UNIVERSITY OF CALIFORNIA

Los Angeles

Robust Modeling through Causal Priors and Data Purification in Machine Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Sunay Gajanan Bhat

2024

ABSTRACT OF THE DISSERTATION

Robust Modeling through Causal Priors
and Data Purification in Machine Learning

by

Sunay Gajanan Bhat
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Gregory J. Pottie, Chair

The continued success and ubiquity of machine learning techniques, particularly Deep Learning, have necessitated research in robust model training to enhance generalization capabilities and security against incomplete data, distributional shifts, and adversarial attacks. This thesis presents two primary sets of contributions to robust modeling in machine learning through the use of causal priors and data purification with generative models such as the Variational Autoencoder (VAE), Energy-Based Model (EBM), and Denoising Diffusion Probabilistic Model (DDPM), focusing on image datasets. In the first set of contributions, we use structural causal priors in the latent spaces of VAEs. Initially, we demonstrate counterfactual synthetic data generation outside the training data distribution. This technique allows for the creation of diverse and novel data points, which is critical to enhancing model robustness and generalization capabilities. We utilize a similar VAE architecture to compare causal structural (graphical) hypotheses, showing that the fit of generated data from various hypotheses on distributionally shifted test data is an effective method for hypothesis comparison. Additionally, we explore using augmentations in the latent space of a VAE as an efficient and effective way to generate realistic augmented data. The second set of contributions focuses on data purification using EBMs and DDPMs. We propose a framework of universal data purification methods to defend against train-time data poisoning attacks. This framework utilizes stochastic transforms realized via iterative Langevin dynamics of EBMs, DDPMs, or both, to purify poisoned data with minimal impact on classifier generalization. Our specially trained EBMs and DDPMs provide state-of-the-art defense against various poisoning attacks while preserving natural accuracy. Preprocessing data with these techniques pushes poisoned images into the natural, clean image manifold, effectively neutralizing adversarial perturbations. The framework achieves state-of-the-art performance without needing attack or classifier-specific information, even when the generative models are trained on poisoned or distributionally shifted data. Beyond defense against data poisoning, our framework also shows promise in applications such as the degradation and removal of unwanted intellectual property. The flexibility and generality of these data purification techniques represent a significant step forward in the adversarial model training paradigm. All of these methods enable new perspectives and approaches to robust machine learning, advancing an essential field in artificial intelligence research.

The dissertation of Sunay Gajanan Bhat is approved.

Jonathan Chau-Yan Kao

Achuta Kadambi

Lieven Vandenberghe

Gregory J. Pottie, Committee Chair

University of California, Los Angeles

2024

To my loving wife and better half, Cindy,

to my supportive parents, Vanita and Gajanan, and proud sister, Sneha

and my goofy dog Matcha.

TABLE OF CONTENTS

1 Thesis Overview **1**

1.1 Motivation 1

1.2 Research Objectives and Contributions 2

1.3 Thesis organization 3

2 Background **5**

2.1 Causality and Causal Machine Learning 5

2.1.1 Origins of Causality and Structural Equation Modeling 5

2.1.2 Structural Causal Modeling 6

2.1.3 Causality and Machine Learning 7

2.2 Deep Learning 9

2.3 Preliminary Work on Structural Causal Neural Networks 11

2.3.1 Introduction 11

2.3.2 Structural Causal Neural Networks (SCNNs) 12

2.3.3 Experiments 13

2.3.4 Experimental Results 15

2.3.5 Discussion 16

2.4 Generative Modeling and Causality 17

2.4.1 Generative Models 17

2.4.2 Reformulating Generalization 22

2.4.3 Causality and Generative Modeling 22

2.5 Adversarial Training and Data Purification 23

2.5.1 Adversarial Optimization 24

2.5.2 Adversarial Defense 24

2.5.3 Connecting Adversarial Training and Causal Machine Learning 25

3 De-Biasing Generative Models using Counterfactual Methods 26

3.1 Introduction 26

3.2 Related Work 27

3.3 Background 28

3.3.1 Counterfactuals and Interventions 28

3.3.2 Counterfactual Models 29

3.3.3 Constructing a Causal Generative Model 30

3.3.4 Causal Estimation 32

3.4 Problem Setting 32

3.4.1 Sun Pendulum Image Dataset 32

3.4.2 Tabular National Study of Learning Mindsets Data 33

3.5 Causal Counterfactual Generative Model 35

3.5.1 Limits of the CausalVAE for Counterfactuals 35

3.5.2 Envisioning Bias-Free Models with CausalVAE 36

3.5.3 General Structure of CCGM 37

3.6 Experiments 37

3.6.1 CausalVAE 38

3.6.2 Label-to-Label 39

3.6.3 Label-to-Image 41

3.6.4 Mindset Data 43

3.7 Conclusion 45

4 Hypothesis Testing using Causal and Causal Variational Generative Models 47

4.1 Introduction 47

4.2 Background 48

4.2.1 Causality and Model Hypothesis Testing 48

4.3 Causal Hypothesis Gen and Variational Model 49

4.3.1 Causal Hypothesis Testing with CSHTest 49

4.3.2 Causal Variational Hypothesis Testing with CSVHTest 50

4.4 Problem Setting 51

4.4.1 Structural Hamming Distance 51

4.4.2 Baseline Models 52

4.4.3 Train/Test Data Splits 53

4.5 Experiments 54

4.5.1 Generalization Ability of CSHTest 55

4.5.2 Simulated DAG Hypothesis Testing 56

4.5.3 Pendulum Hypothesis Testing 57

4.5.4 Medical Data Hypothesis Testing 58

4.6 Conclusion 58

5 Towards Composable Distributions of Latent Space Augmentations . . . 60

5.1 Introduction 60

5.2 Related Work 61

5.3 Background 62

5.3.1 Conditional Variational Autoencoders 62

5.3.2 Priors in Pre-processing 63

5.4 Latent Augmentation VAE 64

5.4.1 Architectural Overview 64

5.4.2 Training LAVAE 64

5.4.3 Utilizing LAVAE 65

5.5 Experiments 67

5.5.1 LAVAE Reconstruction Results 67

5.5.2 LAVAE Transfer Decoders 68

5.5.3 Conditional VAE Comparisons 69

5.5.4 LAVAE Latent Geometries 71

5.6 Conclusion 75

6 PureEBM: Robust Train-Time Poison Purification via Mid-run Dynamics of Energy-Based Models 76

6.1 Introduction 76

6.2 Related Work 79

6.2.1 Targeted Data Poisoning Attack 79

6.2.2 Defense Strategies 80

6.3 PureEBM: Purifying Langevin Defense against Poisoning Attacks 82

6.3.1 Energy-Based Model 83

6.3.2 Classification with Stochastic Transformation 85

6.3.3 Why EBM Langevin Dynamics Purify 85

6.3.4 Erasing Poison Signals via Mid-Run MCMC 86

6.4 Experiments 88

6.4.1 Experimental Details 88

6.4.2 Benchmark Results 89

6.4.3 Results on Additional Models and Datasets 90

6.4.4 Further Experiments 92

6.5 Conclusion 94

7 PureGen: Universal Data Purification Using Generative Model Dynamics 96

7.1 Introduction 96

7.2 PureGen-DDPM: Modified Denoising Diffusion for Data Purification . . . 96

7.3 PureGen: Generative Model Based Data Purification for Poison Defense . . 99

7.3.1 Classification with Stochastic Transformation 99

7.3.2 Comparing PureGen-EBM and PureGen-DDPM Dynamics . . . 99

7.3.3 PureGen-DDPM Results 101

7.4 Comparing PureGen-EBM and PureGen-DDPM 103

7.4.1 Robustness to Distributional Shifts 104

7.4.2 Robustness to Poisoning 105

7.4.3 PureGen Timing and Limitations 106

7.5 PureGen Combinations: PureGen-Naive, PureGen-Reps, PureGen-Filt 107

7.6 PureGen for Intellectual Property Purification 109

7.6.1 Problem Setting 109

7.6.2 Analysis and Results 111

7.7 Conclusion 113

8 Conclusion . . . 115

A De-Biasing Generative Models using Counterfactual Methods . . . 118

A.1 Additonal Results . . . 118

A.2 Tabular Variable Reconstructions . . . 118

B Hypothesis Testing using Causal and Causal Variational Generative Models 121

B.1 DAG Simulation Results . . . 121

B.1.1 DAG Size 5x5: Linear . . . 121

B.2 More about the Pendulum . . . 122

B.2.1 Pendulum Dataset . . . 122

B.2.2 Pendulum Hypotheses . . . 123

B.2.3 Pendulum Training Architecture and Hyperparameters . . . 124

B.2.4 Further Pendulum Results . . . 124

B.3 Background Theory . . . 126

B.3.1 Variational Hypothesis testing and Data Generation with CSVHTest 126

B.3.2 Constructing a Causal Generative Model . . . 127

B.4 Causal Problems and the Investigation of Optimizers . . . 128

B.5 DAG Simulation Settings . . . 130

B.5.1 Training Hyperparameters . . . 130

B.5.2 Test Cases . . . 132

B.6 Final Losses with Sample Variances . . . 133

C Towards Composable Distributions of Latent Space Augmentations . . . 134

C.1 Variational Autoencoders (VAEs) . . . 134

C.2 Conditional VAE (CVAE) 135

C.3 Architecture and Training 137

C.4 Sampling and Interpolation 139

C.5 Additional Augmentation Reconstructions 141

C.6 Additional Transfer Decoder Results 143

C.7 Latent Space Geometries 143

D PureEBM: Universal Poison Purification via Mid-run Dynamics of Energy-Based Models 148

D.1 EBM Further Background 148

D.1.1 Chaotic Dynamics 148

D.1.2 EBM Purification is a Convergent Process 150

D.2 Additional Results 152

D.2.1 Full Results Primary Experiments 152

D.2.2 Extended Poison% Results 154

D.2.3 Full MobileNetV2 and DenseNet121 Results 155

D.2.4 Full CINIC-10 Results 157

D.3 Further Experimental Details 158

D.3.1 EBM Training 158

D.3.2 Poison Sourcing and Implementation 159

D.3.3 Training Parameters 161

D.4 Timing Analysis 161

D.5 Additional Model Interpretability Results 163

D.5.1 Poisoned Parameters Diverge 165

D.6 EBM Langevin Dynamics Grid Searches 166

E PureGen: Universal Data Purification Using Generative Model Dynamics 168

E.1 Potential Social Impacts 168

E.2 Training Parameters 169

E.3 Core Results Compute 170

E.4 Additional Results 170

E.4.1 Full Results Primary Experiments 170

E.4.2 From Scratch 80 Epochs Experiments 172

E.4.3 Full Results for MobileNetV2 and DenseNet121 174

E.4.4 PureGen Combos on Increased Poison Power 175

E.5 PureGen-Naive,PureGen-Reps,PureGen-Filt T Param Sweeps 175

E.6 PureGen for Intellectual Property Purification 178

References 179

LIST OF FIGURES

2.1 Directed Acyclic Graph Example (from [Pea10]) 6

2.2 DNN (left) vs. SCM DAG only (right) 12

2.3 Structural Causal Neural Network (SCNN) 12

2.4 Prelim Exam Structural Model: DAG and Equations 13

2.5 SCNN test losses vs η showing a decrease in endogenous loss as we skew more to local gradients, with very moderate sacrifice of final output loss 17

2.6 Variational Autoencoder Architecture with encoder, decoder, and a mean/variance parameterizing a latent space. 18

3.1 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. 29

3.2 Example of a counterfactual model in which a single path is removed to simulate a new distribution of generated data. 29

3.3 Pendulum toy image dataset DAG and example image 33

3.4 School Mindset DAG 34

3.5 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 of the 0 intervention, other interventions do not seem to make sense, especially with the shadows. 38

3.6 Result which still shows effect propagation (shadow moves) after removing the path from sun location to shadow location in CausalVAE method 39

3.7 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 debiased 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 of the intervened images suggests that the midpoint of the shadow remains constant throughout all of the swept images. However, it is worth noting that the shadow width still responds as if the shadow had moved to its location. 40

3.8 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 than the expected locations given the sun. 41

3.9 Interventions in the label-to-image VAE. Note that shadows respond to pendulum angle and sun position. 41

3.10 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 debiased 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. 42

3.11 Label-to-image (Top) Image response to a sweep of sun positions. Again, notice that the shadow responds to the sun's position. (Bottom) image response after breaking the sun position to shadow position link. 43

3.12 Generated distribution of achievement scores closely matches the distribution in the original dataset. 44

3.13 Bootstrap distribution of ATEs for various methods and baselines sampling the entire dataset (n=10391) 100 times. 44

4.1 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. 51

4.2 Two hypothesized structural causal priors for a medical dataset on trauma patients and the decision to perform surgery, H1 and H2. 54

4.3 a) A 75% data-split on the pendulum angle feature (grey is training angle, green is testing angles b) A 75% data-split on the Diastolic Blood Pressure data. . . . 54

4.4 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. 56

4.5 Final OOD Test Error Rates of Each Hypothesized DAG structure in the Pendulum Problem over Two Splits. See Appendix B.2.4 for numerical values and training trajectories. 58

4.6 **Medical Dataset Results:** *Left:* Medical results of CSHTest for each of the hypothesized causal models. *Right:* The test dataset reconstruction distributions for all models using medical H1 on the Diastolic BP 75% data 59

5.1 Approximate model/DAG *learned* in latent space for *known* image space augmentations 61

5.2 Latent Augmentation VAE Architecture 64

5.3 Eight samples of "Flips" latent augmentations with baseline image space augmentations for comparison 68

5.4 Eight samples of "Flips" latent inverse augmentations with original and augmented images (inputs) for comparison 69

5.5 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. 70

5.6 Eight samples of "Flips" latent augmentations with baseline image space augmentations for comparison 71

5.7 Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations) 72

5.8 Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations) 73

5.9 LAVAE vs CVAE reconstructions 73

5.10 "Flips" (left) and "shear X-direction, canny edge-detect" (right) 2-d projections using PCA, ICA, and T-SNE algorithms. 74

6.1 **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. 78

6.2 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. 87

6.3 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. 93

6.4 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. 94

7.1 **Top** We compare PureGen-DDPM forward steps with the standard DDPM where 250 steps *degrades images for purification but does not reach a noise prior. Note that all model are trained with the same linear β schedule.* **Bottom Left** Generated images from sampled prior for models with 250, 750, and 1000 (Standard) train forward steps *where it is clear the 250-step model does not generate realistic images.* **Bottom Right** Significantly improved poison defense performance of PureGen-DDPM with 250 train steps indicating a trade-off between data purification and generative capabilities. 98

7.2 Plot of ℓ_2 distances for PureGen-EBM (**Left**) and PureGen-DDPM (**Right**) 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 Langevin dynamics 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. . . . 100

7.3 PureGen-EBM vs. PureGen-DDPM with increasingly Out-of-Distribution training data (for generative model training) and purifying target/attacked distribution CIFAR-10. PureGen-EBM is much more robust to distributional shift for natural accuracy while both PureGen-EBM and PureGen-DDPM maintain SoTA poison defense across all train distributions **CIFAR-10 is a "cheating" baseline as clean versions of poisoned images are present in training data.* 105

7.4 Tiny-ImageNet "NFL Logos" dataset samples (64x64) at various mix ratios and logo dimensions. 110

7.5 Both PureGen-EBM and PureGen-DDPM can collapse logo classifier accuracy with though steps for the middle mixing rations (0.4-0.6), with PureGen-DDPM 125 steps extending out to a mixing ratio of 0.6, but also generally sacrificing more natural accuracy the PureGen-EBM. 111

7.6 Examples of PureGen on an IP data sample showing the unique behavior of PureGen-EBM and PureGen-DDPM and their potential logo purification capabilities. 112

A.1 Visualization of the latent space of the causal concepts 118

A.2 School mindset distribution CCGM reconstruction example 119

A.3 Success Expectation distribution CCGM reconstruction example 119

A.4 Intervention probability distribution CCGM reconstruction example. Note that top 30% is used to bin data into binary before calculating ATEs. 120

B.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. 121

B.2 Pendulum toy example. (a) The structural DAG dictating the pendulum tabular dataset. (b) A visual representation of one of the results from the pendulum dataset. 122

B.3 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. 123

B.4 Loss trajectory of the Sun Split OOD Run 125

B.5 Loss trajectory of the Shadow Position Split OOD Run 125

B.6 Comparison of CSHTest, CSVHTest, NN, and VAE in the presence of noise . 129

B.7 Comparison of final test loss of optimizers PSGD and AdaBelief across 176 unique tests 130

B.8 Example of loss curves for optimizers PSGD and AdaBelief inf,1 dB SNR and 0,1 Hamming Distance. 131

C.1 Conditional VAE Architecture 135

C.2 All Augmentations Visualized 138

C.3 Sampled digits and augmentations via bounding box method 139

C.4 Interpolating between two test samples (top and bottom row) with augmentations 139

C.5 “Nested Mini-Image, Edge-Detect” Reconstructions 140

C.6 “90 deg Clockwise Rotation, Flip left/right” Reconstructions 141

C.7 “X-direction shear, Canny edge-detect” Reconstructions 142

C.8 Additional transfer decoder head results 143

C.9 “Flips” Image, Latent, and Reconstructions Image 2-D Projections 144

C.10 “Nested Mini-Image, Edge-Detect” Image, Latent, and Reconstructions Image 2-D Projections 145

C.11 “90° Clockwise Rotation, Flip left/right” Image, Latent, and Reconstructions Image 2-D Projections 146

C.12 “X-direction shear, Canny edge-detect” Image, Latent, and Reconstructions Image 2-D Projections 147

D.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. 150

D.2 Random Noise initialization of purification process 151

D.5 Comparing parameter distances from defended models to poisoned model (same init) for increasingly higher 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). 166

D.6 Grid Search for Langevin steps and temp on Narcissus Fine-Tune Transfer . . . 166

D.7 Grid Search for Langevin steps and temp on Bullseye Polytope Fine-Tune Transfer 167

D.8 Grid Search for Langevin steps and temp on Bullseye Polytope Linear Transfer . 167

E.1 PureGen-Reps Sweeps with HLB Model on Narcissus 176

E.2 PureGen-Naive Sweeps with HLB Model on Narcissus 177

E.3 PureGen-Filt Sweeps with HLB Model on Narcissus 177

LIST OF TABLES

2.1 Validation and Test Losses for SCNN and Baseline FCNNs showing comparable performance with convergence checks but not without checks (SCNN*) in both linear and nonlinear datasets 15

2.2 Test losses showing how endogenous variable loss goes down as we utilize more local gradients (higher η), but sometimes at a marginal loss of global accuracy (prediction of pass variable Y) 16

3.1 Correlation of Mindset Variables 34

3.2 Mindset Average Treatment Effect (ATE) Results showing how CCGM with a latent intervention can produce "de-baised" or causally separated synthetic data 45

4.1 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. 55

5.1 “Flips” Reconstruction Errors (MSE) showing increased performance of LAVAE and composition capabiltiies 71

6.1 Poison success and natural accuracy in all poisoned training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 100 GM experiments, 50 BP experiments, and NS triggers over 10 classes. . . . 90

6.2 Poison success and natural accuracy when training on CINIC-10 Dataset From Scratch Results with NS Poison 91

6.3 MobileNetV2 and DenseNet121 results and HyperlightBench for a novel training paradigm where PureEBM is still effective. 92

7.1 Poison success and natural accuracy in all ResNet poison training scenarios. We report the mean and the standard deviations (as subscripts) of 100 GM experiments, 50 BP experiments, and NS triggers over 10 classes. 102

7.2 Results for additional models (MobileNetV2, DenseNet121, and HLB) and the NTGA data-availability attack. PureGen remains state-of-the-art for all train-time *latent* attacks, while NTGA defense shows near SoTA performance. *All NTGA baselines pulled from [DEL24]. 103

7.3 Both PureGen-EBM and PureGen-DDPM are robust to NS attack even when **fully poisoned (all classes at once)** during model training except for NS Eps-16 for PureGen-EBM 106

7.4 PureGen and baseline Timing Analysis on TPU V3 107

7.5 PureGen-Naive, PureGen-Reps, and PureGen-Filt results showing further performance gains on increased poison power scenarios 109

B.1 Mean and Standard Deviation of the Final Test Loss in Pendulum Hypothesis Testing Experiments. 126

D.1 Narcissus transfer fine-tune results at various poison%'s 154

D.2 BP transfer linear gray-box results at various poison%'s 154

D.3 MobileNetV2 Full Results 155

D.4 DenseNet121 Full Results 156

D.5 CINIC-10 Full Results 157

D.6 Median Wall Clock Train Times From Scratch 162

E.1 Compute Hours TPU V3 170

E.2 From-Scratch 80-Epochs Results (ResNet-18, CIFAR-10) 173

E.3 MobileNetV2 Full Results 174

E.4 DenseNet121 Full Results 174

E.5 PureGen Combos with Narcissus Increased Poison % and ϵ 175

E.6 Full Results for IP Purification at various logo dimension sizes 178

ACKNOWLEDGMENTS

The research presented in this thesis was made possible through the support of many people.

I would first like to express gratitude to my advisor, Professor Greg Pottie, for always being supportive and providing guidance during my time at UCLA. I want to thank my co-authors, Omead Pooladzandi and Jeffrey Jiang, for their close collaboration on the majority of work in this thesis and long discussions about countless research topics. I also would like to thank Alex, our undergraduate research assistant, for his enthusiastic help with running experiments and collecting results for the work in Chapter 7. I also thank the rest of my lab, classmates, and researchers from other groups who have provided assistance or advice in my research and academic endeavours.

My research at UCLA was supported with funding from multiple sources. I received a fellowship from the UCLA graduate division during my first year. I also received funding from the H. Boyd McWhorter Postgraduate Scholarship from the Southeastern Conference (SEC) and the NCAA Postgraduate Scholarship from the National Collegiate Athletics Association. This thesis is also supported with Cloud TPUs from Google's Tensorflow Research Cloud (TFRC) program, specifically the computationally heavy experiments in Chapters 6 and 7.

I'd like to thank all of my committee members for their support and feedback during my exams and the dissertation process. I want to thank UCLA Engineering and Electrical and Computer Engineering departments for support through my TAships over the years and for offering a great academic environment.

Finally, I would like to thank my family. My parents, Vanita and Gajanan Bhat, have always believed in me and given me the the confidence to pursue challenges. My sister, Sneha, has been a constant source of inspiration and support. And finally, my loving wife, Cindy, who continued to encourage me and always see the positive even when I struggled to. She would insist I mention our dog Matcha, who made sure I took research breaks to play ball.

VITA

2017	B.S. (Electrical and Computer Engineering under the Computer Engineering option), University of Tennessee, Knoxville. Summa Cum Laude.
2017–2020	Electro-Optical Engineer at Lockheed Martin, Santa Barbara Focalplane
2021	M.S. (Electrical and Computer Engineering), UCLA.
2021–2024	Teaching Assistant, Electrical and Computer Engineering, UCLA. Writing II TA for ECE-180DW, ECE-182W, and ECE-2W
2022	Machine Learning Engineer at Street Simplified, Inc., Pasadena, CA
2022	Research Assistant, Electrical and Computer Engineering, UCLA. Developed Capstone Engineering Writing Programs Curriculum Conducted Research on Concept Maps and Self-Assessments in undergraduate Engineering Courses
2023–*present*	Data-scientist and Machine Learning Engineer at StreetMetrics, Inc., Birmingham, AL.

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* Jiang J., Pooladzandi O., Bhat S., Pottie G. 2024.

PureGen*: Universal Data Purification for Train-Time Poison Defense via Generative Model Dynamics* Bhat S., Jiang J., Pooladzandi O., Branch A., Pottie G. Submitted to NeurIPS 2024.

CHAPTER 1

Thesis Overview

1.1 Motivation

Machine learning progress in recent years has been rapid, with unprecedented levels of performance achieved in various tasks such as image recognition, natural language processing, planning, and navigation. The public release of large-language and image-generative models was a clear inflection point. State-of-the-art models surpassed what many researchers believed possible from current techniques and made the broader public aware of the possibilities of machine learning and deep learning methods. However, as these models become more complex and are applied to increasingly critical domains, concerns about their interpretability, robustness, and generalizability have grown. The lack of transparency in deep learning models, often referred to as "black boxes," presents numerous challenges in fields where explainability and reliability are crucial, such as healthcare, finance, human interaction, and autonomous systems.

Researchers have investigated these challenges using techniques from multiple other fields, from classical artificial intelligence techniques to communications and information theory, to increase our understanding and trust in deep learning models. Causal inference and modeling have gained traction as one set of techniques to enable more interpretable and generalized model performance. Causal reasoning allows for a deeper understanding of the underlying mechanisms that generate data and can enable generalization beyond the training data domain, which remains a crucial challenge in data-sparse domains. Furthermore, by incorporating causal knowledge, researchers aim to develop models that are accurate and aligned with human intuition.

Another major challenge is the presence of biases and spurious correlations in the training data of machine learning models. Models trained on biased data can learn to perpetuate or amplify these biases, leading to inaccurate predictions. Therefore, it is essential to develop techniques to identify and mitigate biases in data and models, particularly in areas where large and diverse datasets might not exist to counteract biases.

As large and powerful models are deployed in increasingly critical applications, the risk of adversarial attacks on such models is simultaneously increasing. These attacks can cause poor performance or undetectable backdoors, producing unwanted or dangerous behavior. Our poor ability to interpret deep learning models, the massive datasets required for training, and the large parameter spaces of such models combine to create a large attack surface area for adversaries.

The promise of machine learning applications has generated considerable interest in solving these critical challenges. Entire subfields are dedicated to causal machine learning, model robustness, and defense against attacks to ensure generalization and performance despite biases, distributional shifts, or adversarial activity.

1.2 Research Objectives and Contributions

The primary objective of this thesis is to explore and develop methods for incorporating causal priors and data purification techniques into machine learning models, particularly generative architectures, to enhance their robustness, interpretability, and performance. Specifically, the research investigates the integration of causal knowledge into a popular generative architecture to enable the generation of counterfactuals and de-bias models even when bias-free data might not be present. We further look at methods for testing and comparing structural causal hypotheses using generative architectures and how augmentation can be performed in a model's latent space to enhance the generation of diverse, high-quality samples. Finally, we address the critical challenge of train data poisoning attacks and undesired content with generative model dynamics.

The main contributions of this thesis are:

- **CCGM:** A counterfactual-based method for de-biasing generative models with causal priors, enabling the generation of out-of-distribution samples from the training dataset.
- **CSHTest:** A causal structural hypothesis testing framework for comparing and evaluating structural causal models using generative deep learning and out-of-distribution data generalization.
- **LAVAE:** A latent space augmentation technique for generative models that allows for the composable generation of diverse and high-quality augmented samples efficiently.
- **PureEBM:** A robust data-purification defense mechanism against train-time data poisoning attacks that uses the stochastic dynamics of energy-based models.
- **PureGen:** A set of universal data-purification methods against attacks and undesired perturbations that uses the stochastic dynamics of energy-based models and diffusion models separately and in combination.

1.3 Thesis organization

This thesis is organized as follows. Chapter 2 provides a review of the literature, providing background in causality, causal modeling in machine learning, generative models, data poisoning attacks, energy-based models, and diffusion models, along with initial work on combining structural models with deep learning. Chapters 3 and 4 cover the main contributions of causal priors in deep learning. Chapter 5 explores latent augmentation research, and Chapters 6 and 7 discuss a new set of methods for data purification as a universal poison defense and extensions to other modalities and potential applications. Finally, Chapter 8 concludes the thesis, summarizing the main findings and outlining future research directions.

CHAPTER 2

Background

2.1 Causality and Causal Machine Learning

2.1.1 Origins of Causality and Structural Equation Modeling

Causality has a long philosophical history, with the first formal theory coming from Aristotle's *Physics and Metaphysics*, and a more modern treatment coming from 18th century philosopher David Hume. Hume describes causation as a regular connection between two events, A and B, such that B always follows A, but further, that A is always temporally antecedent to B, A and B are close in space and time, and A is necessary for B [Kle12]. Counterfactuals are a consequential aspect of causal theory, in which unobserved worlds are posited with alternate outcomes based on causal relationships (and the arguments we make for sufficient proxies to these counterfactual worlds). While it is tempting to describe causality in terms of pure probabilities, a purely probabilistic approach neglects the directionality inherent in causality, which Hume himself implicitly posited: A is necessary for B, *but B is not necessary for A*.

In the 1920s, Sewall Wright first used path coefficients (linear weights) with a directed graphical model while studying genetics in guinea pigs [Wri21]. Although he only used linear relationships between variables, his key insight was combining mathematical modeling with a directional, graphical model, or Structural Equation Models (SEMs).

2.1.2 Structural Causal Modeling

There are numerous modern treatments of causality, but Judea Pearl's generalization of SEMs from linear to non-parametric models, known as Structural Causal Models (SCMs), is a key aspect of causal machine learning. In Pearl's framework, we draw causal models as Directed-Acyclic Graphs (DAGs) and pair them with a set of equations for each variable or node in the graph. Unlike non-graphical approaches, we can operate on the graph directly, determining independence relationships - as well as corresponding dependencies - with no other assumptions but the graphical model itself.

$$\begin{tikzcd}
Z_1 \arrow[dr] \arrow[dd] & & Z_2 \arrow[dl] \arrow[dd] \\
& Z_3 \arrow[dl] \arrow[dr] & \\
X \arrow[rr] & & Y
\end{tikzcd}$$

Figure 2.1: Directed Acyclic Graph Example (from [Pea10])

Take the DAG example in Figure 2.1 in which we have 5 model variables. From the graph alone, we can write a truncated factorization of the joint distribution as

$$P(X, Z_1, Z_2, Z_3, Y) = P(Z_1)P(Z_2)P(Z_3|Z_1, Z_2)P(X|Z_1, Z_3)P(Y|Z_2, Z_3, X) \tag{2.1}$$

based on the graphical understanding that endogenous nodes are dependent only on their parent variables. Conditioning on variables in the path between two variables breaks their dependence. These simple rules give rise to powerful graphical theory such as the Backdoor Criterion, which tells us what is an admissible set of variables to condition on to measure a direct effect, an effect through the direct, forward path. Also of note is the Markov Boundary, which is the minimal admissible set that satisfies the Backdoor Criterion, which can be important when dealing with practical data limitations and the statistically expensive nature of conditioning [Pea09]. For instance, we see that the direct effect of X on Y can be measured by conditioning on a set $S = \{Z3, Z1\}$ or $S = \{Z3, Z2\}$. Conditioning on Z3 is needed as it is a confounder, but doing so opens a spurious correlation which must be blocked by also conditioning on either Z_1 or Z_2.

Graphical connections do not necessarily mean that an obvious correlation will exist in the data. Such measurements from data are subject to the functional relationships themselves, noise, sampling constraints, and, of course, the certainty in the model itself. From this comes a powerful intuition of information asymmetry, *it is the absence of connection in an SCM that provides us with the most information at the graphical level*, but this absence is also the hardest to prove or justify. Or, stated another way, a DAG could always be described as fully connected (still satisfying acyclicity) with weights of zeros where connections were previously absent. Having the confidence to remove a connection usually requires *overwhelming and domain-invariant* information.

2.1.3 Causality and Machine Learning

We can broadly categorize causal research into two categories: causal discovery and causal inference. Causal discovery involves extracting the graphical structure and estimating structural equations from observational data, while causal inference is concerned with the impact of changing a model variable on an outcome [NPR22]. Both have seen tremendous progress in the AI and deep learning community in recent years.

Causal discovery is arguably the more difficult problem, as there are numerous limitations to learning a DAG solely from observational data, and even in the best case, one can usually only learn a Markov Equivalence Class [HJY15]. This is the set of DAGs that encode the same set of conditional independences, effectively providing the skeleton, or the undirected graph. The absence of certain connections is still powerful information, but *directionality is one of the key aspects of causal priors.* Deep learning has also been applied to this problem space with some interesting and influential results. Perhaps the most notable contribution is a paper that suggested a function of a DAG adjacency matrix $\mathbf{G}$ [ZAR18]. The main use is that $\mathbf{G}$ is a DAG if and only if

$$H(\mathbf{G}) := tr\left[(\mathbf{I} + \mathbf{G} \circ \mathbf{G})^n\right] - n = 0 \tag{2.2}$$

So we can take what was previously a discrete, non-differentiable problem and apply the continuous differentiable loss function (2.3) via an augmented Lagrangian for learning the DAG

$$\ell_h = \lambda H(\mathbf{G}) + \frac{c}{2}|H(\mathbf{G})|^2 \tag{2.3}$$

balancing between the need to preserve acyclicity while ensuring we don't degenerate down to the identity matrix. This core idea of learning the DAG adjacency matrix via a differentiable loss function has been used in numerous follow-up works, including our own, as covered in Chapters 3 and 4.

Causal inference also has a growing body of research, largely concerned with estimating 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's ($index = i$) treated and untreated groups, assuming a binary intervention variable (D), and an outcome variable (Y) as in equation (2.4).

$$\widehat{ATE} = \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i|D_i = 0] \tag{2.4}$$

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 (Inverse Propensity Weighting - IPW) can then be used to weight each sample as in equation (2.5) and thus adjust for the bias of any *measured* confounders.

$$\widehat{ATE}_{IPW} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{D_i Y_i}{\hat{\pi}(X_i)} - \frac{(1-D_i)Y_i}{1-\hat{\pi}(X_i)}\right] \tag{2.5}$$

More recent developments introduce double-robust methods which 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 with a set of outcome models that estimate the outcome variable as a function of the intervention and all covariates [GQ10a]. Both the exposure/propensity score model and the outcome model can be trained as deep neural networks.

2.2 Deep Learning

Deep learning is a sub-field of machine learning that deals with the training of artificial neural networks with many layers [LBH15]. The addition of multiple layers enables the learning of complex, hierarchical representations of data. The origins of deep learning come from the single-layer perceptron, which is effectively a linear transform via a matrix multiplication along with a summation and an 'activation' function to compute some non-linear output value (for a given intermediate or output node). The invention of back-propagation enabled multi-layer perceptron training and the ability to train arbitrarily large artificial neural networks as universal function approximators [RHW86].

Deep learning tasks typically fall into classification, regression/prediction, or generation. Model training can occur in a supervised, unsupervised, or mixed (semi-supervised) setting in which data exist with (supervised) or without (unsupervised) labeling or categorization. Fundamentally, training data is utilized to fit a model for some task, optimized for a loss function. Formally, consider the empirical risk minimization formula

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{|\mathbf{D}_{train}|} \sum_{(x,y)\in\mathbf{D}_{train}} l(y, f(x;\theta)) \tag{2.6}$$

$$Loss(\theta, \mathbf{D}_{test}) = \frac{1}{|\mathbf{D}_{test}|} \sum_{(x,y)\in\mathbf{D}_{test}} l(y, f(x;\theta)) \tag{2.7}$$

in which we minimize a loss function 2.6 with respect to our model parameters θ using training data. The true goal is to minimize an identical loss 2.7 on an unseen test data set [Mur22]. This is *generalization* in a classical machine learning sense, but this presents challenges, as the test data is likely limited with respect to a vast or distributionally shifted intended domain.

Specific architectures were developed to better process certain patterns in data such as Convolutional Neural Networks (CNNs) for images and Recurrent Neural Networks (RNNs) and Transformers on sequential data and language [KSH12, VSP17]. Although such architectures have represented major performance milestones, a consistent theme in deep learning has been large performance gains from increasing model and data set size, typically following power law scaling [HKK20, KMH20]. Today's state-of-the-art (SoTA) models, exceeding trillions of parameters, consuming petabytes of data and exa-FLOPs of specialized compute, have achieved remarkable success in various domains, such as computer vision, natural language processing, and speech recognition, often exceeding human performance [HPB24, SHM16, HZR15, Len24].

Despite their impressive performance, deep learning models are often criticized for their lack of interpretability and robustness, particularly when adversarial attacks are involved [RHC23, LXG23]. The complex, non-linear nature of deep neural networks makes it difficult to understand how they predict or generate, which can be problematic in high-risk applications such as healthcare and autonomous driving. Deep learning models are highly over-parameterized, and can behave like non-parametric models with a tight fit to data [FC19]. They can pick up spurious correlation or noise despite regularization, which might generalize to the highly-overlapping test distribution, but is less applicable to a broader domain of interest. This makes high-performing models "fragile" or readily susceptible to adversarial attacks, where small, often imperceptible, perturbations of the input can cause the model to make incorrect predictions or undesired generations [RKH22, XML19]. These drawbacks inspired numerous research fields to make classical deep learning approaches more robust and easier to interpret while maintaining performance.

2.3 Preliminary Work on Structural Causal Neural Networks

2.3.1 Introduction

In this section, we explore Structural Causal Neural Networks (SCNNs), an early framework for modeling that combines the expressive power and function approximation capabilities of neural networks with causal information and priors expressed in Structural Causal Models (SCMs). As discussed, there is a growing desire for research in interpretability, modularity, and generality for reasons ranging from ethics to data efficacy [OCS20, OSJ18, LPK20, RGG, OSJ18]. SCNNs integrate causal information and insight into modeling offering a compelling path to more robust and data-efficient models. We explore the usage of SCNNs on toy models with a novel method for utilizing global and local gradient information where endogenous/latent variables might be observable.

A simple two-layer DNN architecture is shown in Figure 2.2. Besides high-level hyper-parameter and design choices, the parameters of the hidden nodes - often in the millions of billions for state-of-the-art models - have few constraints. The result is often an over-parameterized models that can achieve high loss performances but have complex,uninterpretable substructure, giving neural networks their nickname as "black-boxes".

In contrast to deep learning methods, causality has largely been researched in the symbolic and graphical context. Structural Causal Models [Pea00, Pea10] provide a systematic way to analyze causal Directed Acyclic Graphs (DAGs), independently of the functional relationships of the variables. Figure 2.2 on the right shows an example of the graphical portion of an SCM where the arrows represent a known causal relationship and, more importantly, the lack of arrows represents the absence of functional relationships (that is, independent variables). Paired with structural equations or functional relationships between variables, the SCM is a concise and highly interpretable representation of a system.

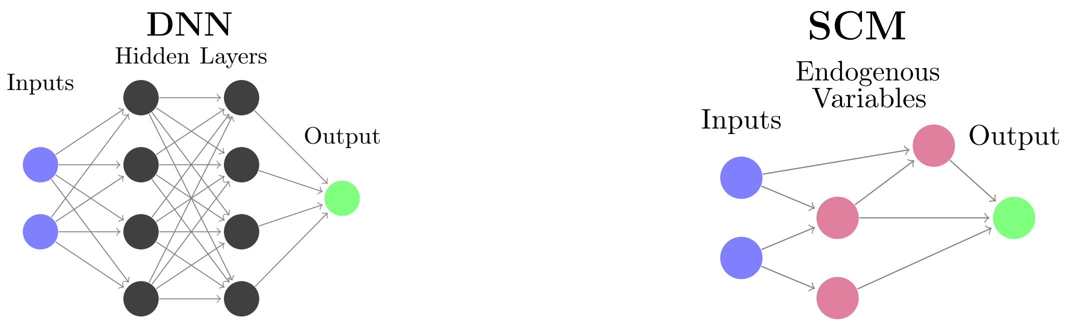


Figure 2.2: DNN (left) vs. SCM DAG only (right)

2.3.2 Structural Causal Neural Networks (SCNNs)

Structural Causal Neural Networks combine SCMs with neural networks and deep learning training methods to attempt to maintain invariances in the causal model with a more flexible training process, utilizing local and global gradient information based on the observability of endogenous variables. This is done by using "sub-neural-networks" as function approximators of the structural equations (functional relationships). Figure 2.3 visualizes an SCNN with many small neural networks placed at the inputs of each endogenous variable in the SCM. In the circumstance where the DAG of a causal model might be known or hypothesized, we can now use backpropagation to train the SCNN after selecting appropriately sized neural networks.

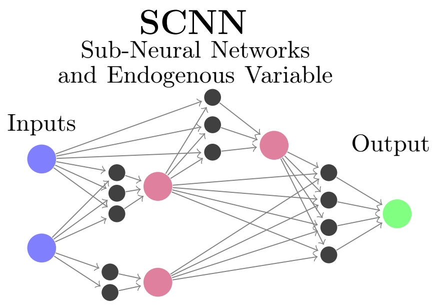


Figure 2.3: Structural Causal Neural Network (SCNN)

2.3.3 Experiments

2.3.3.1 Synthetic Data Model

Analysis is done on two simulated data models, one linear and one nonlinear, both of which share the same causal DAG but have different structural equations. The variables in the simulated toy model are labeled to represent a student taking an oral exam.

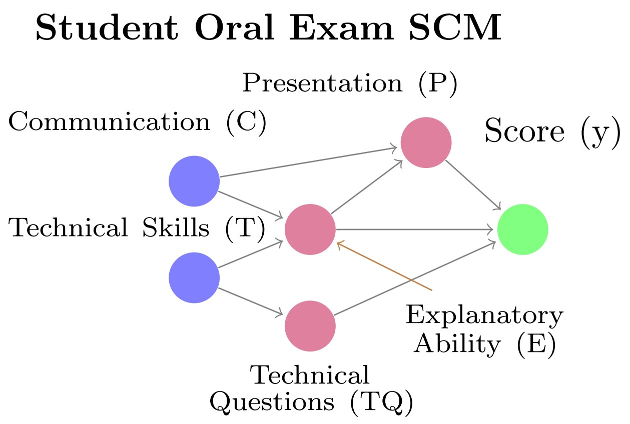


Figure 2.4: Prelim Exam Structural Model: DAG and Equations

The data model has initial inputs from a Gaussian distribution and additive Gaussian noise in all structural equations, where samples $N = 14000$ times for dataset sizes $train \sim 10000, val \sim 2000, test \sim 2000$. The inputs are given by $C = \mathcal{N}(0, 0.1)$ and $T = \mathcal{N}(0, 0.1)$ and the linear and nonlinear equations are given in equations 2.8 and 2.9.

$$\begin{aligned} E &= 2C + T + \mathcal{N}(4, 0.1) \\ TQ &= 2T + \mathcal{N}(0, 0.1) \\ P &= 1.5E + 0.5C + \mathcal{N}(0.5, 0.1) \\ Y &= 1.5P + 3E + TQ + \mathcal{N}(0, 0.1) \end{aligned} \tag{2.8}$$

$$\begin{aligned} E &= 2C^3 + T^2 + \mathcal{N}(0, 0.1) \\ TQ &= 2T + \mathcal{N}(0, 0.1) \\ P &= 1.5\sin(E) + 0.5\sin(C) + \mathcal{N}(0.5, 0.1) \\ Y &= 1.5P^3 + 3e^E + 2.5TQ + \mathcal{N}(0, 0.1) \end{aligned} \tag{2.9}$$

2.3.3.2 Trained Models and Optimization

A Fully Connected neural network (FCNN) was used for our baseline DNN for performance comparison. The FCNNs were optimized across the learning rate, width, and depth of the network - for the linear and nonlinear models. We developed a new class for the SCNNs which initializes and handles an SCNN object with a 'meta-level' of forward and back propagation between the sub-neural-networks. The SCNNs were also optimized across learning rate, width, and depth of the sub-neural-network for validation loss performance.

2.3.3.3 Multi-Gradient Optimization

We introduce a gradient local/global weight parameter η which allows us to slide between a weighted combination of the local and global gradients for each sub-neural-network NN_i^{sub} as in equation 2.10. If the latent/endogenous variables in the SCNN are unobserved, we resort to training as if it was a standard, sparsely connected NN ($\eta = 0$). On the other hand, if the observed variables are known, we could utilize that as a local loss/gradient for the latent variables. If we fully trusted the observed data, we could train only on the local loss ($\eta = 1$), which is equivalent to training each sub-neural-network independently.

$$\nabla_{NN_i^{sub}} = (1-\eta) \cdot \nabla_{NN_i^{sub}} \mathcal{L}_{global} + \eta \cdot \nabla_{NN_i^{sub}} \mathcal{L}_{local} \forall i \tag{2.10}$$

2.3.3.4 Ground Truth Analyses

To measure the ground truth approximation capability of each sub-neural-network, we forward propagate a verification vector which is an evenly spaced set of eleven values spanning the domain of the two inputs $C_{verify}, T_{verify} = [-0.4 : 0.08 : 0.4]$, with the values based on the generative input distribution $\mathcal{N}(0, 0.1)$. We compute the mean squared error difference between the verification values from the original structural equations (without additive noise) and the values of the equivalent nodes in the SCNN. We sum across the four hidden nodes to compute the total 'ground truth loss' $\mathcal{L}^{\mathcal{GT}}$. All experiments are run 10 times to get the mean and standard deviation of the losses.

$$\mathcal{L}^{\mathcal{GT}} = \sum_{latent} \frac{1}{n} \times \sum_{i=1}^{n} (SCNN[latent^i] - latent^i_{verify})^2 \tag{2.11}$$

2.3.4 Experimental Results

2.3.4.1 Basic Results ($\eta = 0$)

In Table 2.1 we show the test losses and final validation losses for both SCNNs, with ($SCNN$) and without ($SCNN*$) convergence checks. Convergence checks include results with validation loss < 0.7 and < 2 for linear and nonlinear, respectively. The two FCNN baselines that performed best on validation data are shown with the dimensions of the hidden layer(s) given in parentheses. These results, when $\eta = 0$ (no information about endogenous variables is known or used), indicate that SCNNs, when naively trained and optimized with a convergence check, produce results comparable to FCNNs (and marginally better in the linear case). But SCNNs are prone to divergence, or catastrophic failure, and without the convergence checks the divergent training runs dramatically increase error.

Table 2.1: Validation and Test Losses for SCNN and Baseline FCNNs showing comparable performance with convergence checks but not without checks (SCNN*) in both linear and nonlinear datasets

	Linear		**Nonlinear**	
	Validation	**Test**	**Validation**	**Test**
SCNN	**0.34 ± 0.02**	**0.34 ± 0.03**	1.74 ± 0.08	1.67 ± 0.07
SCNN*	97.27 ± 155.11	97.81 ± 155.95	3.45 ± 2.47	3.30 ± 2.38
FCNN [4]	0.38 ± 0.01	0.37 ± 0.01	**1.63 ± 0.04**	**1.61 ± 0.03**
FCNN [6, 6]	0.38 ± 0.02	0.38 ± 0.02	1.64 ± 0.05	1.63 ± 0.05

2.3.4.2 η vs. Performance

We further explore training with observable endogenous variable loss and using local and global loss information as in Equation 2.10. In Table 2.2 and Figure 2.5 we see the results of moving the weight η from 0 where we use only global gradient information to 1 where we use only local gradient information. As expected, utilizing local information greatly decreases local loss with a very marginal sacrifice of global loss (on the final Pass/Fail output Y). If all the data is observed, we could allow η to be a trainable parameter, which might tell our confidence in the local vs. global data, or confidence in the structural model itself.

Table 2.2: Test losses showing how endogenous variable loss goes down as we utilize more local gradients (higher η), but sometimes at a marginal loss of global accuracy (prediction of pass variable Y)

	Linear				**Nonlinear**			
	E	**TQ**	**P**	**Y**	**E**	**TQ**	**P**	**Y**
0.00	3.9045 ± 2.95	0.9858 ± 0.91	6.4518 ± 7.70	0.0453 ± 0.00	0.2777 ± 0.35	0.1849 ± 0.08	1.6961 ± 3.03	0.2479 ± 0.00
0.01	0.3857 ± 0.24	0.3924 ± 0.22	1.5012 ± 1.66	**0.0450 ± 0.00**	0.1033 ± 0.08	0.0955 ± 0.05	0.0912 ± 0.08	0.2483 ± 0.00
0.10	0.5255 ± 0.35	0.2620 ± 0.19	0.2337 ± 0.26	0.0461 ± 0.00	0.0102 ± 0.01	0.0192 ± 0.01	0.0085 ± 0.00	**0.2449 ± 0.00**
0.33	0.0749 ± 0.06	0.1253 ± 0.03	0.2606 ± 0.78	0.0452 ± 0.00	0.0033 ± 0.00	0.0055 ± 0.00	0.0064 ± 0.00	0.2494 ± 0.01
0.70	0.0041 ± 0.00	0.0185 ± 0.01	0.3466 ± 0.72	0.0474 ± 0.00	0.0016 ± 0.00	0.0017 ± 0.00	0.0053 ± 0.00	0.2510 ± 0.01
1.00	**0.0014 ± 0.00**	**0.0026 ± 0.00**	**0.0050 ± 0.00**	0.0516 ± 0.01	**0.0015 ± 0.00**	**0.0014 ± 0.00**	**0.0049 ± 0.00**	0.2556 ± 0.00

2.3.5 Discussion

SCNNs are an interesting exercise in integrating causal knowledge and structure into a deep learning paradigm. The results offer us some intuitions, such as how causal priors can often result in better performance but might be harder to optimize and offer less consistent convergence. Although such a naive approach had some promising results, having such complete causal information - an accurate DAG and/or observability of all endogenous variables - is impractical. Further, even with such information, optimization is sporadic and subject to non-convergene at a high rate, which is likely to be a persistent issue as we increase

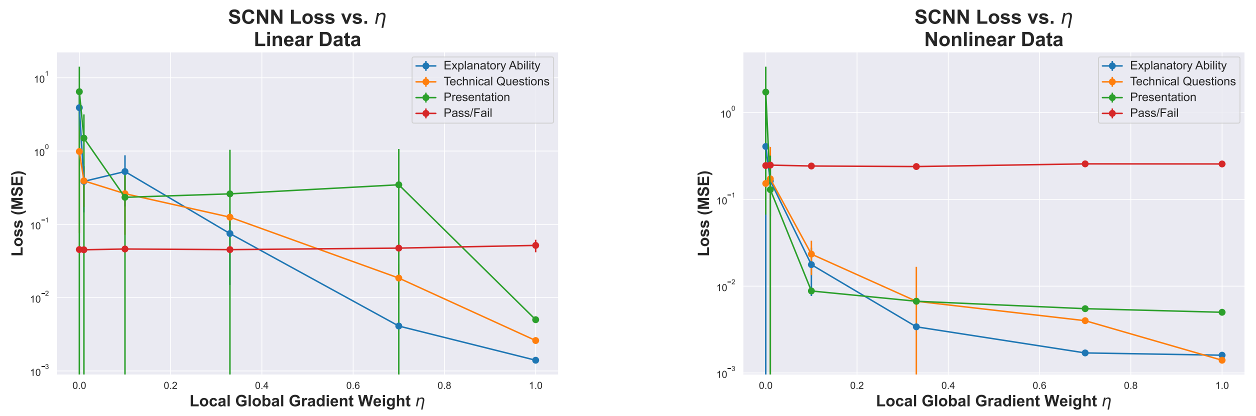


Figure 2.5: SCNN test losses vs η showing a decrease in endogenous loss as we skew more to local gradients, with very moderate sacrifice of final output loss

the scale of the data and models. Later research covered in subsequent chapters makes use of far more complex deep learning architectures integrated with more realistic causal priors, but the basic intuition of embedding structural prior information into deep learning models is maintained.

2.4 Generative Modeling and Causality

2.4.1 Generative Models

Generative models have been crucial to solving many problems in modern machine learning and creating useful synthetic datasets. Two of the early and most influential models include Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [GPM14, KW13a], used primarily for image modeling. Much of the recent progress in generative language models is due to the Transformer architecture [VSP17, RWC19]. More recently, increasing development has occurred on diffusion models (DMs) and energy-based models (EBMs), which use recursive stochastic dynamics to generate or modify data, showing tremendous progress in a wide variety of image generation tasks [HJA20, DM20, RBL22b].

2.4.1.1 Variational Autoencoders (VAEs)

Most of our work integrating causal priors in generative models focuses on VAEs. VAEs are a powerful extension of the vanilla Autoencoder, which is an information bottleneck architecture. A VAE architecture, as seen in Figure 2.6, differs from the Autoencoder by parameterizing the latent space as a Gaussian distribution that can be sampled via a reparameterization trick that allows gradients to still flow[KW13a]. This forces a more continuous, tightly distributed latent space, usually via a Kullback-Leibler divergence loss, in which one can interpolate between training samples in the latent space to get new outputs that differ from any training data or inputs.

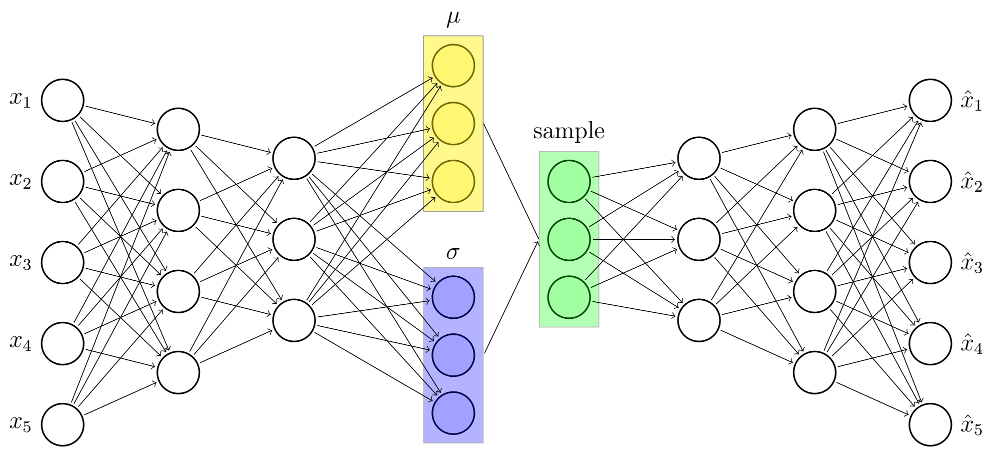


Figure 2.6: Variational Autoencoder Architecture with encoder, decoder, and a mean/variance parameterizing a latent space.

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:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x^{(i)}) &= \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] - \mathrm{KL}\left(e_\phi(z|x^{(i)})||m_\theta(z)\right) \\ &= \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] - \int e_\phi(z|x^{(i)}) \log \frac{e_\phi(z|x^{(i)})}{m_\theta(z)} dz\end{aligned} \tag{2.12}$$

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 $m_\theta(z)$.

2.4.1.2 Energy-Based Models (EBMs)

Energy-based models are a class of generative models that define a probability distribution over the input space by assigning an unnormalized scalar "energy" value to inputs. EBMs are typically formulated as a Gibbs-Boltzmann density, as introduced in [XLZ16]. This model can be mathematically represented as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(-\mathcal{G}_\theta(x))q(x), \tag{2.13}$$

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 nerual network with parameters θ. The normalizing constant, or the partition function, $Z(\theta) = \int \exp\{-\mathcal{G}_\theta(x)\}q(x)dx = \mathbb{E}_q[\exp(-\mathcal{G}_\theta(x))]$, 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.

Inputs with lower energy are considered higher probability under the trained model's distribution. Training an EBM is often achieved by minimizing a contrastive loss function that pushes down on the energy of training data points while pulling up on the energy of sampled negative examples. We can formulate this as an optimization problem.

$$\nabla\mathcal{L}(\theta) = \mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] - \mathbb{E}_{p_\theta}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] \doteq \sum \nabla_\theta \mathcal{G}_\theta(x_i^+) - \sum \nabla_\theta \mathcal{G}_\theta(x_i^-) \tag{2.14}$$

where,

- $\mathcal{L}(\theta)$ is the negative log-likelihood loss function
- $\mathbb{E}p_\theta[\cdot]$ is the expectation over the model distribution
- $\mathcal{G}_\theta(x)$ is the negative energy function, equivalent to the unnormalized log probability, parameterized by θ
- x_i^+ denotes a positive sample from the true data distribution
- x_i^- denotes a negative sample from the model distribution

Since computing the probability requires integrating over the entire input space, which is intractable for high-dimensional data, approximate sampling techniques such as Markov Chain Monte Carlo (MCMC) methods are commonly used. While difficult to train, some advantages of EBMs are their flexibility in architecture and ability to learn complex, multi-modal distributions. Influential work on EBMs includes the Deep Energy Model and Joint Energy-Based Models [KY22, GWJ20].

2.4.1.3 Denoising Diffusion Probabilistic Models

Diffusion models are a class of generative models inspired by thermodynamics where the key idea is to define a forward diffusion process that adds noise and then learn a reverse process that removes the noise to generate samples [HJA20]. The forward process successively adds noise over a sequence of time steps, eventually resulting in values that follow a prior distribution, typically a standard Gaussian. The reverse process is defined as the conditional distribution of the previous variable at a timestep, given the current one. This reverse process is parameterized by a neural network and trained to de-noise a variable from the prior to match the real data distribution. Thus, sampling from a trained diffusion model involves first drawing a sample from the prior distribution, and then running the learned de-noising process to gradually remove noise and yield a final sample.

Formally, let $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ be a clean image sampled from the data distribution. The forward process is defined by a fixed Markov chain with Gaussian transitions for a sequence of timesteps $t = 1, \ldots, T$:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.15}$$

where $\beta_t \in (0, 1)$ is a variance schedule. After T steps, $\mathbf{x}_T$ is nearly an isotropic Gaussian distribution.

The reverse process is defined as a Markov chain with learned Gaussian transitions:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{2.16}$$

The mean $\mu_\theta(\mathbf{x}_t, t)$ and covariance $\Sigma_\theta(\mathbf{x}_t, t)$ are learned using a neural network parameterized by θ. The reverse process is trained to maximize the variational lower bound on the log-likelihood of the data distribution.

Diffusion models have shown remarkable performance in high-fidelity image generation, often outperforming GANs in terms of sample quality and diversity. They have also been successfully applied to audio and video generation. Some milestone diffusion model architectures include Denoising Diffusion Probabilistic Models (DDPMs) and Latent Diffusion Models (LDMs) [RBL22b].

2.4.2 Reformulating Generalization

Causality ideally encodes invariances that exist outside of the train/test dataset domain. Causal priors might also restrict a model's parameters from over-fitting to noise/spurious relationships within the train/test domain to better generalize to out-of-distribution (OOD) data. Let's define an abstract set of causal priors $\hat{\mathbf{C}}$, which might be a DAG or full SCM, and a model or method that utilizes those priors $m_{\hat{\mathbf{C}}}$, which might be an architectural choice or partially fixed parameterization. In order to measure this, we might discuss a new formulation

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{|\mathbf{D}_{train}|} \sum_{(x,y)\in\mathbf{D}_{train}} l(y, m_{\hat{\mathbf{C}}}(x;\theta)) \tag{2.17}$$

$$\begin{gathered} Loss(\theta, \mathbf{D}_{OOD}) = \frac{1}{|\mathbf{D}_{OOD}|} \sum_{(x,y)\in\mathbf{D}_{test}} l(y, m_{\hat{\mathbf{C}}}(x;\theta)) \\ s.t.\;\; d(\mathbf{D}_{train}, \mathbf{D}_{OOD}) >> d(\mathbf{D}_{train}, \mathbf{D}_{test}) \end{gathered} \tag{2.18}$$

where $d(\mathbf{D}_i, \mathbf{D}_j)$ is some statistical distance metric (like a Kullback-Leibler Divergence or Wasserstein Distance) and our constraint is such that the OOD data distribution we wish to generalize to is some *meaningful* distance away from the standard test distribution, representing either a large distributional shift or a counterfactual. In practice, OOD data are usually a more discrete or heuristic-based change, such as generalizing images of actual objects to a similar database of sketched versions or moving between games with similar rules but certain modifications [DGB22, WGL19]. In our research, we similarly often show OOD generalization be either visualizing counterfactuals which explicitly never appear in a training dataset or by splitting data in some non-random manner.

2.4.3 Causality and Generative Modeling

The integration of causality and generative modeling is in its early days, but not without some important attempts. Although GANs are difficult to train, one work utilized plausible structural causal knowledge within a GAN architecture, allowing explicit interventions and counterfactual generation [KSD17]. There has also been work attempting to enforce causal structure in the latent space of a VAE, which some of our work is based on [YLC20]. Contemporary work has been done on causal diffusion models for counterfactual generation [ST22].

One consistency among this research is that causal structure or structural priors are usually learned or utilized, while the functional relationships themselves are parameterized as neural networks. Our causal priors are typically graphical, while the functional relationships are rarely known outside of domains like physics. Thus, many of these models utilize one of the fundamental promises of deep learning architectures, the universal functional approximation, within the bounds of causal priors instead of on the whole problem space or domain. This perspective informs much of our work.

2.5 Adversarial Training and Data Purification

Research on adversarial examples for deep learning models shows that examples can be "crafted" to be imperceptible to humans such that those "poisons" can generalize quite broadly across training paradigms and architectures, sometimes requiring limited or no understanding of the model details itself during crafting [SZS14, GSS15]. Such adversarial attacks can be broadly categorized into inference-time evasion attacks, which manipulate test inputs to cause misclassification, and poisoning attacks, which inject modified data into the training set to degrade model performance or insert hidden backdoors [GTX21]. Data poisoning attacks are particularly concerning, as they can be difficult to detect since they require modifying only a small percentage of training data and have a lasting impact on the trained model [WXS23].

2.5.1 Adversarial Optimization

Fundamentally, an attacker is solving a bi-level optimization problem:

$$\begin{aligned} &\underset{\delta_i \in C}{\operatorname{argmin}} \sum_{(x_\pi, y_\pi) \in \Pi} L(F(x_\pi + \rho; \phi(\delta)), y_{adv}) \\ &\text{s.t.} \quad \phi(\delta) = \underset{\phi}{\operatorname{argmin}} \sum_{(x,y) \in D} L(F(x + \delta_i; \phi), y) \end{aligned} \tag{2.19}$$

The primary objective is to minimize the adversarial loss on a set of target examples Π. 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}. The constraint, utilized for backdoor attacks, ensures that the classifier parameters ϕ are obtained by training on the poisoned dataset, and thus minimally impact classifier performance as the poison percentage is usually low (whereas this constraint might be reversed for untrainable/availability attacks since the goal is to collapse model performance). Typically, additional constraints $\sum_{i=0}^{n} \mathbf{1}_{\delta_i \neq 0} \leq \alpha n$ limit the number of poisoned samples to a fraction α of the total training set (0.1-10% of the data) and $\mathcal{C} = \{\delta \in \mathbb{R}^D : \|\delta\|_\infty \leq \xi\}$ limits the perturbations to be within some imperceptible bound (8/255 in PNG images for example).

2.5.2 Adversarial Defense

To defend against these threats, researchers have proposed various techniques, such as adversarial training, which augments the training data with adversarial examples to improve robustness [BLZ21, BGC21]. However, these methods often come at the cost of reduced accuracy on clean data and can be computationally expensive. More performant techniques typically use filtering methods that identify and remove malicious data points or robust training techniques that modify the learning algorithm to mitigate the impact of poisoned data [PGH20, BLZ21]. In chapters 6 and 7f, we explore our own contributions to a universal poison defense that does not require train-time information and achieves SoTA results in both poison defense and preserved natural accuracy.

2.5.3 Connecting Adversarial Training and Causal Machine Learning

As we shall see, integration of causal priors and adversarial defense share many similar properties around ensuring robustness and trustworthiness in models with hard-to-predict behavior that, even while achieving superhuman results, can baffle us with seemingly obvious failures [BMC23]. Ultimately, modern deep learning operates in high-dimensional spaces, where our ability to directly interpret the relevant feature spaces is difficult. However, the integration of defense techniques and priors can serve as useful stepping stones in the journey to better understand and align machine learning models with our own intelligence and understanding of the world.

CHAPTER 3

De-Biasing Generative Models using Counterfactual Methods

3.1 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) [Pea10]. 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 [KW13b]. 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 presented with the CausalVAE, which allowed the decoder to learn a *causally* disentangled representation of latent space variables [YLC20]. 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. The disentanglement of this causal layer allows 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.

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 to learn a more refined, isolated causal structure that the latent space must go through, which we call Causal Counterfactual Generative Model (CCGM). This allows us to expand the use of the causal layer to more than just single interventions and to also hypothesize and synthesize datasets of counterfactual causal models in interesting and useful ways.

3.2 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 [GQ10b].

3.3 Background

3.3.1 Counterfactuals and Interventions

The SCM literature has long explored the benefits of interventions and counterfactual modeling once a causal model is known. Extensive background on causality and causal models can be found in section 2.1. Pearl introduces interventions using ‘do-calculus’ or the explicit setting of a variable to a specific value and calculating the resulting outputs [Pea10]. In Figure 3.1 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.

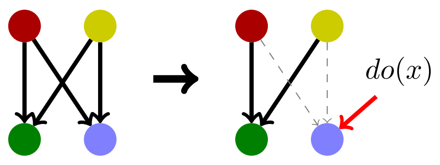


Figure 3.1: Example of a DAG on the left and a mutated counterfactual model on the right with an intervention setting the target variable to an explicit value x.

3.3.2 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 3.2. 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 3.1, 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.

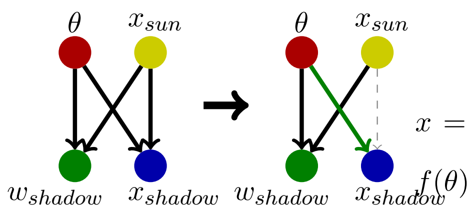


Figure 3.2: Example of a counterfactual model in which a single path is removed to simulate a new distribution of generated data.

3.3.3 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 $p(\cdot)$ [KW13b].

$$\text{ELBO} = \mathbb{E}_{q_\mathcal{X}}\left[\mathbb{E}_{\mathbf{z}\sim q_\phi}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))\right] \tag{3.1}$$

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

$$\mathbf{z} = \mathbf{A}^T\mathbf{z} + \boldsymbol{\epsilon} \tag{3.2}$$

which finds some causal structure of the latent space variables $\mathbf{z}$ with respect to a matrix $\mathbf{A}$. By itself, $\mathbf{A}$ 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 $\mathbf{A}$ is composed of column vectors $\mathbf{A}_i$. For each latent space concept i, define a non-linear function $g_i : \mathbb{R}^n \to \mathbb{R}$ and modify equation (3.2) such that

$$\mathbf{z}_i = g_i(\mathbf{A}_i \circ \mathbf{z}) + \boldsymbol{\epsilon} \tag{3.3}$$

where $\circ$ is the Hadamard product. In this formulation, the view of $\mathbf{A}$ changes from one of function estimation to one of adjacency. That is, if $\mathbf{A}$ is viewed as a binary adjacency matrix, the g_i functions take the responsibility of reconstructing $\mathbf{z}$ given only the the parents, dictated by $\mathbf{A}_i \circ \mathbf{z}$. In the simplest case, if $g_i(\mathbf{v}) = \sum_j v_j$, the summation of all the values of $\mathbf{v}$, then Equation (3.3) degenerates back to Equation (3.2) [NZF19].

Including the causal layer introduces many auxiliary loss functions that we mostly adopt [YLC20]. First is a label loss (3.4), where the adjacency matrix $\mathbf{A}$ should also apply to the labels $\mathbf{u}$. This loss is used in pre-training in its linear form to learn a form of $\mathbf{A}$ 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 $\mathbf{A}$.

$$\ell_u = \mathbb{E}_{q_{\mathcal{X}}}\left[\sum_{i=1}^{n} \|u_i - f_i(\mathbf{A}_i \circ \mathbf{u})\|^2\right] \tag{3.4}$$

The latent loss tries to enforce the SCM, described by Equation (3.3).

$$\ell_z = \mathbb{E}_{\mathbf{z}\sim q_\phi}\left[\sum_{i=1}^{n} \|z_i - g_i(\mathbf{A}_i \circ \mathbf{z})\|^2\right] \tag{3.5}$$

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 (3.6) and apply a scheduling technique to enforce the DAG [ZAR18, YCG19]. The main use is that $\mathbf{G}$ is a DAG if and only if

$$H(\mathbf{G}) := tr\left[(\mathbf{I} + \mathbf{G} \circ \mathbf{G})^n\right] - n = 0 \tag{3.6}$$

The scheduling is done via the augmented Lagrangian

$$\ell_h = \lambda H(\mathbf{G}) + \frac{c}{2}\left|h(\mathbf{G})\right|^2 \tag{3.7}$$

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

$$\lambda_{t+1} = \lambda_t + c_t H(\mathbf{G}_t) \tag{3.8}$$
$$c_{t+1} = \begin{cases} \eta c_t & |H(\mathbf{G}_t)| > \gamma\,|H(\mathbf{G}_{t-1})| \\ c_t & \text{else} \end{cases}$$

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

3.3.4 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 presented in equation (2.4) in background section 2.1.3.

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 (2.5) from background section 2.1.3 and thus adjust for the bias of any measured confounders.

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 [GQ10b].

3.4 Problem Setting

3.4.1 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 3.3 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.

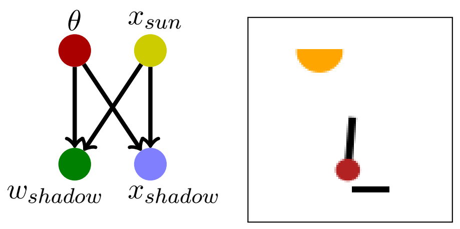


Figure 3.3: Pendulum toy image dataset DAG and example image

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].

3.4.2 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 correlations between all four variables can be seen in Table 3.1.

Table 3.1: 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

Thus we maintain a hypothesized DAG structure as in Figure 3.4 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.

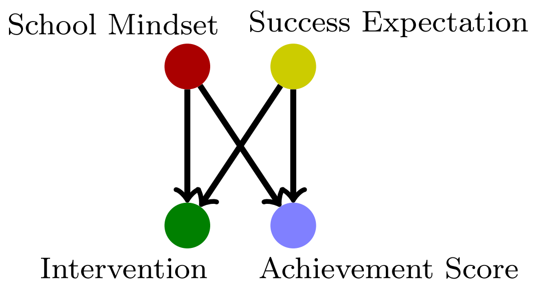


Figure 3.4: School Mindset DAG

The intuitive belief is that a naive estimate of the ATE, calculated as the difference of means as in equation (2.4), 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 naive ATE aligns with the ATE measure using the statistical adjustment methods.

3.5 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.

3.5.1 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 (3.2),

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

Whatever causal structure is learned by $\mathbf{A}$, 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 $\mathbf{A}$ 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 $\mathbf{A} = \mathbf{0}$, which is still a valid DAG. In this case, no remaining causal information remains in the layer and the entire CausalVAE reverts back to a normal β-VAE.

3.5.2 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

$$\mathbf{z} = \mathbf{A}^T\mathbf{z} \tag{3.9}$$

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

$$\mathbf{A} = \underbrace{\mathbf{G}}_{\text{DAG}} + \underbrace{\mathbf{B}}_{\text{diag.}} \tag{3.10}$$

where $\mathbf{B}$ has 1 on the diagonal for exogenous variables and 0 if endogenous. This ensures that the trivial solution where $\mathbf{A} = \mathbf{I}$ 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 (3.3), but dropping the leakage.

$$\mathbf{z}_i = g_i(\mathbf{A}_i \circ \mathbf{z}) \tag{3.11}$$

When separating adjacency and estimation, we necessarily want to have a pre-training step for 5 epochs, where we train $\mathbf{A}$ to recognize the adjacency of the labels before applying the non-linear mask. After the pre-training, we apply training on both the $\mathbf{A}$ 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 $\mathbf{A}$ matrix. The framework proposed by [YLC20] requires one to retrain the entire model to generate a counterfactual $\mathbf{A}$ 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 $\mathbf{A}$'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.

3.5.3 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.

3.6 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 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.

3.6.1 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 of the concepts. However, non-zero interventions did not appear to be working as intended. Figure 3.5 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.

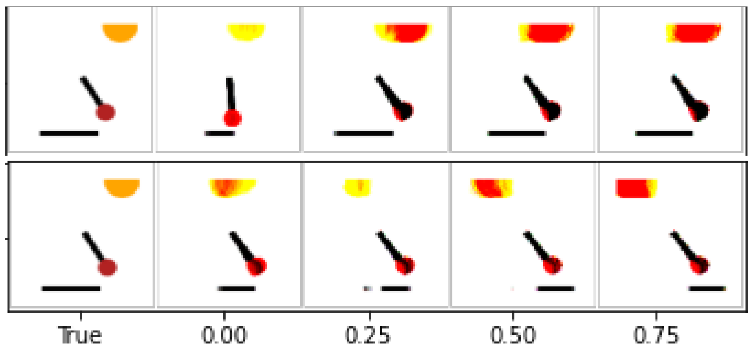


Figure 3.5: A sweep of the pendulum angle (top) and sun position (bottom) in the latent space for CausalVAE. Values are chosen to attempt to see changes. Outside of the 0 intervention, other interventions do not seem to make sense, especially with the shadows.

Furthermore, we noticed little to no change in the results of certain interventions when generating counterfactual models, such as in Figure 3.6 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 $\boldsymbol{\epsilon}$.

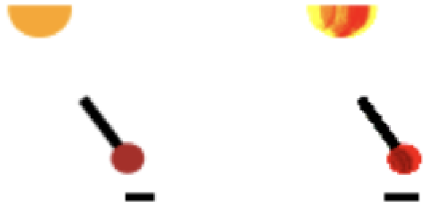


Figure 3.6: Result which still shows effect propagation (shadow moves) after removing the path from sun location to shadow location in CausalVAE method

3.6.2 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 [KW13b].

Now, we subject $\mathbf{z}$ to the learned causal layer. Based on our designation of $\mathbf{u}$, we set $diag(\mathbf{A}) = [1, 1, 0, 0]$, representing the exogenous and endogenous variables of $\mathbf{u}$. Equation 3.11 is applied to 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 are shown in Figures 3.7 and 3.8. 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.

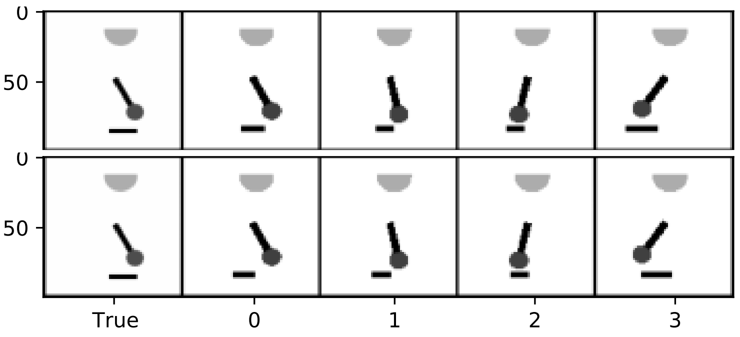


Figure 3.7: Label-to-label (Top) Image response to a sweep of the pendulum angle. Notice that for all interventions, the shadow responds to the pendulum. (Bottom) Image response after shadow position is debiased from pendulum angle. In particular pay attention to the right-most image. While subtle, the shadow positions between the top and bottom image are very noticeable. A quick scan from left to right on all of the intervened images suggests that the midpoint of the shadow remains constant throughout all of the swept images. However, it is worth noting that the shadow width still responds as if the shadow had moved to its location.

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 $\mathbf{A}$, respectively. Afterward, if we do the same interventions, the shadow position no longer responds to that intervention. While some of the results can be subtle, in Figure 3.7, the final image shows a noticeable difference in position before and after the counterfactual model and in Figure 3.8, the first

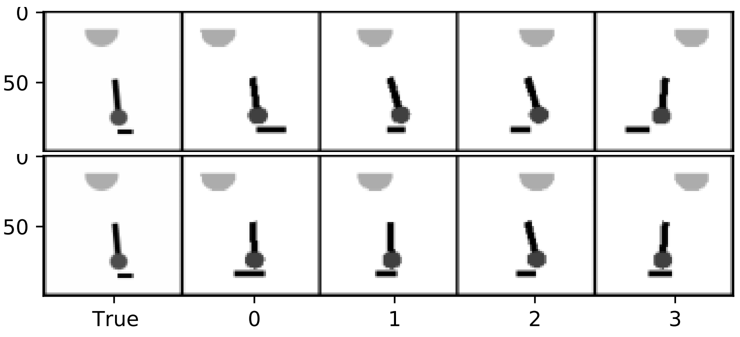


Figure 3.8: Label-to-label (Top) Image response to a sweep of sun positions. Again, notice that the shadow responds to the sun's position. (Bottom) In this case, the changes are more noticeable in that the shadow position remains constant throughout the row and is very different than the expected locations given the sun.

and last interventions both show differences. The connection to shadow width remains, and so the shadow width still responds to the swept variable.

3.6.3 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 a VAE to use the disentanglement granted by the Causal Layer. We can use a pre-trained version of the **A** 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.

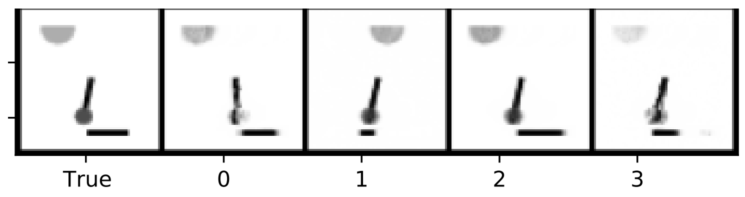


Figure 3.9: Interventions in the label-to-image VAE. Note that shadows respond to pendulum angle and sun position.

Other than this additional pre-training step to learn the **A** 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 3.9. 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 3.10 and 3.11.

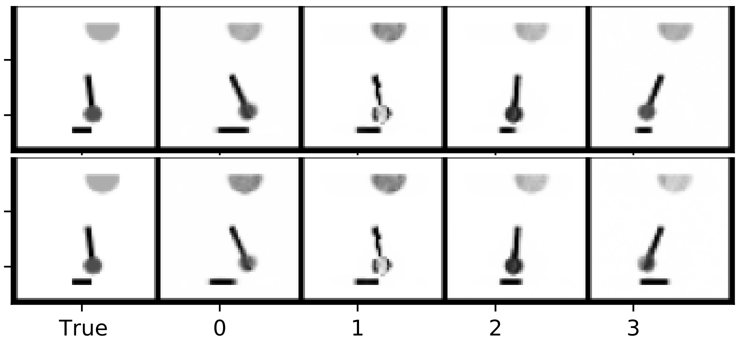


Figure 3.10: Label-to-image (Top) Image response to a sweep of the pendulum angle. Notice that for all interventions, the shadow responds to the pendulum. (Bottom) Image response after shadow position is debiased from pendulum angle. In this case, both the first and last intervened images show noticeable differences from the pre-counterfactual images and one can observe that the shadow midpoint remains consistent across the row.

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 3.10 and 3.11, the first and the last interventional images show noticeable movement from the non-counterfactual interventions.

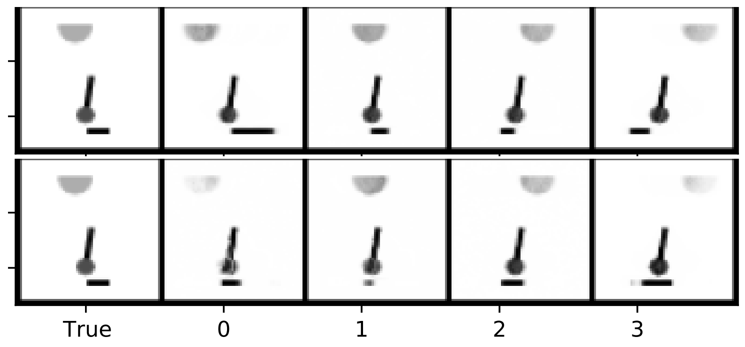


Figure 3.11: Label-to-image (Top) Image response to a sweep of sun positions. Again, notice that the shadow responds to the sun's position. (Bottom) image response after breaking the sun position to shadow position link.

3.6.4 Mindset Data

We begin by first training our CCGM method on the original Student Mindset data. In Figure 3.12, 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 3.13. 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 expect a "de-biased" dataset to have.

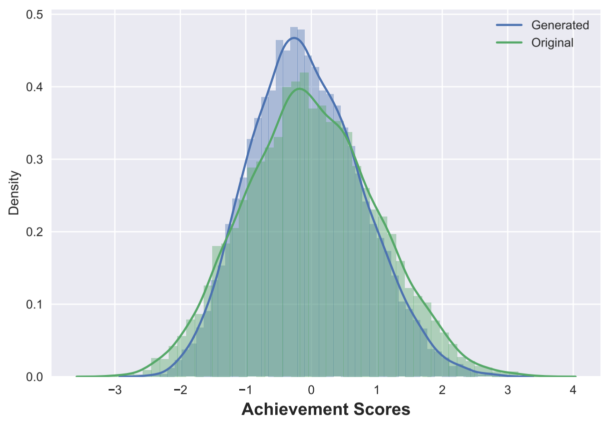


Figure 3.12: Generated distribution of achievement scores closely matches the distribution in the original dataset.

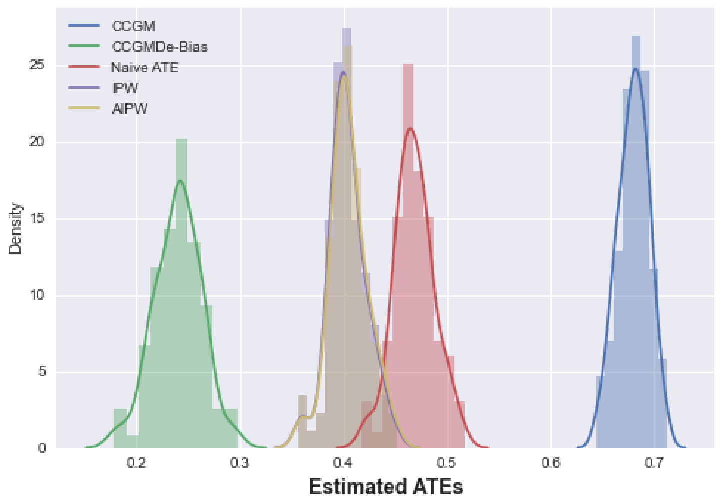


Figure 3.13: Bootstrap distribution of ATEs for various methods and baselines sampling the entire dataset (n=10391) 100 times.

Further, Table 3.2 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

Table 3.2: Mindset Average Treatment Effect (ATE) Results showing how CCGM with a latent intervention can produce "de-baised" or causally separated synthetic data

	Mean ATE	**Std Dev**	**2.5%**	**97.5%**
Naive	0.468	0.019	0.426	0.507
AIPW	0.405	0.018	0.364	0.441
IPW	0.404	0.018	0.363	0.440
CCGM	0.680	0.015	0.650	0.707
CCGM-De-Bias$_{StudentExpectation}$	**0.240**	0.023	0.192	0.283
CCGM-De-Bias$_{Both}$	**0.242**	0.022	0.204	0.282
CCGM-De-Bias$_{SchoolMindset}$	0.677	0.017	0.647	0.712

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 ($iters = 100$).

3.7 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 of the training data, and outside of 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. We believe CCGM is a promising start within a growing field of work in causal generative models.

CHAPTER 4

Hypothesis Testing using Causal and Causal Variational Generative Models

4.1 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) [Pea10]. 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 [KW13b] 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]. In the previous chapter 3, Causal Counterfactual Generative Modeling (CCGM), in which exogeneity priors are included, extends the counterfactual modeling capabilities to test alternative structures and "de-bias" datasets [BJP22].

CausalVAE and CCGM focus on causal discovery concurrently with simulation (i.e. reconstruction error-based training) [YLC20]. 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.

4.2 Background

4.2.1 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 as covered in background section 2.1 [Pea09]. Because the structural priors are not known to be ground truth, we assume a more deterministic functional form and we can make no assumptions about identifiability [Pea09]. 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.

4.3 Causal Hypothesis Gen and Variational Model

4.3.1 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 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:

$$\mathbf{S} = \underbrace{\mathbf{G}}_{\text{DAG}} + \underbrace{\mathbf{B}}_{\text{diag.}} \tag{4.1}$$

$\mathbf{G}$ represents a DAG adjacency matrix, usually referred to as the causal structural model in literature with the modification from the previous chapter that $\mathbf{G} \in \{0,1\}^{d\times d}$, and $\mathbf{B}$ has 1 on the diagonal for exogenous variables and 0 if endogenous ($\mathbf{B} \in \{0,1\}^{d}$). 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:

$$\hat{\boldsymbol{x}}_i = \eta_i(\mathbf{S}_i \circ \boldsymbol{x}) \tag{4.2}$$

where $\mathbf{S}_i$ represents the i-th column vector of $\mathbf{G}$, and $\hat{\boldsymbol{x}}_i$ is the i-th reconstructed output [NZF19]. In the case of exogenous variable $\boldsymbol{x}_i$, a corresponding 1 at $\mathbf{B}_{ii}$, '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 4.1, is trained on a reconstruction loss, defined by $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}})$. We use the L2 loss (Mean Squared Error):

$$\ell_{\text{CSHTest}} = ||\boldsymbol{x} - \eta_i(\mathbf{S}_i \circ \boldsymbol{x})||_2^2 \tag{4.3}$$

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 4.4.3 we demonstrate the ID, OOD train/test splits to test this generalization capacity, and our experimental results provide baselines for this approach.

4.3.2 Causal Variational Hypothesis Testing with CSVHTest

We extend CSHTest to a variational model CSVHTest, that includes sampling functionality like a VAE [KW13b]. 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

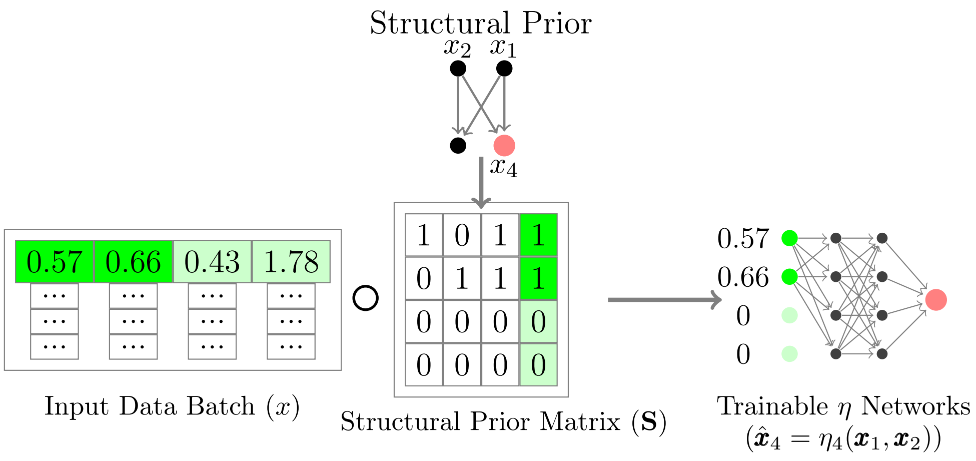

Structural Prior
x_2 x_1
x_4
0.57 0.66 0.43 1.78
...
Input Data Batch (x)
$\circ$
1 0 1 1
0 1 1 1
0 0 0 0
0 0 0 0
Structural Prior Matrix ($\mathbf{S}$)
0.57
0.66
0
0
Trainable η Networks
($\hat{\boldsymbol{x}}_4 = \eta_4(\boldsymbol{x}_1, \boldsymbol{x}_2)$)

Figure 4.1: 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.

CSHTest causal layer and a decoder. Further details are provided in the appendix B.3.1.

4.4 Problem Setting

4.4.1 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 [YCG19, KCW22], often described as the norm of the difference between adjacency matrices:

$$\mathbf{H} = |\mathbf{A}^i - \mathbf{A}^j|_1 \tag{4.4}$$

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 $\mathbf{A}^1$,

$$\mathbf{H}^+(\mathbf{A}^1, \mathbf{A}^0) = |\mathbf{A}^1 > \mathbf{A}^0|_1 \qquad \mathbf{H}^-(\mathbf{A}^1, \mathbf{A}^0) = |\mathbf{A}^1 < \mathbf{A}^0|_1 \tag{4.5}$$

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.

4.4.2 Baseline Models

4.4.2.1 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 4.4.3), and Structural Hamming Distance with iterations at the ground truth and modified DAG levels for robustness. In our experimental results, we calculate the probability a $\mathbf{H}$ of 1 closer to ground truth would have a lower OOD test error as the ratio across our simulations:

$$\Pr(\ell_{\mathrm{CSHT}_{\mathrm{EST}}}(S_j) < \ell_{\mathrm{CSHT}_{\mathrm{EST}}}(S_i)) \,\Big|\, 1 = |\mathbf{A}^i - \mathbf{A}_{GT}|_1 - |\mathbf{A}^j - \mathbf{A}_{GT}|_1 \tag{4.6}$$

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 B.5.

4.4.2.2 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 Appendix B.2.1.

4.4.2.3 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 used 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 4.2 based on knowledge of the selected variables and how they may interact to inform the surgery decision.

4.4.3 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

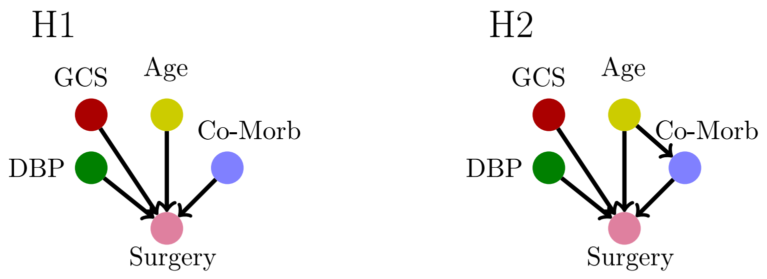
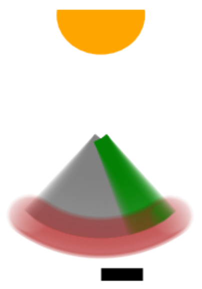


Figure 4.2: Two hypothesized structural causal priors for a medical dataset on trauma patients and the decision to perform surgery, H1 and H2.

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 4.3. 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.



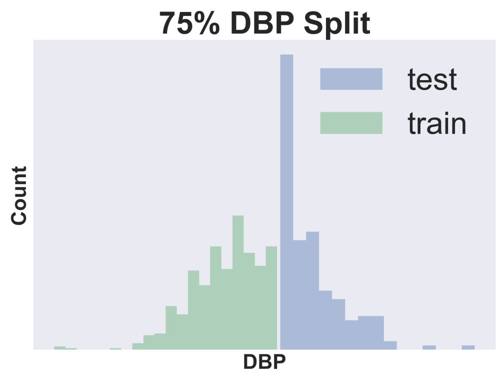


Figure 4.3: a) A 75% data-split on the pendulum angle feature (grey is training angle, green is testing angles b) A 75% data-split on the Diastolic Blood Pressure data.

4.5 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.

4.5.1 Generalization Ability of CSHTest

Table 4.1: 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.

	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

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 4.1. 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 "close-ness" of our model to the true causal model, this would enable the use of the OOD loss as a proxy for causal hypothesis testing.

4.5.2 Simulated DAG Hypothesis Testing

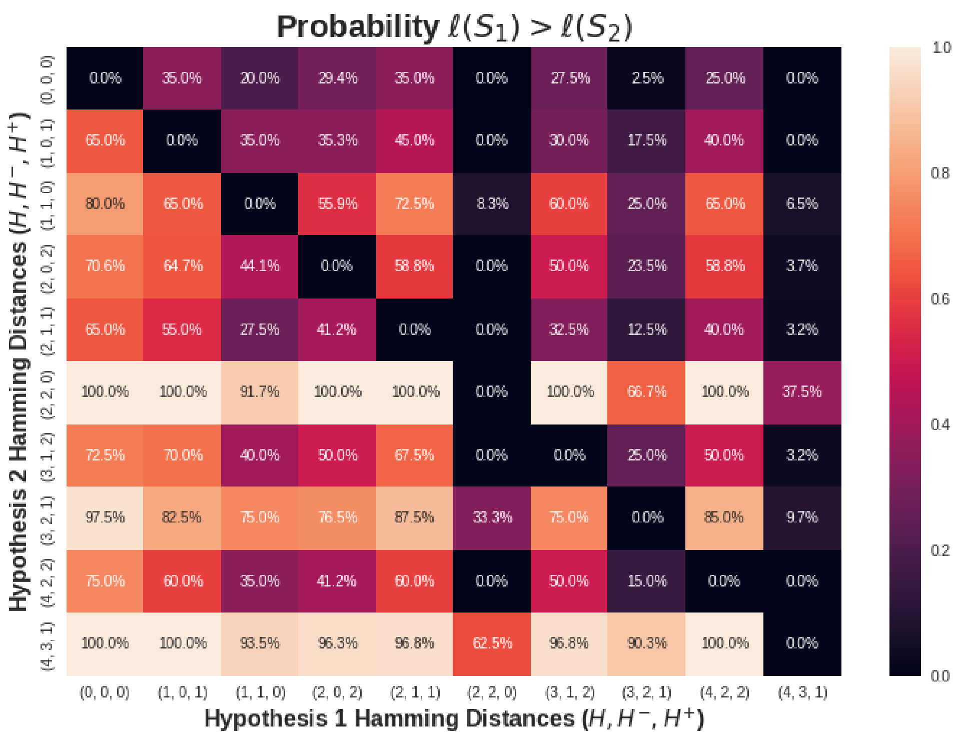


Figure 4.4: Probability table for a 4 node 4 edge DAG size with a linear SEM ground truth model for DAG simulations comparing hypothesis with various Hamming Distance Tuples.

Figure 4.4 shows the type of empirical probability tables we can construct by simulating DAGs of various sizes under numerous conditions detailed in the appendix B.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 DAG size 5x5 can be found in the appendix B.1.

4.5.3 Pendulum Hypothesis Testing

We consider 6 different hypotheses, shown in Figure B.3, detailed in Appendix B.2.2. 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 4.5. 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.

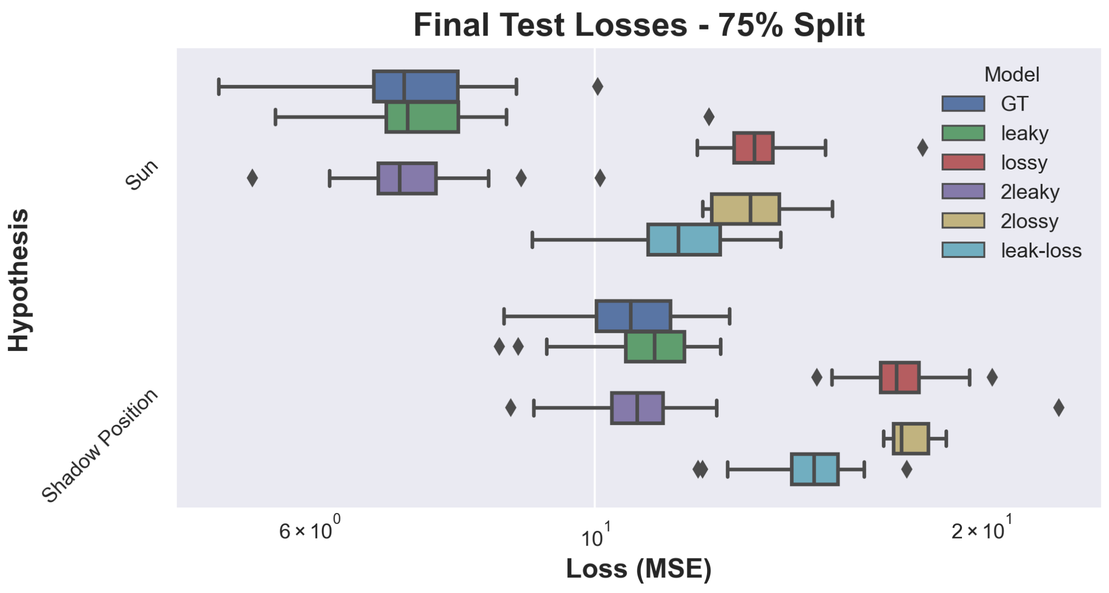


Figure 4.5: Final OOD Test Error Rates of Each Hypothesized DAG structure in the Pendulum Problem over Two Splits. See Appendix B.2.4 for numerical values and training trajectories.

4.5.4 Medical Data Hypothesis Testing

In the medical dataset, the second hypothesis from 4.2, which includes a path from Age to No-Comorbidities 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 testset over the same distribution 4.6.

4.6 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 effectiveness of our methodology on extensive simulated DAGs where ground truth is known, and we

Medical Results		
Hypothesis	**Train**	**Test**
H1	0.024	0.035
H2	0.017	0.025

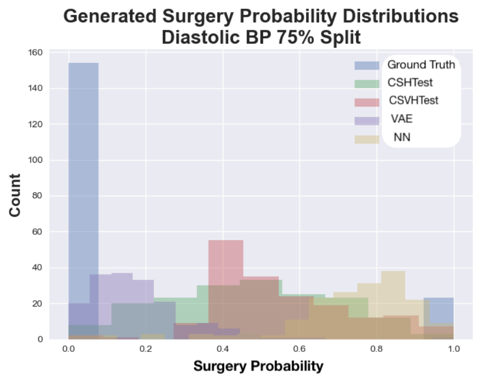


Figure 4.6: **Medical Dataset Results:** *Left:* Medical results of CSHTest for each of the hypothesized causal models. *Right:* The test dataset reconstruction distributions for all models using medical H1 on the Diastolic BP 75% data

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. It would be interesting to differentiate leaky causal models, without constraints on losses, using minimum entropy properties such as in [CGK22] . Further, one could 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 to much further research integrating deep learning causal models with real-world priors and domain knowledge.

CHAPTER 5

Towards Composable Distributions of Latent Space Augmentations

5.1 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.

In this paper, we introduce a novel approach for latent image augmentation using Variational Autoencoder (VAE) architecture called LAVAE. 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 latent 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 5.1 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.

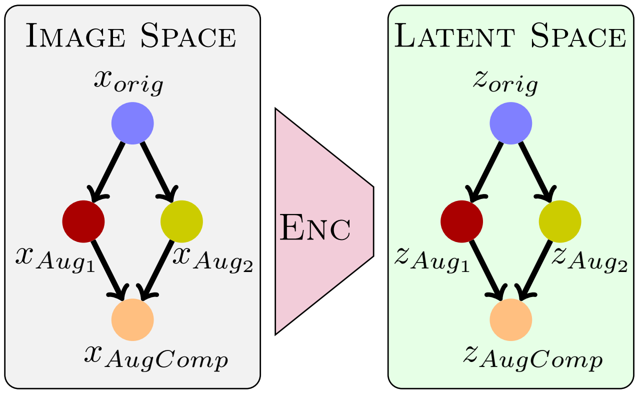


Figure 5.1: Approximate model/DAG *learned* in latent space for *known* image space augmentations

5.2 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 [RBL22a].

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, BPJ22, 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 5.1.

5.3 Background

5.3.1 Conditional Variational Autoencoders

The VAE and variational lower bound is covered in detail in Section 2.4.1.1. 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 Appendix C.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.

5.3.2 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 an parent node of original images such that the resulting images are still within the same class. Succinctly, given the original dataset D_{orig},

$$D_{aug} = \bigcup_{\substack{i \in \mathcal{A}_c \\ d \in D_{orig} \\ c = class(d)}} f_{aug_i}(d) \tag{5.1}$$

$$s.t. \quad \mathcal{A}_c = \{i : class(d) = class(f_{aug_i}(d)); d \in D_{orig}\} \tag{5.2}$$

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.

5.4 Latent Augmentation VAE

5.4.1 Architectural Overview

In the Latent Augmentation VAE, as seen in Figure 5.2, 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.

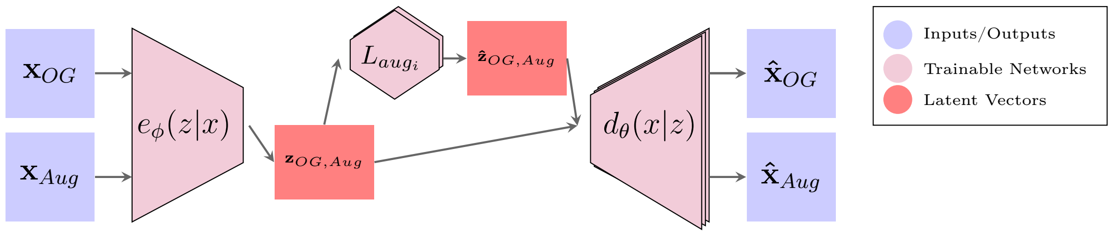


Figure 5.2: Latent Augmentation VAE Architecture

5.4.2 Training LAVAE

For our initial experiments, we focus on pairs of augmentations and their compositions (as in Figure 5.1). 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:

$$\sum_{x \in D_{aug}} \ell_{BCE}(x, \hat{x}) \quad \text{s.t.} \quad \hat{x} = \theta(\phi(x)) \tag{5.3}$$

$$\sum_{x_0 \in D_{orig}} \sum_{k \in \mathcal{A}} (\phi(f_{aug_k}(x_0)) - \phi(x_0) \cdot L_{aug_k})^2 \tag{5.4}$$

$$\sum_{x_0 \in D_{orig}} \sum_{\substack{g(k) \in \mathcal{A}_T \\ k \in \mathcal{A}}} \ell_{BCE}(f_{aug_{g(k)}}(x_0), \theta_T(\phi(x_0) \cdot L_{aug_k})) \tag{5.5}$$

where the trained parameters at each step are highlighted in blue. $\mathcal{A}, \mathcal{A}_T$ 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 C.1, with respective weights on KL and reconstruction $\lambda_{KL} = 5$, $\lambda_{recon} = 1$.

We performed experiments with non-linear latent augmentation networks, which showed slightly better performance, but lacked composability and simple invertability. We also experimented with combining training stages 1 and 2, but this degraded final performance and reconstruction.

5.4.3 Utilizing LAVAE

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

$$\hat{x} = \theta(\phi(x)) \; \forall x \in D_{aug} \tag{5.6}$$

We can augment the original images in the latent space:

$$\begin{gathered} \hat{z_i} = \phi(x_0) \cdot L_{aug_i} \\ \hat{x_i} = \theta(\hat{z_i}) \quad | \quad i \in \mathcal{A} \end{gathered} \tag{5.7}$$

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.

$$\begin{gathered} \hat{\mathring{z}} = \phi(x_0) \cdot L_{aug_1} \cdot L_{aug_2} \\ \hat{\mathring{x}} = \theta(\hat{\mathring{z}}) \end{gathered} \tag{5.8}$$

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

$$\begin{gathered} \hat{\mathring{z}}_r = \phi(x_0) \cdot L_{aug_2} \cdot L_{aug_1} \\ \hat{\mathring{x}} \approx \theta(\hat{\mathring{z}}_r) \\ \hat{\mathring{z}}_r = zL_{aug_2}L_{aug_1} \approx zL_{aug_1}L_{aug_2} = \hat{\mathring{z}} \end{gathered} \tag{5.9}$$

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 a original image, giving us ‘any-to-any’ functionality:

$$\hat{x}_0 = \theta(\phi(x_i) \cdot L_{aug_i}^{-1}) \quad | \quad i \in \{1,2\} \tag{5.10}$$
$$\hat{x}_0 = \theta(\phi(\mathring{x}) \cdot (L_{aug_1} \cdot L_{aug_2})^{-1})$$

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.

$$\begin{aligned} \hat{x}_0^{(k)} &= \theta(\phi(\hat{x}_i^{(k)}) \cdot L_{aug_i}) \\ \hat{x}_i^{(k+1)} &= \theta(\phi(\hat{x}_0^{(k)}) \cdot L_{aug_i}) \\ k &\in [1, n] \end{aligned} \tag{5.11}$$

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

5.5 Experiments

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

5.5.1 LAVAE Reconstruction Results

Figure 5.3 shows the basic and augmented reconstruction results for the “Flips” (flip left/right, flip up/down) augmentation pair. Figure 5.4 shows the inverse reconstruction results for the Flips augmentation pair.

Figure 5.5 displays two examples of recursive augmentation using flip left/right, where 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

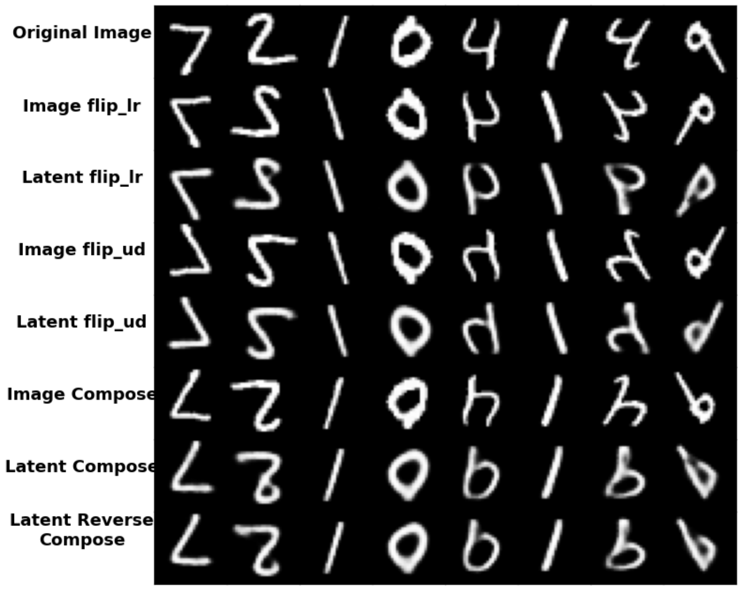


Figure 5.3: Eight samples of “Flips” latent augmentations with baseline image space augmentations for comparison

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 C.4.

5.5.2 LAVAE Transfer Decoders

LAVAE includes multiple decoder heads to enable the transfer of a trained latent space to any pair of augmentations. Figure 5.6 shows the transfer reconstruction results from “Flips”

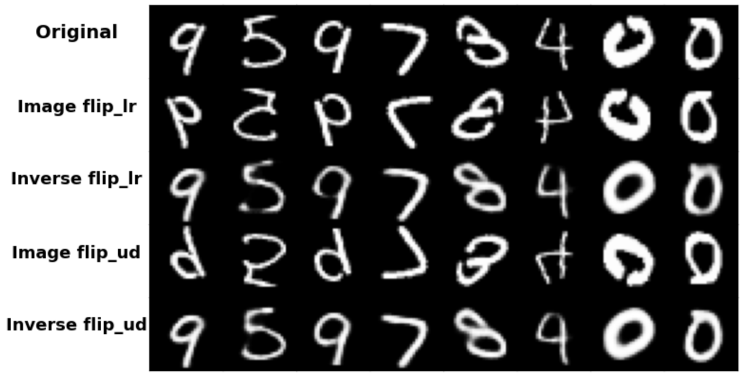


Figure 5.4: Eight samples of "Flips" latent inverse augmentations with original and augmented images (inputs) for comparison

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 5.7 shows a heat-map matrix of reconstruction error (Mean-Squared Error in image space) with initial augmentation pair 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 C.3.

5.5.3 Conditional VAE Comparisons

As we discussed, we realized 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 wanted to partition just like with the LAVAE with respect to augmentation.

Thus, we first trained the CVAE in the traditional way where the conditional is the

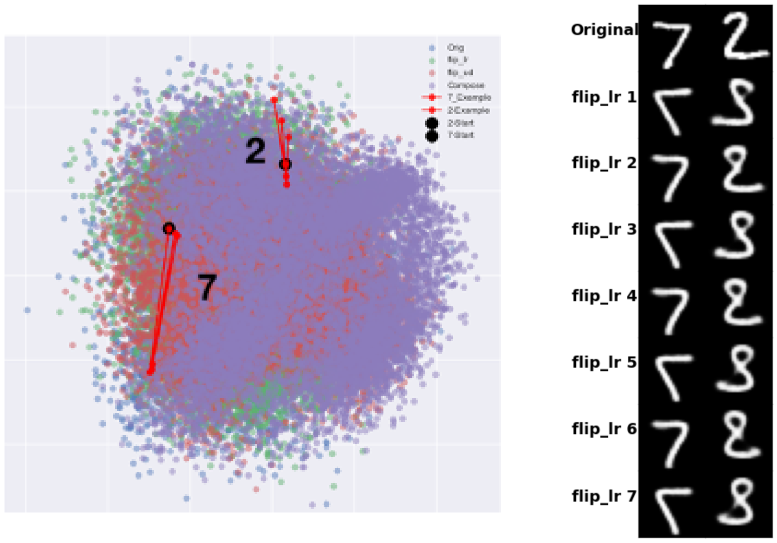


Figure 5.5: Two samples latent trajectory (2-D projection) and reconstructions of recursive flip left/right augmentation. The '7' is stable, but the' 2' diverges both in latent space trajectory and reconstructions.

augmentation type. Example results are shown in 5.8. 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, we can say that the conditional variables and the latent variables are "entangled." This suggests that the CVAE cannot naively handle causally-linked 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

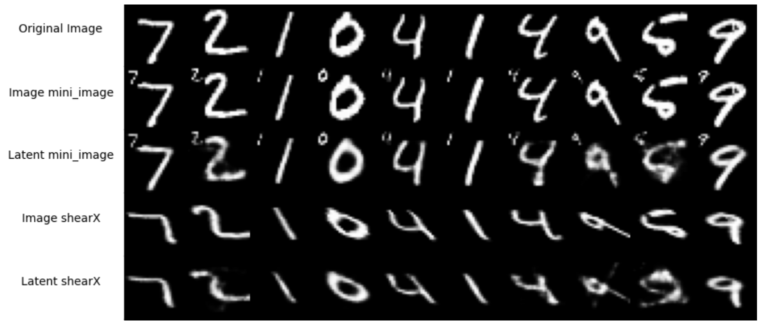


Figure 5.6: Eight samples of “Flips” latent augmentations with baseline image space augmentations for comparison

reconstruction loss compared to the LAVAE as can be seen in Figure 5.9. Furthermore, the composition property does not emerge in the same way that it does for the LAVAE.

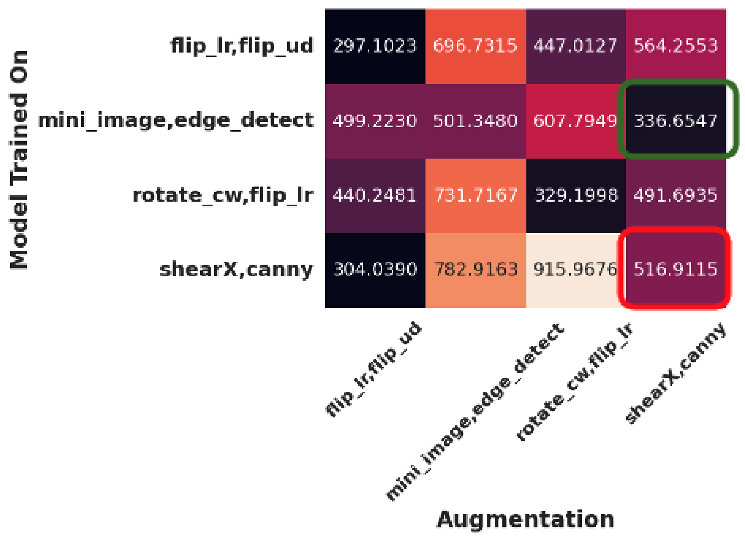
In Table 5.1, we compare the reconstruction errors of the LAVAE against both methods of training the CVAE on the “Flips” augmentation pair, showing the superior performance of LAVAE.

Table 5.1: “Flips” Reconstruction Errors (MSE) showing increased performance of LAVAE and composition capabiltiies

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

5.5.4 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 5.10. We leave a



	flip_lr,flip_ud	mini_image,edge_detect	rotate_cw,flip_lr	shearX,canny
flip_lr,flip_ud	297.1023	696.7315	447.0127	564.2553
mini_image,edge_detect	499.2230	501.3480	607.7949	336.6547
rotate_cw,flip_lr	440.2481	731.7167	329.1998	491.6935
shearX,canny	304.0390	782.9163	915.9676	516.9115

Figure 5.7: Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations)

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 pairs, the separation between regions and the relative areas of regions vary significantly across pairs. Additional images can be found in Appendix C.7.

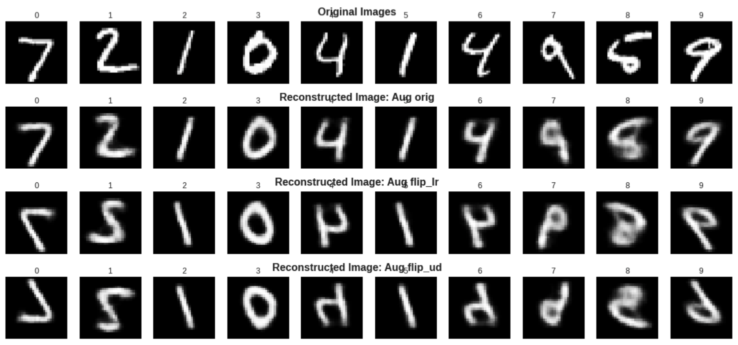

Original Images
0
1
2
3
4
5
6
7
8
9
Reconstructed Image: Aug orig
Reconstructed Image: Aug flip_lr
Reconstructed Image: Aug flip_ud

Figure 5.8: Initial augmentation pair choice vs. transferred augmentation MSE reconstruction error (across all augmentations)

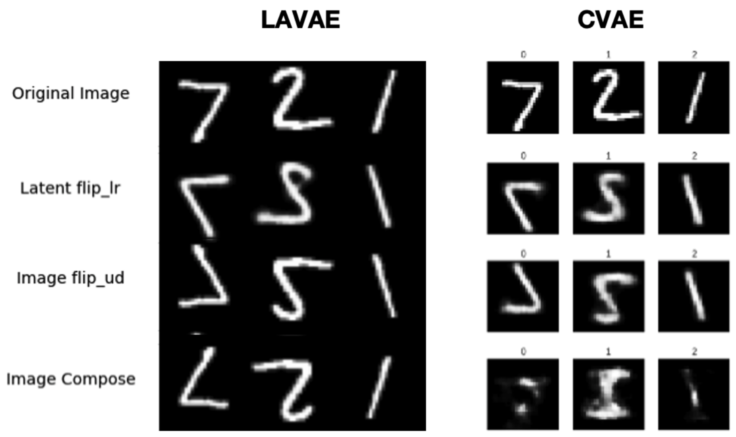


Figure 5.9: LAVAE vs CVAE reconstructions

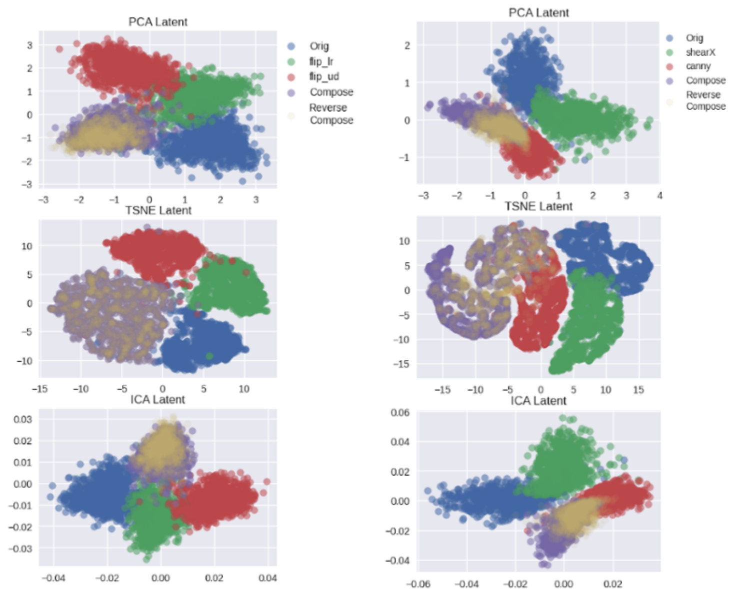


Figure 5.10: "Flips" (left) and "shear X-direction, canny edge-detect" (right) 2-d projections using PCA, ICA, and T-SNE algorithms.

5.6 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 flexibilities 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 augmentation model. For future work, it would be interesting 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, our approach could be applied to other datasets, including those where the image model might not be known, such as 2D to 3D reconstruction or perspective shift tasks.

CHAPTER 6

PureEBM: Robust Train-Time Poison Purification via Mid-run Dynamics of Energy-Based Models

6.1 Introduction

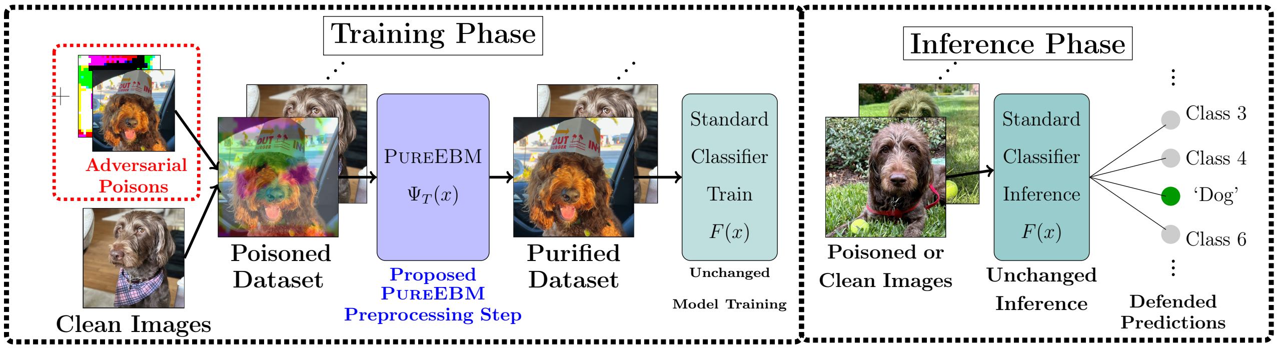
Large datasets empower modern, over-parameterized deep learning models. An adversary can easily insert a small number of powerful, but imperceptible, poisoned images into these datasets, often scraped from the open Internet, and manipulate a Neural Network's (NN) behavior at test time with a high success 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.

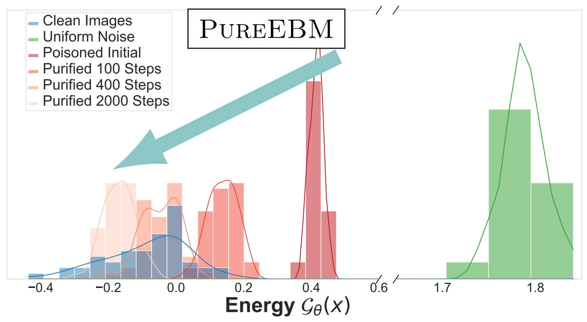
Numerous methods of poisoning deep learning systems have been proposed in recent years. These disruptive techniques typically fall into two distinct categories: backdoor, triggered data poisoning, or triggerless poisoning attacks. Triggered attacks conceal an imperceptible trigger pattern in the samples 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 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 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 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 6.1. 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





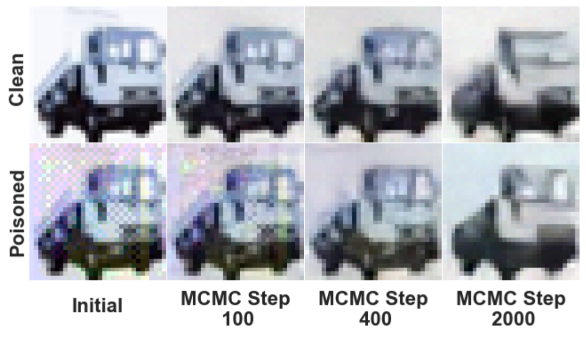


Figure 6.1: **Top** The full PureEBM pipeline is shown where we apply our method as a preprocessing step with no further downstream changes to the classifier training or inference. *Poisoned images are moderately exaggerated to show visually.* **Bottom Left** Energy distributions of clean, poisoned, 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.

- 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

6.2 Related Work

6.2.1 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:

$$F(x) = \begin{cases} y & x \in \{x : (x, y) \in \mathcal{D}_{test}\} \\ y^{\text{adv}} & x \in \{x + \rho : (x, y) \in \mathcal{D}_{test}, y \neq y^{\text{adv}}\} \end{cases} \tag{6.1}$$

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:

$$F(x) = \begin{cases} y & x \in \{x : (x, y) \in \mathcal{D}_{test} \setminus \Pi\} \\ y^{\text{adv}} & x \in \{x : (x, y) \in \Pi\} \end{cases} \tag{6.2}$$

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].

6.2.2 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.

6.3 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 $i = 1, \dots, n$. 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:

$$p_{data} = (1 - \alpha)p_{clean} + \alpha p_{poison} \tag{6.3}$$

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:

$$\begin{aligned} &\underset{\substack{\delta_i \in \mathcal{C}_\delta, \rho \in \mathcal{C}_\rho \\ \sum_{i=0}^{n} \mathbb{1}_{\delta_i \neq \mathbf{0}} \leq \alpha n}}{\operatorname{argmin}} \sum_{(x^\pi, y^\pi) \in \Pi} \mathcal{L}\left(F(x^\pi + \rho; \phi(\delta)), y^{\text{adv}}\right) \\ &s.t. \quad \phi(\delta) = \underset{\phi}{\operatorname{argmin}} \sum_{(x,y) \in \mathcal{D}} \mathcal{L}\left(F(x + \delta_i; \phi), y\right) \end{aligned} \tag{6.4}$$

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.,

$$\nabla \mathcal{L}\left(F_\phi\left(x^\pi\right), y^{\text{adv}}\right) \propto \sum_{i:\delta_i \neq \mathbf{0}} \nabla \mathcal{L}\left(F_\phi(x_i + \delta_i), y_i\right) \tag{6.5}$$

Minimizing the training loss on RHS of Equation 6.5 also minimizes the adversarial loss objective of Equation 6.4.

For the triggered poison, Narcissus (NS), we find the most representative patch ρ for class π given $\mathcal{C}$, defining Equation 6.4 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].

6.3.1 Energy-Based Model

Background on Energy-Based Models (EBMs) can be found in Section 2.4.1.2. Since, 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 in 2.14. The derivative of this log-likelihood, crucial for parameter updates, is given by:

$$\begin{aligned}\nabla \mathcal{L}(\theta) &= \mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] - \mathbb{E}_{p_\theta}\left[\nabla_\theta \mathcal{G}_\theta(x)\right] \\ &\doteq \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{G}_\theta(x_i^+) - \frac{1}{k}\sum_{i=1}^{k} \nabla_\theta \mathcal{G}_\theta(x_i^-), \end{aligned} \tag{6.6}$$

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:

$$x_{\tau+1} = x_\tau - \Delta\tau \nabla_{x_\tau} \mathcal{G}_\theta(x_\tau) + \sqrt{2\Delta\tau}\epsilon_\tau, \tag{6.7}$$

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 D.3.1 for details.

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 $\tau$ 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
```

Theoretical perspectives on the memoryless property of our defense can be found in section 6.3.3. 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 MLE method is unsurprisingly not affected by targeted poisons.

6.3.2 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 (6.7). 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.

6.3.3 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 (6.7), 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. D.1.1.

6.3.4 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 6.1, 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.

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 6.2 we explore the MCMC trajectory's impacts on ℓ_2 distance of both purified clean and poisoned images

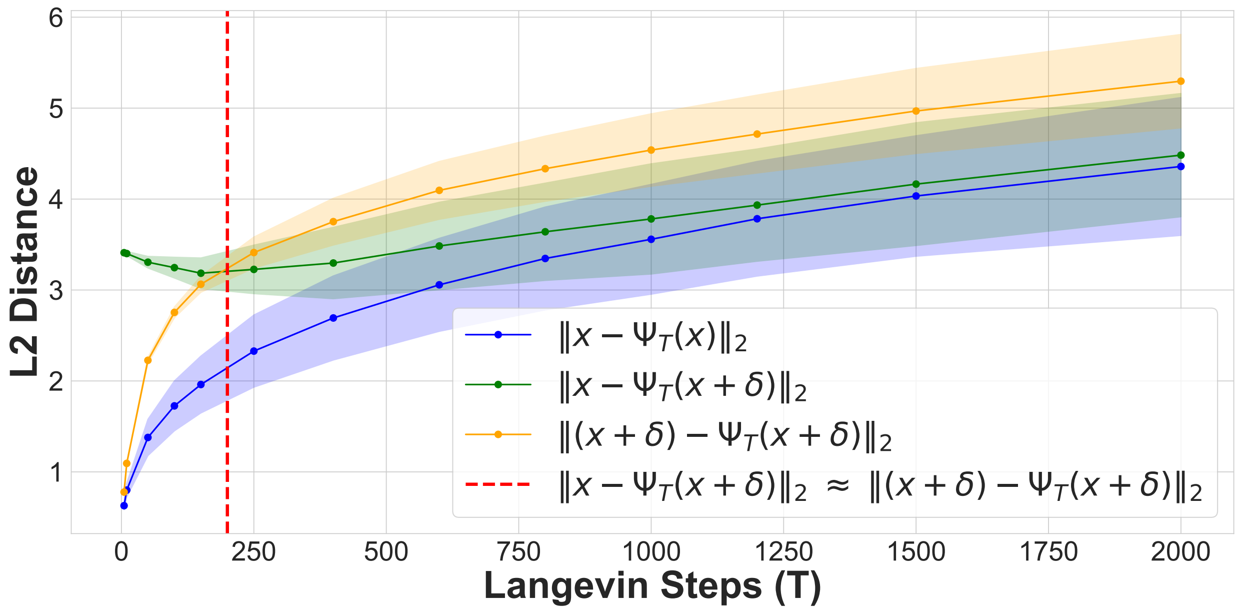


Figure 6.2: Plot of ℓ_2 distances between clean images and clean purified (blue), clean images and poisoned purified (green), and poisoned images and poisoned purified images (orange) at points on the MCMC sampling trajectory. Purifying poisoned images for less than 250 steps moves a poisoned image closer to its clean image with a minimum around 150, preserving the natural image while removing the adversarial features.

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 6.1.

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.

6.4 Experiments

6.4.1 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. 6.3.1 with specific datasets for our experimental results:

1. **PureEBM**: To ensure EBM training is blind to poisoned images, we excluded the 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 $_{\textbf{CN-10}}$**: 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-10 indices [DCA18]. This model investigates the effectiveness of EBM purification when trained on a distributionally similar dataset.

4. **PureEBM $_{\textbf{IN}}$**: 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$_{\textbf{CN-10}}$**: 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. D.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. D.3.2.

6.4.2 Benchmark Results

Table 6.1 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.

Table 6.1: Poison success and natural accuracy in all poisoned training scenarios (ResNet18, CIFAR-10). We report the mean and the standard deviations (as subscripts) of 100 GM experiments, 50 BP experiments, and NS triggers over 10 classes.

	From Scratch									
	200 - Epochs					**80 - Epochs**				
	Gradient Matching-1%		**Narcissus-1%**			**Gradient Matching-1%**		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	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	47.00	$93.79_{0.2}$	$32.51_{30.3}$	$93.76_{0.2}$	79.43
EPIc	10.00	$85.14_{1.2}$	$27.31_{34.0}$	$82.20_{1.1}$	84.71	27.00	$90.87_{0.4}$	$21.53_{28.8}$	$88.05_{1.1}$	80.75
FrieNDs	**0.00**	**$91.15_{0.4}$**	$8.32_{22.3}$	$91.01_{0.4}$	83.03	**1.00**	$90.09_{0.4}$	**$1.37_{0.9}$**	$90.01_{0.2}$	3.18
PureEBM	**0.00**	**$92.26_{0.2}$**	**$1.27_{0.6}$**	**$92.91_{0.2}$**	**2.16**	**1.00**	**$91.36_{0.3}$**	**$1.46_{0.8}$**	**$91.83_{0.3}$**	**2.49**
PureEBM-P	NA	NA	**$1.38_{0.7}$**	**$92.70_{0.2}$**	**2.78**	NA	NA	**$1.63_{1.0}$**	**$91.49_{0.3}$**	3.47
PureEBM $_{CN-10}$	**0.00**	**$92.99_{0.2}$**	**$1.43_{0.8}$**	**$92.90_{0.2}$**	**3.06**	**1.00**	**$92.02_{0.2}$**	**$1.50_{0.9}$**	**$92.03_{0.2}$**	**2.52**
PureEBM $_{IN}$	1.00	**$92.98_{0.2}$**	**$1.39_{0.8}$**	**$92.92_{0.2}$**	**2.50**	**1.00**	**$92.02_{0.2}$**	**$1.52_{0.8}$**	**$92.02_{0.3}$**	**2.81**
PureEBM-P$_{CN-10}$	NA	NA	**$1.64_{0.01}$**	**$92.86_{0.20}$**	**4.34**	NA	NA	**$1.68_{1.0}$**	**$92.07_{0.2}$**	3.34

	Transfer Learning								
	Fine-Tune					**Linear - Bullseye Polytope**			
	Bullseye Polytope-10%		**Narcissus-10%**			**BlackBox-10%**		**WhiteBox-1%**	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	46.00	$89.84_{0.9}$	$33.41_{33.9}$	$90.14_{2.4}$	98.27	93.75	$83.59_{2.4}$	98.00	$70.09_{0.2}$
EPIc	42.00	$81.95_{5.6}$	$20.93_{27.1}$	$88.58_{2.0}$	63.00	66.67	$84.34_{3.8}$	91.00	$64.79_{0.7}$
FrieNDs	8.00	$87.82_{1.2}$	$3.04_{5.1}$	$89.81_{0.5}$	17.32	33.33	$85.18_{2.3}$	19.00	$60.90_{0.6}$
PureEBM	**0.00**	**$88.95_{1.1}$**	**$1.98_{1.7}$**	**$91.40_{0.4}$**	**5.98**	**0.00**	**$92.89_{0.2}$**	**6.00**	**$64.51_{0.6}$**
PureEBM-p	NA	NA	**$3.66_{4.63}$**	**$90.89_{0.31}$**	16.04	NA	NA	NA	NA
PureEBM $_{CN-10}$	**0.00**	**$88.67_{1.2}$**	**$2.97_{2.5}$**	**$90.99_{0.3}$**	**7.95**	**0.00**	**$92.82_{0.1}$**	**6.00**	**$64.44_{0.4}$**
PureEBM $_{IN}$	**0.00**	**$87.52_{1.2}$**	**$2.02_{1.0}$**	**$89.78_{0.6}$**	**3.85**	**0.00**	**$92.38_{0.3}$**	**6.00**	**$64.98_{0.3}$**

6.4.3 Results on Additional Models and Datasets

Table 6.2 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 6.3 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. D.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 6.3 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 6.3 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 a 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. D.4 for further timing.

Table 6.2: 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**	$\mathbf{91.98}_{0.16}$

Table 6.3: 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**	$478.29_{0.47}$

6.4.4 Further Experiments

Model Interpretability Using the Captum interpretability library, in Figure 6.3, 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 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. Aditionally we see that the NN trained with PureEBM is less sensitive to input perturbations compared to all other NNs. See App. D.5 for additional examples.

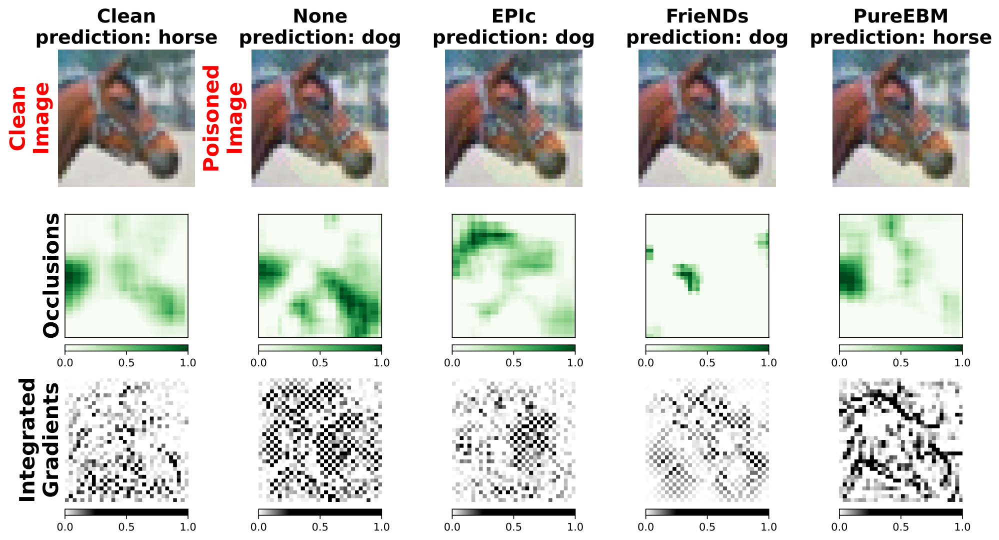


Figure 6.3: Defense Interpretability: Model using PureEBM focuses on the outline of the horse in the occlusions analysis and to a higher degree on the primary features in the gradient space than even the clean model on clean data.

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 [Li19, Li22] 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. D.5.1, we find that PUREEBM and FRIENDS models' parameters diverge from poisoned models more so than EPIC.

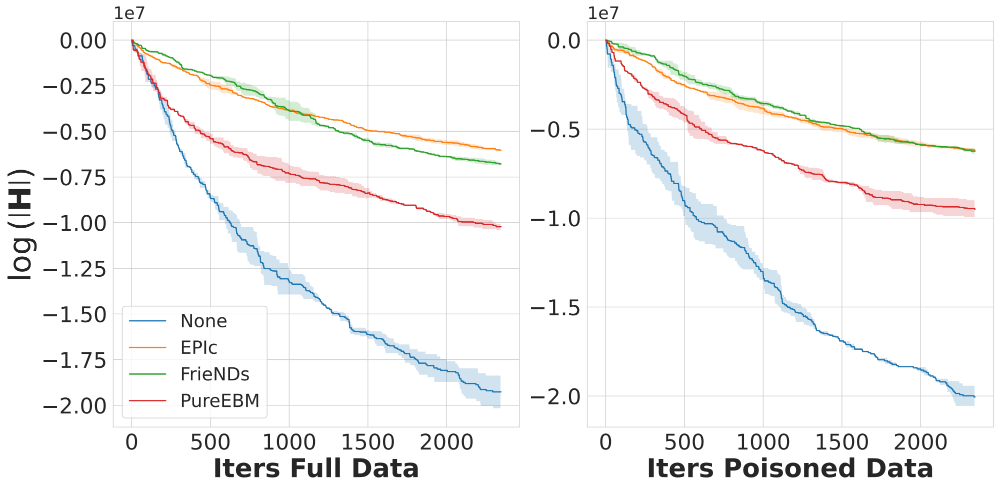


Figure 6.4: Estimate loss curvature - classifier robustness - with $\log(|\mathbf{H}|)$ against both full and poisoned subset of training data. Model trained with PUREEBM has the lowest curvature compared to SoTA defense methods.

6.5 Conclusion

In this chapter, 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 stochastic preprocssing 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.

CHAPTER 7

PureGen: Universal Data Purification Using Generative Model Dynamics

7.1 Introduction

In this chapter, we introduce PureGen, a set of techniques for data purification using the dynamics of generative models. PureGen encompasses our previously discussed PureEBM method, now referred to as PureGen-EBM, as well as PureGen-DDPM, a modified version of Denoising Diffusion Probabilistic Models (DDPMs) tailored for purification tasks. We explore the robustness of these methods to distributional shifts and poisoning in the generative model training data. Furthermore, we propose three novel combinations EBM and DDPM dynamics: PureGen-Naive, PureGen-Reps, PureGen-Filt to further enhance the purification process. Finally, we apply PureGen variants to the task of purifying intellectual property (IP) from image datasets, demonstrating its effectiveness in removing undesired visible perturbations while preserving the natural accuracy of the dataset.

7.2 PureGen-DDPM: Modified Denoising Diffusion for Data Purification

As discussed in detail in Section 2.4.1.3 Denoising Diffusion Probabilistic Models (DDPMs) [HJA20] are a class of generative models that have shown remarkable performance in high-fidelity image generation. The key idea in DDPMs is to define a forward diffusion process that gradually adds Gaussian noise to an image over a fixed number of timesteps, and learn a reverse process that removes this noise to generate a clean sample.

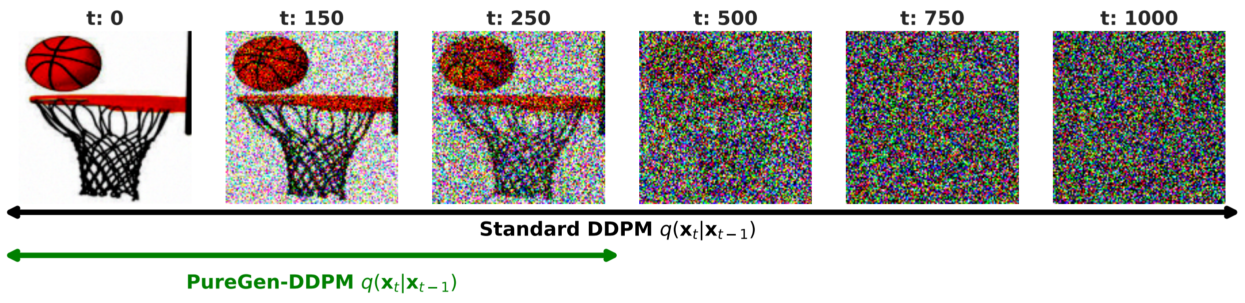
In the standard DDPM formulation, the number of diffusion steps T is usually set to a large value like 1000 to ensure that $\mathbf{x}_T$ is sufficiently noisy and close to an isotropic Gaussian. However, for the purpose of data purification, we make a key modification in PureGen-DDPM- we limit the number of forward diffusion steps in training to a much smaller value, typically around 250. The intuition behind this change is as follows:

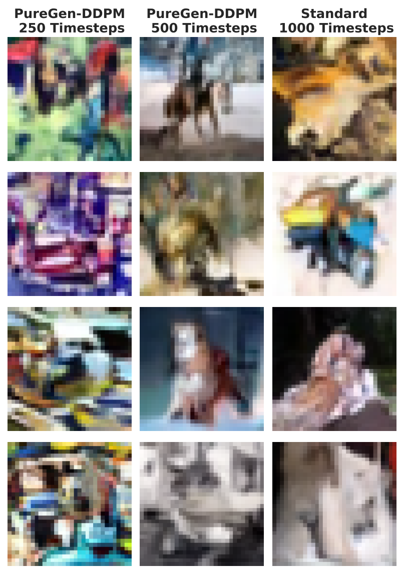
1. By stopping the diffusion process earlier, we ensure that the corrupted image $\mathbf{x}_T$ still retains some structure and information from the original clean image $\mathbf{x}_0$. This is in contrast to running the full 1000 steps, which would essentially destroy all information about $\mathbf{x}_0$.

2. During training, limiting the diffusion steps allows the model to dedicate more of its capacity towards learning to restore the moderately corrupted images back to their clean versions, rather than learning to generate images from pure noise. This shift in focus is more aligned with our goal of purification, and sacrifices "generative" capabilities for better purification and image restoration.

Experimentally, we found that using a maximum of around 250 forward process training steps provided the optimal balance at inference time for removing adversarial perturbations while preserving the original image content. Using too few steps did not provide the inference capacity to eliminate the poisons sufficiently, while using too many steps dedicated less model capacity to restoration, and we lost natural accuracy as well as defense capabilities. Examples of this can be seen in 7.1.

By modifying the forward diffusion process and focusing the model capacity on purification rather than generation, PureGen-DDPM provides a more effective and tailored defense against adversarial poisoning attacks on image datasets.


t: 0
t: 150
t: 250
t: 500
t: 750
t: 1000
Standard DDPM $q(\mathbf{x}_t|\mathbf{x}_{t-1})$
PureGen-DDPM $q(\mathbf{x}_t|\mathbf{x}_{t-1})$



Narcissus $\epsilon = 16$ 1%				
Purify Steps	75	100	125	150
Forward Train Steps	Avg Natural Accuracy (%) ↑			
150	90.96 ± 0.15	90.21 ± 0.20	89.18 ± 0.11	88.46 ± 0.22
250	**91.04 ± 0.17**	**90.55 ± 0.19**	**89.75 ± 0.17**	**89.60 ± 0.17**
500	90.48 ± 0.21	89.77 ± 0.20	88.99 ± 0.19	88.19 ± 0.15
750	90.25 ± 0.12	89.06 ± 0.18	88.14 ± 0.10	87.19 ± 0.21
1000	90.11 ± 0.16	89.00 ± 0.25	87.98 ± 0.18	86.83 ± 0.10
Forward Train Steps	Avg Poison Success (%) ↓			
150	8.03 ± 6.36	6.36 ± 5.84	5.51 ± 4.07	5.43 ± 4.51
250	**7.14 ± 6.94**	**5.58 ± 5.25**	**4.36 ± 3.63**	**4.15 ± 3.24**
500	8.88 ± 7.31	6.34 ± 5.10	5.45 ± 4.22	4.93 ± 4.36
750	9.27 ± 6.26	7.01 ± 5.19	5.96 ± 4.64	5.36 ± 3.42
1000	9.12 ± 6.61	7.01 ± 4.82	6.43 ± 5.12	5.12 ± 3.18

Figure 7.1: **Top** We compare PureGen-DDPM forward steps with the standard DDPM where 250 steps *degrades images for purification but does not reach a noise prior. Note that all model are trained with the same linear β schedule.* **Bottom Left** Generated images from sampled prior for models with 250, 750, and 1000 (Standard) train forward steps *where it is clear the 250-step model does not generate realistic images.* **Bottom Right** Significantly improved poison defense performance of PureGen-DDPM with 250 train steps indicating a trade-off between data purification and generative capabilities.

7.3 PureGen: Generative Model Based Data Purification for Poison Defense

7.3.1 Classification with Stochastic Transformation

We update our notation from the previous Section 6.3.2, $\Psi_T : \mathbb{R}^D \to \mathbb{R}^D$ which is a stochastic pre-processing transformation. We now define $T = (T_{\text{EBM}}, T_{\text{DDPM}}, T_{\text{Reps}}) \in \mathbb{R}^3$, where T_{EBM} represents the number of EBM MCMC steps, T_{DDPM} represents the number of diffusion steps, and T_{REPS} represents the number of times these steps are repeated. Thus for PureGen-EBM $T_{\text{DDPM}} = 0$ and for PureGen-DDPM $T_{\text{EBM}} = 0$ and $T_{\text{REPS}} = 1$ for both.

To incorporate EBM filtering, we order $\mathcal{D}$ by $\mathcal{G}_\theta(x)$ and partition the ordering based on k into $\mathcal{D}^{(k)}_{max} \cup \mathcal{D}^{(1-k)}_{min}$, where $\mathcal{D}^{(k)}_{max}$ contains $k|\mathcal{D}|$ datapoints with the maximal energy (where $k = 1$ results in purifying everything and $k = 0$ is traditional training). Then, with some abuse of notation,

$$\Psi_{T,k}(\mathcal{D}) = \Psi_T(\mathcal{D}^{(k)}_{max}) \cup \mathcal{D}^{(1-k)}_{min} \tag{7.1}$$

One can compose a stochastic transformation $\Psi_{T,k}(x)$ with a randomly initialized deterministic classifier $f_\phi(x) \in \mathbb{R}^J$ (for us, a naturally trained classifier) to define a new deterministic classifier $F_\phi(x) \in \mathbb{R}^J$ as

$$F_\phi(x) = \mathbb{E}_{\Psi_{T,k}(x)}[f_{\phi_0}(\Psi_{T,k}(x))] \tag{7.2}$$

which is trained with cross-entropy loss via SGD to realize $F_\phi(x)$. As this is computationally infeasible we take $f_\phi(\Psi_{T,k}(x))$ as the point estimate of $F_\phi(x)$, which is valid because $\Psi_{T,k}(x)$ has low variance.

7.3.2 Comparing PureGen-EBM and PureGen-DDPM Dynamics

PureGen-EBM is akin to a noisy gradient descent over the unconditional energy landscape of a learned data distribution. This is more implicit in the PureGen-DDPM dynamics. As T increases, poisoned images move from their initial higher energy towards more realistic lower-energy samples that lack poison perturbations. In from-scratch $\epsilon = 8$ poison scenarios, 150 EBM Langevin steps or 75 DDPM steps fully purifies the majority of the dataset with minimal feature loss to the original image. In Figure 7.2, we compare the Langevin 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 ($\|(x+\delta) - \Psi_T(x+\delta)\|_2$) for both PureGen-EBM and PureGen-DDPM. Both poisoned and clean distance trajectories converge to similar distances away from the original clean image ($\lim_{T\to\infty} \|x - \Psi_T(x)\|_2 = \lim_{T\to\infty} \|x - \Psi_T(x+\delta)\|_2$), and 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 EBM and 75 DDPM steps, indicating when purification has moved the poisoned image closer to the original clean image than the poisoned version of the image.

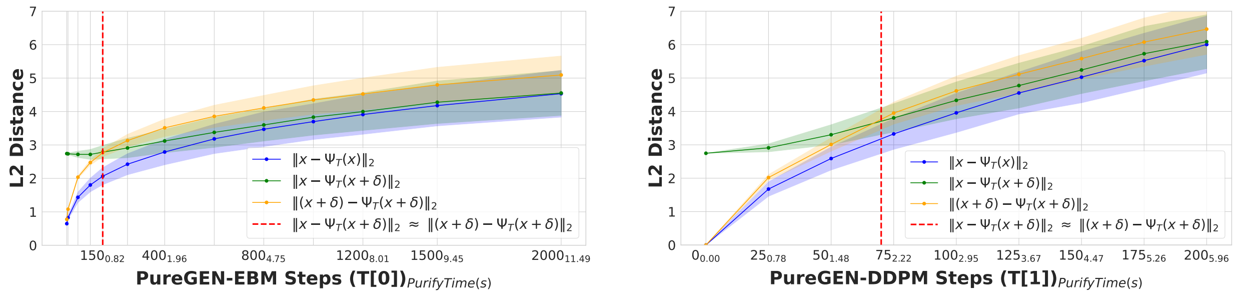


Figure 7.2: Plot of ℓ_2 distances for PureGen-EBM (**Left**) and PureGen-DDPM (**Right**) between clean images and clean purified (blue), clean images and poisoned purified (green), and poisoned images and poisoned purified images (orange) at points on the Langevin dynamics trajectory. Purifying poisoned 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.

These dynamics provide a **concrete intuition for choosing step counts** that best balance poison defense with natural accuracy (given a poison ϵ), hence why we use 150-1000 EBM steps of 75-125 (specifically 150 EBM, 75 DDPM steps in from-scratch scenarios) shown in App. E.2. Further, PureGen-EBM dynamics stay closer to the original images, while

PureGen-DDPM moves further away as we increase the steps as the EBM has explicitly learned a probable data distribution, while the DDPM restoration is highly dependent on the conditional information in the degraded image. These dynamics align with empirical results showing that EBMs better maintain natural accuracy and poison defense with smaller perturbations and across larger distributional shifts, but DDPM dynamics are better suited for larger poison perturbations. Finally, we note the purify times in the x-axes of Fig. 7.2, where PureGen-EBM is much faster for the same step counts to highlight the computational differences for the two methods, which we further explore Section 7.4.3.

7.3.3 PureGen-DDPM Results

We begin by considering many of the same train-time poisoning attacks from the previous Chapter 6 to now compare the performance of PureGen-EBM and PureGen-DDPM. We add baseline defense JPEG compression, which offers powerful defense against poison attacks in numerous scenarios [LZL23].

We also add data availability attack Neural Tangent Generalization Attacka (NTGA), which is a clean-label, black-box data availability attack that can collapse model test accuracies [YW21]. Background for data availability attacks can be found in [YZC22]. We do not focus on such attacks since they are realized in model results during training. They do not pose a latent security risk in deployed models, and arguably have ethical applications within data privacy and content creator protections as discussed in App. E.1.

Our EBMs and DDPMs are trained on the ImageNet (70k) portion of the CINIC-10 dataset, CIFAR-10, and CalTech-256 for poisons scenarios using CIFAR-10, CINIC-10, and Tiny-ImageNet respectively, to ensure no overlap of PureGen train and attacked classifier train datasets [Ale09, DCA18, GHP22].

Table 7.1 shows our primary results using ResNet18 (*ResNet34 for Tiny-IN*) in which **PureGen achieves state-of-the-art (SoTA) poison defense and natural accuracy**

Table 7.1: Poison success and natural accuracy in all ResNet poison training scenarios. We report the mean and the standard deviations (as subscripts) of 100 GM experiments, 50 BP experiments, and NS triggers over 10 classes.

From Scratch										
	CIFAR-10 (ResNet-18)					**CINIC-10 (ResNet-18)**			**Tiny ImageNet (ResNet-34)**	
	Gradient Matching-1%		**Narcissus-1%**			**Narcissus-1%**			**Gradient Matching-0.25%**	
	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Nat Acc (%) ↑
None	44.00	$94.84_{0.2}$	$43.95_{33.6}$	$94.89_{0.2}$	93.59	$62.06_{0.21}$	$86.32_{0.10}$	90.79	26.00	$65.20_{0.5}$
EPIc	10.00	$85.14_{1.2}$	$27.31_{34.0}$	$82.20_{1.1}$	84.71	$49.50_{0.27}$	$81.91_{0.08}$	91.35	18.00	$60.55_{0.7}$
FrieNDs	**0.00**	$\mathbf{91.15}_{0.4}$	$8.32_{22.3}$	$91.01_{0.4}$	83.03	$11.17_{0.25}$	$77.53_{0.60}$	82.21	2.00	$42.74_{7.5}$
JPEG	**0.00**	$90.00_{0.19}$	$\mathbf{1.78}_{1.17}$	$\mathbf{92.94}_{0.15}$	4.13	$18.89_{27.46}$	$81.06_{0.18}$	92.12	10.00	$60.01_{0.47}$
PureGen-DDPM	**0.00**	$\mathbf{90.93}_{0.20}$	$\mathbf{1.64}_{0.82}$	$90.99_{0.22}$	2.83	$\mathbf{4.76}_{2.37}$	$\mathbf{79.35}_{0.08}$	**7.74**	**0.00**	$50.50_{0.80}$
PureGen-EBM	1.00	$\mathbf{92.98}_{0.2}$	$\mathbf{1.39}_{0.8}$	$\mathbf{92.92}_{0.2}$	**2.50**	$7.73_{0.08}$	$\mathbf{82.37}_{0.14}$	29.48	2.00	$\mathbf{63.27}_{0.4}$

Transfer Learning (CIFAR-10, ResNet-18)									
	Fine-Tune					**Linear - Bullseye Polytope**			
	Bullseye Polytope-10%		**Narcissus-10%**			**BlackBox-10%**		**WhiteBox-1% (CIFAR-100)**	
	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Poison Success (%) ↓	Avg Nat Acc (%) ↑
None	46.00	$89.84_{0.9}$	$33.41_{33.9}$	$90.14_{2.4}$	98.27	93.75	$83.59_{2.4}$	98.00	$70.09_{0.2}$
EPIc	42.00	$81.95_{5.6}$	$20.93_{27.1}$	$88.58_{2.0}$	91.72	66.67	$84.34_{3.8}$	63.00	$60.86_{1.5}$
FrieNDs	8.00	$87.82_{1.2}$	$3.04_{5.1}$	$89.81_{0.5}$	17.32	33.33	$85.18_{2.3}$	19.00	$60.90_{0.6}$
JPEG	**0.00**	$\mathbf{90.40}_{0.44}$	$2.95_{3.71}$	$87.63_{0.49}$	12.55	**0.00**	$92.44_{0.47}$	8.00	$50.42_{0.73}$
PureGen-DDPM	**0.00**	$\mathbf{91.53}_{0.15}$	$\mathbf{1.88}_{1.12}$	$\mathbf{90.69}_{0.26}$	**3.42**	**0.00**	$\mathbf{93.81}_{0.08}$	9.0	$54.53_{0.64}$
PureGen-EBM	**0.00**	$87.52_{1.2}$	$\mathbf{2.02}_{1.0}$	$\mathbf{89.78}_{0.6}$	**3.85**	**0.00**	$92.38_{0.3}$	**6.00**	$\mathbf{64.98}_{0.3}$

in all poison scenarios. Both PureGen methods show large improvements over baselines in triggered NS attacks (PureGen matches or exceeds previous SoTA with a 1-6% poison defense reduction and 0.5-1.5% less degradation in natural accuracy), while maintaining perfect or near-perfect defense with improved natural accuracy in triggerless BP and GM scenarios. Note that PureGen-EBM does a better job maintaining natural accuracy in the 100 class scenarios (BP-WhiteBox and Tiny-IN), while PureGen-DDPM tends to get much better poison defense when the PureGen-EBM is not already low.

Table 7.2 shows selected results for additional models MobileNetV2, DenseNet121, and Hyperlight Benchmark (HLB) [Bal23]. **In all results, PureGen is again SoTA**, except

Table 7.2: Results for additional models (MobileNetV2, DenseNet121, and HLB) and the NTGA data-availability attack. PureGen remains state-of-the-art for all train-time *latent* attacks, while NTGA defense shows near SoTA performance. *All NTGA baselines pulled from [DEL24].

Scenario	From Scratch NS($\epsilon = 8$)-1%						Linear Transfer BlackBox BP-10%			
Model	MobileNetV2		DenseNet121		Hyperlight Bench		MobileNetV2		DenseNet121	
Defense	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Poison Success (%) ↓	Avg Nat Acc (%) ↑
None	$32.70_{0.25}$	$93.92_{0.13}$	$46.52_{32.2}$	$95.33_{0.1}$	$76.39_{16.35}$	$93.95_{0.10}$	81.25	$73.27_{0.97}$	73.47	$82.13_{1.62}$
EPIc	$22.35_{0.24}$	$78.16_{9.93}$	$32.60_{29.4}$	$85.12_{2.4}$	$10.58_{18.35}$	$24.88_{6.04}$	56.25	$54.47_{5.57}$	66.67	$70.20_{10.15}$
FrieNDs	$\mathbf{2.00}_{0.01}$	$88.82_{0.57}$	$8.60_{21.2}$	$91.55_{0.3}$	$11.35_{18.45}$	$87.03_{1.52}$	41.67	$68.86_{1.50}$	60.42	$80.22_{1.90}$
JPEG	$\mathbf{2.30}_{1.20}$	$86.60_{0.13}$	$\mathbf{1.90}_{1.54}$	$92.03_{0.22}$	$\mathbf{1.73}_{0.97}$	$90.92_{0.22}$	2.08	$73.14_{0.71}$	**0.00**	$78.67_{1.60}$
PureGen-DDPM	$\mathbf{2.13}_{1.02}$	$86.91_{0.23}$	$\mathbf{1.71}_{0.94}$	$90.94_{0.23}$	$\mathbf{1.75}_{0.90}$	$89.34_{0.25}$	**0.00**	$\mathbf{83.15}_{0.02}$	**0.00**	$\mathbf{89.02}_{0.15}$
PureGen-EBM	$\mathbf{1.64}_{0.01}$	$\mathbf{91.75}_{0.13}$	$\mathbf{1.42}_{0.7}$	$\mathbf{93.48}_{0.1}$	$\mathbf{1.89}_{1.06}$	$\mathbf{91.94}_{0.14}$	**0.00**	$78.57_{1.37}$	**0.00**	$\mathbf{89.29}_{0.94}$

	NTGA Data Availability Attack								
Defense	None	FAutoAug.*	Median Blur*	TVM*	Grayscale*	Avatar *	**JPEG**	**PureGen-DDPM**	**PureGen-EBM**
Avg Natural Accuracy (%) ↑	$11.49_{0.69}$	$27.56_{2.45}$	$28.43_{1.41}$	$41.41_{1.37}$	$\mathbf{81.27}_{0.27}$	$\mathbf{86.22}_{0.38}$	$79.22_{0.25}$	$83.48_{0.43}$	$85.22_{0.38}$

for NTGA data-availability attack, where PureGen is just below SoTA method Avatar (which is also a diffusion based approach). But we again emphasize data-availability attacks are not the focus of PureGen which secures against 'latent' attacks.

7.4 Comparing PureGen-EBM and PureGen-DDPM

Energy-based models (EBMs) and denoising diffusion probabilistic models (DDPMs) are two distinct approaches to generative modeling, each with its own strengths and weaknesses. In the context of data purification, understanding these differences is crucial for selecting the most appropriate method for a given task and for developing effective combination strategies.

PureGen-EBM, based on energy-based models, learns to capture the underlying data manifold by assigning low energy values to samples that resemble the training data and high energy values to outliers or anomalous samples. This approach allows PureGen-EBM to effectively model the inherent structure of the data, making it robust to distributional shifts and capable of identifying and purifying corrupted samples. However, the training process of EBMs can be challenging, particularly in terms of stability and convergence, and the purification process may be computationally expensive due to the need for Markov chain Monte Carlo (MCMC) sampling.

On the other hand, PureGen-DDPM, based on denoising diffusion probabilistic models, learns to generate clean samples by reversing a gradual noising process. This approach is highly effective at modeling complex data distributions and can generate high-quality samples. In the context of data purification, PureGen-DDPM excels at removing complex perturbations and artifacts from corrupted samples. However, the purification process may result in a slight loss of fidelity to the original data distribution, as the model's primary objective is to remove noise rather than to strictly adhere to the underlying data manifold.

In the following subsections, we explore a comprehensive comparison of PureGen-EBM and PureGen-DDPM in terms of their performance on poison defense, their robustness to distributional shifts and poisoning in training, and their computational efficiency. By understanding the strengths and weaknesses of each approach, we aim to provide insights into their suitability for different purification scenarios and to motivate the development of effective combination strategies.

7.4.1 Robustness to Distributional Shifts

An important consideration for PureGen is the distributional shift between the data used to train the generative models and the target dataset to be purified. Figure 7.3 explores this by training PureGen-EBM and PureGen-DDPM on increasingly out-of-distribution (OOD) datasets while purifying the CIFAR-10 dataset (NS attack). We quantify the distributional shift using the Fréchet Inception Distance (FID) [HRU18] between the original CIFAR-10 training images and the OOD datasets. Notably, **both methods maintain SoTA or near SoTA poison defense across all training distributions,** highlighting their effectiveness even under distributional shift. The results show that PureGen-EBM is more robust to distributional shift in terms of maintaining natural accuracy, with only a slight drop in

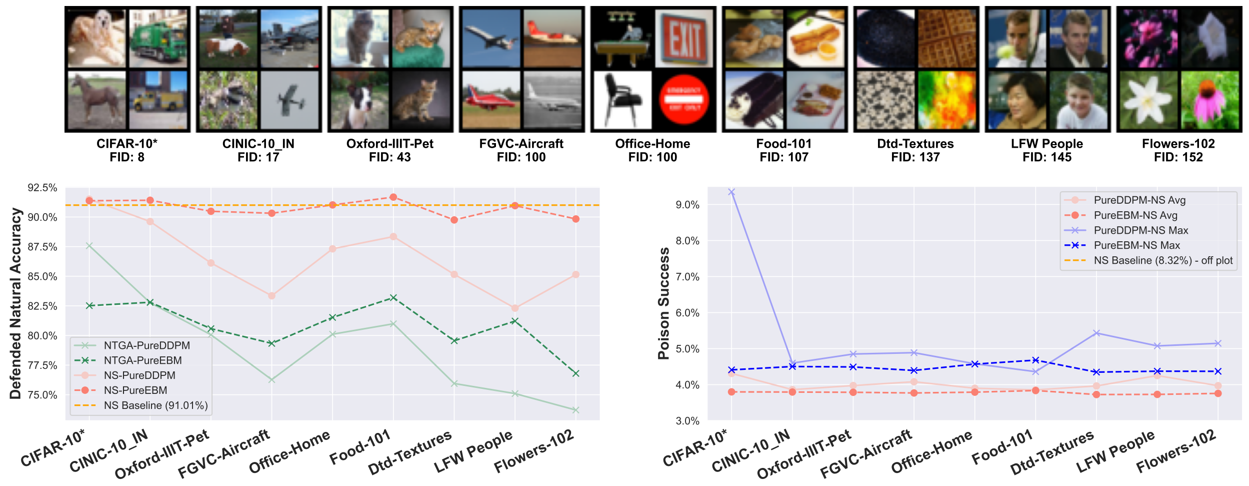


Figure 7.3: PUREGEN-EBM vs. PUREGEN-DDPM with increasingly Out-of-Distribution training data (for generative model training) and purifying target/attacked distribution CIFAR-10. PUREGEN-EBM is much more robust to distributional shift for natural accuracy while both PUREGEN-EBM and PUREGEN-DDPM maintain SoTA poison defense across all train distributions **CIFAR-10 is a "cheating" baseline as clean versions of poisoned images are present in training data.*

performance even when trained on highly OOD datasets like Flowers-102 and LFW people. In contrast, PUREGEN-DDPM experiences a more significant drop in natural accuracy as the distributional shift increases. Note that the CIFAR-10 is a "cheating" baseline, as clean versions of the poisoned images are present in the generative model training data, but it provides an upper bound on the performance that can be achieved when the generative models are trained on perfectly in-distribution data.

7.4.2 Robustness to Poisoning

Another important consideration is the robustness of PUREGEN when the generative models themselves are trained on poisoned data. Table 7.3 shows the performance of PUREGEN-EBM and PUREGEN-DDPM when their training data is fully poisoned with the Narcissus (NS)

Table 7.3: Both PureGen-EBM and PureGen-DDPM are robust to NS attack even when **fully poisoned (all classes at once)** during model training except for NS Eps-16 for PureGen-EBM

Classifier NS Attack Eps		**8**			**16**		
PureGen w/NS Training Poison		**Nat Acc (%) ↑**	**Poison Success (%) ↓**	**Max Poison (%) ↓**	**Nat Acc (%) ↑**	**Poison Success (%) ↓**	**Max Poison (%) ↓**
0	**PureDDPM**	**91.51±0.13**	**2.62±3.75**	12.70	**90.31±0.18**	**4.61±3.99**	**12.86**
	PureEBM	**91.37±0.14**	**1.60±0.82**	**2.82**	**88.21±0.15**	**8.73±6.29**	**23.05**
8	**PureDDPM**	88.99±0.16	**1.65±0.79**	**2.87**	85.24±0.10	**4.79±2.83**	**10.53**
	PureEBM	**91.11±0.18**	**1.55±0.89**	**2.87**	**87.60±0.18**	**5.35±3.30**	**12.05**
16	**PureDDPM**	88.02±0.21	**1.57±0.79**	**2.79**	83.74±0.21	**2.90±1.54**	**6.11**
	PureEBM	**90.76±0.14**	**1.28±0.86**	**3.43**	85.58±0.40	17.73±14.62	44.15

attack, *meaning that all classes are poisoned simultaneously.* The results demonstrate that both PureGen-EBM and PureGen-DDPM are highly robust to poisoning during model training, maintaining SoTA poison defense and natural accuracy with only exception being PureGen-EBM's performance on the more challenging NS $\epsilon = 16$ attack when poisoned with the same perturbations. While it is unlikely an attacker would have access to both the the generative model and classifier train datasets, these findings highlight the inherent robustness of PureGen, as the generative models can effectively learn the underlying clean data distribution even in the presence of poisoned samples during training. This is a key advantage of PureGen compared to other defenses, especially when there is no secure dataset

7.4.3 PureGen Timing and Limitations

Table 7.4 presents the training times for using PureGen-EBM and PureGen-DDPM (923 and 4181 seconds respectively to purify) on CIFAR-10 using a TPU V3. Although these times may seem significant, PureGen is a universal defense applied once per dataset, making its cost negligible when reused across multiple tasks and poison scenarios. To highlight this, we also present the purification times amortized over the 10 and 100 NS and GM poison scenarios, demonstrating that the cost becomes negligible when the purified dataset is used multiple times relative to baselines like FRIENDS which require retraining for each specific task and poison scenario (while still utilizing the full dataset unlike EPIC). PUREGEN-EBM generally has lower purification times compared to PUREGEN-DDPM, making it more suitable for subtle and rare perturbations. Conversely, PUREGEN-DDPM can handle more severe perturbations but at a higher computational cost and potential reduction in natural accuracy.

Table 7.4: PUREGEN and baseline Timing Analysis on TPU V3

	Train Time (seconds)		
	Single Classifier (Median)	**Gradient Matching 100 Classifiers**	**Narcissus 10 Classifiers**
None, JPEG	**3690**	3673_{48}	$\mathbf{5288}_{29}$
EPIC	3650	3624_{153}	5212_{295}
FRIENDS	11502	11578_{627}	$12868s_{573}$
PUREGEN-EBM $_{T=[150,0,1]}$	4613	3699_{48}	$\mathbf{5380}_{32}$
PUREGEN-DDPM $_{T=[0,75,1]}$	7871	3731_{48}	5706_{37}

Training the generative models for PUREGEN involves substantial computational cost and data requirements. However, as shown in Table 7.3 and Figure 7.3, these models remain effective even when trained on poisoned or out-of-distribution data. This universal applicability justifies the initial training cost, as the models can defend against diverse poisoning scenarios. So while JPEG is a fairly effective baseline, the added benefits of PUREGEN start to outweigh the compute as the use cases of the dataset increase.

7.5 PUREGEN Combinations: PUREGEN-NAIVE, PUREGEN-REPS, PUREGEN-FILT

While both PUREGEN-EBM and PUREGEN-DDPM have shown impressive results in data purification tasks, they each have their own strengths and weaknesses. PUREGEN-EBM excels at capturing the underlying data manifold and is more robust to distributional shifts, but it can struggle with more complex perturbations. On the other hand, PureGen-DDPM is highly effective at removing complex perturbations but may sacrifice some fidelity to the original data distribution.

To leverage the strengths of both PureGen-EBM and PureGen-DDPM, we propose PureGen combinations:

1. PureGen-Naive ($\Psi_{T|T_{EBM}>0, T_{DDPM}>0, T_{Reps}=1}$): Apply a fixed number of PureGen-EBM steps followed by PureGen-DDPM steps. While this approach does improve the purification results compared to using either method alone, it does not fully exploit the synergy between the two techniques.

2. PureGen-Reps ($\Psi_{T|T_{EBM}>0, T_{DDPM}>0, T_{Reps}>1}$): To better leverage the strengths of both methods, we propose a repetitive combination, where we alternate between a smaller number of PureGen-EBM and PureGen-DDPM steps for multiple iterations.

3. PureGen-Filt ($\Psi_{T,k|T_{EBM}\geq 0, T_{DDPM}\geq 0, 0<k<1}$): In this combination, we first use PureGen-EBM to identify a percentage of the highest energy points in the dataset, which are more likely to be samples with poisoned perturbations as shown in Fig. 6.1. We then selectively apply PureGen-EBM or PureGen-DDPM purification to these high-energy points.

We note that methods 2 and 3 require extensive hyperparameter search with performance sweeps using the HLB model in App E.5, as there was little intuition for the amount of reps (T_{Reps}) or the filtering threshold (k) needed. Thus, we do not include these methods in our core results, but instead show the added performance gains on higher power poisons in Table 7.5, both in terms of increased perturbation size $\epsilon = 16$ and increased poison % (and both together). We note that 10% would mean the adversary has poisoned the entire class in CIFAR-10 with an NS trigger, and $\epsilon = 16$ is starting to approach visible perturbations, but both are still highly challenging scenarios worth considering for purification.

Table 7.5: PUREGEN-NAIVE, PUREGEN-REPS, and PUREGEN-FILT results showing further performance gains on increased poison power scenarios

	Narcissus $\epsilon = 8$ 10%			**Narcissus $\epsilon = 16$ 1%**			**Narcissus $\epsilon = 16$ 10%**		
	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓
None	96.27$_{6.62}$	84.57$_{0.60}$	99.97	83.63$_{12.09}$	93.67$_{0.11}$	97.36	99.35$_{0.81}$	84.58$_{0.63}$	99.97
Best Baseline	16.95$_{9.72}$	84.66$_{1.51}$	33.96	11.85$_{12.60}$	87.72$_{0.19}$	36.90	71.28$_{22.90}$	82.83$_{0.43}$	99.25
PUREGEN-DDPM	**6.38**$_{5.16}$	**85.86**$_{0.46}$	16.29	**5.21**$_{3.35}$	86.16$_{0.19}$	13.32	69.38$_{16.73}$	83.58$_{1.02}$	89.35
PUREGEN-EBM	52.48$_{23.29}$	86.14$_{1.82}$	99.86	7.35$_{4.46}$	85.61$_{0.25}$	16.94	77.50$_{7.01}$	78.79$_{0.93}$	90.84
PUREGEN-NAIVE	10.43$_{8.58}$	88.20$_{0.54}$	27.42	**5.20**$_{2.61}$	85.95$_{0.23}$	**9.80**	63.01$_{15.24}$	83.14$_{0.90}$	87.17
PUREGEN-REPS	**3.75**$_{2.28}$	**85.56**$_{0.22}$	**7.74**	**4.95**$_{2.48}$	85.79$_{0.18}$	10.75	**53.79**$_{17.14}$	**83.92**$_{1.02}$	**81.09**
PUREGEN-FILT	**6.47**$_{6.98}$	**86.08**$_{2.00}$	18.81	**5.74**$_{4.05}$	**90.52**$_{0.18}$	16.08	69.13$_{12.94}$	85.47$_{1.45}$	87.66

7.6 PUREGEN for Intellectual Property Purification

Data purification can extend well beyond the realm of invisible perturbations for attacks. There are many practical applications for purifying image datasets of visible perturbations such as Intellectual Property (IP) logos or watermarks. These perturbations can be undesirable for various reasons, such as copyright infringement or visual distraction. Manually detecting and removing these perturbations might be expensive or time consuming. In this section, we explore the application of PUREGEN to the task of automatically purifying IP from image datasets with minimal degradation to the broad features of the dataset.

7.6.1 Problem Setting

We use the Tiny ImageNet dataset, which consists of 100,000 images across 200 classes, and combine it with a dataset of 32 NFL Logos at various mixing ratios and sizes. The logos are randomly placed on the images to ensure that they are uncorrelated with the image classes. This is done by first inserting the logo into an image in a unique random location per epoch. Let α be the mixing ratio, defined as the proportion of images with logos in the dataset, and the final image is an interpolation (as seen in Figure 7.4):

$$x_{train} = x_{orig} \times (1 - \alpha) + x_{logo} \times \alpha \tag{7.3}$$

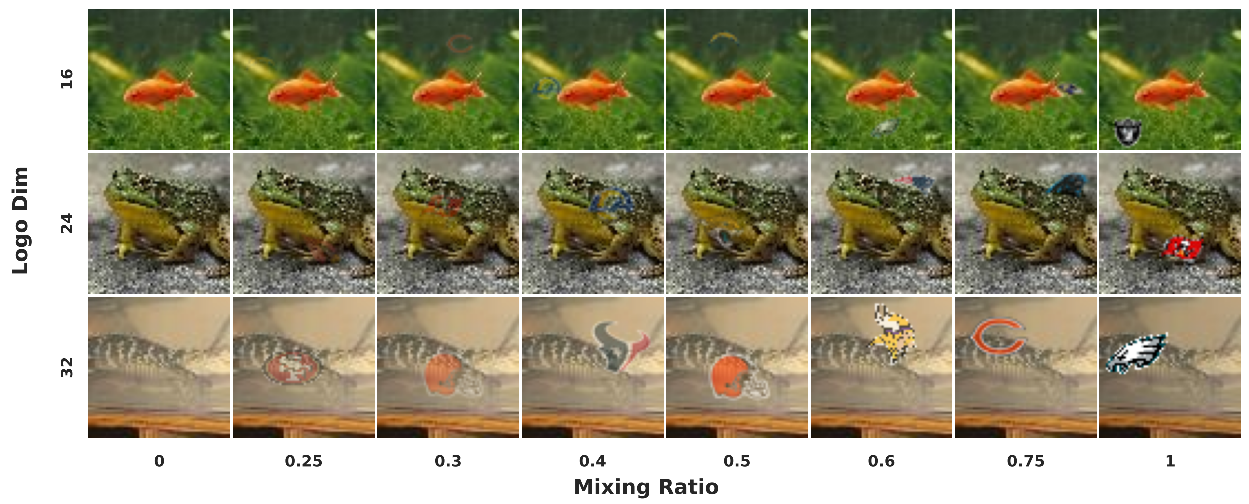


Figure 7.4: Tiny-ImageNet "NFL Logos" dataset samples (64x64) at various mix ratios and logo dimensions.

We then split the Tiny ImageNet "NFL Logos" dataset into two parts for each class: one for training the generative models PureGen which lacks any logos, $\alpha = 0$, and the other for training the ResNet18 classifiers. We train two classifiers: one to classify the original image classes and another to classify the presence of logos. The objective is now to reduce the classification accuracy of the logos classifier while minimally impacting the natural accuracy for the classes. Formally:

$$\underset{\text{PureGen}}{\operatorname{argmax}} A_{classes} - A_{logo} \tag{7.4}$$

where $A_{classes}$ and A_{logo} are the accuracies of the natural classes and the logo classes respectively and we abuse notation to find the PureGen method and settings that maximizes this classification difference.

7.6.2 Analysis and Results

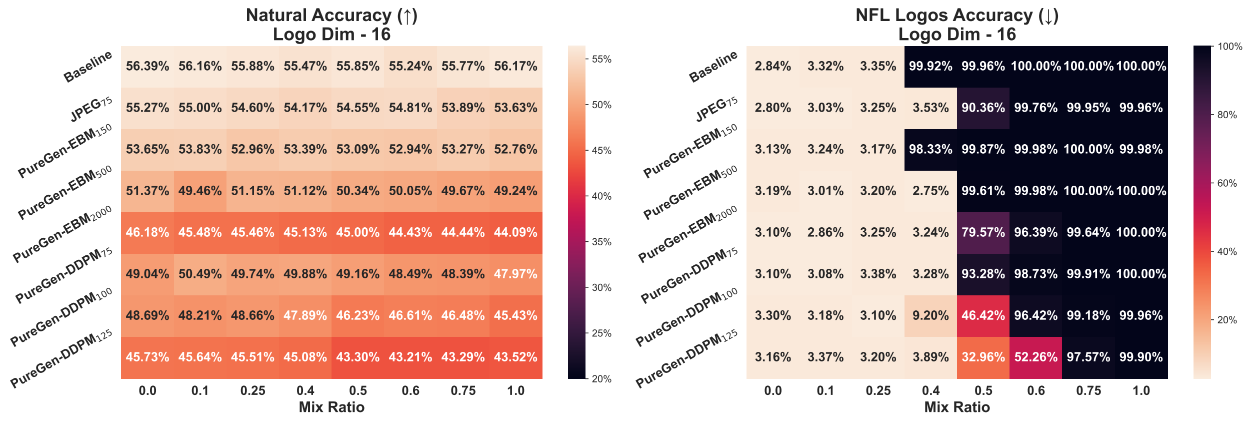


Figure 7.5: Both PureGen-EBM and PureGen-DDPM can collapse logo classifier accuracy with though steps for the middle mixing rations (0.4-0.6), with PureGen-DDPM 125 steps extending out to a mixing ratio of 0.6, but also generally sacrificing more natural accuracy the PureGen-EBM.

Results for logo dimension of 16 are heatmapped in Figure 7.5. Note that A_{logo} is largely bimodal, achieving a nearly perfect classification accuracy (>99%) or collapsing to a very low or "untrainable" amount (3%). In addition, the logos classifier can only be trained after $\alpha > 0.25$.

We see that PureGen purifies IP from image datasets with various successes over baseline comparisons JPEG compression and without any processing. **Applying PureGen purification techniques, particularly with higher numbers of EBM or DDPM steps, can push out the mixing ratio at which the logo classifier collapses while maintaining a reasonable level of $A_{classes}$, although not without noticeable sacrifice**. For PureGen-EBM, 2000 steps can purify $\alpha = 0.4$ (and is starting to do so at $\alpha = 0.5$) but $A_{classes}$ decreases anywhere from 10-12% at the various α's. We can start to collapse A_{logo} even further at $\alpha = 0.6$ with PureGen-DDPM and 125 steps, but we sacrifice slightly more natural accuracy. Full results across multiple logo sizes are in Appendix E.6.

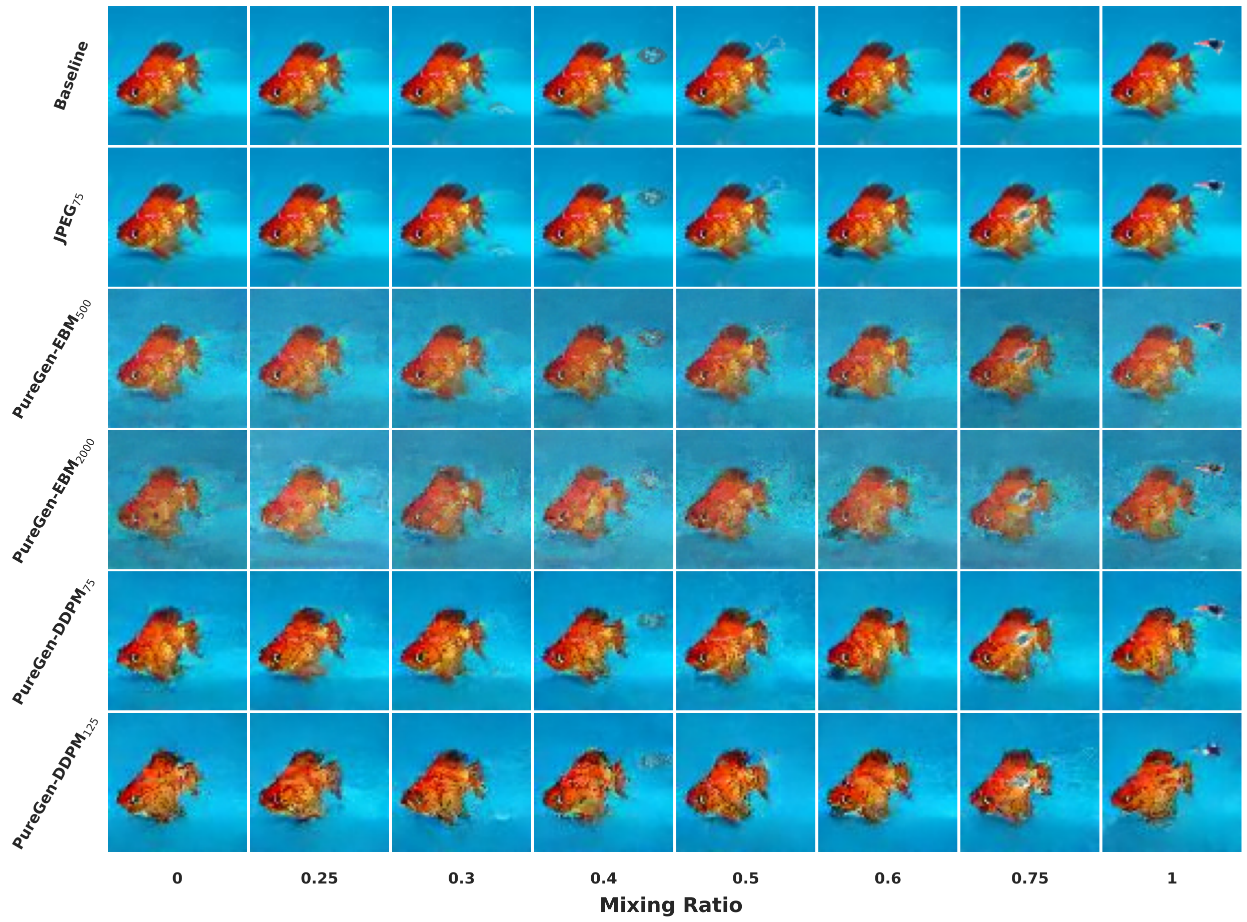


Figure 7.6: Examples of PureGen on an IP data sample showing the unique behavior of PureGen-EBM and PureGen-DDPM and their potential logo purification capabilities.

In Figure 7.6 we show the visual impact of baselines and various PureGen methods. The distinct nature of PureGen-EBM and PureGen-DDPM is apparent. PureGen-EBM purified images have more diffuse impacts, with noticeable color perturbations across the image, while PureGen-DDPM images are more similar in the color space but occasionally contain greater structural deficiencies. This aligns with analysis from Section 7.3.2 where PureGen-DDPM can move images much further in the pixel space, particularly with the degradation in the forward noise process, thus also allowing for greater purification and disruption of undesired perturbations. But this can come at the sacrifice of fidelity to the underlying data distribution, where the relevant features of the images are not restored ideally.

The IP purification exercise offers further insight into the capabilities of PureGen to extend beyond poison defense and invisible perturbations. There are many avenues for further research. For instance, we could explore combinations of PureGen-DDPM and PureGen-EBM as we did in previous sections. We could also explore the addition of negative training examples in EBM training or look to further incentivize degradation of IP in generative model training. These initial results show that PureGen can offer additional capabilities as a potential generalized IP purification technique with further exploration and improvement.

7.7 Conclusion

In this chapter, we introduced PureGen, a suite of data purification techniques that harness the power of generative models, specifically energy-based models (PureGen-EBM) and denoising diffusion probabilistic models (PureGen-DDPM). Using the unique strengths of these two approaches, PureGen provides a robust and effective framework for defending against adversarial poison attacks and removing unwanted perturbations from image datasets.

Through experiments, we demonstrated the effectiveness of PureGen in various purification tasks, showcasing the significant reduction in poison success rates while maintaining high natural accuracy. We also explored the robustness of the various techniques to distributional shifts and poisoning in generative model training data, highlighting the generalization capability of these methods.

In addition, we proposed several combination strategies that take advantage of the synergies between PureGen-EBM and PureGen-DDPM, allowing for more efficient and effective purification processes. These combinations, such as PureGen-Reps andPureGen-Filt, allow for more gradual or targeted purification. Finally, we applied PureGen to the task of purifying intellectual property from image datasets, demonstrating the potential utility to remove visible perturbations while preserving natural accuracy.

The PureGen framework is still in its early days and provides potential for much future work. The combination PureGen techniques are extensive but far from comprehensive. The train datasets and size of the models themselves are always opportunities for improvements, and increasing both model and dataset size routinely results in capability improvements across deep learning research. We could also look at novel architecture that can capitalize on the capabilities of both techniques, such as a 'U-Net' architecture EBM model that could do pixel-wise energy and ideally purify the undesired perturbations more explicitly.

The PureGen framework represents a significant advance in the field of data purification, providing a powerful and flexible tool-set to ensure the integrity and reliability of image data sets in the face of adversarial attacks and unwanted perturbations. As the importance of data quality and security continues to grow in the era of deep learning, PureGen offers a promising solution for researchers and practitioners who seek to build robust and reliable machine learning systems.

CHAPTER 8

Conclusion

Throughout this thesis, we have explored the integration of causal priors and data purification techniques in machine learning models, with a focus on generative architectures, to enhance their robustness, interpretability, and performance. Our research has spanned several key areas, including the incorporation of causal knowledge in generative models for counterfactual reasoning and de-biasing, the development of a causal structural hypothesis testing framework, the exploration of latent space augmentations for improved data generation, and the introduction of novel data purification methods for defending against adversarial poisoning attacks.

The main contributions of this thesis can be summarized as follows:

1. We proposed a counterfactual-based method, CCGM, for de-biasing generative models using causal priors, enabling the generation of out-of-distribution samples from the training dataset. This approach allows for the simulation of "what-if" scenarios and the generation of diverse, bias-free data.

2. We developed a causal structural hypothesis testing framework, CSHTest and CSVHTest, which leverage generative deep learning and out-of-distribution data generalization to compare and evaluate structural causal models. This framework provides a principled approach for assessing the validity of causal hypotheses in the absence of ground truth.

3. We introduced a latent space augmentation technique, LAVAE, which enables the composable generation of diverse and high-quality augmented samples efficiently. By

learning augmentations in the latent space, LAVAE offers greater control and interpretability compared to traditional image space augmentations.

4. We proposed PureGen, a set of novel data purification methods that utilize the stochastic dynamics of energy-based models and denoising diffusion probabilistic models, independently and in combination, to defend against adversarial train-time poisoning attacks. These methods achieve state-of-the-art performance in terms of poison defense and natural accuracy, while maintaining broad applicability and robustness to distributional shifts and poisoning of the generative models themselves.

The overarching theme of this thesis is the pursuit of robustness and improved generalization in machine learning models. By incorporating causal priors and data purification techniques, we aim to build models that are not only accurate but also aligned with human intuition and resilient to adversarial attacks.

The connection between robustness, poison defense, and causality is of interest to us. Causal reasoning allows us to capture the underlying mechanisms that generate the data, rather than relying solely on statistical associations. By incorporating causal knowledge, we can build models that are more robust to distributional shifts and less susceptible to spurious correlations. This is especially important in the context of adversarial attacks, where the attacker may exploit the model's reliance on superficial patterns to manipulate its behavior.

Moreover, the lack of explicit structure and semi/self-supervision in traditional deep learning models can lead to overly narrow definitions derived from the training data alone. This makes the models vulnerable to poisoning attacks, as the attacker can carefully craft examples that fit the model's narrow definition but lead to unintended consequences. By incorporating causal priors and data purification techniques, we can guide the model towards more robust and meaningful representations that are less easily fooled by adversarial perturbations.

The research presented in this thesis also highlights the importance of interpretability and transparency in machine learning. By developing methods that allow for counterfactual reasoning, hypothesis testing, and controlled data generation, we can gain insights into the model's decision-making process and build trust with end-users. This is particularly crucial in high-stakes applications, such as healthcare and autonomous systems, where the consequences of model failures can be severe.

This thesis contributes to the ongoing effort to build interpretable and trustworthy machine learning models. By integrating causal reasoning and data purification techniques, we have shown that it is possible to create models that are not only accurate but also aligned with human intuition and resilient to adversarial attacks. As machine learning continues to be applied in increasingly critical domains, the importance of these properties cannot be overstated. We hope that the research presented in this thesis will inspire further work in this direction and contribute to the development of more reliable and responsible artificial intelligence systems.

APPENDIX A

De-Biasing Generative Models using Counterfactual Methods

A.1 Additonal Results

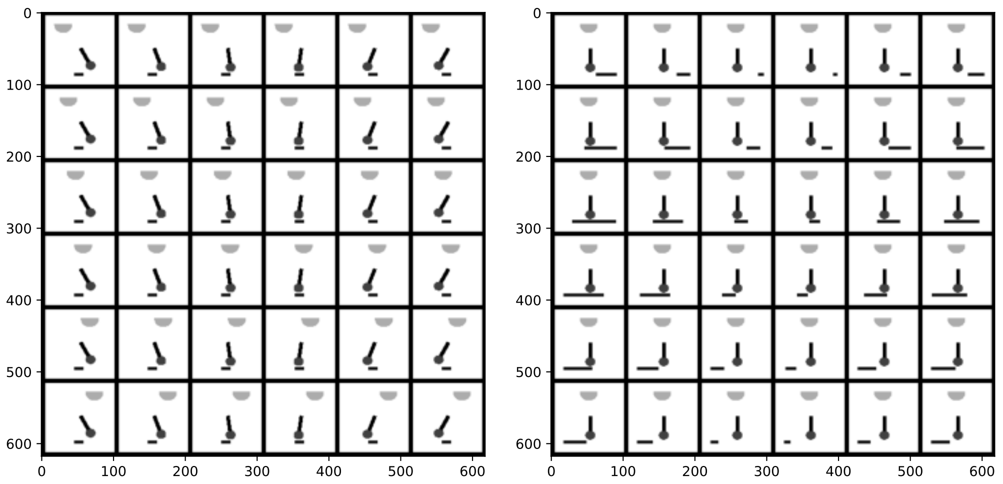


Figure A.1: Visualization of the latent space of the causal concepts

A.2 Tabular Variable Reconstructions

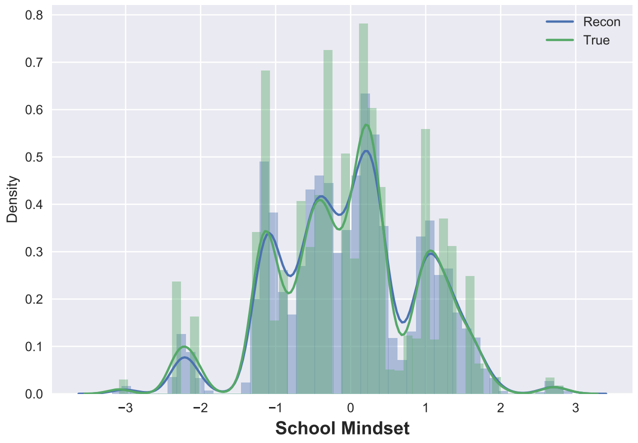


Figure A.2: School mindset distribution CCGM reconstruction example

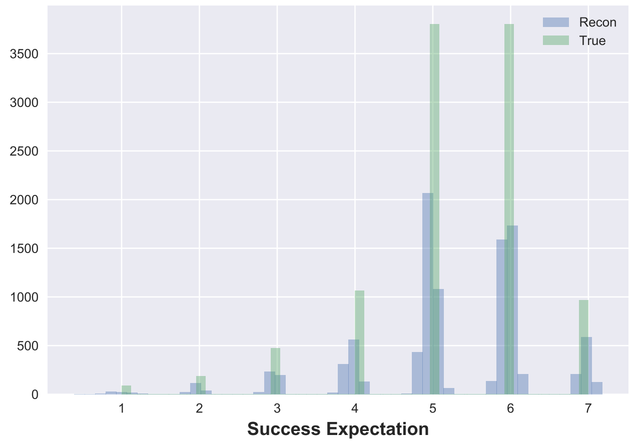


Figure A.3: Success Expectation distribution CCGM reconstruction example

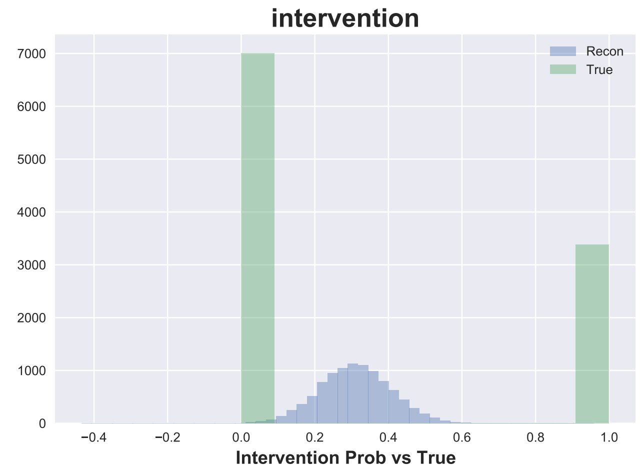


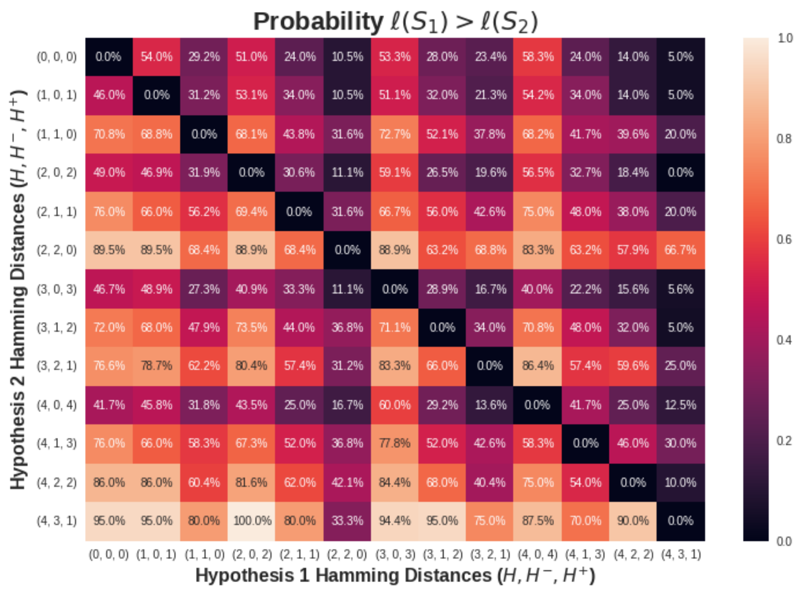
Figure A.4: Intervention probability distribution CCGM reconstruction example. Note that top 30% is used to bin data into binary before calculating ATEs.

APPENDIX B

Hypothesis Testing using Causal and Causal Variational Generative Models

B.1 DAG Simulation Results

B.1.1 DAG Size 5x5: Linear



	(0, 0, 0)	(1, 0, 1)	(1, 1, 0)	(2, 0, 2)	(2, 1, 1)	(2, 2, 0)	(3, 0, 3)	(3, 1, 2)	(3, 2, 1)	(4, 0, 4)	(4, 1, 3)	(4, 2, 2)	(4, 3, 1)
(0, 0, 0)	0.0%	54.0%	29.2%	51.0%	24.0%	10.5%	53.3%	28.0%	23.4%	58.3%	24.0%	14.0%	5.0%
(1, 0, 1)	46.0%	0.0%	31.2%	53.1%	34.0%	10.5%	51.1%	32.0%	21.3%	54.2%	34.0%	14.0%	5.0%
(1, 1, 0)	70.8%	68.8%	0.0%	68.1%	43.8%	31.6%	72.7%	52.1%	37.8%	68.2%	41.7%	39.6%	20.0%
(2, 0, 2)	49.0%	46.9%	31.9%	0.0%	30.6%	11.1%	59.1%	26.5%	19.6%	56.5%	32.7%	18.4%	0.0%
(2, 1, 1)	76.0%	66.0%	56.2%	69.4%	0.0%	31.6%	66.7%	56.0%	42.6%	75.0%	48.0%	38.0%	20.0%
(2, 2, 0)	89.5%	89.5%	68.4%	88.9%	68.4%	0.0%	88.9%	63.2%	68.8%	83.3%	63.2%	57.9%	66.7%
(3, 0, 3)	46.7%	48.9%	27.3%	40.9%	33.3%	11.1%	0.0%	28.9%	16.7%	40.0%	22.2%	15.6%	5.6%
(3, 1, 2)	72.0%	68.0%	47.9%	73.5%	44.0%	36.8%	71.1%	0.0%	34.0%	70.8%	48.0%	32.0%	5.0%
(3, 2, 1)	76.6%	78.7%	62.2%	80.4%	57.4%	31.2%	83.3%	66.0%	0.0%	86.4%	57.4%	59.6%	25.0%
(4, 0, 4)	41.7%	45.8%	31.8%	43.5%	25.0%	16.7%	60.0%	29.2%	13.6%	0.0%	41.7%	25.0%	12.5%
(4, 1, 3)	76.0%	66.0%	58.3%	67.3%	52.0%	36.8%	77.8%	52.0%	42.6%	58.3%	0.0%	46.0%	30.0%
(4, 2, 2)	86.0%	86.0%	60.4%	81.6%	62.0%	42.1%	84.4%	68.0%	40.4%	75.0%	54.0%	0.0%	10.0%
(4, 3, 1)	95.0%	95.0%	80.0%	100.0%	80.0%	33.3%	94.4%	95.0%	75.0%	87.5%	70.0%	90.0%	0.0%

Figure B.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.

B.2 More about the Pendulum

B.2.1 Pendulum Dataset

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 B.2 shows the true DAG and an example generated image. Here, the sun and pendulum variables are exogenous, and the shadow variables are endogenous. 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.

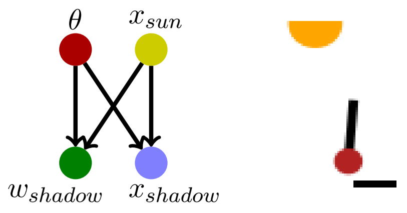


Figure B.2: Pendulum toy example. (a) The structural DAG dictating the pendulum tabular dataset. (b) A visual representation of one of the results from the pendulum dataset.

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 in operating on various trigonometric functions. In the shadow width case, we also deal with an maximum as we don't want the width to go below 0. 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.

B.2.2 Pendulum Hypotheses

We introduce 6 enumerated hypotheses for the pendulum dataset, enumerated in Figure B.3. 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. Two points of interest. 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 SCM's 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.

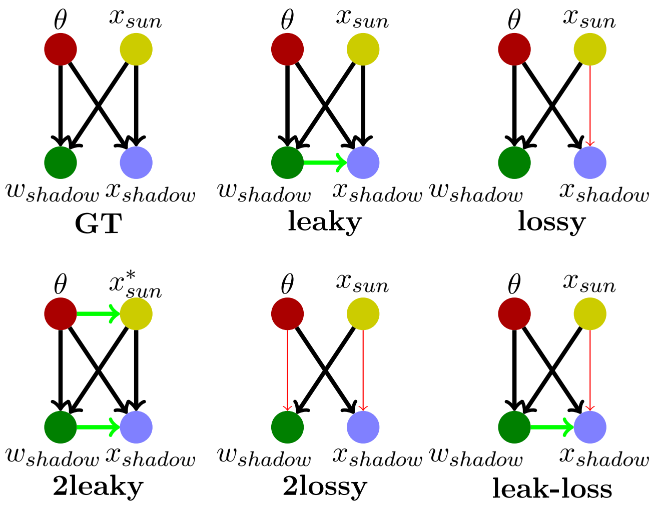


Figure B.3: The 6 enumerated pendulum hypotheses that we try out. Red and thin arrows are arrows that we remove from the true DAG and green arrows are arrows that we add to the true DAG. In the case of 2leaky, we maintain x_{sun} as an exogenous variable, but allow θ information to also influence its value.

We also tried a couple other hypotheses, such as adding the link from θ to x_{sun} by dropping the exogenousity of x_{sun} and the inverse DAG (which is just the same ground truth DAG except with inverted arrows). Neither of them performed well, so we removed them from further analysis.

B.2.3 Pendulum Training Architecture and Hyperparameters

Unfortunately, there is a limitation with regards 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 initializations.
- Activation: Soft Leaky ReLU.
- Optimizer: PSGD (discussed further in B.4). Initial Learning Rate of 0.01.
- Cosine Annealing Scheduler. Warm Restarts implemented for non-variational models.

B.2.4 Further Pendulum Results

Full numerical results of Figure B.1. Train and Loss trajectory curves are shown in Figures B.4 and B.5.

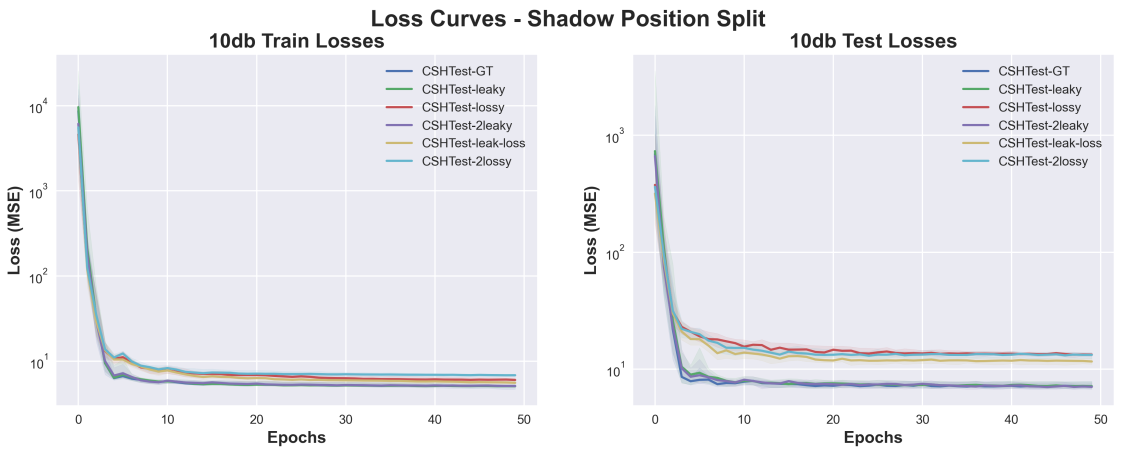


Figure B.4: Loss trajectory of the Sun Split OOD Run

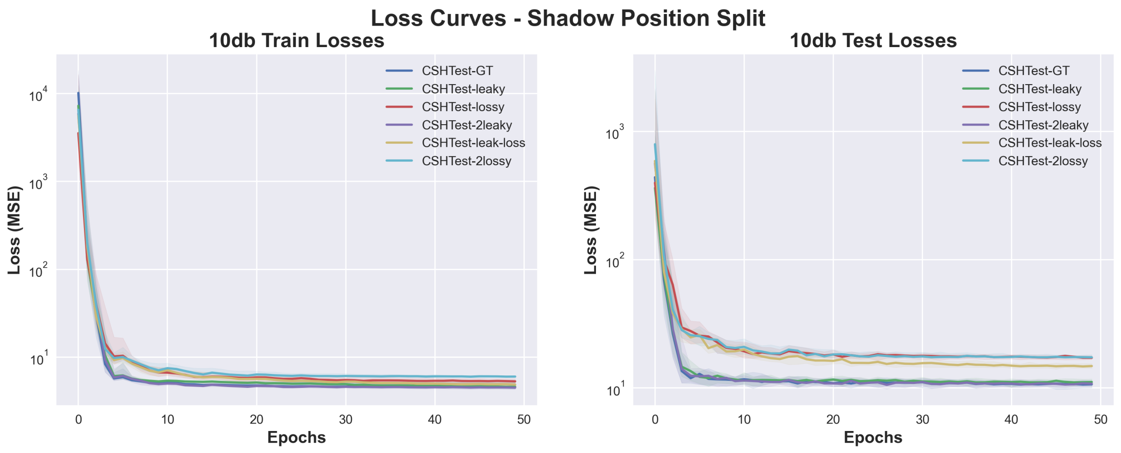


Figure B.5: Loss trajectory of the Shadow Position Split OOD Run

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

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

B.3 Background Theory

B.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 [KW13b]. Thus CSVHTest can generate new data points that are not deterministic on the inputs, allowing for synthetic data generation. CSVHTest consists of an encoder, a CSHTest causal layer and a decoder.

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

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

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 approximators on these transformations.

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

$$\ell_{latent} = \ell(\boldsymbol{z}, \hat{\boldsymbol{z}})$$

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

$$\ell_{\text{CSHT}_{\text{EST}}} = \ell_{MSE} + \lambda_{KL} * \ell_{KL} + \lambda_{latent} * \ell_{latent}$$

B.3.2 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 $p(\cdot)$ [KW13b].

$$\text{ELBO} = \mathbb{E}_{q_{\mathcal{X}}}\left[\mathbb{E}_{\mathbf{z}\sim q_\phi}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))\right] \tag{B.1}$$

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

$$\mathbf{z} = \mathbf{S}^T\mathbf{z} + \boldsymbol{\epsilon} \tag{B.2}$$

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 is 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 (B.2) such that

$$\mathbf{z}_i = g_i(\mathbf{S}_i \circ \mathbf{z}) + \boldsymbol{\epsilon} \tag{B.3}$$

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 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 (B.3) degenerates back to Equation (B.2) [NZF19].

Including the causal layer introduces many auxiliary loss functions that we mostly adopt [YLC20]. First is a label loss (B.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 $\mathbf{S}$.

$$\ell_u = \mathbb{E}_{q\mathcal{X}} \left[\sum_{i=1}^{n} \| u_i - f_i(\mathbf{S}_i \circ \mathbf{u}) \|^2 \right] \tag{B.4}$$

The latent loss tries to enforce the SCM, described by Equation (3.3).

$$\ell_z = \mathbb{E}_{\mathbf{z} \sim q_\phi} \left[\sum_{i=1}^{n} \left\| z_i - g_i(\mathbf{S_i} \circ \mathbf{z})^2 \right\| \right] \tag{B.5}$$

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.

We notice that the Variational version CSVHTest often works better than CSHTest in the presence of noise. Results are shown in Figure B.6.

B.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.

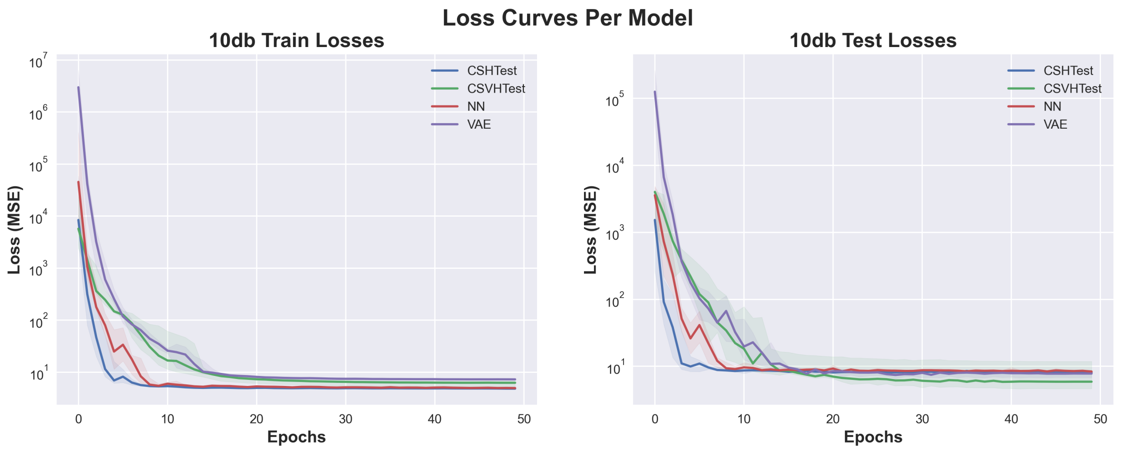


Figure B.6: Comparison of CSHTest, CSVHTest, NN, and VAE in the presence of noise

Initially optimizers Adam, SGD, Adabelief [ZTD20], and PSGD [Li19, Li18] are compared primarily for mean OOD MSE test loss across iterations in a limited number of test cases. Due to better performance, Adabelief and PSGD are compared in a more robust set of cases. Figure B.7 shows a scatter plot of the final losses across all test cases. 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 (Sturctural Hamming Distance): 0 (ground truth), 1
- Split: ID (in-dsitribution or random), OOD (out-of-dist. or 75% quantile split)
- Split Number: Which node/variable is data being split on (not relevant to ID)

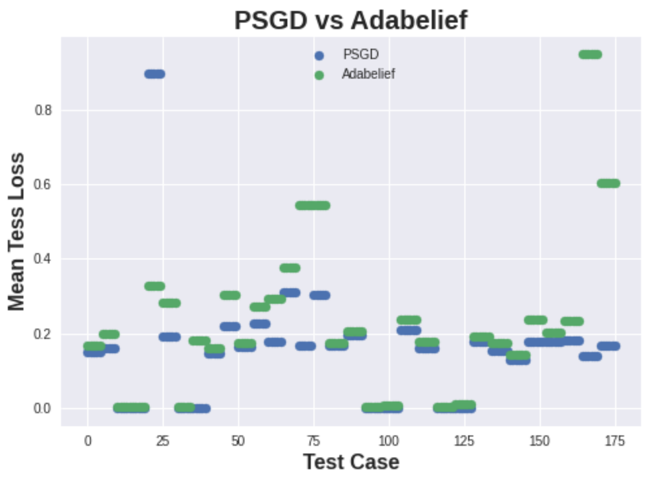


Figure B.7: Comparison of final test loss of optimizers PSGD and AdaBelief across 176 unique tests

B.5 DAG Simulation Settings

B.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 $\mathbf{N}(0, 1)$ for linear model weights
- Linear model η nets size: [4,4] MLP

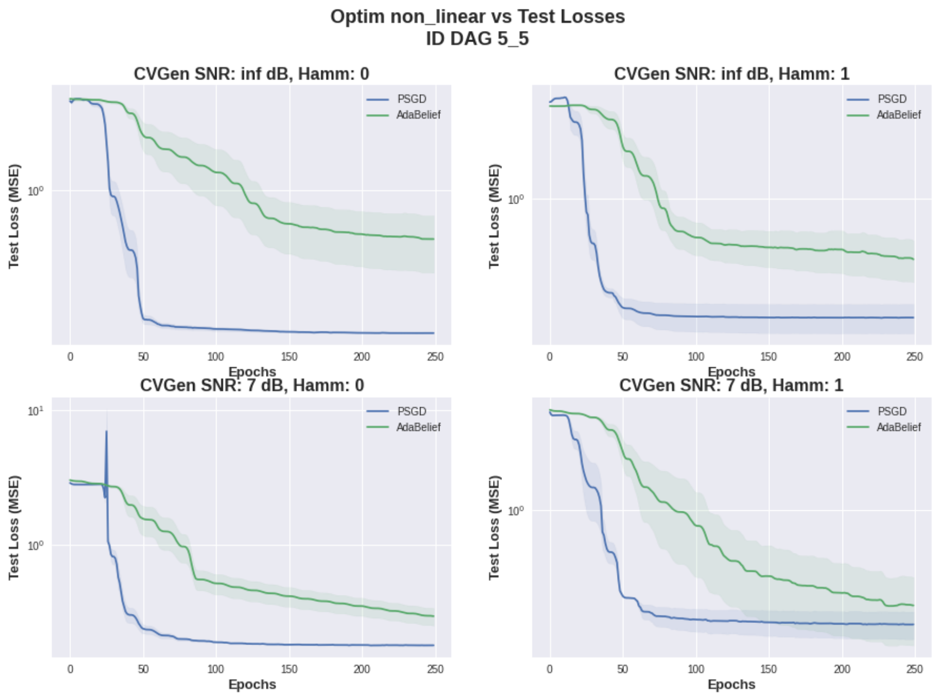


Figure B.8: Example of loss curves for optimizers PSGD and AdaBelief inf,1 dB SNR and 0,1 Hamming Distance.

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

B.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)

B.6 Final Losses with Sample Variances

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

APPENDIX C

Towards Composable Distributions of Latent Space Augmentations

C.1 Variational Autoencoders (VAEs)

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:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] - \mathrm{KL}\left(e_\phi(z|x^{(i)})||m_\theta(z)\right) \tag{C.1}$$

$$= \mathbb{E}_{e_\phi(z|x^{(i)})}\left[\log d_\theta(x^{(i)}|z)\right] - \int e_\phi(z|x^{(i)}) \log \frac{e_\phi(z|x^{(i)})}{m_\theta(z)} dz \tag{C.2}$$

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 $m_\theta(z)$.

C.2 Conditional VAE (CVAE)

Let y be the conditional input, and let x be the data that we want to generate. The encoder in the CVAE takes both y and x as input and maps them to the latent space z as seen in Figure C.1. Specifically, the encoder outputs a mean vector $\mu_\phi(z|y,x)$ and a variance vector $\sigma^2_\phi(z|y,x)$ that parameterize a Gaussian distribution $e_\phi(z|y,x)$ over the latent variables z. The decoder takes the latent variables z and the conditional input y as input and maps them to the data space. Specifically, the decoder outputs a mean vector $\mu_\theta(x|y,z)$ that parameterizes a Gaussian distribution $d_\theta(x|y,z)$ over the data variables x.

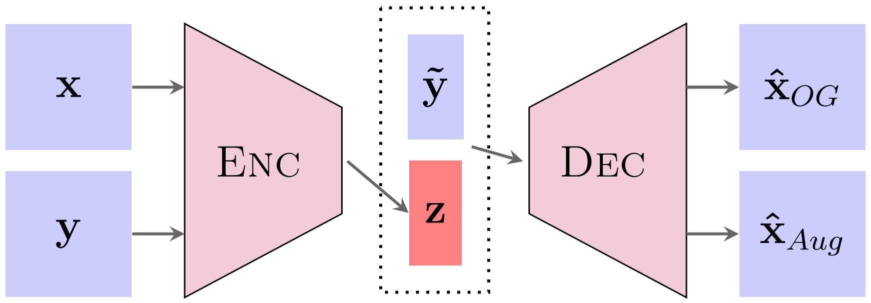


Figure C.1: Conditional VAE Architecture

The lower bound for the CVAE can be derived in a similar way to the VAE, by introducing a variational lower bound on the log likelihood. The conditional version of the lower bound is given by:

$$\mathcal{L}(\theta,\phi;x^{(i)},y^{(i)}) = \mathbb{E}e\phi(z|y^{(i)},x^{(i)}) \left[\log d_\theta(x^{(i)}|y^{(i)},z)\right] - \mathrm{KL}\left(e_\phi(z|y^{(i)},x^{(i)})||m_\theta(z|y^{(i)})\right) \tag{C.3}$$

$$= \mathbb{E}e\phi(z|y^{(i)},x^{(i)}) \left[\log d_\theta(x^{(i)}|y^{(i)},z)\right] - \int e_\phi(z|y^{(i)},x^{(i)}) \log \frac{e_\phi(z|y^{(i)},x^{(i)})}{m_\theta(z|y^{(i)})} dz \tag{C.4}$$

where $x^{(i)}$ and $y^{(i)}$ are a single training example and its corresponding condition, respectively. The first term in the lower bound, $\mathbb{E}e\phi(z|y^{(i)},x^{(i)}) \left[\log d_\theta(x^{(i)}|y^{(i)},z)\right]$, measures the reconstruction loss, i.e., the difference between the reconstructed data and the original data given the condition $y^{(i)}$. The second term, $\mathrm{KL}\left(e_\phi(z|y^{(i)}, x^{(i)})||m_\theta(z|y^{(i)})\right)$, measures the KL divergence between the approximate posterior distribution $e_\phi(z|y^{(i)}, x^{(i)})$ and the prior distribution $m_\theta(z|y^{(i)})$ over the latent variables z given the condition $y^{(i)}$.

The CVAE can be used for a variety of tasks, such as image and text generation, where the conditional input y corresponds to a class label and the data x is an image respectively. In the CVAE framework, the encoder network takes both the input data x and the conditional input y as inputs and produces the approximate posterior distribution $e_\phi(z|x, y)$ over the latent variable z. Similarly, the decoder network takes both z and y as inputs and produces the reconstructed output $d_\theta(x|z, y)$.

The objective function for the CVAE can be derived from the VAE's lower bound by conditioning on the conditional input y:

$$\mathcal{L}_{\text{CVAE}}(\theta, \phi; x^{(i)}, y^{(i)}) = \mathbb{E}e_\phi(z|x^{(i)}, y^{(i)}) \left[\log d_\theta(x^{(i)}|z, y^{(i)})\right] - \mathrm{KL}\left(e_\phi(z|x^{(i)}, y^{(i)})||m_\theta(z|y^{(i)})\right) \tag{C.5}$$

$$= \mathbb{E}e\phi(z|x^{(i)}, y^{(i)}) \left[\log d_\theta(x^{(i)}|z, y^{(i)})\right] - \int e_\phi(z|x^{(i)}, y^{(i)}) \log \frac{e_\phi(z|x^{(i)}, y^{(i)})}{m_\theta(z|y^{(i)})} dz \tag{C.6}$$

where $y^{(i)}$ is the conditional input for the i-th training example and θ and ϕ are the parameters of the decoder and encoder, respectively. The first term in the lower bound, $\mathbb{E}e\phi(z|x^{(i)}, y^{(i)}) \left[\log d_\theta(x^{(i)}|z, y^{(i)})\right]$, measures the difference between the reconstructed output and the original output, given the input and the conditional input. The second term, $\mathrm{KL}\left(e_\phi(z|x^{(i)}, y^{(i)})||m_\theta(z|y^{(i)})\right)$, measures the difference between the approximate posterior distribution and the prior distribution of the latent variable, given the conditional input.

C.3 Architecture and Training

- Model: (Total Parameter Count:) Encoder/Decoder: (Parameter Count:)
  - 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 16×16 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

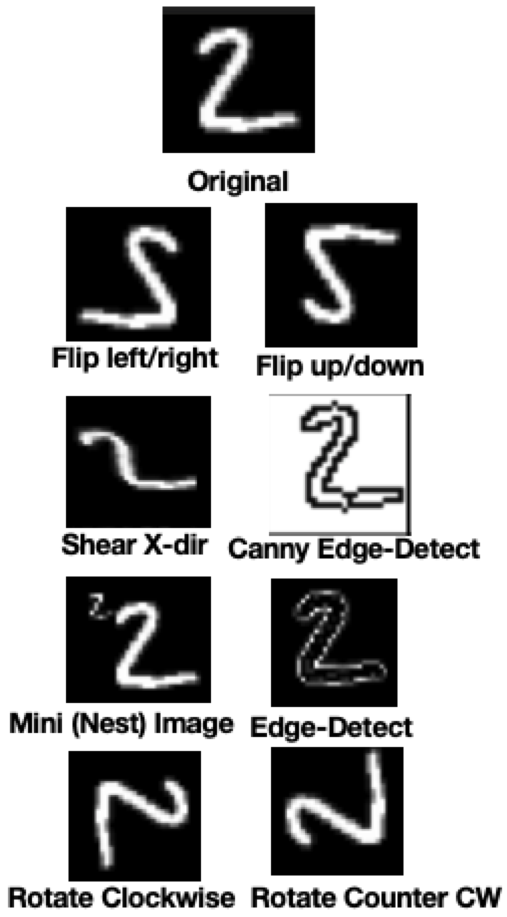


Figure C.2: All Augmentations Visualized

C.4 Sampling and Interpolation

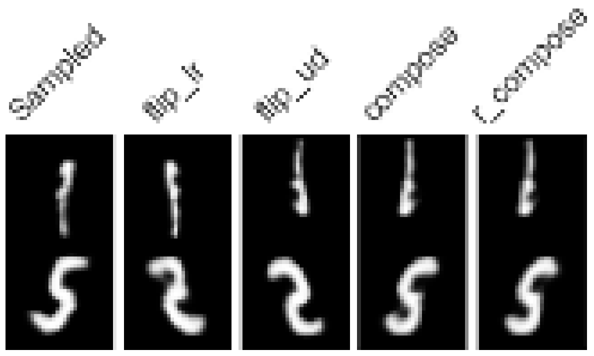


Figure C.3: Sampled digits and augmentations via bounding box method

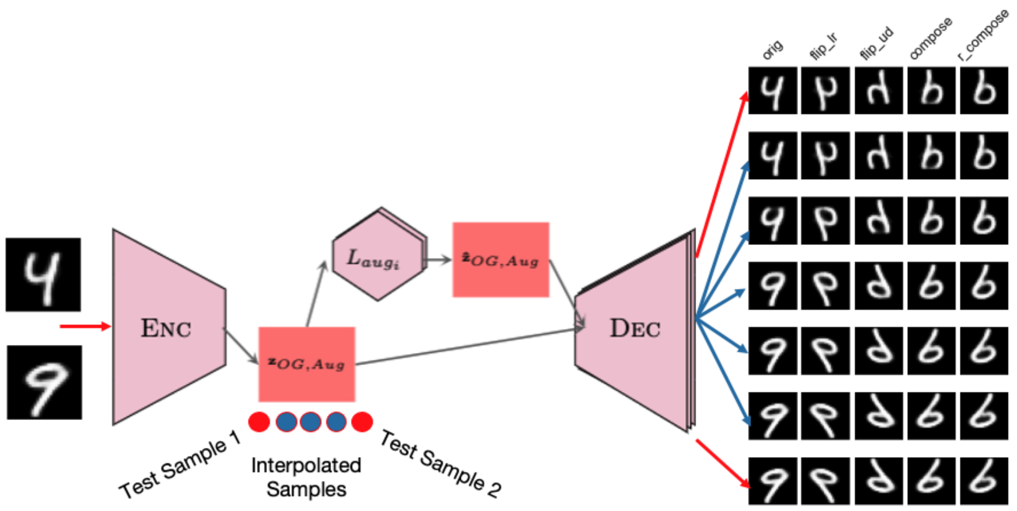


Figure C.4: Interpolating between two test samples (top and bottom row) with augmentations

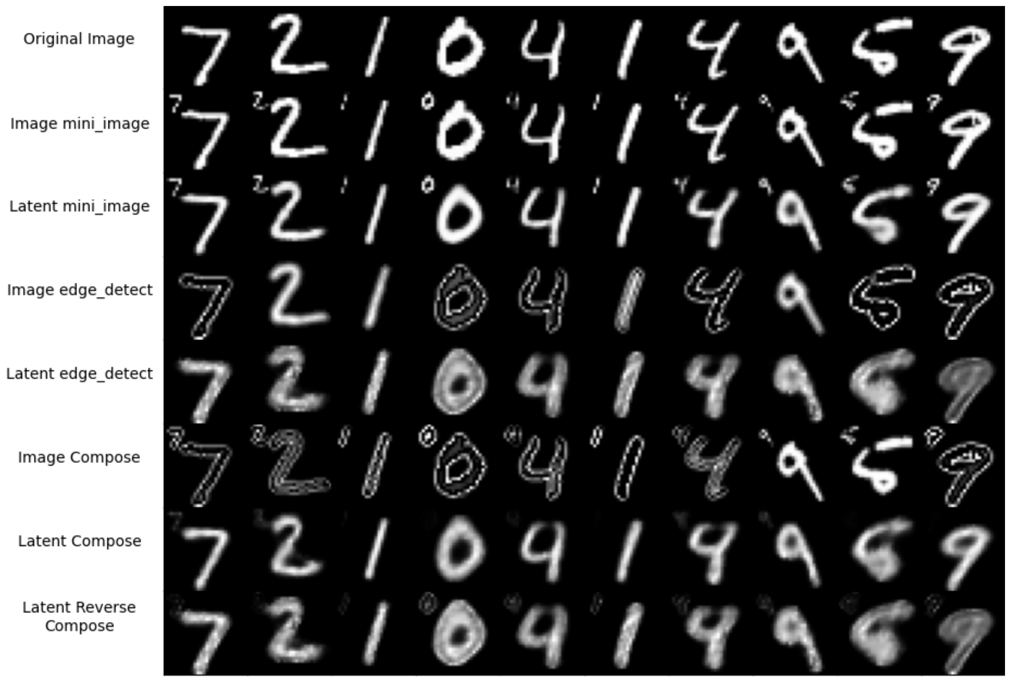


Figure C.5: “Nested Mini-Image, Edge-Detect” Reconstructions

C.5 Additional Augmentation Reconstructions

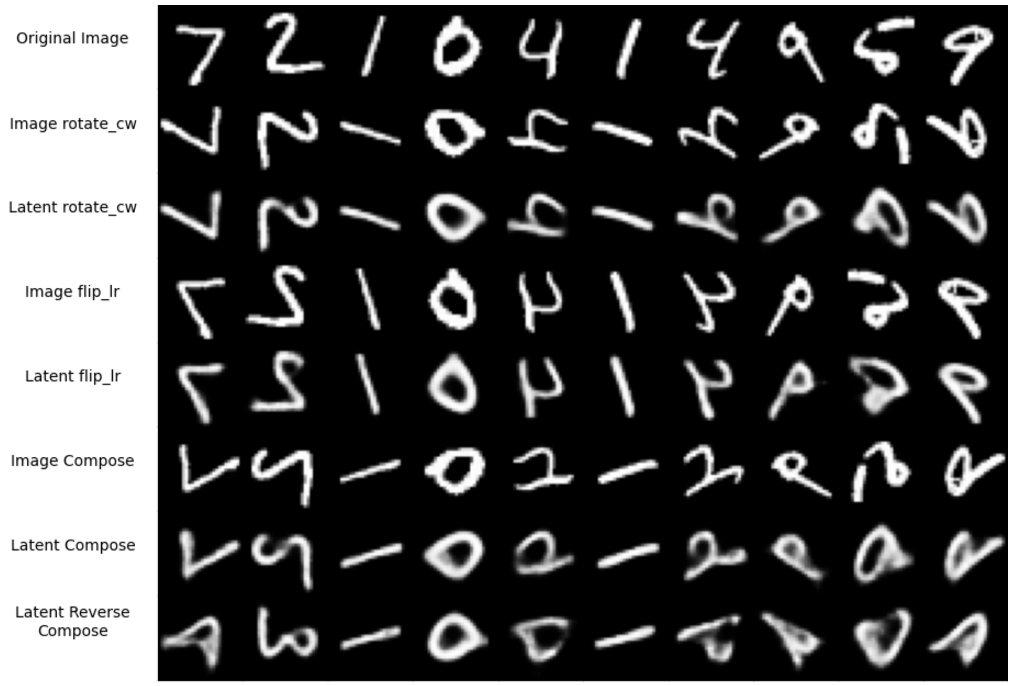


Figure C.6: “90 deg Clockwise Rotation, Flip left/right” Reconstructions

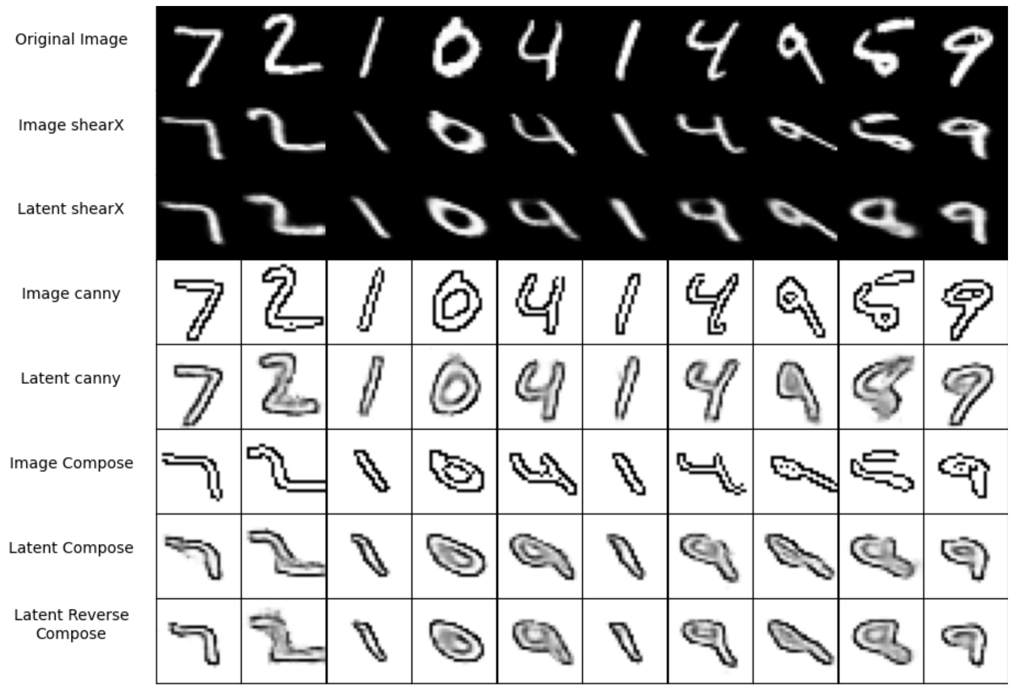
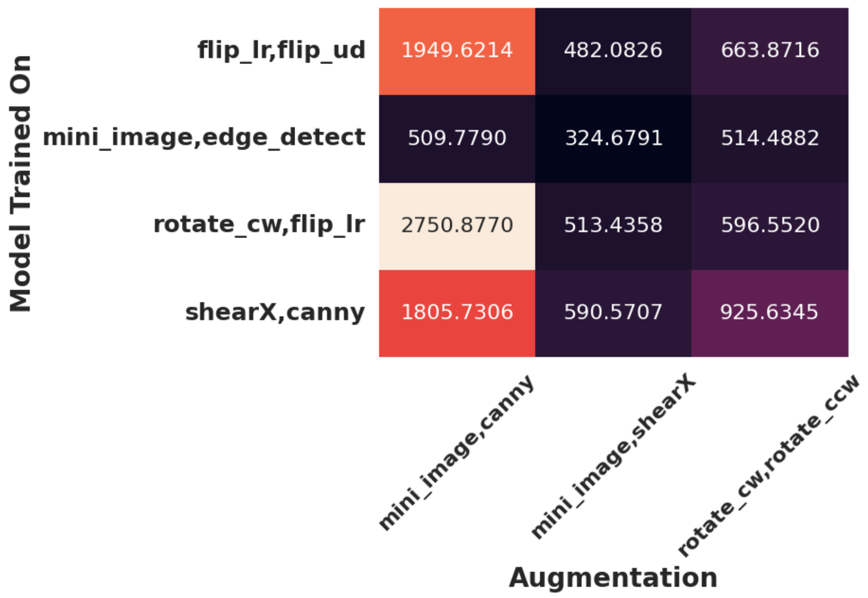


Figure C.7: “X-direction shear, Canny edge-detect” Reconstructions

C.6 Additional Transfer Decoder Results



	mini_image,canny	mini_image,shearX	rotate_cw,rotate_ccw
flip_lr,flip_ud	1949.6214	482.0826	663.8716
mini_image,edge_detect	509.7790	324.6791	514.4882
rotate_cw,flip_lr	2750.8770	513.4358	596.5520
shearX,canny	1805.7306	590.5707	925.6345

Figure C.8: Additional transfer decoder head results

C.7 Latent Space Geometries

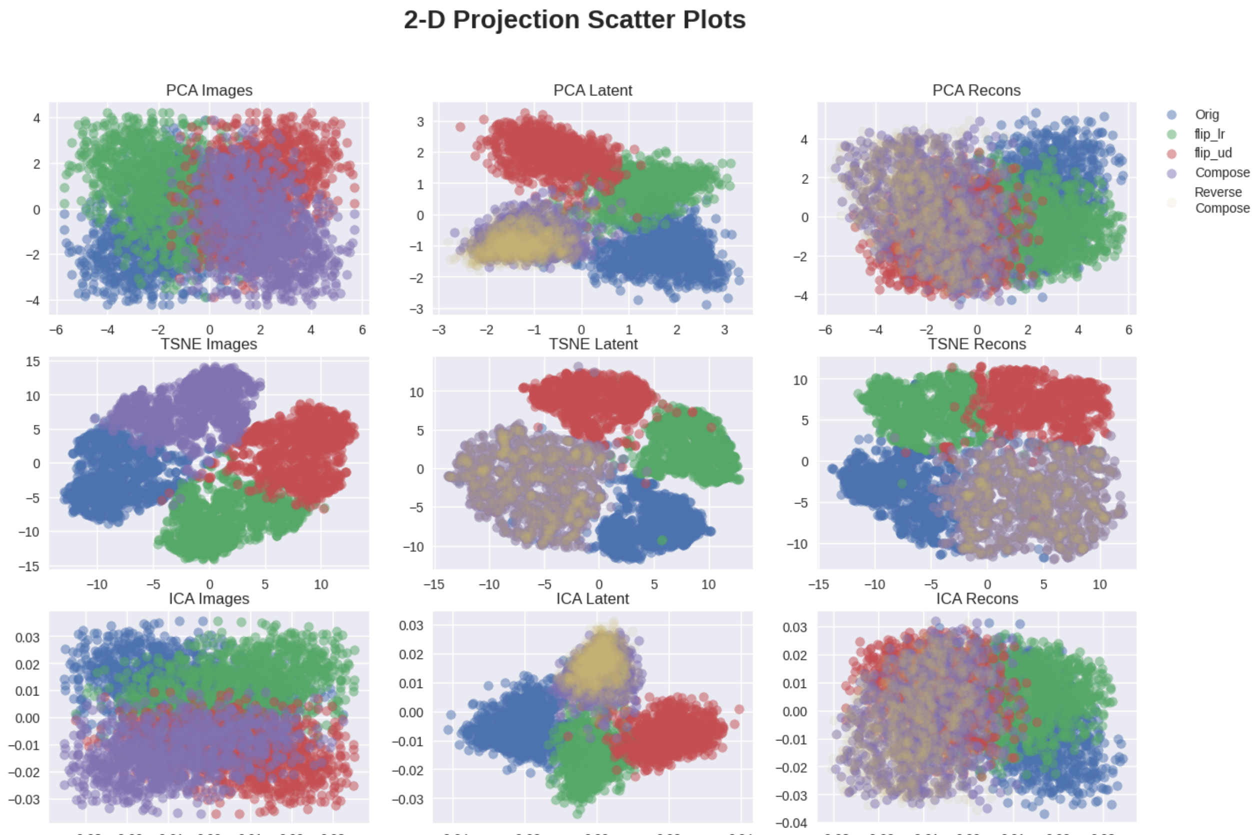


Figure C.9: “Flips” Image, Latent, and Reconstructions Image 2-D Projections

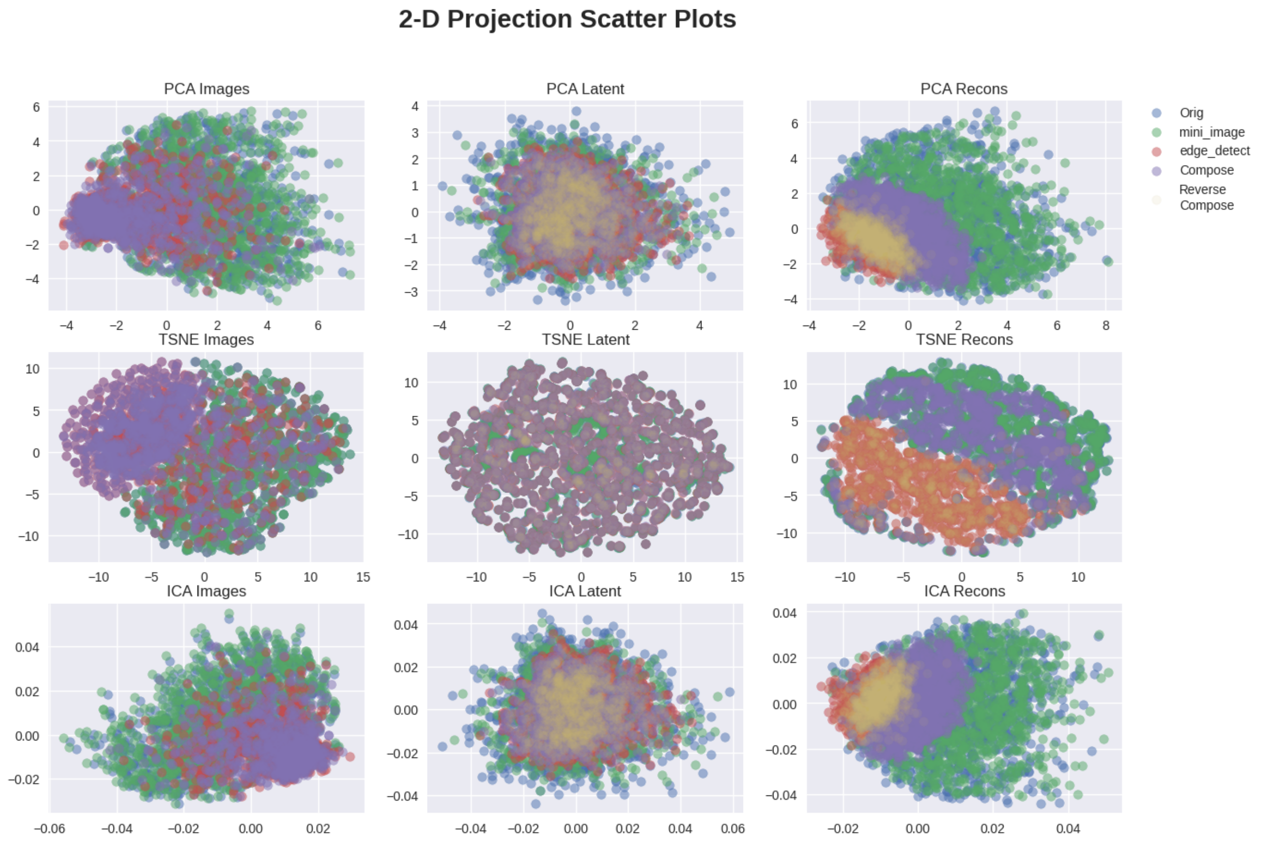


Figure C.10: "Nested Mini-Image, Edge-Detect" Image, Latent, and Reconstructions Image 2-D Projections

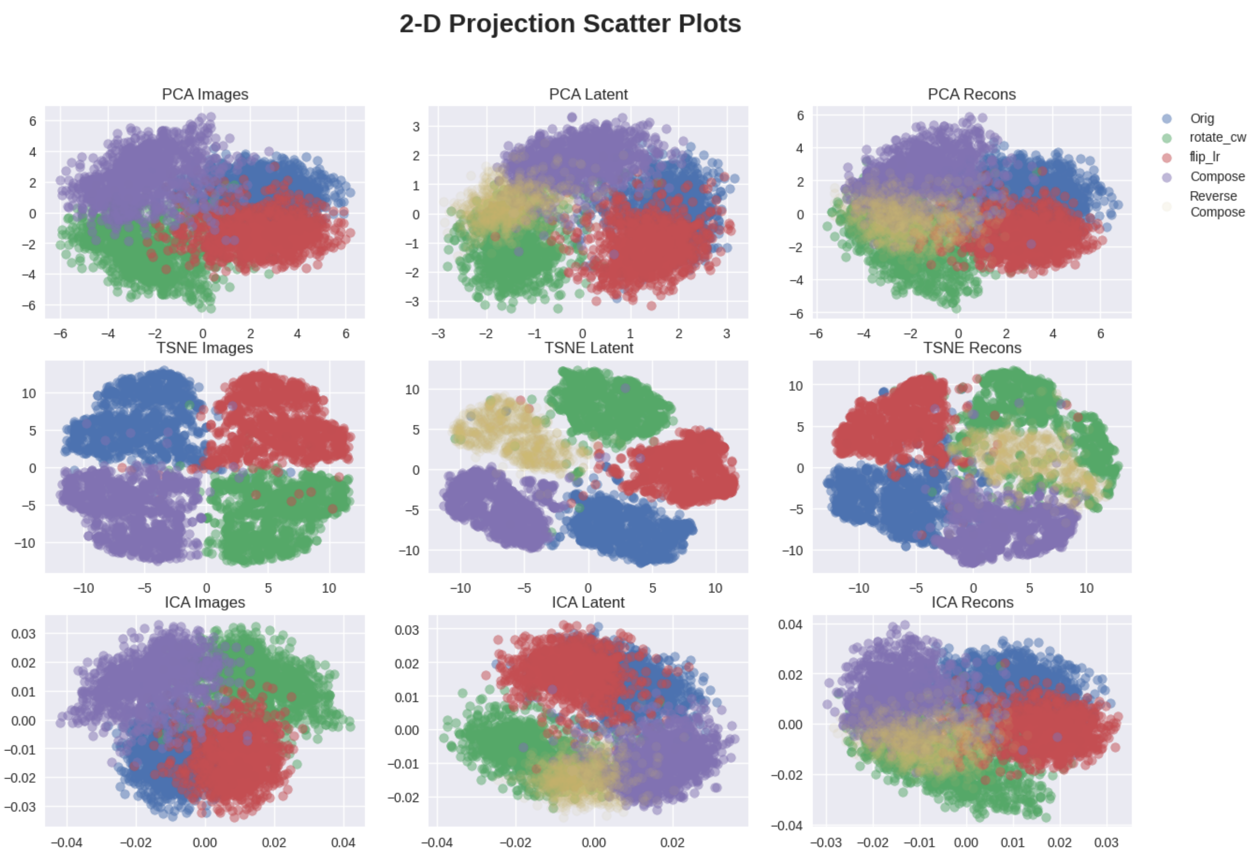


Figure C.11: “90° Clockwise Rotation, Flip left/right” Image, Latent, and Reconstructions Image 2-D Projections

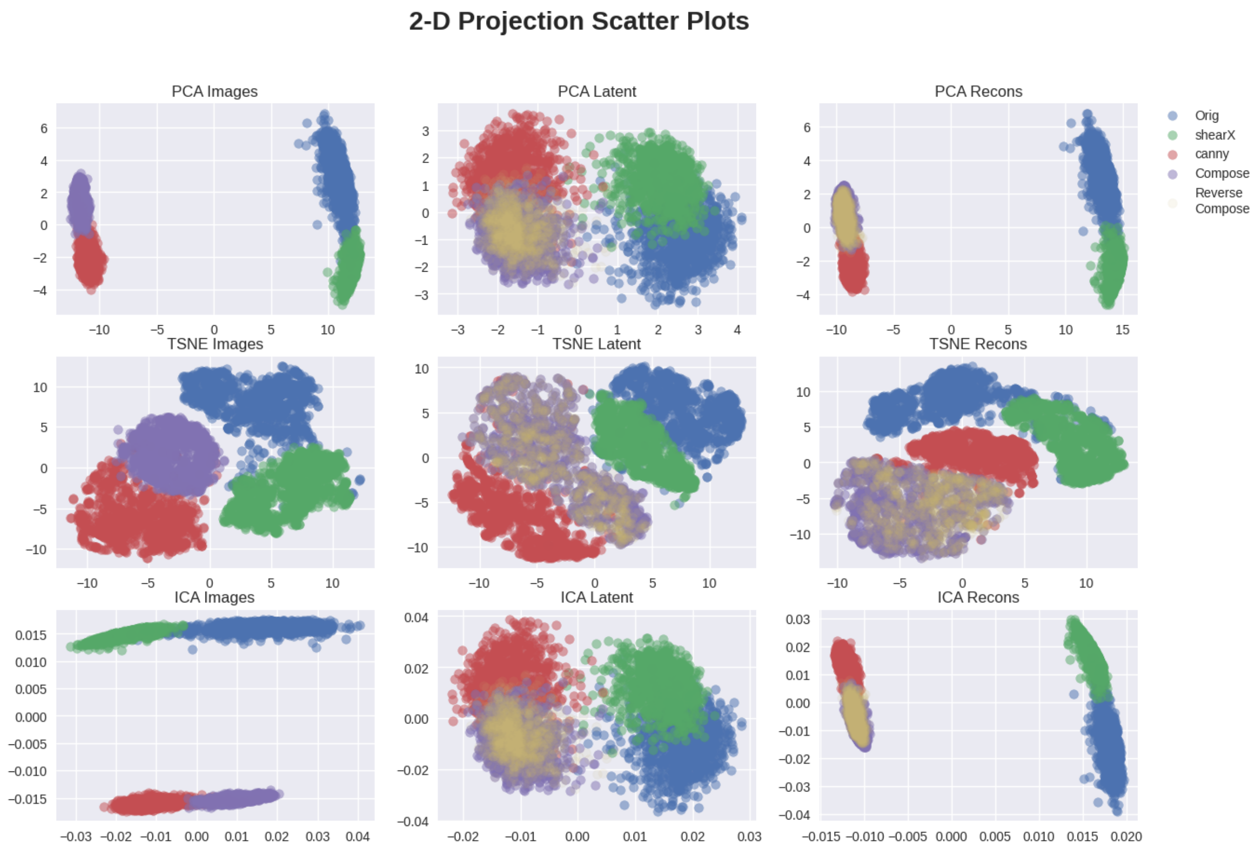


Figure C.12: "X-direction shear, Canny edge-detect" Image, Latent, and Reconstructions Image 2-D Projections

APPENDIX D

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

D.1 EBM Further Background

D.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:

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

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:

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

where $\delta X_{\eta_{noise}}(t)$ represents an infinitesimal perturbation in the system state at time t, evolved according to Eq. D.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:

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

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 D.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,

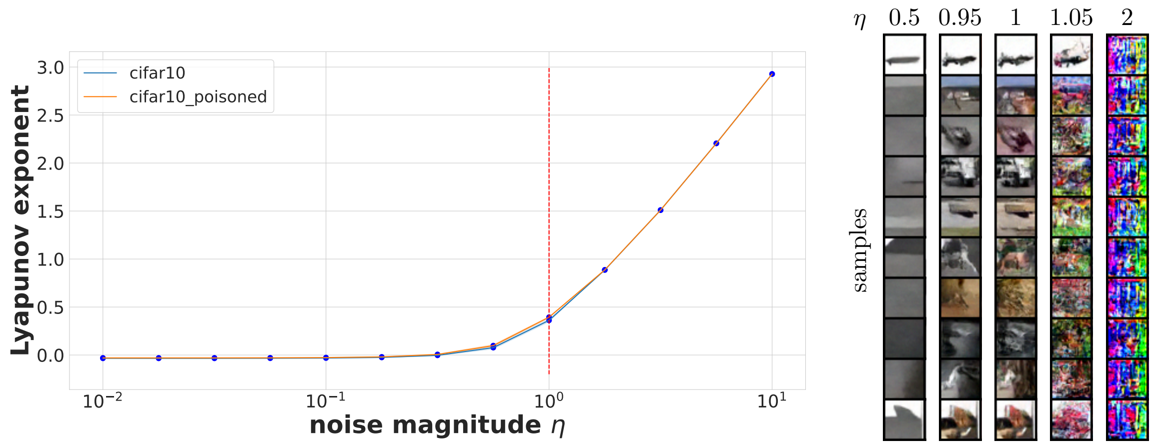


Figure D.1: *Left:* The maximal Lyapunov exponent varies significantly with different values of the noise parameter η_{noise}. Notably, at $\eta_{noise} = 1$, which is the setting used in our training and defense dynamics, there is a critical transition observed. This transition is from an ordered region, 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.

D.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

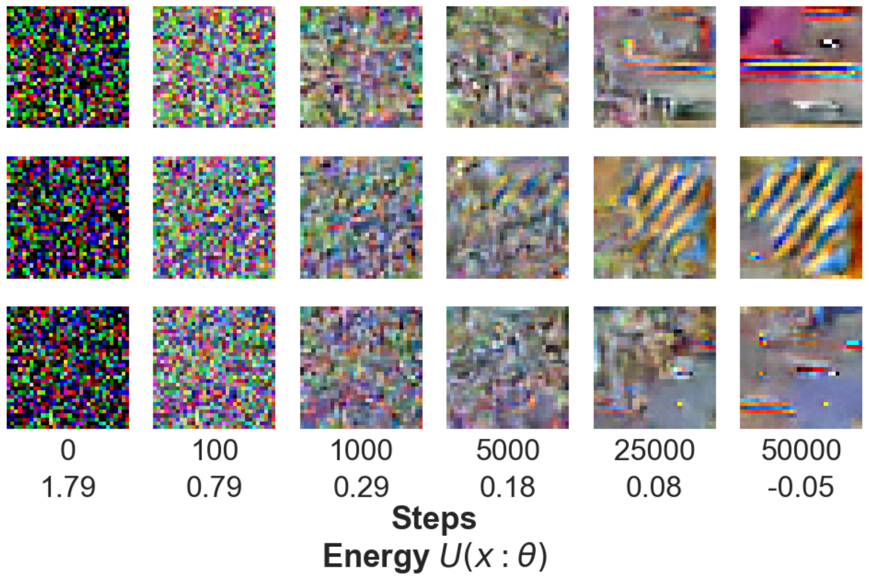


Figure D.2: Random Noise initialization of purification process

samples are generated from a random initialization using a conditional or unconditional 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 D.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.

D.2 Additional Results

D.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 (bernouilli or gaussian added noise trasnform), and all natural PureEBM versions. Asterisk (*) indicates a baseline defense that was selected for the main paper results table due to best poison defense performance.

We note that the implemention 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 $_{reported}$.

	From Scratch									
	200 - Epochs					**80 - Epochs**				
	Gradient Matching-1%		**Narcissus-1%**			**Gradient Matching-1%**		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	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	47.00	93.79$_{0.2}$	32.51$_{30.3}$	93.76$_{0.2}$	79.43
EPIc-0.1*	34.00	91.27$_{0.4}$	30.18$_{32.2}$	91.17$_{0.2}$	81.50	27.00	90.87$_{0.4}$	24.15$_{30.1}$	90.92$_{0.4}$	79.42
EPIc-0.2	21.00	88.04$_{0.7}$	32.50$_{33.5}$	86.89$_{0.5}$	84.39	28.00	91.02$_{0.4}$	23.75$_{29.2}$	89.72$_{0.3}$	74.28
EPIc-0.3*	10.00	85.14$_{1.2}$	27.31$_{34.0}$	82.20$_{1.1}$	84.71	44.00	92.46$_{0.3}$	21.53$_{28.8}$	88.05$_{1.1}$	80.75
EPIc $_{reported}$	1.00	90.26	NA	NA	NA	NA	NA	NA	NA	NA
FrieNDs-B	1.00	**91.16**$_{0.4}$	8.32$_{22.3}$	91.01$_{0.4}$	71.76	2.00	90.07$_{0.4}$	**1.42**$_{0.8}$	90.06$_{0.3}$	2.77
FrieNDs-G*	**0.00**	**91.15**$_{0.4}$	9.49$_{25.9}$	91.06$_{0.2}$	83.03	**1.00**	90.09$_{0.4}$	**1.37**$_{0.9}$	90.01$_{0.2}$	3.18
PureEBM	**0.00**	**92.26**$_{0.2}$	**1.27**$_{0.6}$	**92.91**$_{0.2}$	**2.16**	**1.00**	**91.36**$_{0.3}$	**1.46**$_{0.8}$	**91.83**$_{0.3}$	**2.49**
PureEBM-P	NA	NA	**1.38**$_{0.7}$	**92.70**$_{0.2}$	**2.78**	NA	NA	**1.63**$_{1.0}$	**91.49**$_{0.3}$	3.47
PureEBM $_{CN-10}$	**0.00**	**92.99**$_{0.2}$	**1.43**$_{0.8}$	**92.90**$_{0.2}$	**3.06**	**1.00**	**92.02**$_{0.2}$	**1.50**$_{0.9}$	**92.03**$_{0.2}$	**2.52**
PureEBM $_{IN}$	1.00	**92.98**$_{0.2}$	**1.39**$_{0.8}$	**92.92**$_{0.2}$	**2.50**	**1.00**	**92.02**$_{0.2}$	**1.52**$_{0.8}$	**92.02**$_{0.3}$	**2.81**
PureEBM-P$_{CN-10}$	NA	NA	**1.64**$_{0.01}$	**92.86**$_{0.20}$	**4.34**	NA	NA	**1.68**$_{1.0}$	**92.07**$_{0.2}$	3.34

	Transfer Learning								
	Fine-Tune					**Linear - Bullseye Polytope**			
	Bullseye Polytope-10%		**Narcissus-10%**			**BlackBox-10%**		**WhiteBox-1%**	
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑
None	46.00	89.84$_{0.9}$	33.41$_{33.9}$	90.14$_{2.4}$	98.27	93.75	83.59$_{2.4}$	98.00	70.09$_{0.2}$
EPIc-0.1	50.00	89.00$_{1.8}$	32.40$_{33.7}$	90.02$_{2.2}$	98.95	91.67	83.48$_{2.9}$	98.00	69.35$_{0.3}$
EPIc-0.2*	42.00	81.95$_{5.6}$	20.93$_{27.1}$	88.58$_{2.0}$	91.72	66.67	84.34$_{3.8}$	91.00	64.79$_{0.7}$
EPIc-0.3	44.00	86.75$_{6.3}$	28.01$_{34.9}$	84.36$_{6.3}$	99.91	66.67	83.23$_{3.8}$	63.00	60.86$_{1.5}$
FrieNDs-B	8.00	87.80$_{1.1}$	3.34$_{5.7}$	89.62$_{0.5}$	19.48	35.42	84.97$_{2.2}$	19.00	60.85$_{0.6}$
FrieNDs-G*	8.00	87.82$_{1.2}$	3.04$_{5.1}$	89.81$_{0.5}$	17.32	33.33	85.18$_{2.3}$	19.00	60.90$_{0.6}$
PureEBM	**0.00**	**88.95**$_{1.1}$	**1.98**$_{1.7}$	**91.40**$_{0.4}$	**5.98**	**0.00**	**92.89**$_{0.2}$	**6.00**	**64.51**$_{0.6}$
PureEBM-P	NA	NA	**3.66**$_{4.63}$	**90.89**$_{0.31}$	16.04	NA	NA	NA	NA
PureEBM $_{CN-10}$	**0.00**	**88.67**$_{1.2}$	**2.97**$_{2.5}$	**90.99**$_{0.3}$	**7.95**	**0.00**	**92.82**$_{0.1}$	**6.00**	**64.44**$_{0.4}$
PureEBM $_{IN}$	**0.00**	**87.52**$_{1.2}$	**2.02**$_{1.0}$	**89.78**$_{0.6}$	**3.85**	**0.00**	**92.38**$_{0.3}$	**6.00**	**64.98**$_{0.3}$

D.2.2 Extended Poison% Results

Table D.1: Narcissus transfer fine-tune results at various poison%'s

Poison-%	**1%**			**2.5%**			**10%**		
	Avg Poison Success (%) ↓	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**	**Avg Poison Success (%) ↓**	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**	**Avg Poison Success (%) ↓**	**Avg Natural Accuracy (%) ↑**	**Max Poison Success (%) ↓**
None	17.06$_{27.0}$	93.18$_{0.1}$	81.97	22.22$_{30.1}$	93.35$_{0.1}$	89.74	33.41$_{33.9}$	90.14$_{2.4}$	98.27
EPIc-0.1	15.58$_{25.5}$	92.75$_{0.2}$	73.65	19.77$_{27.5}$	92.72$_{0.3}$	87.51	32.40$_{33.7}$	90.02$_{2.2}$	98.95
EPIc-0.2	12.33$_{23.8}$	85.86$_{2.9}$	74.32	24.26$_{31.2}$	85.59$_{3.3}$	96.07	20.93$_{27.1}$	88.58$_{2.0}$	91.72
EPIc-0.3	12.74$_{21.2}$	91.37$_{4.0}$	67.45	12.32$_{18.7}$	92.24$_{0.4}$	61.33	28.01$_{34.9}$	84.36$_{6.3}$	99.91
FrieNDs-B	**1.44**$_{0.8}$	90.61$_{0.2}$	**2.49**	2.25$_{3.3}$	90.44$_{0.3}$	11.46	3.34$_{5.7}$	89.62$_{0.5}$	19.48
FrieNDs-G	**1.34**$_{0.7}$	90.50$_{0.2}$	**2.50**	2.43$_{3.6}$	90.51$_{0.2}$	12.61	3.04$_{5.1}$	89.81$_{0.5}$	17.32
PureEBM	**1.50**$_{1.4}$	**91.65**$_{0.1}$	5.19	**1.60**$_{1.2}$	**91.27**$_{0.1}$	**4.76**	**1.98**$_{1.7}$	**91.40**$_{0.4}$	**5.98**
PureEBM-P	4.50$_{7.4}$	89.61$_{0.3}$	24.43	7.93$_{12.4}$	90.26$_{0.2}$	39.59	**3.66**$_{4.63}$	**90.89**$_{0.31}$	16.04
PureEBM $_{CN-10}$	**1.77**$_{1.2}$	**91.56**$_{0.1}$	4.07	**2.21**$_{1.6}$	**91.45**$_{0.1}$	**5.02**	**2.97**$_{2.5}$	**90.99**$_{0.3}$	**7.95**
PureEBM $_{IN}$	**1.62**$_{0.9}$	90.91$_{0.1}$	3.35	**1.85**$_{0.9}$	90.85$_{0.2}$	**3.39**	**2.02**$_{1.0}$	89.78$_{0.6}$	**3.85**
PureEBM-P$_{CN-10}$	4.33$_{6.2}$	90.99$_{0.2}$	21.25	5.95$_{8.5}$	90.80$_{0.2}$	28.88	11.84$_{19.9}$	88.77$_{1.3}$	66.63

Table D.2: BP transfer linear gray-box results at various poison%'s

Poison-%	**1%**		**2%**		**5%**		**10%**	
	Poison Success (%) ↓	**Natural Accuracy (%) ↑**	**Poison Success (%) ↓**	**Natural Accuracy (%) ↑**	**Poison Success (%) ↓**	**Natural Accuracy (%) ↑**	**Poison Success (%) ↓**	**Natural Accuracy (%) ↑**
None	26.00	93.60$_{0.2}$	32.00	93.60$_{0.2}$	66.00	92.89$_{0.4}$	93.75	83.59$_{2.4}$
EPIc-0.1	12.00	93.34$_{0.4}$	50.00	92.79$_{0.6}$	70.00	92.43$_{0.8}$	91.67	83.48$_{2.9}$
EPIc-0.2	18.00	92.53$_{1.4}$	34.00	92.86$_{1.4}$	76.00	91.72$_{2.0}$	66.67	84.34$_{3.8}$
EPIc-0.3	18.00	92.80$_{0.9}$	24.00	92.89$_{1.0}$	62.00	90.95$_{2.7}$	66.67	83.23$_{3.8}$
FrieNDs-B	4.00	**94.09**$_{0.1}$	4.00	**94.11**$_{0.1}$	26.00	**93.72**$_{0.2}$	35.42	84.97$_{2.2}$
FrieNDs-G	4.00	**94.12**$_{0.1}$	4.00	**94.13**$_{0.1}$	22.00	**93.73**$_{0.2}$	33.33	85.18$_{2.3}$
PureEBM	**0.00**	93.18$_{0.0}$	**0.00**	92.94$_{0.1}$	**0.00**	92.92$_{0.1}$	**0.00**	**92.89**$_{0.2}$
PureEBM $_{CN-10}$	**0.00**	93.14$_{0.1}$	**0.00**	92.61$_{0.1}$	**0.00**	93.00$_{0.1}$	**0.00**	**92.82**$_{0.1}$
PureEBM $_{IN}$	**0.00**	92.09$_{0.1}$	**0.00**	91.51$_{0.1}$	**0.00**	92.75$_{0.1}$	**0.00**	**92.38**$_{0.3}$

D.2.3 Full MobileNetV2 and DenseNet121 Results

Table D.3: MobileNetV2 Full Results

From Scratch - MobileNetV2										
	200 - Epochs					**80 - Epochs**				
	Gradient Matching-1%		**Narcissus-1%**			**Gradient Matching-1%**		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Success (%) ↓	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	30.00	$92.54_{0.2}$	$27.26_{26.5}$	$92.53_{0.2}$	74.82
EPIc-0.1	37.50	$91.28_{0.2}$	$40.09_{27.1}$	$91.15_{0.2}$	79.74	16.00	$90.45_{0.3}$	$31.37_{30.9}$	$90.51_{0.3}$	89.36
EPIc-0.2	19.00	$91.24_{0.2}$	$38.55_{27.5}$	$87.65_{0.5}$	74.72	22.00	$89.90_{0.3}$	$29.22_{27.6}$	$89.91_{0.3}$	76.54
EPIc-0.3	9.78	$87.80_{1.6}$	$22.35_{23.9}$	$78.16_{9.9}$	69.52	14.00	$90.23_{0.3}$	$30.69_{30.6}$	$90.30_{0.3}$	82.92
FrieNDs-B	6.00	$84.30_{2.7}$	$\mathbf{2.00}_{1.3}$	$88.82_{0.6}$	4.88	**1.00**	$87.89_{0.3}$	$\mathbf{1.98}_{1.1}$	$87.90_{0.4}$	4.00
FrieNDs-G	5.00	$88.84_{0.4}$	$\mathbf{2.05}_{1.7}$	$88.93_{0.3}$	6.33	3.00	$87.90_{0.4}$	$\mathbf{2.00}_{1.4}$	$88.09_{0.3}$	5.07
PureEBM	**1.00**	$\mathbf{90.93}_{0.2}$	$\mathbf{1.64}_{0.8}$	$\mathbf{91.75}_{0.1}$	**2.91**	**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**	$\mathbf{78.57}_{1.4}$

Table D.4: DenseNet121 Full Results

From Scratch - DenseNet121										
	200 - Epochs					**80 - Epochs**				
	Gradient Matching-1%		**Narcissus-1%**			**Gradient Matching-1%**		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Succes (%) ↓	Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Avg Poison Success (%) ↓	Avg Natural Accuracy (%) ↑	Max Poison Succes (%) ↓
None	14.00	$95.30_{0.1}$	46.52±32.2	$95.33_{0.1}$	91.96	19.00	$94.38_{0.2}$	$38.01_{36.3}$	$94.49_{0.1}$	89.11
EPIc-0.1	14.00	$93.0_{0.3}$	43.38±32.0	$93.07_{0.2}$	88.97	16.00	$92.78_{0.3}$	$32.85_{33.0}$	$92.87_{0.3}$	79.42
EPIc-0.2	7.00	$90.67_{0.5}$	41.97±33.2	$90.23_{0.6}$	86.85	13.00	$92.69_{0.3}$	$30.67_{28.1}$	$92.82_{0.2}$	65.46
EPIc-0.3	4.00	$88.3_{1.0}$	32.60±29.4	$85.12_{2.4}$	71.50	15.00	$93.35_{0.2}$	$36.80_{36.0}$	$93.34_{0.2}$	90.41
FrieNDs-B	1.00	**91.33**$_{0.4}$	8.60±21.2	$91.55_{0.3}$	68.57	**1.00**	$89.93_{0.4}$	$5.60_{11.6}$	$90.01_{0.4}$	38.08
FrieNDs-G	1.00	**91.33**$_{0.4}$	10.13±25.2	$91.32_{0.4}$	81.47	**1.00**	$89.97_{0.4}$	$7.59_{18.7}$	$89.89_{0.4}$	60.68
PureEBM	**0.00**	**92.85**$_{0.2}$	**1.42±0.7**	**93.48**$_{0.1}$	**2.60**	2.00	**91.88**$_{0.3}$	**1.59**$_{0.9}$	**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	**87.06**$_{1.0}$	$5.34_{9.9}$	**88.62**$_{0.8}$	33.42	60.42	$80.22_{1.9}$
FrieNDs-G	2.00	**87.37**$_{0.9}$	$5.55_{10.4}$	**88.75**$_{0.6}$	34.91	56.25	$80.12_{1.8}$
PureEBM	**0.00**	$84.39_{1.0}$	**2.48**$_{1.9}$	**88.75**$_{0.5}$	**7.41**	**0.00**	**89.29**$_{0.9}$

D.2.4 Full CINIC-10 Results

Table D.5: CINIC-10 Full Results

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**	$\mathbf{91.98}_{0.16}$
	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**	$\mathbf{91.33}_{0.20}$

D.3 Further Experimental Details

D.3.1 EBM Training

```
Algorithm 2 ML with SGD for Convergent Learning of EBM (2.13)
Require: ConvNet potential $\mathcal{G}_\theta(x)$, number of training steps $J = 150000$, initial weight $\theta_1$,
    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 $\theta_{J+1}$ for energy $\mathcal{G}_\theta(x)$.

    Set optimizer $g \leftarrow \text{SGD}(\gamma_{\text{SGD}})$. Initialize persistent image bank as $N_{\text{data}}$ 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
```

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.

D.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:

$$PSR_{notr}(F,i) = \mathbb{1}_{F(x_i^{\pi})=y_i^{\text{adv}}} \tag{D.4}$$

$$PSR_{tr}(F,y^{\pi}) = \frac{\sum_{(x,y)\in\mathcal{D}_{test}\setminus\mathcal{D}_{test}^{\pi}} \mathbb{1}_{F(x+\rho^{\pi})=y^{\pi}}}{|\mathcal{D}_{test}\setminus\mathcal{D}_{test}^{\pi}|} \tag{D.5}$$

D.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-box scenario targeting ResNet18 and DenseNet121 architectures, and grey-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 samples with strong poison success, but the undefended natural accuracy baseline is much lower.

D.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 datapoint of whether the target was correctly moved into the poisoned target class and the attack success rate is across all 100 datasets provided.

D.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).

D.3.2.4 Neural Tangent Availability Attacks

For Neural Tangent Availability Attack, the full NTGA dataset (all samples poisoned) is sourced from the authors of the original NTGA attack paper [YW21]. Baseline defenses are pull from AVATAR [DEL24].

D.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 and we only substituted in a poisoned dataloader [Bal23].

Parameter	Shared	From Scratch	Transfer Linear	Transfer Fine-Tune
Device Type	TPU-V3	-	-	-
Weight Decay	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

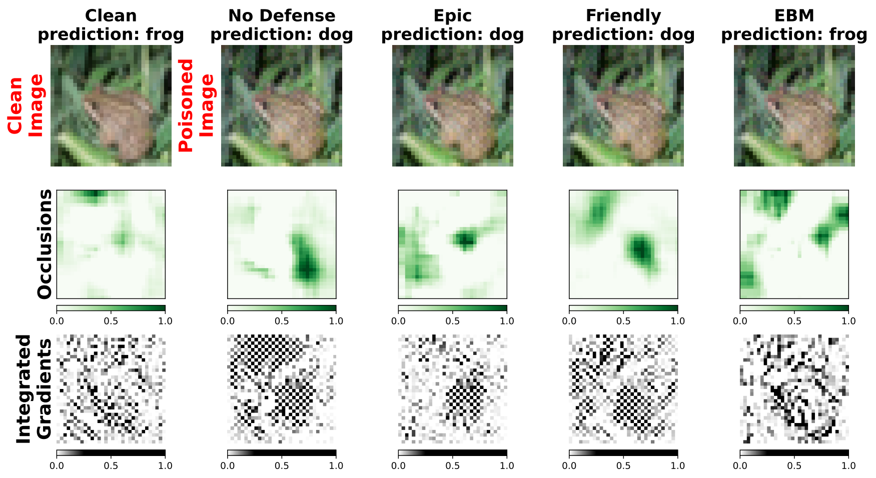
D.4 Timing Analysis

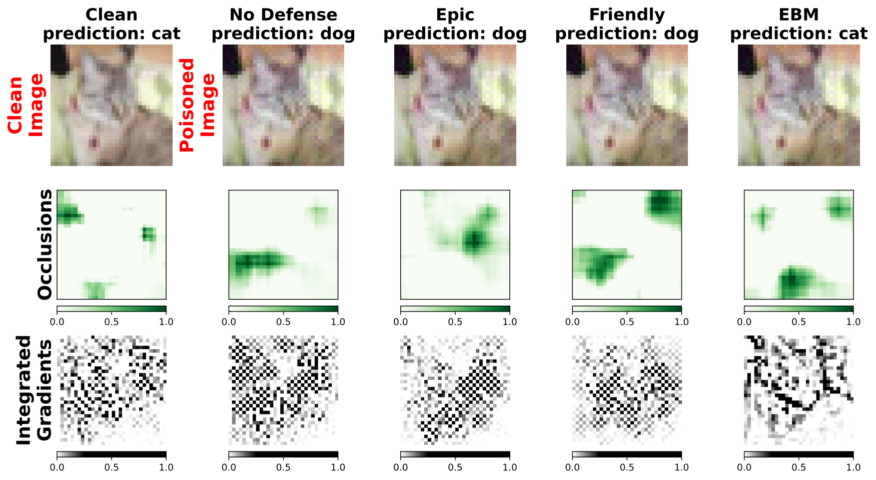
Table D.6 shows the training times for each poison defense in the from-scratch scenario on a TPU-V3. As PureEBM is a preprocessing step, the purification time (∼400 seconds) is shared across poison scenarios, making it increasingly comparable to no defense as the number of models/scenarios increase. Although EBM training is a compute intensive process, noted in detail in App. D.3.1, we share results in the section Table 6.1 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 offers superior performance and flexibility to the classifier training pipeline.

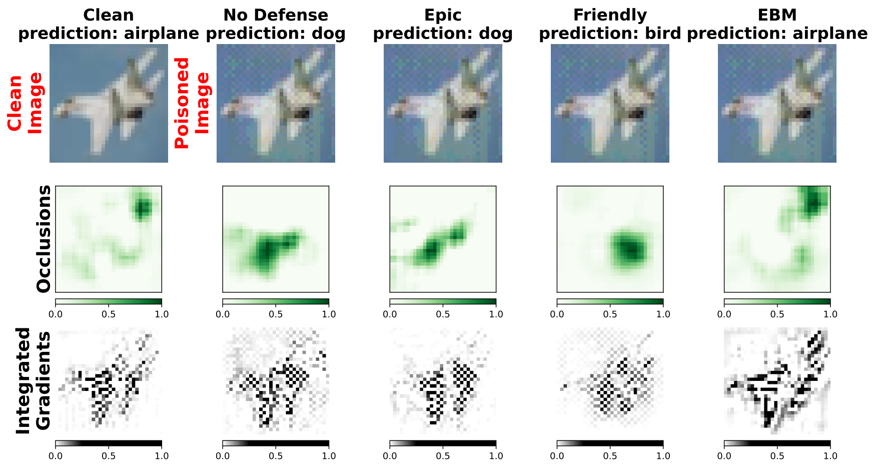
Table D.6: Median Wall Clock Train Times From Scratch

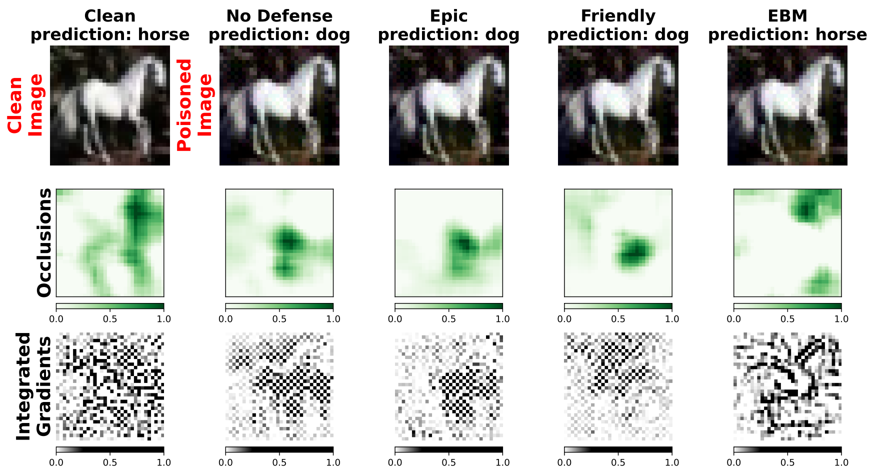
Train Time (seconds)				
	Gradient Matching		**Narcissus**	
epochs	**80**	**200**	**80**	**200**
None	$\mathbf{2202}_{16}$	5482_{49}	$\mathbf{2936}_{94}$	7154_{194}
EPIc	$\mathbf{2256}_{97}$	$\mathbf{5006}_{253}$	3564_{213}	$\mathbf{6359}_{462}$
FrieNDs	7740_{394}	11254_{413}	8728_{660}	12868_{573}
PureEBM	$\mathbf{2213}_{36}$	5520_{47}	$\mathbf{2962}_{92}$	7293_{219}

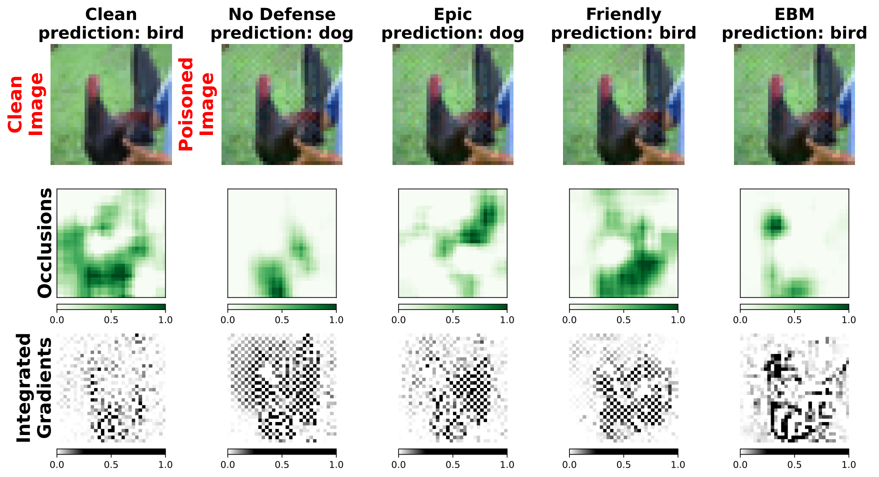
D.5 Additional Model Interpretability Results

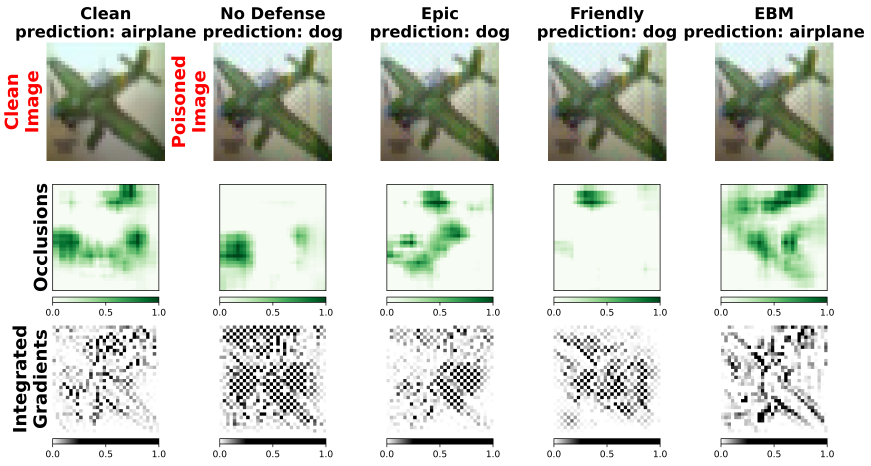







Clean
prediction: horse
No Defense
prediction: dog
Epic
prediction: dog
Friendly
prediction: dog
EBM
prediction: horse
Clean Image
Poisoned Image
Occlusions
Integrated Gradients
0.0 0.5 1.0


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

D.5.1 Poisoned Parameters Diverge

[YLM22] proposes a subset selection method EPIC which rejects poison points through training. This defense method 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 D.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 sub-optimal defense.

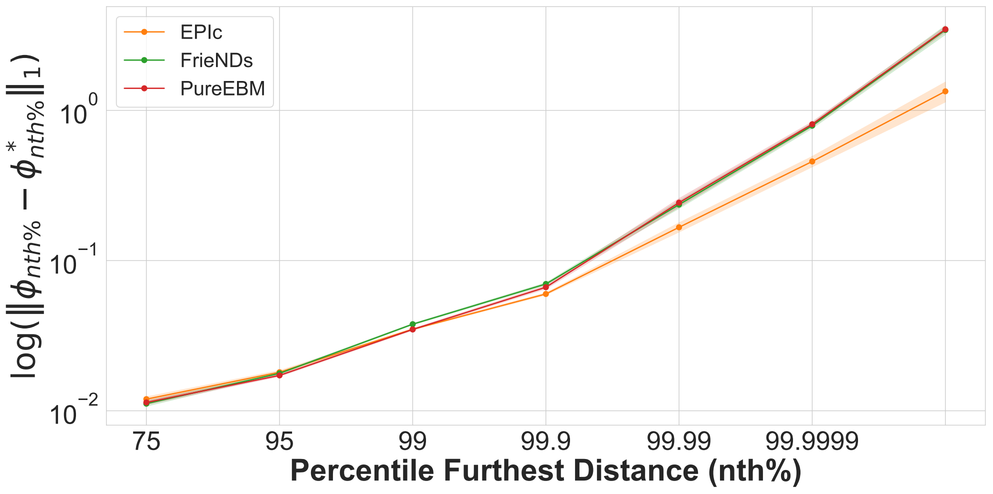


Figure D.5: Comparing parameter distances from defended models to poisoned model (same init) for increasingly higher percentiles of the most moved parameters. PureEBM trained models show the least movement in the tail of parameter which poisons are theorized to impact most (followed very closely by FrieNDs but well above EPIc).

D.6 EBM Langevin Dynamics Grid Searches

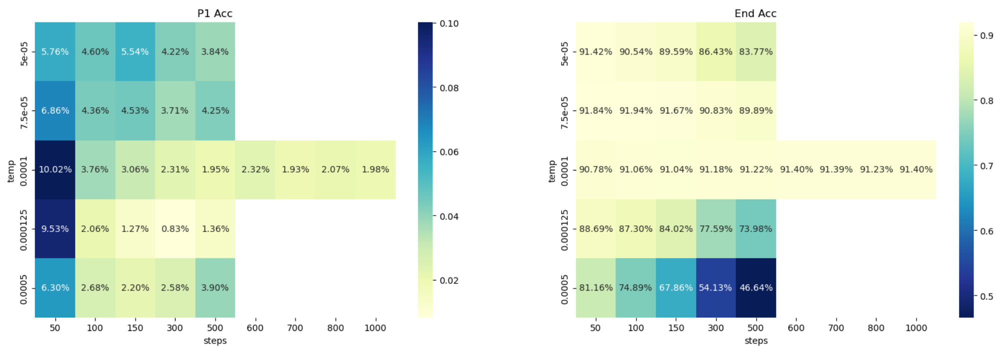


Figure D.6: Grid Search for Langevin steps and temp on Narcissus Fine-Tune Transfer

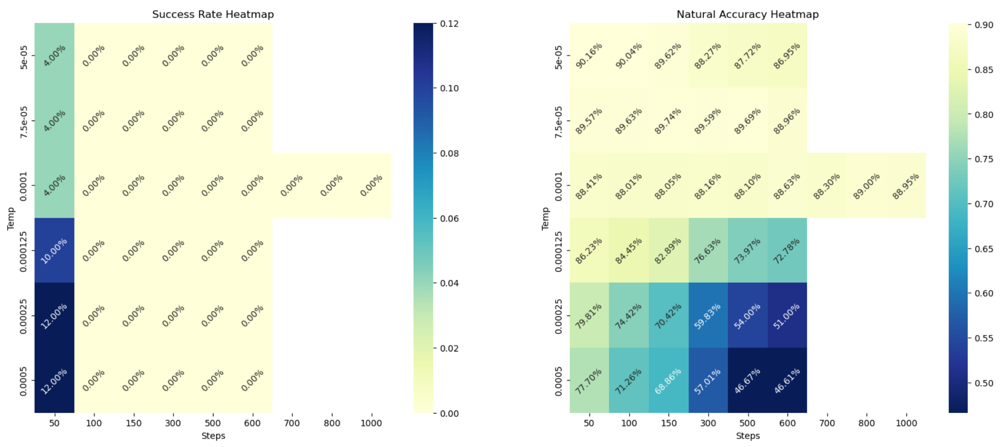


Figure D.7: Grid Search for Langevin steps and temp on Bullseye Polytope Fine-Tune Transfer

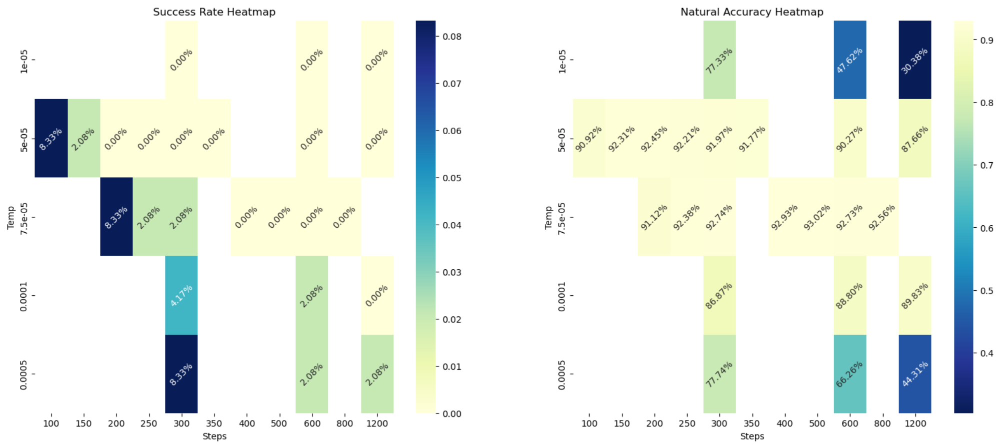


Figure D.8: Grid Search for Langevin steps and temp on Bullseye Polytope Linear Transfer

APPENDIX E

PureGen: Universal Data Purification Using Generative Model Dynamics

E.1 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. In particular, we specifically avoid extensive testing on data avilability attacks, which do not introduce latent vulnerabilities in a model and represent a much smaller risk to the machine learning community. Such attacks have been explored to protect artists and content creators from having their work product be used as training data for models without their permission and with an evolving and unclear legal landscape. Thus we limit research on such attacks where it is unclear if the risk outweigh the benefits.

This goal of secure model training is challenging enough withouts 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.

E.2 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 and we only substituted in a poisoned dataloader [Bal23].

Parameter	From Scratch	Transfer Linear	Transfer Fine-Tune
PureGen-EBM Steps (T_{EBM})	150	500	1000
PureGen-DDPM Steps (T_{DDPM})	75	125	125
PureGen-Reps (T_{Reps})	7	-	-
PureGen-Filt (k)	0.5	-	-
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
Reinitialize Linear Layer	NA	True	True

E.3 Core Results Compute

Training compute for *core result only* which is in Table 7.1 on a TPU V3.

Table E.1: Compute Hours TPU V3

	From Scratch			**Transfer**				**Total**
	Narcissus	**Gradient Matching**	**NTGA**	**Fine Tune BP BlackBox**	**Fine Tune Narc**	**Linear BP BlackBox**	**Linear BP WhiteBox**	
Train Time (Hours)	1959.91	4155.74	283.75	73.82	45.89	548.15	70.76	7138.03

E.4 Additional Results

E.4.1 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 (bernouilli or gaussian added noise trasnform), and all natural JPEG versions (compression ratios). Green highlight indicates a baseline defense that was selected for the main paper results table *chosen by the best poison defense performance that did not result in significant natural accuracy degradaion.* For both TinyImageNet and CINIC-10 from-scratch results, best performing baseline settings were used from respective poison scenarios in CIFAR-10 for compute reasons (so there are no additonal results and hence they are removed from this table).

From Scratch					
	CIFAR-10 (ResNet-18)				
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	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	$91.16_{0.4}$	$8.32_{22.3}$	$91.01_{0.4}$	71.76
FrieNDs-G	**0.00**	$91.15_{0.4}$	$9.49_{25.9}$	$91.06_{0.2}$	83.03
JPEG-25	**0.00**	$90.00_{0.19}$	**1.67**$_{0.88}$	$90.15_{0.26}$	3.38
JPEG-50	**0.00**	$91.70_{0.18}$	**1.70**$_{0.98}$	$91.83_{0.20}$	3.83
JPEG-75	2.00	$92.73_{0.20}$	**1.78**$_{1.17}$	**92.94**$_{0.15}$	4.13
JPEG-85	5.00	$93.43_{0.16}$	$5.76_{13.24}$	$93.43_{0.20}$	43.36
PureGen-DDPM	**0.00**	$90.93_{0.20}$	**1.64**$_{0.82}$	$90.99_{0.22}$	**2.83**
PureGen-EBM	1.00	$92.98_{0.2}$	**1.39**$_{0.8}$	**92.92**$_{0.2}$	**2.50**

	Transfer Learning (CIFAR-10, ResNet-18)								
	Fine-Tune					**Linear - Bullseye Polytope**			
	Bullseye Polytope-10%		**Narcissus-10%**			**BlackBox-10%**		**WhiteBox-1%**	
	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Poison Success (%) ↓	Avg Nat Acc (%) ↑
None	46.00	$89.84_{0.9}$	$33.41_{33.9}$	$90.14_{2.4}$	98.27	93.75	$83.59_{2.4}$	98.00	$70.09_{0.2}$
EPIc-0.1	50.00	$89.00_{1.8}$	$32.40_{33.7}$	$90.02_{2.2}$	98.95	91.67	$83.48_{2.9}$	98.00	$69.35_{0.3}$
EPIc-0.2	42.00	$81.95_{5.6}$	$20.93_{27.1}$	$88.58_{2.0}$	91.72	66.67	$84.34_{3.8}$	91.00	$64.79_{0.7}$
EPIc-0.3	44.00	$86.75_{6.3}$	$28.01_{34.9}$	$84.36_{6.3}$	99.91	66.67	$83.23_{3.8}$	63.00	$60.86_{1.5}$
FrieNDs-B	8.00	$87.80_{1.1}$	$3.34_{5.7}$	$89.62_{0.5}$	19.48	35.42	$84.97_{2.2}$	19.00	$60.85_{0.6}$
FrieNDs-G	8.00	$87.82_{1.2}$	$3.04_{5.1}$	$89.81_{0.5}$	17.32	33.33	$85.18_{2.3}$	19.00	$60.90_{0.6}$
JPEG-25	0.00	$88.93_{0.66}$	$2.95_{3.71}$	$87.63_{0.49}$	12.55	**0.00**	$\mathbf{92.44}_{0.47}$	8.0	$50.42_{0.73}$
JPEG-50	**0.00**	$\mathbf{90.40}_{0.44}$	$3.51_{4.64}$	$88.41_{0.58}$	15.76	**0.00**	$86.03_{2.23}$	16.0	$53.49_{0.54}$
JPEG-75	4.00	$90.11_{0.78}$	$18.28_{25.83}$	$89.12_{0.51}$	86.39	16.67	$84.23_{1.76}$	36.0	$56.02_{0.50}$
JPEG-85	5.00	$93.43_{0.16}$	$25.19_{31.44}$	$88.63_{1.59}$	94.41	54.17	$83.61_{1.36}$	51.0	$58.08_{0.54}$
PureGen-DDPM	**0.00**	$\mathbf{91.53}_{0.15}$	$\mathbf{1.88}_{1.12}$	$\mathbf{90.69}_{0.26}$	**3.42**	**0.00**	$\mathbf{93.81}_{0.08}$	9.0	$54.53_{0.64}$
PureGen-EBM	**0.00**	$87.52_{1.2}$	$\mathbf{2.02}_{1.0}$	$\mathbf{89.78}_{0.6}$	**3.85**	**0.00**	$\mathbf{92.38}_{0.3}$	**6.00**	$\mathbf{64.98}_{0.3}$

E.4.2 From Scratch 80 Epochs Experiments

Baseline FrieNDs [LYM23] includes an 80-epoch from-scratch scenario to show poison defense on a faster training schedule. None of these results are included in the main paper, but we show again SoTA or near SoTA for PureGen against all baselines (and JPEG is again introduced as a baseline).

Table E.2: From-Scratch 80-Epochs Results (ResNet-18, CIFAR-10)

From Scratch (80 - Epochs)					
	Gradient Matching-1%		**Narcissus-1%**		
	Poison Success (%) ↓	Avg Nat Acc (%) ↑	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	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
JPEG-25	1.00	$88.73_{0.24}$	$\mathbf{1.66}_{0.92}$	$90.01_{0.20}$	3.18
JPEG-50	2.00	$90.55_{0.23}$	$\mathbf{1.76}_{1.07}$	$90.56_{0.28}$	3.67
JPEG-75	**0.00**	$91.69_{0.23}$	$\mathbf{1.68}_{1.14}$	$91.67_{0.23}$	3.79
JPEG-85	4.00	$92.31_{0.23}$	$\mathbf{1.87}_{1.41}$	$\mathbf{92.42}_{0.16}$	5.13
PUREGEN-DDPM	1.00	$89.82_{0.26}$	$\mathbf{1.54}_{0.82}$	$90.00_{0.13}$	**2.52**
PUREGEN-EBM	1.00	$\mathbf{92.02}_{0.2}$	$\mathbf{1.52}_{0.8}$	$\mathbf{92.02}_{0.3}$	**2.81**

E.4.3 Full Results for MobileNetV2 and DenseNet121

Table E.3: MobileNetV2 Full Results

	From Scratch					Transfer Learning				
	Gradient Matching-1%		**Narcissus-1%**			**Fine-Tune Narcissus-10%**			**Linear BP BlackBox-10%**	
	Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Avg Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Max Poison Success (%) $\downarrow$	Avg Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Max Poison Success (%) $\downarrow$	Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$
None	20.00	93.86$_{0.2}$	32.70$_{24.5}$	93.92$_{0.1}$	73.97	23.59$_{23.2}$	88.30$_{1.2}$	66.54	81.25	73.27$_{1.0}$
EPIc-0.1	37.50	91.28$_{0.2}$	40.09$_{27.1}$	91.15$_{0.2}$	79.74	23.25$_{22.8}$	88.35$_{1.0}$	65.97	81.25	69.78$_{2.0}$
EPIc-0.2	19.00	91.24$_{0.2}$	38.55$_{27.5}$	87.65$_{0.5}$	74.72	19.95$_{19.2}$	87.67$_{1.3}$	50.05	56.25	54.47$_{5.6}$
EPIc-0.3	9.78	87.80$_{1.6}$	22.35$_{23.9}$	78.16$_{9.9}$	69.52	21.70$_{28.1}$	78.17$_{6.0}$	74.96	58.33	58.74$_{9.0}$
FRIENDs-B	6.00	84.30$_{2.7}$	**2.00**$_{1.3}$	88.82$_{0.6}$	4.88	**2.21**$_{1.5}$	**83.05**$_{0.7}$	**5.63**	41.67	68.86$_{1.5}$
FRIENDs-G	5.00	88.84$_{0.4}$	**2.05**$_{1.7}$	88.93$_{0.3}$	6.33	**2.20**$_{1.4}$	**83.04**$_{0.7}$	**5.42**	47.92	68.94$_{1.5}$
JPEG-25	1.00	85.18$_{0.31}$	**2.43**$_{1.16}$	85.00$_{0.24}$	4.40	6.28$_{9.05}$	80.00$_{1.04}$	29.25	2.08	73.14$_{0.71}$
JPEG-50	1.00	86.82$_{0.34}$	**2.30**$_{1.20}$	86.60$_{0.13}$	3.99	6.76$_{10.23}$	83.70$_{0.94}$	33.34	12.50	76.12$_{1.75}$
JPEG-75	1.00	88.08$_{0.34}$	2.46$_{1.42}$	87.88$_{0.23}$	4.87	13.84$_{18.74}$	84.67$_{1.41}$	52.96	68.75	73.11$_{1.53}$
JPEG-85	2.00	88.83$_{0.31}$	10.03$_{16.93}$	88.61$_{0.34}$	52.46	14.93$_{18.42}$	85.46$_{1.31}$	56.82	77.08	71.99$_{1.07}$
PureGen-EBM	**1.00**	**90.93**$_{0.2}$	**1.64**$_{0.8}$	**91.75**$_{0.1}$	**2.91**	**3.66**$_{5.4}$	**84.18**$_{0.5}$	18.85	**0.00**	**78.57**$_{1.4}$
PureGen-DDPM	1.00	86.79$_{0.26}$	**2.13**$_{1.02}$	86.91$_{0.23}$	**3.74**	**3.41**$_{4.82}$	**86.92**$_{0.39}$	16.79	**0.00**	**83.14**$_{0.20}$

Table E.4: DenseNet121 Full Results

	From Scratch					Transfer Learning				
	Gradient Matching-1%		**Narcissus-1%**			**Fine-Tune Narcissus-10%**			**Linear BP BlackBox-10%**	
	Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Avg Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Max Poison Success (%) $\downarrow$	Avg Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$	Max Poison Success (%) $\downarrow$	Poison Success (%) $\downarrow$	Avg Nat Acc (%) $\uparrow$
None	14.00	95.30$_{0.1}$	46.52$_{32.2}$	95.33$_{0.1}$	91.96	56.52$_{38.6}$	87.03$_{2.8}$	99.56	73.47	82.13$_{1.6}$
epic-0.1	14.00	93.0$_{0.3}$	43.38$_{32.0}$	93.07$_{0.2}$	88.97	53.97$_{39.0}$	87.04$_{2.8}$	99.44	62.50	78.88$_{2.1}$
epic-0.2	7.00	90.67$_{0.5}$	41.97$_{33.2}$	90.23$_{0.6}$	86.85	43.66$_{36.5}$	85.97$_{2.6}$	97.17	41.67	70.13$_{5.2}$
epic-0.3	4.00	88.3$_{1.0}$	32.60$_{29.4}$	85.12$_{2.4}$	71.50	43.24$_{43.0}$	72.76$_{10.8}$	100.00	66.67	70.20$_{10.1}$
friends-B	1.00	91.33$_{0.4}$	8.60$_{21.2}$	91.55$_{0.3}$	68.57	5.34$_{9.9}$	88.62$_{0.8}$	33.42	60.42	80.22$_{1.9}$
friends-G	1.00	91.33$_{0.4}$	10.13$_{25.2}$	91.32$_{0.4}$	81.47	5.55$_{10.4}$	88.75$_{0.6}$	34.91	56.25	80.12$_{1.8}$
jpeg-25	**0.00**	90.09$_{0.17}$	1.68$_{1.10}$	90.15$_{0.26}$	3.62	2.46$_{2.92}$	83.82$_{0.81}$	9.79	0.00	78.67$_{1.60}$
jpeg-50	**0.00**	91.94$_{0.20}$	1.90$_{1.54}$	92.03$_{0.22}$	5.41	3.07$_{2.69}$	85.92$_{1.07}$	8.70	12.50	84.24$_{1.39}$
jpeg-75	**0.00**	**93.08**$_{0.19}$	2.73$_{3.12}$	93.16$_{0.07}$	8.88	32.21$_{35.67}$	87.19$_{1.28}$	98.64	27.08	81.31$_{1.34}$
jpeg-85	7.00	93.85$_{0.19}$	13.36$_{28.10}$	93.77$_{0.27}$	90.77	38.70$_{37.98}$	86.25$_{2.29}$	97.19	70.83	80.57$_{1.40}$
pgenEBM	**0.00**	**92.85**$_{0.2}$	**1.42**$_{0.7}$	**93.48**$_{0.1}$	**2.60**	**2.48**$_{1.9}$	**88.75**$_{0.5}$	**7.41**	**0.00**	**89.29**$_{0.9}$
pgenDDPM	3.00	91.09$_{0.24}$	**1.71**$_{0.94}$	90.94$_{0.23}$	2.97	**2.79**$_{2.51}$	**88.21**$_{0.62}$	**8.95**	0.00	**89.02**$_{0.15}$

E.4.4 PureGen Combos on Increased Poison Power

Table E.5: PureGen Combos with Narcissus Increased Poison % and ϵ

	Narcissus $\epsilon = 8$ **1%**			**Narcissus** $\epsilon = 8$ **10%**			**Narcissus** $\epsilon = 16$ **1%**			Narcissus $\epsilon = 16$ 10%		
	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓	Avg Poison Success (%) ↓	Avg Nat Acc (%) ↑	Max Poison Success (%) ↓
None	$40.33_{38.63}$	$93.61_{0.12}$	90.26	$96.27_{6.62}$	$84.57_{0.60}$	99.97	$83.63_{12.09}$	$93.67_{0.11}$	97.36	$99.35_{0.81}$	$84.58_{0.63}$	99.97
EPIc-0.1	$36.39_{34.52}$	$92.02_{0.63}$	87.34	$98.63_{2.06}$	$83.12_{0.69}$	99.98	$74.81_{13.75}$	$92.22_{0.33}$	94.26	$98.81_{3.04}$	$82.89_{0.59}$	99.96
EPIc-0.2	$29.75_{31.82}$	$88.37_{0.37}$	82.41	$96.65_{3.73}$	$79.75_{0.94}$	99.78	$73.11_{14.02}$	$88.24_{0.59}$	91.51	$98.54_{1.43}$	$79.73_{0.93}$	99.57
EPIc 03	$28.53_{33.15}$	$83.00_{1.87}$	93.68	$93.54_{12.99}$	$76.31_{1.54}$	99.99	$56.16_{17.91}$	$82.94_{1.53}$	81.59	$97.40_{3.51}$	$75.95_{1.59}$	99.97
FrieNDs-G	$10.01_{26.92}$	$91.18_{0.30}$	86.53	$32.40_{43.26}$	$84.95_{1.70}$	99.89	$18.90_{20.93}$	$91.15_{0.34}$	71.46	$73.56_{20.71}$	$82.82_{0.42}$	97.31
FrieNDs-B	$7.89_{20.37}$	$91.25_{0.26}$	65.78	$30.87_{41.67}$	$85.17_{1.70}$	100.00	$18.06_{16.88}$	$91.16_{0.29}$	54.17	$71.28_{22.90}$	$82.83_{0.43}$	99.25
JPEG-25	**$1.82_{0.98}$**	$87.76_{0.15}$	3.46	$16.95_{9.72}$	$84.66_{1.51}$	33.96	$11.85_{12.60}$	$87.72_{0.19}$	36.90	$79.28_{8.05}$	$79.87_{0.79}$	91.99
JPEG-50	**$1.75_{1.07}$**	$89.35_{0.27}$	3.71	$31.14_{14.99}$	$84.15_{1.63}$	50.97	$21.67_{17.81}$	$89.32_{0.19}$	57.58	$90.48_{6.66}$	$81.14_{0.86}$	100.00
JPEG-75	**$1.66_{0.92}$**	**$90.70_{0.14}$**	3.30	$56.99_{27.60}$	$84.07_{1.51}$	99.53	$34.06_{21.16}$	$90.77_{0.13}$	65.14	$92.00_{6.90}$	$82.20_{0.73}$	99.86
JPEG-85	$4.67_{9.82}$	$91.74_{0.12}$	32.52	$69.75_{25.47}$	$83.72_{1.39}$	100.00	$42.21_{26.43}$	$91.66_{0.24}$	82.82	$93.22_{6.90}$	$82.90_{0.53}$	99.83
PureGen-DDPM	**$1.72_{0.92}$**	$89.61_{0.14}$	3.20	**$6.38_{5.16}$**	**$85.86_{0.46}$**	16.29	**$5.21_{3.35}$**	$86.16_{0.19}$	13.32	$69.38_{16.73}$	$83.58_{1.02}$	89.35
PureGen-EBM	**$1.59_{0.86}$**	**$91.41_{0.16}$**	3.01	$52.48_{23.29}$	$86.14_{1.82}$	99.86	$7.35_{4.46}$	$85.61_{0.25}$	16.94	$77.50_{7.01}$	$78.79_{0.93}$	90.84
PureGen-Naive $T{=}[150,75,1]$	**$1.71_{0.89}$**	$88.84_{0.16}$	3.17	$10.43_{8.58}$	$88.20_{0.54}$	27.42	**$5.20_{2.61}$**	$85.95_{0.23}$	9.80	$63.01_{15.24}$	$83.14_{0.90}$	87.17
PureGen-Reps $T{=}[10,50,5]$	**$1.73_{0.84}$**	$87.25_{0.18}$	3.25	**$3.75_{2.28}$**	**$85.56_{0.22}$**	**7.74**	**$4.95_{2.48}$**	**$85.79_{0.18}$**	10.75	**$53.79_{17.14}$**	**$83.92_{1.02}$**	**81.09**
PureGen-Filt $T{=}[0,125,1], k{=}0.5$	**$1.73_{0.89}$**	**$90.61_{0.16}$**	**2.88**	**$6.47_{6.98}$**	**$86.08_{2.00}$**	18.81	**$5.74_{4.05}$**	**$90.52_{0.18}$**	16.08	$69.13_{12.94}$	**$85.47_{1.45}$**	87.66

E.5 PureGen-Naive,PureGen-Reps,PureGen-Filt T Param Sweeps

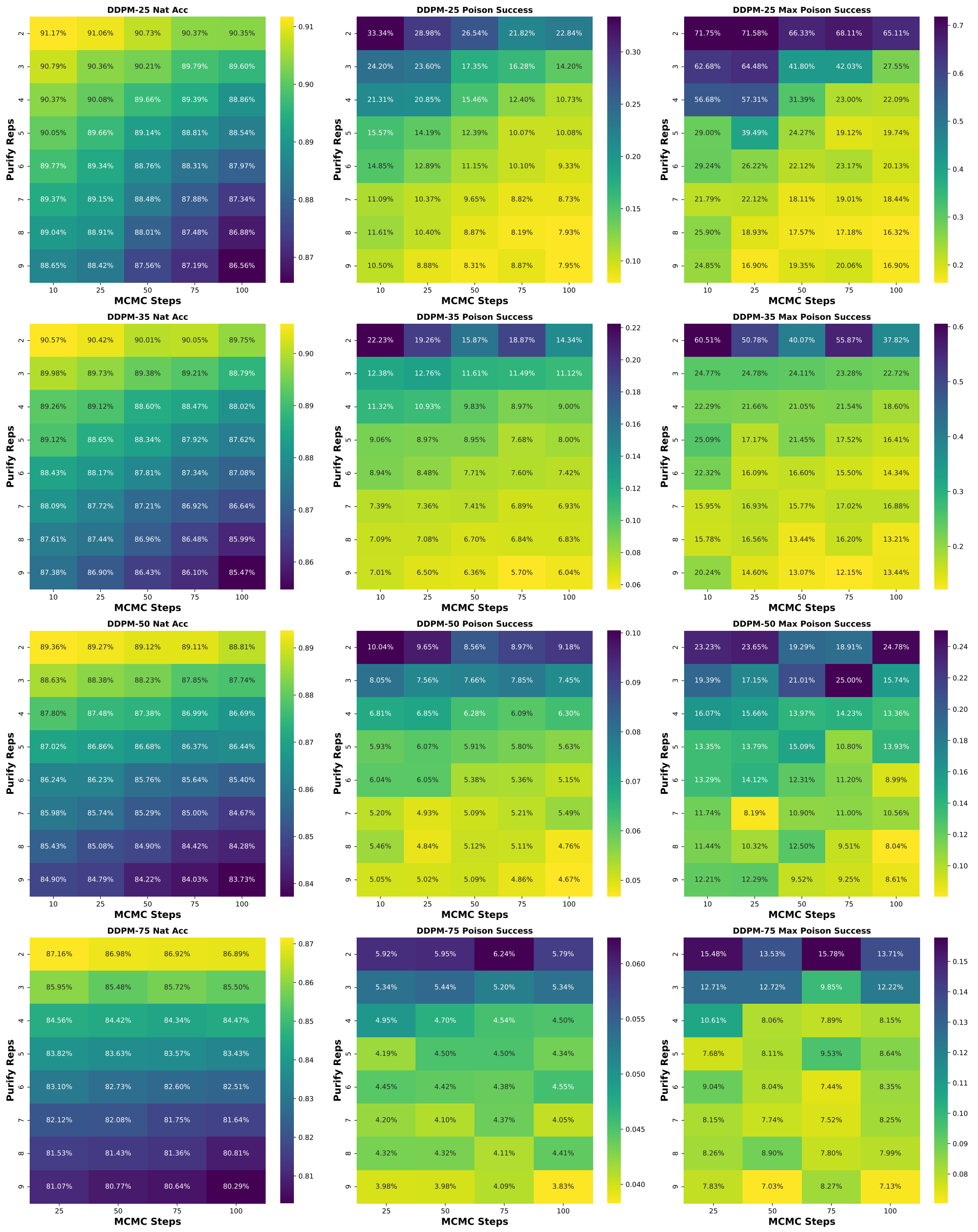


Figure E.1: PureGen-Reps Sweeps with HLB Model on Narcissus

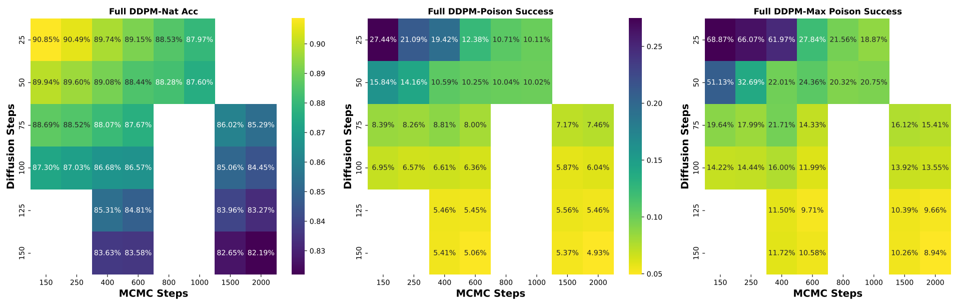


Figure E.2: PureGen-Naive Sweeps with HLB Model on Narcissus

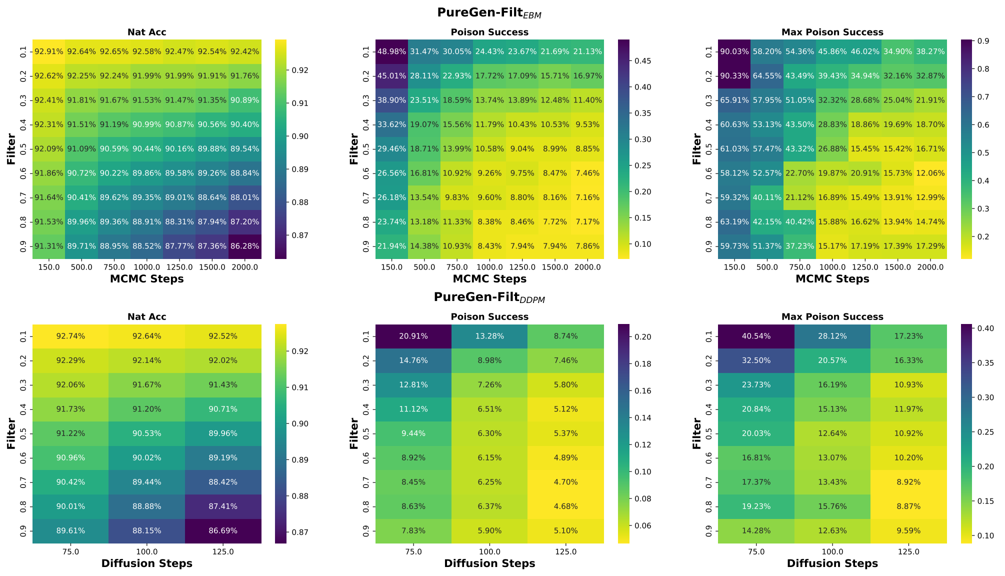


Figure E.3: PureGen-Filt Sweeps with HLB Model on Narcissus

E.6 PureGen for Intellectual Property Purification

Table E.6: Full Results for IP Purification at various logo dimension sizes

	16x16 Logos															
	Natural Accuracy (%) ↑								Logos Accuracy (%) ↓							
Mix Ratio	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**
Baseline	56.39	56.16	55.88	55.47	55.85	55.24	55.77	56.17	**2.84**	**3.32**	**3.35**	99.92	99.96	100.00	100.00	100.00
JPEG $_{75}$	55.27	55.00	54.60	54.17	54.55	54.81	53.89	53.63	**2.80**	**3.03**	**3.25**	**3.53**	90.36	99.76	99.95	99.96
PureGen-EBM $_{500}$	51.37	51.15	50.69	51.12	50.34	50.05	49.67	49.24	**3.19**	**3.20**	62.74	**2.75**	99.61	99.98	100.00	100.00
PureGen-EBM $_{1250}$	49.14	48.69	48.08	47.94	47.06	47.21	46.80	46.78	**3.10**	**3.09**	**3.02**	**2.97**	95.09	99.16	99.91	99.99
PureGen-EBM $_{2000}$	46.18	45.46	44.99	45.13	45.00	44.43	44.44	44.09	**3.10**	**3.25**	**3.02**	**3.24**	**79.57**	96.39	99.64	100.00
PureGen-DDPM $_{75}$	49.04	50.49	49.74	49.88	49.16	48.49	48.39	47.97	**3.10**	**3.08**	**3.38**	**3.28**	93.28	98.73	99.91	100.00
PureGen-DDPM $_{100}$	48.69	48.21	48.66	47.89	46.23	46.61	46.48	45.43	**3.30**	**3.18**	**3.10**	**9.20**	**46.42**	96.42	99.18	99.96
PureGen-DDPM $_{125}$	45.73	45.64	45.51	45.08	43.30	43.21	43.29	43.52	**3.16**	**3.37**	**3.20**	**3.89**	**32.96**	**52.26**	97.57	99.90

	24x24 Logos															
	Natural Accuracy (%) ↑								Logos Accuracy (%) ↓							
Mix Ratio	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**
Baseline	55.42	54.00	54.34	53.45	53.45	52.37	53.45	53.93	**2.91**	**3.33**	99.99	99.99	100.00	99.99	100.00	100.00
JPEG $_{75}$	55.27	53.33	53.42	53.39	52.45	52.36	52.72	51.95	**2.80**	**3.16**	**3.38**	99.83	99.97	100.00	100.00	100.00
PureGen-EBM $_{500}$	51.37	50.02	50.17	48.85	48.57	48.20	48.11	47.50	**3.19**	**3.26**	96.30	99.95	100.00	100.00	100.00	100.00
PureGen-EBM $_{1250}$	49.04	50.31	48.90	46.53	47.58	46.88	47.20	46.77	**3.10**	**39.55**	**3.37**	99.00	99.96	99.99	99.99	100.00
PureGen-EBM $_{2000}$	46.18	44.00	44.45	42.80	42.95	41.85	41.93	41.89	**3.10**	**21.35**	69.76	92.23	99.53	99.96	100.00	100.00
PureGen-DDPM $_{75}$	49.04	50.31	48.90	46.53	47.58	46.88	47.20	46.77	**3.10**	**39.55**	**3.37**	99.00	99.96	99.99	99.99	100.00
PureGen-DDPM $_{100}$	48.69	46.24	46.44	44.86	45.12	44.71	43.43	43.77	**3.30**	**3.26**	**3.62**	97.32	99.61	99.95	99.98	99.99
PureGen-DDPM $_{125}$	45.73	44.67	44.05	43.33	43.26	42.31	41.40	40.88	**3.16**	**3.25**	**3.39**	80.70	98.20	99.74	99.98	100.00

	32x32 Logos															
	Natural Accuracy (%) ↑								Logos Accuracy (%) ↓							
Mix Ratio	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**	**0.00**	**0.10**	**0.25**	**0.40**	**0.50**	**0.60**	**0.75**	**1.00**
Baseline	55.42	52.13	52.74	51.93	50.72	50.11	51.64	51.71	**2.91**	99.98	99.98	99.99	99.99	99.99	99.99	100.00
JPEG $_{75}$	55.27	53.74	51.35	51.72	50.30	50.65	49.50	50.62	**2.80**	99.05	99.72	99.94	100.00	100.00	100.00	99.99
PureGen-EBM $_{500}$	51.37	48.10	47.66	46.49	45.78	44.02	45.27	46.09	**3.19**	99.43	99.89	99.99	100.00	100.00	100.00	100.00
PureGen-EBM $_{1250}$	49.14	45.59	44.77	43.20	42.48	41.56	42.59	42.56	**3.10**	96.35	99.11	99.96	99.99	100.00	100.00	100.00
PureGen-EBM $_{2000}$	46.18	43.51	42.39	41.06	39.46	38.97	38.62	39.41	**3.10**	**89.20**	96.89	99.82	99.97	100.00	100.00	100.00
PureGen-DDPM $_{75}$	49.04	47.68	46.64	44.83	43.16	43.88	44.13	44.62	**3.10**	98.18	99.67	99.96	100.00	100.00	100.00	100.00
PureGen-DDPM $_{100}$	48.69	45.64	44.65	42.20	41.99	41.18	40.89	41.40	**3.30**	93.70	98.71	99.89	99.97	100.00	100.00	100.00
PureGen-DDPM $_{125}$	45.73	42.25	42.00	39.49	39.39	38.83	37.98	38.64	**3.16**	**81.45**	95.18	99.55	99.98	99.99	100.00	100.00

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 Jastrzębski, 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.

[Ale09] et al. Alex Krizhevsky, Geoffrey Hinton. “Learning multiple layers of features from tiny images. Technical report, Citeseer,.” `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`, 2009.

[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.

[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.

[BGC21] Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. “DP-InstaHide: Provably Defusing Poisoning and Backdoor Attacks with Differentially Private Data Augmentations.”, 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.

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

[BLZ21] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. "Recent Advances in Adversarial Training for Adversarial Robustness.", 2021.

[BMC23] Alessandro Bruno, Pier Luigi Mazzeo, Aladine Chetouani, Marouane Tliba, and Mohamed Amine Kerkouri. "Insights into Classifying and Mitigating LLMs' Hallucinations.", 2023.

[BPJ22] Sunay Gajanan Bhat, Omead Pooladzandi, Jeffrey Jiang, and Gregory Pottie. "Hypothesis Testing using Causal and Causal Variational Generative Models." In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*, 2022.

[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.

[DEL24] Hadi M. Dolatabadi, Sarah Erfani, and Christopher Leckie. "The Devil's Advocate: Shattering the Illusion of Unexploitable Data using Diffusion Models.", 2024.

[DGB22] Manfred Diaz, Charlie Gauthier, Glen Berseth, and Liam Paull. "Generalization Games for Reinforcement Learning." In *ICLR Workshop on Agent Learning in Open-Endedness*, 2022.

[DM20] Yilun Du and Igor Mordatch. "Implicit Generation and Generalization in Energy-Based Models.", 2020.

[FC19] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks." In *International Conference on Learning Representations*, 2019.

[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.

[GHP22] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech 256.", Apr 2022.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[GQ10a] Adam N. Glynn and Kevin M. Quinn. "An Introduction to the Augmented Inverse Propensity Weighted Estimator." *Political Analysis*, **18**(1):36–56, 2010.

[GQ10b] Adam N. Glynn and Kevin M. Quinn. "An Introduction to the Augmented Inverse Propensity Weighted Estimator." *Political Analysis*, **18**(1):36–56, 2010.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples.", 2015.

[GTX21] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses.", 2021.

[GWJ20] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. "Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One.", 2020.

[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.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models.", December 2020. arXiv:2006.11239 [cs, stat].

[HJY15] Yangbo He, Jinzhu Jia, and Bin Yu. "Counting and Exploring Sizes of Markov Equivalence Classes of Directed Acyclic Graphs." *JMLR.org*, **16**(1):2589–2609, jan 2015.

[HKK20] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. "Scaling Laws for Autoregressive Generative Modeling.", 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.

[HPB24] Nathaniel Hudson, J. Gregory Pauloski, Matt Baughman, Alok Kamatar, Mansi Sakarvadia, Logan Ward, Ryan Chard, André Bauer, Maksim Levental, Wenyi Wang, Will Engler, Owen Price Skelly, Ben Blaiszik, Rick Stevens, Kyle Chard, and Ian Foster. "Trillion Parameter AI Serving Infrastructure for Scientific Discovery: A Survey and Vision.", 2024.

[HRU18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.", 2018.

[HZR15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.", 2015.

[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.

[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.", October 2022. arXiv:2204.04875 [cs, stat].

[Kle12] Samantha Kleinberg. *A Brief History of Causality*, p. 11–42. Cambridge University Press, 2012.

[KMH20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling Laws for Neural Language Models.", 2020.

[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.

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

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[KW13a] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes.", 2013.

[KW13b] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes.", 2013.

[KY22] Beomsu Kim and Jong Chul Ye. "Energy-Based Contrastive Learning of Visual Representations.", 2022.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature*, **521**(7553):436–444, May 2015.

[Len24] Mariana Lenharo. "Google AI has better bedside manner than human doctors — and makes better diagnoses." *Nature*, **625**(7996):643–644, January 2024.

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

[Li18] Xi-Lin Li. "Online Second Order Methods for Non-Convex Stochastic Optimizations.", 2018.

[Li19] X. L. 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.

[LPK20] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable AI: A Review of Machine Learning Interpretability Methods." *Entropy*, **23**(1):18, December 2020.

[LXG23] Yanjie Li, Bin Xie, Songtao Guo, Yuanyuan Yang, and Bin Xiao. "A Survey of Robustness and Safety of 2D and 3D Deep Learning Models Against Adversarial Attacks.", 2023.

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

[LZL23] Zhuoran Liu, Zhengyu Zhao, and Martha Larson. "Image Shortcut Squeezing: Countering Perturbative Availability Poisons with Compression.", 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.

[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.

[Mur22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[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.

[NPR22] Ana Rita Nogueira, Andrea Pugnana, Salvatore Ruggieri, Dino Pedreschi, and João Gama. "Methods and tools for causal discovery and causal inference." *WIREs Data Mining and Knowledge Discovery*, **12**(2), March 2022.

[NZF19] Ignavier Ng, Shengyu Zhu, Zhuangyan Fang, Haoyang Li, Zhitang Chen, and Jun Wang. "Masked Gradient-Based Causal Structure Learning.", 2019.

[OCS20] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. "Zoom In: An Introduction to Circuits." *Distill*, **5**(3):10.23915/distill.00024.001, March 2020.

[OSJ18] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. "The Building Blocks of Interpretability." *Distill*, **3**(3):10.23915/distill.00010, March 2018.

[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.

[Pea00] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, Cambridge, U.K. ; New York, 2000.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, Cambridge, UK, 2 edition, 2009.

[Pea10] Judea Pearl. "An Introduction to Causal Inference." *The International Journal of Biostatistics*, **6**(2), January 2010.

[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.", 2020.

[RBL22a] 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.

[RBL22b] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models.", 2022.

[RGG] Eric Ronco, Henrik Gollee, and Peter Gawthrop. "Modular Neural Networks and Self-Decomposition.".

[RHC23] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. "Toward Transparent AI: A Survey on Interpreting the Inner Structures of Deep Neural Networks.", 2023.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Nature*, **323**(6088):533–536, October 1986.

[RKH22] Miguel A. Ramirez, Song-Kyoo Kim, Hussam Al Hamadi, Ernesto Damiani, Young-Ji Byon, Tae-Yeon Kim, Chung-Suk Cho, and Chan Yeob Yeun. "Poisoning Attacks and Defenses on Artificial Intelligence: A Survey.", 2022.

[RWC19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language Models are Unsupervised Multitask Learners.", 2019.

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

[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, Chris 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**(7587):484–489, January 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.

[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.

[ST22] Pedro Sanchez and Sotirios A. Tsaftaris. “Diffusion Causal Models for Counterfactual Estimation.”, 2022.

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

[SZS14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. “Intriguing properties of neural networks.”, 2014.

[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.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is All you Need.” In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WGL19] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. “Learning Robust Global Representations by Penalizing Local Predictive Power.” In *Advances in Neural Information Processing Systems*, pp. 10506–10518, 2019.

[Wri21] Sewall Wright. “Systems of Mating. I. the Biometric Relations between Parent and Offspring.” *Genetics*, **6,2**, 1921.

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

[WXS23] Ganghua Wang, Xun Xian, Jayanth Srinivasa, Ashish Kundu, Xuan Bi, Mingyi Hong, and Jie Ding. “Demystifying Poisoning Backdoor Attacks from a Statistical Perspective.”, 2023.

[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.

[XML19] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. “Adversarial Attacks and Defenses in Images, Graphs and Text: A Review.”, 2019.

[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.

[YW21] Chia-Hung Yuan and Shan-Hung Wu. “Neural Tangent Generalization Attacks.” In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12230–12240. PMLR, 18–24 Jul 2021.

[YZC22] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. “Availability Attacks Create Shortcuts.” In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD ’22. ACM, August 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.