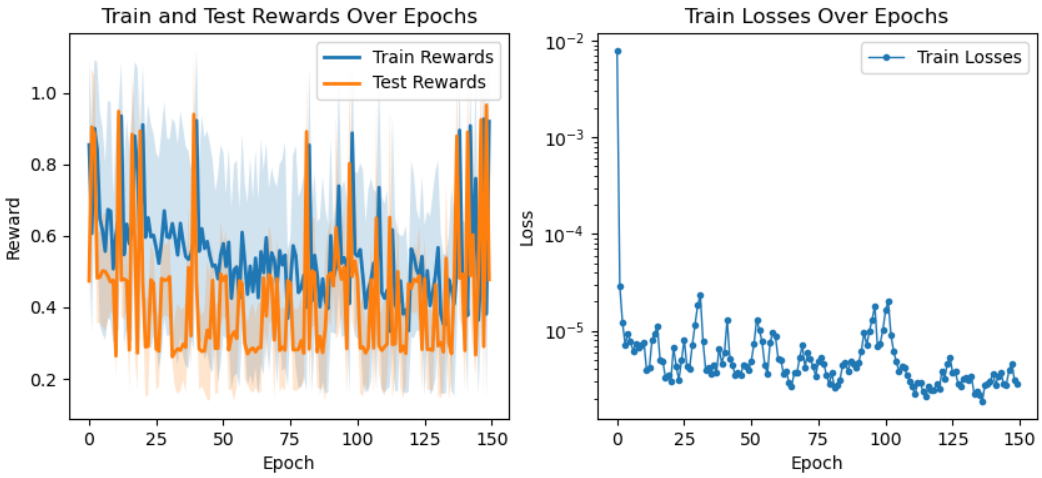


Figure D.2: Example DQN Training Trajectory for DQN with full probing capabilities on an unobserved course. Error bars represent the one standard deviation away from the mean across 1000 simulated students.

However, most of the inconsistency is likely due to the difficulty of partial observability for RL. In particular, we noticed that with a DQN that is capable of (oracle) probing, such as the one in Figure D.2, the inconsistencies are even greater, and they tend to perform worse, even though they have actions that should help it within the dynamics. In the implementation, the DQN without probing capabilities has a reduced action space, which overall reduces the required search space. The low immediate impact of individual probes introduces several problems. First, introducing probes elongates the total number of steps while training, diluting rewards even further backward and requiring additional searches. Second, because probes can change the state so little, they can produce loop structures in the RL search space, causing possible confusion and continuous probing. Overall, the intuitive thought where RL is greedier in terms of immediate rewards shines through with the variability of results. We acknowledge that further hyperparameter tuning is required

and the RL agents with the larger action space likely require longer training time.