Generative Modeling With Adversarial Training

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Xuwang Yin

December 2022

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Xuwang Yin

This Dissertation has been read and approved by the examing committee:

Advisor: Gustavo Rohde

Advisor:

Committee Member: Miaomiao Zhang

Committee Member: Jundong Li

Committee Member: Farzad Farnoud

Committee Member: Soheil Kolouri

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

December 2022

Generative Modeling With Adversarial Training

To my beloved parents

Acknowledgments

I would like to first thank my advisor, Gustavo Rohde, without whom this thesis would not have been possible. Gustavo has been very supportive of my research ideas and inclinations, and gave me the freedom to pursue my interests in adversarial machine learning. From Gustavo I have learned the arts of critical thinking, experiment design and analysis, technical writing, and many other valuable skills. I am grateful for his advice, support, and mentorship.

I would like to thank my first research mentor at UVA, Vicente Ordóñez Román, for leading me into the world of computer vision & natural language processing research. From Vicente I have learned how to think about high-level research problems and directions, and the arts of making scientific plots and effective presentation.

I want to thank my dissertation committee members, Miaomiao Zhang, Jundong Li, Soheil Kolouri, and Farzad Hassanzadeh, for providing invaluable suggestions and advice for my dissertation. I want to especially thank Soheil for his help with my first paper on adversarial machine learning.

I would like to thank the administrative staff at CS and CPE, especially Natalie Ann Edwards, for their help and support.

I would also like to thank my friends and colleagues from the CS, ECE, and BME departments who have helped me through difficult and challenging times in graduate school.

Finally, I would like to thank my lab mates at the Imaging & Data Science Lab, both current and past. I am grateful for all of the encouragement and support I received.

Abstract

A key limitation of existing generative models, specifically explicit density models such as autoregressive models, normalizing flows, VAEs, and energy-based models (EBMs), is that they tend to assign higher likelihood values on out-of-distribution data than on in-distribution data. In this work we investigate an adversarial training-based generative model that overcomes this limitation. Inspired by recent work that shows adversarially robust classifiers learn high-level, interpretable features, we investigate training a binary classifier to discriminate in-distribution data from adversarially perturbed out-of-distribution data. Our analysis shows that in this setup, the binary classifier learns the support of the in-distribution data, and the learning process is closely related to MCMC-based maximum likelihood learning of EBMs. The training objective of the binary classifier can also be interpreted as a maximin two-player zero-sum game, and is related to GANs' minimax game. Based on the above analysis, we propose improved training techniques for generative modeling with adversarial training (AT), and show that this AT generative model is capable of generating diverse and realistic images, and at the same time has the expected behavior on (normal and worst-case) out-of-distribution inputs. We further investigate the AT generative model's applications to image restoration (denoising and inpainting), image-to-image translation, detecting adversarial examples, (worst-case) out-of-distribution detection, and generative classification.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 3

1.3 Outline 5

2 Learning Energy-Based Model With Adversarial Training **7**

2.1 Introduction 7

2.2 Background 9

2.2.1 Generative models 9

2.2.2 Energy-based models 10

2.2.3 Binary adversarial training 11

2.3 Related Work 12

2.4 Binary AT Generative Model 14

2.4.1 Optimal solution to the binary AT problem 15

2.4.2 Learning mechanism 17

2.4.3 Maximum likelihood learning interpretation 19

2.4.4 Connection with GANs 21

2.4.5 Improved training of AT generative model 24

2.5 Experiments 26

2.5.1 Setups 27

2.5.2 Image generation . . . 28
2.5.3 Applications to image restoration and image translation . . . 33
2.5.4 Nearest neighbor and interpolation . . . 37
2.5.5 Sampling efficiency . . . 39
2.5.6 Training stability analysis . . . 40
2.6 Conclusion . . . 40

3 Applications **43**

3.1 Adversarial Machine Learning Background . . . 43
3.1.1 On the existence of adversarial examples . . . 44
3.1.2 Adversarial attacks . . . 45
3.1.3 Defense mechanisms . . . 49
3.2 Adversarial Training for Detecting Adversarial Examples . . . 54
3.2.1 Proposed approach . . . 54
3.2.2 Evaluation methodology . . . 59
3.2.3 Experiments . . . 62
3.2.4 Conclusion . . . 69
3.3 Worst-Case Out-Of-Distribution Detection . . . 70
3.3.1 Introduction . . . 70
3.3.2 Experiments . . . 72
3.3.3 Conclusion . . . 74
3.4 Generative Robust Classification . . . 75
3.4.1 Introduction . . . 75
3.4.2 Generative robust classification . . . 77
3.4.3 Experiments . . . 80
3.4.4 Ablation study . . . 85
3.4.5 Conclusion . . . 90

4 Conclusions and Future Work **93**

Bibliography **97**

List of Figures

2.1 Plots of contours and (normalized) gradient vector fields of the D functions learned with different p_0 data. 19

2.2 Plots of contours and (normalized) gradient vector fields of the D functions learned with different p_0 data. 24

2.3 FIDs obtained with models trained with different p_0 datasets on CIFAR-10. . . . 25

2.4 FID scores obtained with different Ks in Algorithm 1 on CIFAR-10. 26

2.5 Progressive training vs. training with fixed-K on CIFAR-10. 26

2.6 The effect of R_1 regularization on CelebA-HQ. 27

2.7 Uncurated CIFAR-10 generated samples. 31

2.8 Source images (top panel) and generated images (bottom panel, 256×256 resolution) on CelebA-HQ, AFHQ-CAT, and LSUN Church. 32

2.9 Uncurated generation samples on CelebAHQ 256. 32

2.10 Uncurated generation samples on AFHQ-CAT 256 and LSUN-Church 256. . . . 33

2.11 Image-to-image translation demonstration. 34

2.12 Uncurated image translation results on CelebA-HQ 256 and AFHQ-CAT 256. . . 34

2.13 Uncurated denoising and inpainting results on CelebA-HQ 256. 36

2.14 Inpainting result obtained with nearest neighbor (NN) search. 36

2.15 Nearest neighbors of generated samples on CIFAR-10. 37

2.16 Interpolation results on CelebA-HQ 256 and AFHQ-CAT 256. 37

2.17 Nearest neighbors of generated samples on CelebA-HQ 256. 38

3.1 A schematic illustration of the proposed adversarial example detection method. . 55
3.2 Performances of integrated detection and generative detection under $L_\infty, \epsilon = 0.3$ constrained attack. 65
3.3 Clean samples and corresponding perturbed samples produced by performing targeted attacks against the generative classifier and the softmax robust classifier. 65
3.4 Performances of generative detection and integrated detection under $L_\infty, \epsilon = 8$ attack. 68
3.5 Clean samples and corresponding perturbed samples by performing targeted attack against the generative classifier and the robust classifier 68
3.6 Seed images and generated images produced by performing targeted adversarial attack on the generative classifier and the softmax robust classier. 69
3.7 The performance of the generative classifier and the softmax robust classifier evaluated under different test perturbation sizes. 82
3.8 Seed images and generated images generated by performing targeted adversarial attack on the generative classifier and the robust discriminative classifier. 85
3.9 Seed images and counterfactuals generated by performing targeted adversarial attack on the generative classifier and the robust discriminative classifier. 85
3.10 Integrated gradients (IG) masks generated by different classifiers 86
3.11 Histogram of d_k's outputs on samples of class k and samples of other classes, $k = 1, 2, 3$. 87
3.12 Adversarial AUC score of the class 1 models of different capacities. 88
3.13 Standard and adversarial performances of the class 1 model when trained with different perturbation sizes. 89
3.14 Adversarial AUC score of the class 1 model when trained with different data augmentation policies. 90
3.15 Clean and adversarial AUC scores of the class 1 model when trained with combined AT and out-distribution AT only. 90

List of Tables

2.1 Key differences between binary AT and maximum likelihood EBMs. 19

2.2 Network architecture for the D model used in CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256. 28

2.3 Training hyperparameters. 28

2.4 Sample generation setting. 29

2.5 IS and FID scores on CIFAR-10. 30

2.6 FID scores on CelebA-HQ 256, AFHQ-CAT [32], and LSUN Church 256. 31

2.7 SSIM [184] of the first 10 noise image and denoised images in Fig. 2.13. 35

2.8 SSIM [184] of the first 10 occluded image and recovered images by the proposed model (Fig. 2.13) and the nearest neighbor (NN) baseline (Fig. 2.14). 35

2.9 The number of update steps in the PGD attack (our method) and Langevin dynamics (other methods). 39

2.10 Number of steps and wall-clock time to generate 50 CIFAR-10 samples. Data of NCSN and VAEBM are from [193]. 39

2.11 FID scores of samples generated using different combinations of number of steps and step-sizes. 39

3.1 Training setups for MNIST models. 62

3.2 AUC scores of the first two binary classifiers d_1, d_2 evaluated under different configurations of PGD attacks. 63

3.3 AUC scores of the first two binary classifiers d_1, d_2 evaluated under different configurations of PGD attacks. 63
3.4 AUC scores of the first two binary classifiers d_1, d_2 under cross-norm and cross-perturbation attacks. 63
3.5 AUC scores of d_1 evaluated under fixed-start and multiple random-restarts attacks. 64
3.6 Mean L_2 distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on the perturbed MNIST test set and 0.95 TPR on the clean MNIST test set. 65
3.7 AUC scores of the first two CIFAR-10 $L_\infty, \epsilon = 8$ binary classifiers d_1, d_2 under $L_\infty, \epsilon = 8$ constrained PGD attacks of different steps and step-sizes. 66
3.8 AUC scores of the first two CIFAR-10 $L_2, \epsilon = 80$ binary classifiers d_1, d_2 under $L_2, \epsilon = 80$ constrained PGD attacks of different steps and step-sizes. 67
3.9 AUC scores of the first CIFAR-10 $L_\infty, \epsilon = 8$ binary classifier d_1 evaluated under PGD attacks of different norms and perturbation limits. 67
3.10 AUC scores of CIFAR-10 d_1 under fixed start and multiple random restarts attacks. 67
3.11 CIFAR-10 mean L_2 distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on perturbed set and 0.95 TPR on clean set. . . . 68
3.12 CIFAR-10 standard and worst-case OOD detection results (AUC scores). 73
3.13 Standard and worst-case OOD detection results (AUC scores) on 256×256 datasets. 74
3.14 CIFAR10 standard accuracy and robust accuracy ($\epsilon_{\text{test}} = 0.5$, L_2 norm). 82
3.15 FID of generated samples and counterfactuals of the generative classifier and softmax robust classifier under different PGD attacks. 84
3.16 Accuracies on the training set and test set before and after calibration. 87
3.17 The values of the learned $\{c_1, ..., c_K\}$. 87

Chapter 1

Introduction

1.1 Motivation

Generative modeling is one of the most active research areas of artificial intelligence. The task of generative modeling can be broadly categorized as *data generation* and *density estimation*. In data generation, the goal is to learn a generator function that can generate more data samples from the same training data distribution. A prominent example is generative adversarial networks (GANs) [56], whose primary purpose is to learn a generator that can produce more data samples. Data generation can be used in many applications, such as data augmentation, art creation, code/text generation, audio synthesis, and simulation in reinforcement learning. Some applications, although cannot be directly solved using data generation, can be reformulated so that they can be solved using a generator. For instance, super-resolution, inpainting, denoising, and image-translation can all be solved using GANs.

We can also use generative models to perform density estimation, that is, to determine the probability density function of the training data. This is a classical problem in statistics and machine learning. Traditional approaches such as kernel density estimators and histogram estimators only work well when the data is in low dimensional space. More complex neural network-based models such as autoregressive models, normalizing flows, variational autoen-

coders (VAEs), and energy-based models (EBMs) have been developed to address density estimation in high-dimensional space. Density estimation has a broad range of applications. For instance, once we have obtained the density function, we can use it to analyze properties of the studied probability distribution. We can also use the density function to identity observations that lie in low-density areas of the density function. This task is called *out-of-distribution (OOD) detection*, also known as *outlier detection*, *novelty detection*, and *anomaly detection*. OOD detection has many practical applications, such as fraud detection in financial transactions, fault detection in manufacturing, intrusion detection in a computer network, spotting potential risk or medical problems in health data, etc.

Unfortunately, most of the generative models fail at the task of OOD detection. It is observed that the learned density functions of many generative models cannot properly estimate the likelihood of out-of-distribution inputs — state-of-the-art explicit density models including Glow [93], PixelCNN [133], VAEs [91, 146], and RealNVP [43], tend to assign a higher likelihood to out-distribution data than to in-distribution data [75, 94, 128, 128, 158]. This result is very counterintuitive: since the estimated density function is normalized, by maximum likelihood training on in-distribution data, the density function should in principle be able to outputs higher likelihood values on in-distribution data.

There is one exception — energy-based models (EBMs) can in fact work reasonably well on OOD detection. EBMs are in fact unnormalized density models, so they cannot be directly optimized with maximum likelihood estimation. Standard EBMs training makes use of the gradient of the log-likelihood which can be approximated with MCMC sampling. The learning process allows EBMs to explicitly suppress spurious modes outside the support of the target data distribution, so the learned density function are able to output lower (unnormalized) likelihood on out-of-distribution data than on in-distribution data. However, when the OOD samples are perturbed by adding small adversarial noise, EBMs become much less effective [9]. The failure of EBMs on *adversarial (or worst-case) OOD detection* indicates that there still exists a large amount of spurious modes in the learned density function.

Inspired by recent work that shows adversarially robust classifiers learn high-level interpretable features, we investigate an adversarial training-based generative model that overcomes the above limitation. Our theoretical analysis shows that in a setting where a binary classifier is trained to distinguish in-distribution (target distribution) data from adversarially perturbed out-of-distribution data, the binary classifier learns a special kind of energy function that models the support of target distribution, and the learning process is closely related to MCMC-based maximum likelihood learning of EBMs. The training objective of the binary classifier is also related to GANs's objective: they both solve a two player zero-sum game with the same utility function, with one solving a maximin game, and the other solving a minimax game. The above analysis explains why adversarially trained (binary) classifier has a strong generative property, and motivates us to apply (binary) adversarial training to generative modeling. Based on our interpretation of the binary classifier as an energy-based model, we propose improved training techniques for generative modeling with AT, and show that this AT generative model is capable of generating realistic and diverse data samples, and at the same time has low probability outputs on normal and worst-case out-of-distribution inputs. In addition to the theoretical analysis and evaluation of the proposed model on the standard image generation task, we investigate the AT generative model's applications to denoising, inpainting, image-to-image translation, out-of-distribution detection, adversarial example detection, and generative classification. Our results suggest that the proposed AT generarive model provides competitive solutions to the above problems.

1.2 Contributions

This thesis introduces a new approach to learning energy-based models. The specific contributions are:

- **Contribution 1: Adversarial training-based approach to learning energy-based models.** We show that (binary) AT learns a special kind of energy function that models the support of the data distribution. The above result is based on an analysis of the optimal

solution to the training objective and a comparative analysis of the training algorithms and training objectives of the proposed approach and standard EBMs. We further develop several techniques to improve the generative modeling performance of the AT generative model. Aside from the evaluation of the proposed model on image generation, we demonstrate the model's applications to denoising, inpainting, and image-to-image translation. Compared to regular EBMs, the proposed AT approach provides the following benefits: the model is stable to train, the learned energy function has the expected behavior on worst-case OOD inputs, and the model can be directly applied to image-to-image translation. We in addition provide a game theory-based analysis of the training objective of the AT generative model, and show its close connection with GANs.

- **Contribution 2: Adversarial training for detecting adversarial examples.** We propose a novel approach to detecting adversarial examples that can withstand adaptive attacks. The idea is to partition the input space into subspaces based on the classifier's decision boundary, train a binary classifier in each subspace to distinguish in-class samples from adversarially perturbed samples of other classes, and then use the binary classifiers to perform clean/adversarial example classification in the subspaces. We provide a comprehensive evaluation of the proposed approach, including a thorough evaluation of the robustness of individual binary classifiers and the overall performance of proposed adversarial example detection approach under different adaptive attacks. The proposed approach not only outperforms existing methods by a large margin, but also shows strong interpretability.
- **Contribution 3: Worst-case out-of-distribution detection.** On CIFAR-10 our approach achieves comparable performance to state-of-the-art approaches that make uses of class labels of in-distribution data. Our results on high resolution datasets show that the model not only is capable of generating diverse and realistic images, but also achieves a good worst-case OOD detection performance on novel OOD datasets. This suggests that the model has correctly captured the data distribution, and demonstrates the benefits of the

proposed approach to learning energy-based model by contrasting in-distribution data with adversarially perturbed real OOD data.

- **Contribution 4: Generative classification.** Training adversarially robust discriminative (i.e., softmax) classifier has been the dominant approach to robust classification. Building on our work on AT generative models, we explore using AT to learn unnormalized class-conditional density models and then performing generative robust classification. Our result shows that, under the conditions of similar model capacity, the generative robust classifier achieves comparable performance to a baseline softmax robust classifier when the test data is clean or when the test perturbation is small, and much better performance when the test perturbation exceeds the training perturbation. The generative classifier is also able to generate samples and counterfactuals that more closely resemble the training data, suggesting that the generative classifier can better capture the class-conditional distributions.

1.3 Outline

The thesis is organized into two parts, where in part I (Chapter 2) we investigate theoretical properties of the proposed AT generative model, evaluate its performance on the standard image generation task, and demonstrate its application to denoising, inpainting, and image translation, and in part II (Chapter 3) we investigate the model's applications to detecting adversarial examples, out-of-distribution detection, and generative classification. We briefly summarize each parts as follows.

Part I: Learning energy-based model with adversarial training. We study a new approach to learning energy-based models (EBMs) based on adversarial training (AT). We show that (binary) AT learns a special kind of energy function that models the support of the data distribution, and the learning process is closely related to MCMC-based maximum likelihood learning of EBMs. We also provide a discussion about the connection between the studied approach and GANs. We further propose improved techniques for generative modeling with AT,

and demonstrate that this new approach is capable of generating diverse and realistic images. We further demonstrate the studied approach's applications to image restoration including denoising and inpainting, and image-to-image translation.

Part II: Applications. We investigate several applications of the proposed generative model, with a focus on the adversarial scenario where the data is adversarially modified to reduce the performance of the model on the given task. We first provide a background review on adversarial machine learning, and then discuss the specific applications to detecting adversarial examples, detecting worst-case OOD inputs, and generative robust classification.

Finally, Chapter 3 concludes this work with a brief summary and directions for future research.

Chapter 2

Learning Energy-Based Model With Adversarial Training

2.1 Introduction

In unsupervised learning, energy-based models (EBMs) [101] are a class of generative model that uses an energy function to model the probability distribution of the observed data. Unlike explicit density models, EBMs model the unnormalized density function, which makes it difficult to evaluate the likelihood function. Maximum likelihood learning of EBMs hence makes use of the likelihood function's gradient which can be approximated using Monte Carlo methods. Each iteration of the learning process involves first generating synthesized data by sampling from the current model, and then updating the model to maximize the energy difference between synthesized data and observed data. This process leads to an energy function that outputs low energies on the data manifold and high energies on other regions. EBMs find applications in image restoration (denoising, inpainting, etc), out-of-distribution detection, and various sample generation tasks. The main difficulties of training EBMs lie in the computational challenges from the sampling procedure and some training stability issues [45, 46, 61, 62, 130, 131, 193, 216].

Another line of work on adversarial training (AT) show that adversarially robust classifiers

learn high-level, interpretable features, and can be utilized to solve various computer vision tasks including generation, inpainting, super-resolution, and image-to-image translation [50, 80, 154, 177]. Compared to state-of-the-art generative models, this AT approach does not provide a competitive generation performance and is therefore of limited value in many of these tasks. Nonetheless, the generative properties of the robust classifier suggest that the model has captured the distribution of the training data, although the underlying learning mechanism is not yet well understood.

At a high level, both EBMs training and AT are based on the idea of first using gradient-based optimization to generate samples that reach high activation under the current model, and then optimizing the model to minimize its activation on the generated samples. In addition, both approaches synthesize new samples by performing gradient descent on the trained model. These similarities suggest that there are some connections between these two approaches.

In this work we investigate the mechanism by which AT learns data distributions, and propose improved techniques for generative modeling with AT. We focus on binary AT [206] which does not requires class labels and hence naturally fits the generative modeling task. We first analyze the binary AT objective and the corresponding training algorithm, and show that binary AT learns a special kind of energy function that models the support of the observed data. We then draw a connection between AT and MCMC-base maximum likelihood learning of EBMs by showing that the binary AT objective can be interpreted as a gradient-scaled version of the likelihood objective in EBMs training, and the PGD attack can be viewed as an non-convergent sampler of the model distribution. This connection provides us with intuition of how AT learns data distributions from a maximum likelihood learning perspective, and suggests that binary AT can be viewed as an approximate maximum likelihood learning algorithm.

We further propose improved techniques for generative modeling with AT based on the above analysis. Our empirical evaluation shows that this AT approach provides competitive generation performance to explicit EBMs, and at the same time is stable to train (just like regular adversarial training), is well-suited for image translation tasks, and exhibits strong out-of-distribution adver-

sarial robustness. The main limitation of the studied approach is that it cannot properly learn the underlying density function of the observed data. However, this problem is not unique to the studied approach - most existing work on learning EBMs relies on short-run non-convergent sampler to improve the training efficiency, and the learned model typically does not have a valid steady-state that reflects the distribution of the observed data [130, 131].

In summary, the contributions of this work are: 1) We show that binary AT learns a special kind of energy function that models the support of the data distribution, and the learning process is closely related to MCMC-based maximum likelihood learning of EBMs. 2) We propose improved techniques for generative modeling with AT, and demonstrate competitive image generation performance to state-of-the-art explicit EBMs. 3) We show that the studied approach is stable to train, has competitive training and test time sampling efficiency, and can be applied to denoising, inpainting, image translation, and worst-case out-of-distribution detection.

2.2 Background

2.2.1 Generative models

The high-level idea of generative modeling is to learn a representation of the data distribution in order to perform density estimation and/or sample generation. Existing generative models can be roughly categorized as density-based models and generator-based models. Density-based approaches explicitly model the density function of the data with a parametric function, and learn the parameters of the function by maximum likelihood estimation. Most of the existing generative models, such as autoregressive models, normalizing flows, variational autoencoders (VAEs), energy-based methods (EBMs), and diffusion models, are density-based models. The Generator-based approach aims to learn a generator function that can produce more samples from the same training distribution. A prominent example of the generator-based approach is generative adversarial networks (GANs) [56]. Density-based approaches can be further divided

into tractable density models (e.g., autoregressive models, normalizing flows, diffusion models), unnormalized density models (e.g., EBMs), and approximate density models (e.g., VAEs). Tractable density models can be directly trained by maximum likelihood estimation. EBMs is typically trained by making use of the gradient of the log-likelihood which can be approximated by MCMC sampling. VAEs is trained by maximizing the evidence lower bound (ELBO) of the log-likelihood. VAEs is a widely used generative model, but samples generated by VAEs tend to be blurry and are generally considered inferior to those produced by state-of-the-art models such as GANs. Diffusion models is a class of generative model that formulates sample generation as a denoising process. Diffusion models is currently one of the best performing generative models for high-resolution image generation.

2.2.2 Energy-based models

Energy-based models (EBMs) [101] represent probability distributions by converting the outputs of a scalar function $f_{\boldsymbol{\theta}}$ into probabilities through a Gibbs distribution:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{Z(\theta)}, \tag{2.1}$$

where the normalizing constant $Z(\theta)$, also known as the partition function, is an integral over the unnormalized probability of all states: $Z(\theta) = \int \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}$. The energy function is defined as $E_\theta(\boldsymbol{x}) = -f_{\boldsymbol{\theta}}(\boldsymbol{x})$, and thus has the property of attributing low energy outputs on the support of the target data distribution and high energy outputs in other regions.

For many interesting models, the partition function $Z(\theta)$ is intractable, and therefore maximum likelihood estimation (MLE) of the model parameters θ is not directly applicable. Standard maximum likelihood learning of EBMs makes use of the gradient of the log likelihood function. Denote the distribution of the observed data as p_{data}, the gradient of the log likelihood takes the form

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})] = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})]. \tag{2.2}$$

Intuitively, maximizing log-likelihood with this gradient causes $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ to increase on p_{data} samples and decrease on samples drawn from $p_{\boldsymbol{\theta}}$; when $p_{\boldsymbol{\theta}}$ matches p_{data}, the gradient cancels out and the training terminates.

Evaluating $\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})$ requires sampling from the model distribution. This can be done with Markov chain Monte Carlo (MCMC) methods. Recent work scaling EBMs training to high-dimensional data [45, 130, 131, 197] makes use of the SGLD method [185] which samples the model distribution by

$$\boldsymbol{x}_0 \sim p_0, \quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\lambda}{2}\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \lambda), \tag{2.3}$$

where p_0 is some random noise distribution. A proper SGLD sampler requires a large number of update steps in order for the distribution of sampled data to match $p_{\boldsymbol{\theta}}$. Due to the high computational cost of this sampling process, many authors resort to short-run non-convergent MCMC to improve the sampling efficiency[45, 61, 130, 131, 197]. The resulting model typically does not have a valid steady-state that reflects the distribution of the observed data, but is still capable of generating realistic and diverse samples [130, 131].

2.2.3 Binary adversarial training

Binary adversarial training [206] is a method for detecting adversarial examples. In a K class classification problem, the detection method consists of K binary classifiers, with the k-th binary classifier trained to distinguish clean data of class k from adversarially perturbed data of other classes. A committee of K binary classifiers then provides a complete solution for detecting adversarially perturbed samples of any classes.

Denote the data distribution of class k as p_{data}, the mixture distribution of other classes as $p_0 = \frac{1}{K-1}\sum_{i=1,\ldots,K,i\neq k} p_i$, the k-th binary classifier is trained by maximizing the objective

$$J(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_0}[\min_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon)} \log(1 - D(\boldsymbol{x}'))], \tag{2.4}$$

where $D : \mathcal{X} \subseteq \mathbb{R}^d \to [0, 1]$ is the classification function, and $\mathbb{B}(\boldsymbol{x}, \epsilon)$ is a neighborhood of $\boldsymbol{x}$: $\mathbb{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{x}' \in \mathcal{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_2 \leq \epsilon\}$. In practice, D is defined by applying a logistic sigmoid function to the output of a neural network:

$$D(\boldsymbol{x}) = \sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})), \tag{2.5}$$

where $f_{\boldsymbol{\theta}}$ is a neural network with a single output node and parameters θ.

The inner minimization in Eq. (2.4) is solved using the PGD attack [99, 112], a first-order method that employs an iterative update rule of (L_2-based attack):

$$\boldsymbol{x}_0 \sim p_0, \quad \boldsymbol{x}_{i+1} = \text{Proj}(\boldsymbol{x}_i - \lambda \frac{\nabla_{\boldsymbol{x}} \log(1 - D(\boldsymbol{x}_i))}{\|\nabla_{\boldsymbol{x}} \log(1 - D(\boldsymbol{x}_i))\|_2}), \tag{2.6}$$

where λ is some step size, and Proj is the operation of projecting onto the feasible set $\mathbb{B}(\boldsymbol{x}, \epsilon)$. Because the gradient vector in Eq. (2.6) is normalized to have unit norm, we can equivalently implement the attack by directly performing gradient ascent on $f_{\boldsymbol{\theta}}$:

$$\boldsymbol{x}_0 \sim p_0, \quad \boldsymbol{x}_{i+1} = \text{Proj}(\boldsymbol{x}_i + \lambda \frac{\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\|\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2}). \tag{2.7}$$

2.3 Related Work

Learning EBMs

Due to the intractability of the normalizing constant in the EBMs likelihood function, maximum likelihood learning of EBMs makes use of the gradient of the log-likelihood which can be approximated using MCMC sampling. Recent work [45, 130, 131, 197] scaling EBMs training to high-dimensional data performs sampling using SGLD [185] and initialize the chain from a noise distribution. The sampling process involves estimating the model's gradient with respect to the current sample at each step and therefore has high computational cost. To improve the sam-

pling efficiency, many authors consider short-run non-convergent SGLD sampler in combination with a persistent sampling buffer [45, 46, 61, 62, 131, 193]. Although a short-run sampler is sufficient for learning a generation model, the resulting energy function typically does not have a valid steady-state [130, 131]. The mixing time of the sampling procedure also depends on how close the chain-initialization distribution is to the model distribution. A recent trend hence considers initializing the sampling chain from samples produced by a generator fitted on the target distribution [6, 62, 68, 69, 98, 132, 134, 193, 198, 202, 203].

Maximum likelihood learning of EBMs also has some training stability issues, and various techniques have been developed to address these issues. These techniques include 1) using weight normalization [151], Swish activation [144], gradient clipping, and weight decay (see [193]), 2) gradient norm clipping on model parameters and using a KL term in the training objective (see [46]), 3) adjusting learning rate and SGLD steps during training and adding Gaussian noise to input images (see [61]), 4) gradient clipping on SGLD and model parameters and spectral normalization (see [45]), and 5) multiscale training and smooth activation functions (see [216]). Overall, there does not seem to have a consensus on how to stabilize EBMs training. Due to the computational challenge of MCMC sampling and stability issues, the successful application of EBMs to modeling high-dimensional data such as 256×256 images is only achieved in some very recent works [193, 216].

Aside from MCMC-based maximum likelihood learning of EBMs, alternative approaches for learning EBMs exist. Score matching [79] circumvents the difficulty of estimating the partition function by directly modeling the derivatives of the data distribution. Score matching has recently been successfully applied to modeling large natural images and achieves competitive performance to state-of-the-art generative models such as GANs [78, 162, 163, 164]. Noise contrastive estimation (NCE) [67] learns data distributions by contrasting the observed data with data from a known noise distribution. Similar to our approach, NCE makes use of a logistic regression model. The main difference is that in NCE, the logit of the classifier is the difference in log probabilities of the model distribution and the noise distribution, whereas in our approach the

logit directly defines the estimator (i.e., the energy function). Unlike other EBMs, NCE typically does not scale well to high-dimensional data [26, 67, 147].

Maximin interpretation of EBMs.

When the noise term in the SGLD sampler is disabled, the learning process of EBMs can be interpreted as solving a *maximin* game [199, 200, 201]. This interpretation coincides with our formulation in Eq. (2.12). The key differences lie in the value function, the setting of the sampler (SGLD vs. PGD attack), and the Markov chain initiation distribution.

Understanding and improving AT generative models

Our work is related to [182] which is also an attempt to understand and improve AT's generative capability. The authors' focus is on the *supervised setting*, where they make a connection between *the standard multiclass* AT [112] and the Contrastive Energy-based Model (CEM) proposed by the authors. Our analysis is in an *unsupervised setting*, where we connect *binary* AT with the standard EBMs formulation (Eq. (2.1)). Although [182] also considers the unsupervised scenario, their generative model is a contrastive learning model [87] which requires training samples to do test time sampling. Our generative model is based on binary AT and follows the standard practice of MCMC sampling from the learned energy function. In addition to the theoretical analysis, we propose improve training techniques which allow us to obtain a significantly better FID on CIFAR-10 (Tab. 2.5) and successfully scale the training to 256x256 datasets.

2.4 Binary AT Generative Model

In this section we develop a generative model based on binary AT. We first analyze the optimal solution to the binary AT problem, and then investigate the mechanism by which binary AT learns the data distribution, and finally interpret the learning process from the maximum likelihood learning perspective. Our main result is that under a proper configuration of perturbation limit

and p_0 data, binary AT learns a special kind of energy function that models the support of p_{data}. Based on these theoretical insights, we proposed improved training techniques.

2.4.1 Optimal solution to the binary AT problem

We consider the optimal solution of Eq. (2.4) under the scenario of unbounded perturbation: $\mathbb{B}(\boldsymbol{x}, \epsilon) = \mathcal{X}$. This allows us to further simplify the PGD attack by removing the Proj operator:

$$\boldsymbol{x}_0 \sim p_0, \quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda \frac{\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\|\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2}. \tag{2.8}$$

Perturbing p_0 samples can be thought of as moving p_0 samples via a translation function $T(\boldsymbol{x}) = \boldsymbol{x} + \Delta_{\boldsymbol{x}}$, with $\Delta_{\boldsymbol{x}}$ being the perturbation computed on sample $\boldsymbol{x}$. We can write the density function of the perturbed distribution p_T using random variable transformation:

$$p_T(\boldsymbol{z}) = \int_{\mathcal{X}} p_0(\boldsymbol{x}) \delta(\boldsymbol{z} - T(\boldsymbol{x})) d\boldsymbol{x}. \tag{2.9}$$

The inner problem in Eq. (2.4) can then be interpreted as determining the distribution which has the lowest expected value of $\log(1 - D(\boldsymbol{x}))$:

$$p_T^* = \arg\min_{p_T} \mathbb{E}_{\boldsymbol{x} \sim p_T}[\log(1 - D(\boldsymbol{x}))]. \tag{2.10}$$

The objective of the outer problem is then the log-likelihood in a logistic regression model which discriminates p_{data} samples from p_T^* samples:

$$J(D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_T^*}[\log(1 - D(\boldsymbol{x}))]. \tag{2.11}$$

We can equivalently formulate Eq. (2.4) as a *maximin* problem

$$\max_D \min_{p_T} U(D, p_T) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_T}[\log(1 - D(\boldsymbol{x}))], \tag{2.12}$$

and obtain its optimal solution by following the standard approach to solving maximin problems:

Proposition 1. *The optimal solution of* $\max_D \min_{p_T} U(D, p_T)$ *is* $U(D^*, p_T^*) = -\log(4)$*, where* D^* *outputs* $\frac{1}{2}$ *on Supp*(p_{data}) *and* $\leq \frac{1}{2}$ *outside Supp*(p_{data})*, and* p_T^* *is supported in the contour set* $\{D = \frac{1}{2}\}$.

Proof. Let

$$p_T^* = \arg\min_{p_T} \mathbb{E}_{\boldsymbol{x}\sim p_T}[\log(1 - D(\boldsymbol{x}))], \tag{2.13}$$

then

$$\max_D \min_{p_T} U(D, p_T) = \max_D U(D, p_T^*). \tag{2.14}$$

We solve $\max_D U(D, p_T^*)$ by first deriving its upper bound. Let $\alpha = \max_{\mathcal{X}} D$, then $\mathbb{E}_{\boldsymbol{x}\sim p_T^*}[\log(1-D(\boldsymbol{x}))]$ is minimized when p_T^* is supported in $\{\boldsymbol{x} : D(\boldsymbol{x}) = \alpha\}$. With this result, we can derive an upper bound of $U(D, p_T^*)$:

$$\begin{aligned}
&U(D, p_T^*) \\
&= \int_{\mathcal{X}} p_{\text{data}}(\boldsymbol{x}) \log D(\boldsymbol{x}) d\boldsymbol{x} + \int_{\mathcal{X}} p_T^*(\boldsymbol{x}) \log(1 - D(\boldsymbol{x})) d\boldsymbol{x} \\
&= \int_{\mathcal{X}} p_{\text{data}}(\boldsymbol{x}) \log D(\boldsymbol{x}) d\boldsymbol{x} + \int_{\mathcal{X}} p_T^*(\boldsymbol{x}) \log(1 - \alpha) d\boldsymbol{x} \\
&\leq \int_{\mathcal{X}} p_{\text{data}}(\boldsymbol{x}) \log(\alpha) d\boldsymbol{x} + \int_{\mathcal{X}} p_T^*(\boldsymbol{x}) \log(1 - \alpha) d\boldsymbol{x} \\
&= \log(\alpha) + \log(1 - \alpha) \\
&\leq -\log(4),
\end{aligned} \tag{2.15}$$

where the last inequality follows from the fact that the function $f(\alpha) = \log(\alpha) + \log(1 - \alpha)$ achieves its maximum value of $-\log(4)$ at $\alpha = \frac{1}{2}$. It is not hard to see that equality holds if and only if i) $\max_{\mathcal{X}} D = \frac{1}{2}$, ii) $D = \frac{1}{2}$ on Supp(p_{data}), and iii) Supp$(p_T^*) \subseteq \{\boldsymbol{x} : D(\boldsymbol{x}) = \frac{1}{2}\}$. In

summary, $\max_D \min_{p_T} U(D, p_T)$ achieves its optimal value of $-\log(4)$ at (D^*, p_T^*) where

$$D^*(\boldsymbol{x}) = \begin{cases} \frac{1}{2} & \boldsymbol{x} \in \text{Supp}(p_{\text{data}}) \\ \leq \frac{1}{2} & \boldsymbol{x} \in \mathcal{X} \setminus \text{Supp}(p_{\text{data}}) \end{cases}, \tag{2.16}$$

and p_T^* is supported in the contour set $\{D = \frac{1}{2}\}$. □

The above maximin problem can also be interpreted as a two-player zero-sum game, and is closely related to GANs [56]'s *minimax* game which has the form

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\text{z} \sim p_z}[\log(1 - D(G(z)))]. \tag{2.17}$$

The game-theory point of view provides a convenient way to understand their differences. We provide a game theory-based analysis of $\max_D \min_{p_T} U(D, p_T)$ and a comparative analysis of GANs in Sec. 2.4.4.

2.4.2 Learning mechanism

Proposition 1 states that by solving $\max_D \min_{p_T} U(D, p_T)$ we can obtain a D that outputs $\frac{1}{2}$ on the support of p_{data} and $\leq \frac{1}{2}$ on other regions. This result is obtained by assuming that for any D, the inner minimization Eq. (2.10) is always perfectly solved. In practice, when D is randomly initialized, it has many local maxima outside the support of p_{data}. Because the inner minimization is solved by taking p_0 samples and then performing gradient ascent on D with Eq. (2.8), this process can get trapped in different local maxima of D. Hence we can think of this process as searching for these local maxima and then put the perturbed p_0 data in these regions. Then in the model update stage (outer maximization), D is updated by increasing its outputs on p_{data} samples and decreasing its outputs on the perturbed p_0 data. By repeating this process, local maxima get suppressed and the model learns to correctly model $\text{Supp}(p_{\text{data}})$.

The algorithm for solving the maximin problem is described in Algorithm 1. Fig. 2.1 left

panel shows the 2D simulation result of the algorithm when the p_0 dataset contains random samples from the uniform distribution. It can be seen that when the algorithm converges, local maxima outside Supp(p_{data}) are suppressed, and D (approximately) outputs $\frac{1}{2}$ on Supp(p_{data}) as predicted by Proposition 1. Meanwhile, D retains the gradient information for translating out-distribution samples to Supp(p_{data}).

Because the PGD attack is deterministic gradient ascent, its ability to discover different local maxima depends on the diversity of p_0 samples. Fig. 2.1 right panel shows that when p_0 data is concentrated in the bottom left corner, the final D still has local maxima outside the support of p_{data}. These local maxima are not suppressed because they were never discovered by the perturbed p_0 data.

The above analysis reveals how binary AT learns data distributions: *the learning starts with a randomly-initialized D solution, and then iteratively refine the solution by suppressing local maxima outside the support of the observed data*. This process is similar to EBMs training where the model distribution's spurious modes are constantly discovered by MCMC sampling and subsequently suppressed in the model update stage. However, unlike the EBMs likelihood objective Eq. (2.2), the AT objective Eq. (2.12) cannot properly learn the density function, but can only capture its support. This is corroborated by the 2D experiment where D outputs $\frac{1}{2}$ uniformly on the support of p_{data} (blue points).

Algorithm 1 Binary Adversarial Training

1: **repeat**
2: Draw samples $\{\boldsymbol{x}_i\}_{i=1}^m$ from p_{data}, and samples $\{\boldsymbol{x}_i^0\}_{i=1}^m$ from p_0.
3: Update $\{\boldsymbol{x}_i^0\}_{i=1}^m$ by performing K steps PGD attack Eq. (2.8) on each sample. Denote the resulting samples as $\{\boldsymbol{x}_i^*\}_{i=1}^m$.
4: Update D by maximizing $\frac{1}{m}\sum_{i=1}^m \log D(\boldsymbol{x}_i) + \frac{1}{m}\sum_{i=1}^m \log(1 - D(\boldsymbol{x}_i^*))$ (single step).
5: **until** D convergences

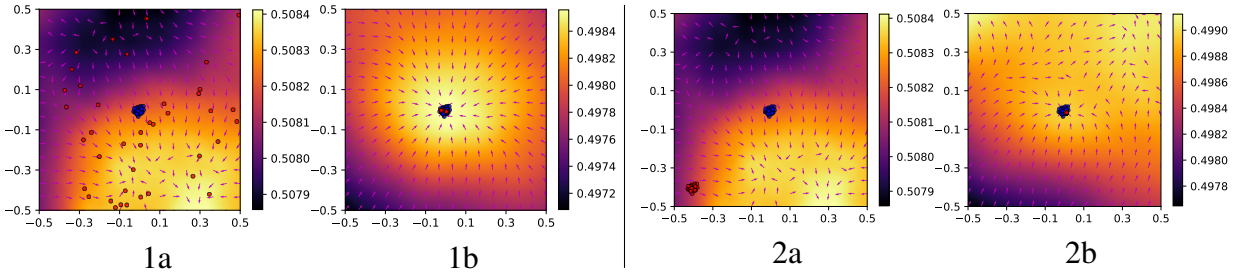


Figure 2.1: Plots of contours and (normalized) gradient vector fields of the D functions learned with different p_0 data. Left and right panel respectively show the initial state (1a and 2a) and final state (1b and 2b) of D when p_0 data is respectively uniformly distributed (red points in 1a) and concerntrated in the lower left corner (red points in 2a). p_{data} is a Gaussian distribution centered at $(0, 0)$ (blue points).

2.4.3 Maximum likelihood learning interpretation

We next consider the learning process of binary AT from a maximum likelihood learning point of view. Both binary AT and MCMC-based EBMs learning employ an iterative optimization algorithm, where in each iteration the contrastive data is computed by performing gradient ascent on the current model, and then the model is updated by maximizing its outputs on the observed data and minimizing its outputs on the contrastive data. The following analysis shows that the PGD attack can be viewed as a non-convergent sampler of the model distribution, and the binary AT objective Eq. (2.11) can be interpreted as a gradient-scaled version of the EBMs objective Eq. (2.2). Tab. 2.1 summaries their key differences.

Table 2.1: Key differences between binary AT and maximum likelihood EBMs.

Objective gradient	EBMs: $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})]$
	Binary AT: $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[(1-\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_T^*}[\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})]$
Contrastive data	EBMs: $\boldsymbol{x}_0 \sim p_0, \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\lambda}{2}\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \epsilon, \epsilon \sim \mathcal{N}(0, \lambda)$
	Binary AT: $\boldsymbol{x}_0 \sim p_0, \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda \frac{\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\|\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2}$
p_0 **data**	EBMs: A noise distribution or a distribution close to p_{data}
	Binary AT: A real and diverse out-distribution dataset (80 million tiny images for CIFAR-10 and ImageNet for 256×256 datasets).

Contrastive data computation

In EBMs training, the contrastive data is computed by MCMC-sampling, typically with Langevin dynamics Eq. (2.3). In binary AT, the contrastive data is computed using the PGD attack Eq. (2.8). Comparing Eq. (2.8) with Eq. (2.3), we find that both approaches compute the contrastive data by first initializing from some out-distribution data, and then performing gradient ascent on $f_{\boldsymbol{\theta}}$. The main differences are that the PGD attack does not have the noise term, and makes use of normalized gradient. Intuitively, the noise term enables the sampler to explore different modes by helping gradient ascent escape local maxima. Although the PGD attack does not have the noise term, its ability to explore different modes can be enhanced by using a diverse p_0 dataset (Fig. 2.3).

In the PGD attack, as the normalized gradient vector has unit norm, the perturbation imposed on $\boldsymbol{x}_i$ is λ; in a K iterations of the update, the overall perturbation $\|\boldsymbol{x}_i^* - \boldsymbol{x}_i\|_2$ is always $\leq \lambda K$. Hence with the PGD attack we can more easily control the distribution of the contrastive data. In contrast, Langevin dynamics adjusts $\boldsymbol{x}_i$ in a scale that corresponds to the magnitude of the gradient of $f_{\boldsymbol{\theta}}$ at $\boldsymbol{x}_i$; when $f_{\boldsymbol{\theta}}$ is updated during training, the overall perturbation may undergo a large change. This behavior of Langevin dynamics can be a source of some training stability issues [131].

Gradient of the training objective

By definition Eq. (2.5), the gradient of D's training objective Eq. (2.11) takes the form

$$\nabla_{\boldsymbol{\theta}} J(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[(1-\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_T^*}[\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})]. \tag{2.18}$$

Comparing the above equation with Eq. (2.2) we find both equations consisting of gradient terms that yield similar effects: the first term causes $f_{\boldsymbol{\theta}}$ outputs on p_{data} samples to increase, and the second causes $f_{\boldsymbol{\theta}}$ outputs on the contrastive samples to decrease. Specifically, as $(1-\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})))$

and $\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}))$ are scalars in the range 0 to 1, the two gradient terms in Eq. (2.18) are respectively the scaled versions of the gradient terms in Eq. (2.2). It should be noted that although these scalars do not change the gradient update direction of individual terms in the model parameter space, the overall gradient update directions of Eq. (2.18) and Eq. (2.2) can be different.

Eq. (2.18) also helps to understand why binary AT can only learn the support of the observed data. In Eq. (2.2), when $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ matches p_{data}, the gradient cancels out and training terminates, whereas in Eq. (2.18), when p_T^* matches p_{data} the gradient becomes $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[(1 - 2\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})]$ and only vanishes when $\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x})) = \frac{1}{2}$ everywhere on the support of p_{data}. This result is consistent with Proposition 1 and the 2D experiment result.

2.4.4 Connection with GANs

Generative adversarial networks (GANs) [56] is a popular approach to generative modeling. GANs learn a generator function G and a discriminator function D by solving the following two-player minimax game:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.19}$$

The generator G implicitly defines a distribution p_g by mapping a prior distribution $p_{\boldsymbol{z}}$ from a low-dimensional latent space $\mathcal{Z} \subseteq \mathbb{R}^z$ to the high-dimensional data space $\mathcal{X} \subseteq \mathbb{R}^d$. $D : \mathcal{X} \to [0, 1]$ is a function that distinguish the target distribution p_{data} samples from p_g samples. It can be shown that in the optimal solution to this minimax game, $p_g = p_{\text{data}}$.

To make a connection to our formulation in Eq. (2.12), we use p_g to define GANs' minimax problem:

$$\min_{p_g} \max_D U(D, p_g) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{2.20}$$

Comparing Eq. (2.12) with Eq. (2.20) we find both problems making use of the same objective function, but order of minimization and maximization is reversed. In fact, both formulations

solve a two-player zero-sum game, a mathematical representation of a situation in which one player's gain is balanced by another player's loss. Such a game is described by its *payoff function* $f : \mathbb{R}^{p+q} \to \mathbb{R}$, which represents the amount of payment that one player (player 1) makes to the other player (player 2). The goal of player 1 is to choose a strategy $u \in \mathbb{R}^p$ such that the payoff is minimized, while the goal of player 2 is to choose a strategy $u \in \mathbb{R}^q$ such that the payoff is maximized. The best strategies for both players, and the resulting payoff, can be solved via $\min_u \max_v f(u, v)$ or $\max_v \min_u f(u, v)$. Depending on the order of maximization and minimization, the game can be played by following two different rules.

In Eq. (2.12), U is the payoff function, and the goal of player p_T is to choose a strategy p_T^* such that the payoff is minimized, whereas the goal of player D is to choose a strategy D^* such that the payoff is maximized. This *maximin* game is played by following such a rule: player D makes the first move by choosing a D; player p_T, after learning that player D has made the move, will choose a p_T to minimize its payment, which results in a payoff of $\min_{p_T} U(D, p_T)$; player D, who is informed of player p_T's strategy, will chooses a D such that the worse case payoff $\min_{p_T} U(D, p_T)$ is maximized, which results in an overall payoff of $\max_D \min_{p_T} U(D, p_T)$. The best strategies of both players and the maximum payoff can be derive from Proposition 1 : In the maximin game $\max_D \min_{p_T} U(D, p_T)$, the best strategy for player D is to choose a D^* that outputs $\frac{1}{2}$ on Supp(p_{data}) and $\leq \frac{1}{2}$ outside of Supp(p_{data}), the best strategy for player p_T is to choose a p_T^* which is supported in $\{\boldsymbol{x} : D(\boldsymbol{x}) = \frac{1}{2}\}$, and the maximum payoff is $-\log(4)$.

In Eq. (2.20), V is the payoff function. Similar to Eq. (2.12), the goal of player p_g is to minimize the payoff, and the goal of player D is to maximize the payoff. In contrast to Eq. (2.12), player p_g makes the first move. The solution to this minimax game is analyzed in [56]: the best strategy of player p_g is to choose a p_g^* which minimizes the Jensen-Shannon divergence (JSD) between p_g and p_{data}: $p_g^* = \arg\min_{p_g} \text{JSD}(p_g \parallel p_{\text{data}}) = p_{\text{data}}$, and the best strategy of player D is to choose $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g^*(\boldsymbol{x})} = \frac{1}{2}$. Under these strategies, the payoff function U measures the JSD between p_g and p_{data}: $U(D^*, p_g^*) = -\log(4) + 2 \cdot \text{JSD}(p_g^* \parallel p_{\text{data}}) = -\log(4)$, which coincides with the U solution in the maximin game. Note that in the minimax game, D^* does

not need to be defined outside of $\text{Supp}(p_g) \cup \text{Supp}(p_{\text{data}})$ [56].

The main differences between the maximin game solution and minimax game solution can be summarized as: 1) In the minimax game, $p_T^* = p_{\text{data}}$, whereas in the maximin game, p_T^* is only required to be supported in $\{\boldsymbol{x} : D(\boldsymbol{x}) = \frac{1}{2}\}$. (2) While both D^*s output $\frac{1}{2}$ in $\text{Supp}(p_{\text{data}})$, their outputs outside of $\text{Supp}(p_{\text{data}})$ are different: in the maximin game, D^* outputs $\leq \frac{1}{2}$ outside of $\text{Supp}(p_{\text{data}})$, whereas in the maximin game, D^* is undefined outside of $\text{Supp}(p_{\text{data}})$.

Due to the above differences these two formulations give rise to different applications. The minimax formulation, which is the formulation used by GANs, is ideal for learning a generator model that can produce a distribution that matches p_{data}. The discriminator, because of its undefined behavior outside of $\text{Supp}(p_{\text{data}})$, may not be very useful for certain downstream tasks like out-of-distribution detection. In the maximin formulation, when the model is trained with a diverse p_0 dataset, we can obtain a D solution where spurious modes are suppressed and therefore can be used for out-of-distribution detection. The D solution at the same time retains gradient information for translating out-distribution samples to $\text{Supp}(p_{\text{data}})$, and thus can be used in generation, image-to-image translation, and some image restoration tasks like denoising and inpainting.

The pseudo code for solving the minimax problem is outlined in Algorithm 2. Fig. 2.2 shows the simulation results in two settings where p_0 data is respectively uniformly distributed (left panel) and concentrated in the lower left corner (right panel). In can be seen that in both cases p_T^* matches p_{data} when the algorithm converges. The right panel shows that when p_0 data is concentrated in the lower left corner, the D solution has undefined outputs outside of $\text{Supp}(p_{\text{data}})$.

Algorithm 2 Solving the minimax problem

1: Draw samples $\{\boldsymbol{x}_i\}_{i=1}^m$ from p_{data}, and samples $\{\boldsymbol{x}_i^*\}_{i=1}^m$ from p_0.
2: **repeat**
3: Update D by maximizing $\frac{1}{m}\sum_{i=1}^m \log D(\boldsymbol{x}_i) + \frac{1}{m}\sum_{i=1}^m \log(1 - D(\boldsymbol{x}_i^*))$ (until converge).
4: For each $\boldsymbol{x} \in \{\boldsymbol{x}_i^*\}_{i=1}^m$, update its value by
$\boldsymbol{x} \leftarrow \boldsymbol{x} + \lambda \frac{\nabla D(\boldsymbol{x})}{\|\nabla D(\boldsymbol{x})\|_2}$ (single step).
5: **until** $\{\boldsymbol{x}_i^*\}_{i=1}^m = \{\boldsymbol{x}_i\}_{i=1}^m$

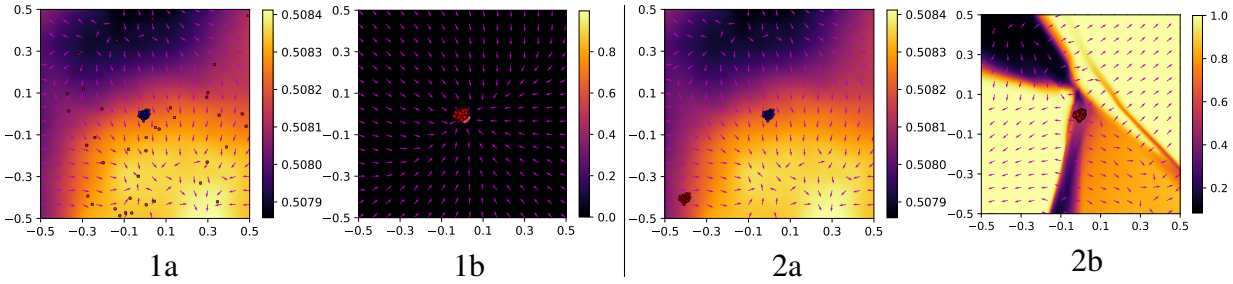


Figure 2.2: Plots of contours and (normalized) gradient vector fields of the D functions learned with different p_0 data. Left and right panel respectively show the initial state (1a and 2a) and final state (1b and 2b) of D when p_0 data is respectively uniformly distributed (red points in 1a) and concerntrated in the lower left corner (red points in 2a). p_{data} is a Gaussian distribution centered at $(0, 0)$ (blue points).

2.4.5 Improved training of AT generative model

Diverse p_0 data

As discussed in Sec. 2.4.2, a diverse p_0 dataset improve the PGD attack's ability to explore different local maxima of D. To validate this 2D intuition generalizes to high dimensions, we evaluate the CIFAR-10 image generation performance under different settings of p_0. It can be seen from Fig. 2.3 that the best FID is obtained when p_0 is the 80 Million Tiny Images dataset [172], the most diverse dataset among the three p_0 datasets.

We follow existing work on adversarial training and use a p_0 dataset that contains real data samples to train the model. Using a real dataset (as opposed to a noise distribution) helps the model achieve out-of-distribution adversarial robustness (Sec. 2.5.3 OOD detection) and learn informative gradient for transforming real out-distribution samples (not just noise samples) into valid samples of p_{data}. The latter can be a useful feature in image translation applications (Sec. 2.5.3). The setting of p_0 in our experiments can be found in Tab. 2.1.

Training with unconstrained perturbations

Existing work on using adversarial training to train robust classifiers uses a small, fixed perturbation limit [112]. In the generative modeling task, we would like the perturbed p_0 data to travel

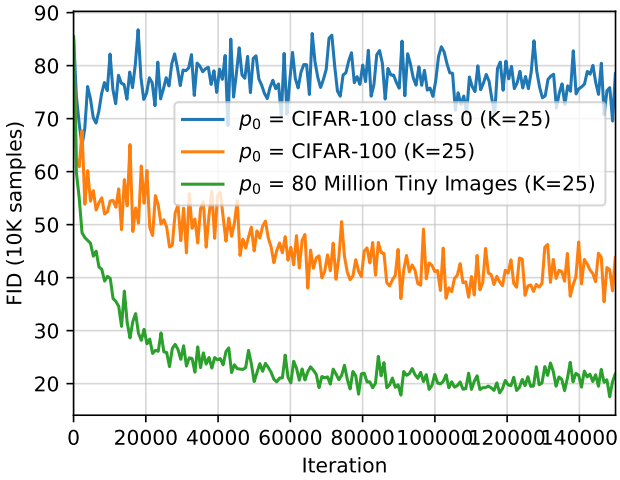


Figure 2.3: FIDs obtained with models trained with different p_0 datasets on CIFAR-10. We use the same source images and PGD attack configuration to do generation in these experiments.

in a larger space to find more local maxima. This can be achieved by taking more PGD attack steps (K) in step 3 of Algorithm 1. Fig. 2.4 shows that a larger K results in better FID scores.

The downside of a large K is that it converges slower because more gradient steps are taken in each iteration (Fig. 2.5 $K = 25$ vs. $K = 5$). To improve the training efficiency we propose a mixed scenario in which we progressively increase the K value during training. We observe that this progressive training scenario converges faster than training with fixed-K (Fig. 2.5 $K = 0, 1, ..., 25$ vs. $K = 25$). The pseudo code for progressive training is in Algorithm 3.

Algorithm 3 Progressive Binary Adversarial Training

1: **for** K in $[0, 1, \ldots, N]$ **do**
2: **for** number of training iterations **do**
3: Draw samples $\{\boldsymbol{x}_i\}_{i=1}^m$ from p_{data}, and samples $\{\boldsymbol{x}_i^0\}_{i=1}^m$ from p_0.
4: Update $\{\boldsymbol{x}_i^0\}_{i=1}^m$ by performing K steps unconstrained PGD attack Eq. (2.8) on each sample. Denote the resulting samples as $\{\boldsymbol{x}_i^*\}_{i=1}^m$.
5: Update D by maximizing $\frac{1}{m}\sum_{i=1}^m \log D(\boldsymbol{x}_i) + \frac{1}{m}\sum_{i=1}^m \log(1 - D(\boldsymbol{x}_i^*))$.
6: **end for**
7: **end for**

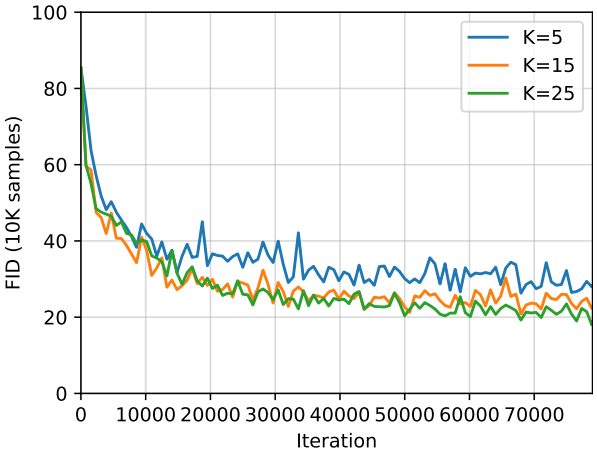


Figure 2.4: FID scores obtained with different Ks in Algorithm 1 on CIFAR-10.

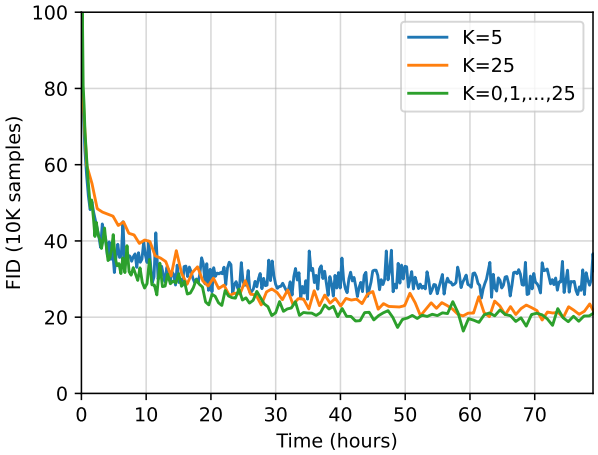


Figure 2.5: Progressive training vs. training with fixed-K on CIFAR-10.

Regularization

While other generative models typically require some forms of regularization, the proposed model can be trained successfully without using any regularization. One trick that we find beneficial for achieving better FID (Fig. 2.6, R_1 reg = 30, pretrained vs. from scratch) is to pretrain the D model on the ImageNet classification task. (This requires adding auxiliary output nodes which are ignored when later training the D model.) When using the pretrained model, we find it necessary to use R_1 regularization [119], otherwise the FID stops improving after a few hundred iterations (Fig. 2.6, R_1 reg = 0 pretrained). Note that when D is trained from scratch, R_1 regularization is not strictly required, but adding the regularizer does not hurt the performance (Fig. 2.6, R_1 reg = 0 from scratch vs. R_1 reg = 30 from scratch).

2.5 Experiments

In this section we provide an empirical evaluation of the proposed AT generative model. We first evaluate the approach's image generation performance, then demonstrate its applications to denoising, inpainting, and image-to-image translation, and finally provide training stability analysis, nearest neighbor and interpolation analysis, and an discussion on sampling efficiency.

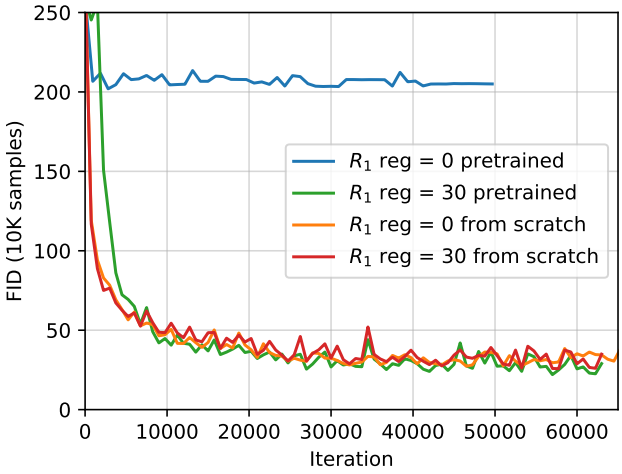


Figure 2.6: The effect of R_1 regularization on CelebA-HQ.

2.5.1 Setups

We evaluate our method on CIFAR-10 [96] (50K training samples), CelebA-HQ 256 [89] (30K training samples), AFHQ-CAT [32] dataset (5153 training samples), and LSUN-Church [208] (126227 training samples). AFHQ [32] is a recently introduced benchmark dataset for image-to-image translation.

On CIFAR-10 we use the standard ResNet50 [70] architecture with ReLU activation for the D model. On CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256 we use a customized architecture (Tab. 2.2) adapted from [32].

We use Algorithm 3 to train the models. The training hyperparameters for each task can be found in Tab. 2.3. For the 256×256 tasks, we pretrain the D model on the ImageNet classification task. To mitigate overfitting, we perform random resized cropping, random horizontal flipping on p_{data} samples. The performance (FID score) of the model is monitored during training and the best-performing model to used to report the final FID score. For CIFAR-10, we use the 80 million tiny images dataset as the p_0 dataset, and for the rest datasets we use the ImageNet p_0.

We use Inception Score (IS) [153] and FID score [77] to evaluate the quality of generated samples. We follow [90] and compute the FID score between 50k generated samples and all training samples (IS is also calculated on the generated 50K samples).

Table 2.2: Network architecture for the D model used in CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256.

Layer	Resample	Output shape
Conv1×1	-	$256 \times 256 \times 64$
ResBlock	AvgPool	$128 \times 128 \times 128$
ResBlock	AvgPool	$64 \times 64 \times 256$
ResBlock	AvgPool	$32 \times 32 \times 512$
ResBlock	AvgPool	$16 \times 16 \times 512$
ResBlock	AvgPool	$8 \times 8 \times 512$
ResBlock	AvgPool	$4 \times 4 \times 512$
LeakyReLU	-	$4 \times 4 \times 512$
Conv4×4	-	$1 \times 1 \times 512$
LeakyReLU	-	$1 \times 1 \times 512$
Reshape	-	512
Linear	-	1

Table 2.3: Training hyperparameters. We use $\beta_1 = 0.0, \beta_2 = 0.99$ for the Adam optimizer.

	CIFAR-10	CelebA-HQ 256	AFHQ-CAT 256	LSUN-Church 256
Batch size	32	40	40	32
Training iterations	172K	218K	225K	215K
Optimizer	Adam	Adam	Adam	Adam
Learning rate	5e-4	5e-5	5e-5	5e-5
K	0,...,25	0,...,40	0,...,25	0,...,35
Epochs per K	5	5	50	1
PGD attack step-size	0.1	2.0	2.0	2.0
R_1 regularization	0.01	30	100	100

2.5.2 Image generation

The generated samples for FID and IS evaluation are produced by performing PGD attacks on 50K samples randomly drawn from the p_0 dataset. The settings for the p_0 dataset and the PGD attack can be found in Table 2.4.

Tab. 2.5 shows that on CIFAR-10 [96] our approach achieves the best Inception Score (IS) [153] and FID [77] among AT generative models. Our approach also improves over state-of-the-art explicit EBMs in terms of IS, and at the same time has a slightly worse FID. Compared to

Table 2.4: Sample generation setting.

Task	p_0 dataset	PGD step size	PGD steps
CIFAR-10	80 million tiny images [172]	0.2	32
CelebA-HQ 256	ImageNet [40]	8.0	20
AFHQ-CAT 256	ImageNet [40]	8.0	14
LSUN-Church 256	ImageNet [40]	8.0	17

VAEBM [193], our method does not require an auxiliary model to train, and has better test time sampling efficiency (Sec. 2.5.5). Diffusion Recovery [53] trains a sequence of conditional EBMs, with each one defining the conditional distribution of a noisy sample given the same sample at a higher noise level. Similar to score-based approaches, these conditional EBMs do not directly model the data distribution of the observed data, so it is unclear how these models can be applied to tasks which require explicit knowledge of the data distribution (e.g., OOD detection). Uncurated CIFAR-10 generation results can be found in Fig. 2.7.

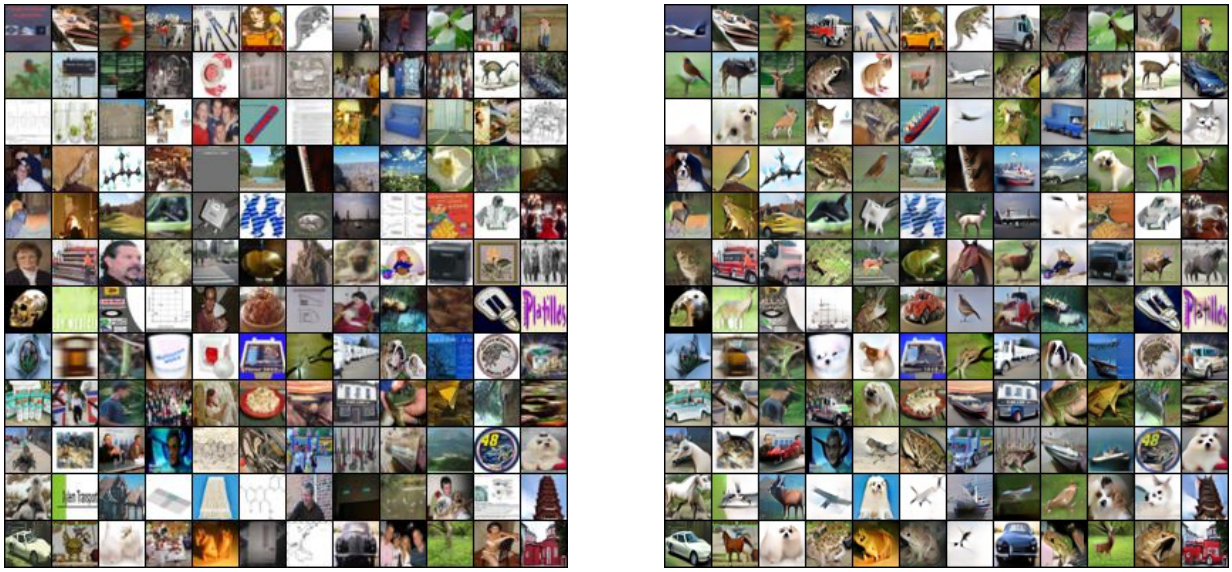
Tab. 2.6 shows that on CelebA-HQ 256 [89] our method outperforms or is on par with state-of-the-art generative models except GANs. On LSUN Church [208] our method outperforms a latest energy-based model VAEBM [193] (the authors only provided the 64×64 result), but falls bellow DDPM and GANs. Fig. 2.8 shows sample image generation results. Fig. 2.9 left panel shows uncurated generation samples on CelebA-HQ 256. We find that some generated images contain significant artifacts. By first applying Gaussian smoothing ($\sigma = 10$) to the source images (p_0 data), we are able to obtain more visually pleasing results (Fig. 2.9 right panel). The generated samples contain less artifacts, but have a slightly worse FID. The smoothing filters out high frequency components, and seems to be playing a similar role as reduced-temperature sampling [178, 193] and the "truncation trick" [19], where better-looking results (typically with reduced diversity) can be generated from latent noise sampled from the high density area of the latent space. Uncurated generation samples on AFHQ-CAT 256 and LSUN-Church 256 can be found in Fig. 2.10.

Table 2.5: IS and FID scores on CIFAR-10.

	Method	IS↑	FID↓
AT	Ours	9.10	13.21
	CEM [182]	8.68	36.4
	Adversarially Robust Classifier [154]	7.5	-
Explicit EBMs	Diffusion Recovery [53]	8.30	9.58
	VAEBM [193]	8.43	12.19
	CoopFlow(pretrained Flow) [203]	-	15.80
	CF-EBM [216]	-	16.71
	ImprovedCD [46]	7.85	25.1
	VERA [62]	-	27.5
	EBMs + VAE [202]	6.65	36.2
	JEM [61]	8.76	38.4
	IGEBM (Ensemble) [45]	6.78	38.2
	Short-Run EBMs [130]	6.21	44.16
GANs	StyleGAN2 w/o ADA [90]	8.99	9.9
	BigGAN [19]	9.22	14.73
	SNGAN [123]	8.22	21.7
	WGAN-GP [65]	7.86	36.4
Score-based	Stochastic Differential Equations [164]	9.89	2.20
	DDPM [78]	9.46	3.17
	NCSNv2 [163]	8.4	10.87
	NCSN [162]	8.87	25.32

Table 2.6: FID scores on CelebA-HQ 256, AFHQ-CAT [32], and LSUN Church 256.

Dataset	Method	FID↓
CelebA-HQ 256	Ours	17.31
	VAEBM [193]	20.38
	CF-EBM [216] (128×128)	23.50
	NVAE [178]	45.11
	GLOW [93]	68.93
	Adversarial Latent Autoencoders [140]	19.21
	ProgressiveGAN [89]	8.03
AFHQ-CAT	Our (256×256)	13.35
	StyleGAN2 (512×512) [90]	5.13
LSUN Church	VAEBM (64×64) [193]	13.51
	Our (64×64)	10.84
	Our (256×256)	14.87
	DDPM (256×256) [78]	7.89
	ProgressiveGAN (256×256) [89]	6.42



Source images Generated images

Figure 2.7: Uncurated CIFAR-10 generated samples.

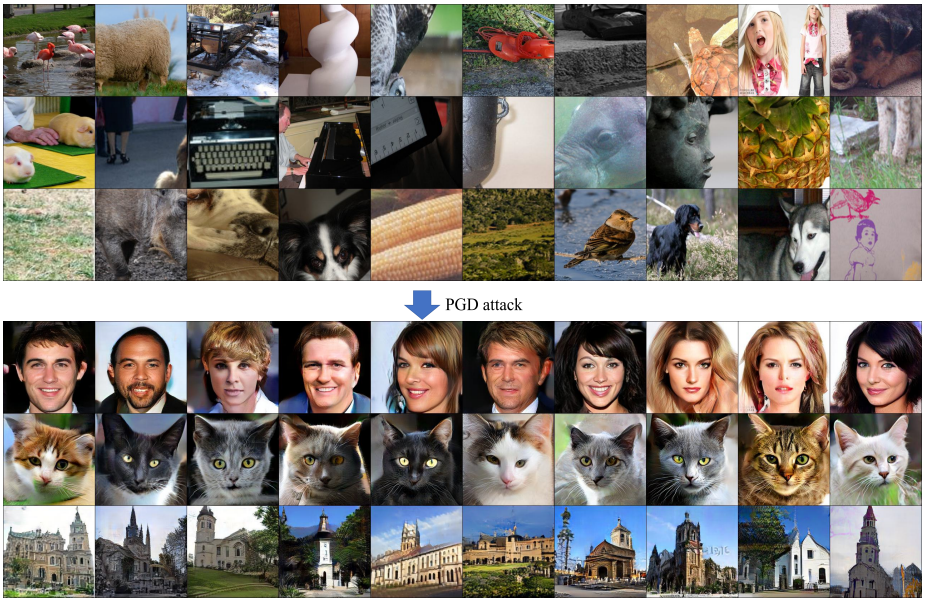


Figure 2.8: Source images (top panel) and generated images (bottom panel, 256×256 resolution) on CelebA-HQ, AFHQ-CAT, and LSUN Church.

Generated samples using original source images

Generated samples using blurred source images

Figure 2.9: Uncurated generation samples on CelebAHQ 256.

Figure 2.10: Uncurated generation samples on AFHQ-CAT 256 and LSUN-Church 256.

2.5.3 Applications to image restoration and image translation

Image-to-image translation

Fig. 2.11 and Fig. 2.12 show that the AFHQ-CAT model can be used to transform CelebA-HQ images into cat images, and vice-versa. Note that these two models are trained independently without knowledge of the source domain, indicating that our approach may generalize better to unseen data than approaches (e.g., pix2pix [81], CycleGAN [220], and StarGAN [32]) that explicitly use the source domain dataset to train the model. The translation results may be further improved by finetuning the trained model on the source domain dataset, or including the source domain data in the p_0 dataset during training. The proposed approach is also more flexible than approaches that employs a fixed generator, as it allows the user to choose how much transformation to apply, and/or create cinematic effect from intermediate transformation results.

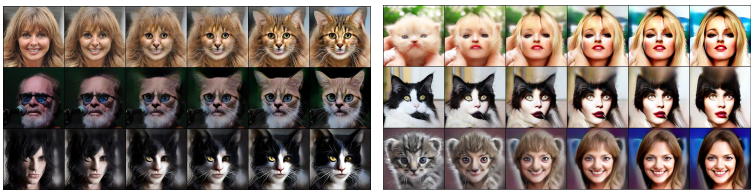


Figure 2.11: Image-to-image translation demonstration.

Figure 2.12: Uncurated image translation results on CelebA-HQ 256 and AFHQ-CAT 256.

Denoising and inpainting

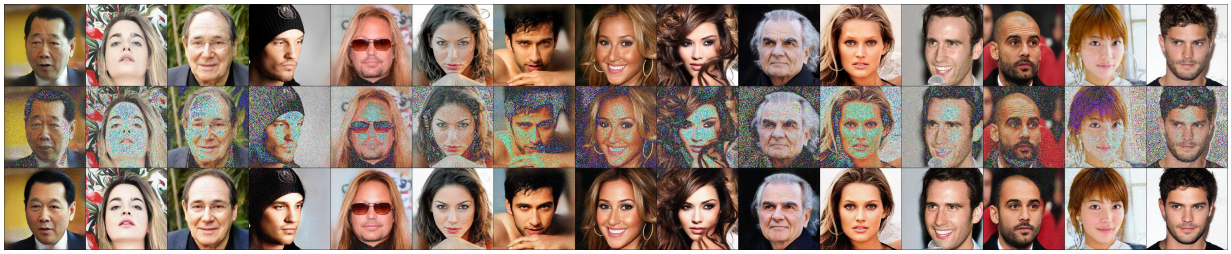
Fig. 2.13 shows the denoising and inpainting results on CelebAHQ 256. To perform denoising or inpainting, we take the noise images or occluded images and then perform gradient ascent on the model. The strength of the denoising or inpainting can be controlled by limiting the number of gradient update steps. Tab. 2.7 shows that the denoised images have significantly better structural similarities to the groundtruth images than the noise images. For inpainting we further consider a nearest neighbor (NN) baseline. As can be seen in Fig. 2.14, in some cases, the patch found by NN is not semantically consistent with other parts of the image. The recovered images by NN search using the original images or occluded images as input are different from the original images. Tab. 2.8 shows that on average, the proposed approach outperforms the NN baseline in terms of SSIM.

Table 2.7: SSIM [184] of the first 10 noise image and denoised images in Fig. 2.13.

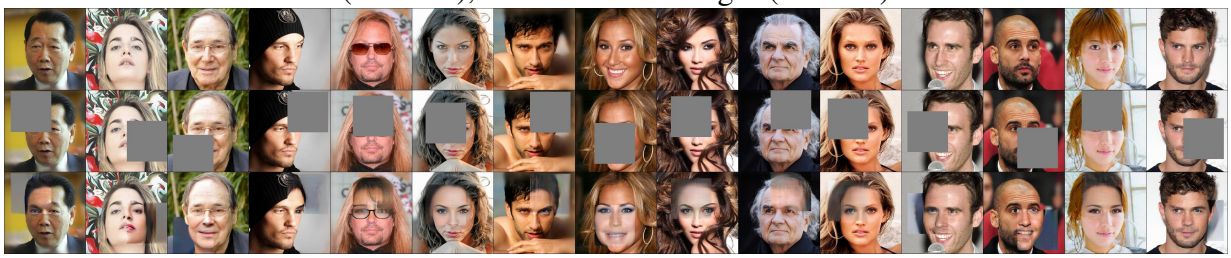
	1	2	3	4	5	6	7	8	9	10
Noise image	0.186	0.342	0.265	0.132	0.259	0.262	0.161	0.244	0.269	0.218
Denoised image	0.301	0.522	0.421	0.297	0.388	0.467	0.349	0.416	0.526	0.378

Table 2.8: SSIM [184] of the first 10 occluded image and recovered images by the proposed model (Fig. 2.13) and the nearest neighbor (NN) baseline (Fig. 2.14).

	1	2	3	4	5	6	7	8	9	10	Average
Occluded	0.846	0.817	0.836	0.881	0.809	0.822	0.769	0.799	0.773	0.817	0.817
Recovered (ours)	0.827	0.815	0.811	0.902	0.829	0.839	0.804	0.832	0.806	0.813	0.828
Recovered (NN-patch)	0.863	0.823	0.867	0.879	0.809	0.857	0.794	0.826	0.807	0.826	0.835
Recovered (NN-original image)	0.391	0.245	0.256	0.433	0.316	0.278	0.339	0.322	0.216	0.316	0.311
Recovered (NN-occluded image)	0.428	0.220	0.249	0.360	0.320	0.285	0.225	0.322	0.160	0.316	0.288



Original images (1st row), images with additive Gaussian noise of standard deviation of 0.1 (2nd row), and recovered images (last row)



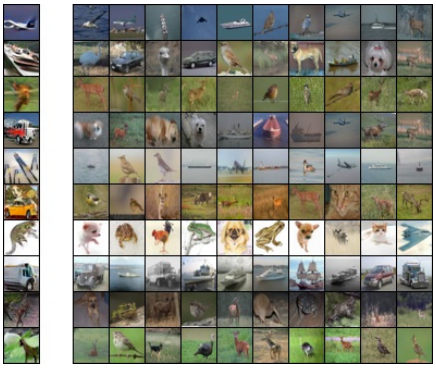
Original image (1st row), occluded images (2nd row), and recovered images (last row)

Figure 2.13: Uncurated denoising and inpainting results on CelebA-HQ 256.

Figure 2.14: Inpainting result obtained with nearest neighbor (NN) search. 1st row: original image, 2nd row: occluded images, 3rd row: recovered images by NN search with the groundtruth patch, 4th row: recovered images by NN search with the original image, 5th row: recovered images by NN search using the occluded image. To perform inpainting with NN using the groundtruth patch, we search the dataset for the patch that has the minimum L_2 distance to the groundtruth patch, and then copy this patch to the occluded region.

2.5.4 Nearest neighbor and interpolation

Fig. 2.15 and Fig. 2.17 show the pixel space and inception feature space nearest neighbors of the generated samples on CIFAR-10, CelebA-HQ 256. Note that none of the nearest neighbors resemble the generated samples, suggesting that the models have not memorized the training data. The interpolation results in Fig. 2.16 show that the models are able to smoothly interpolate between generated samples.



Generated samples (left panel) and pixel space nearest neighbors (right panel)

Generated samples (left panel) and Inception feature space nearest neighbors (right panel)

Figure 2.15: Nearest neighbors of generated samples on CIFAR-10.

CelebA-HQ 256

AFHQ-CAT 256

Figure 2.16: Interpolation results on CelebA-HQ 256 and AFHQ-CAT 256. Intermediate images are generated by performing PGD attacks on linear interpolations between the source images used to generate the leftmost and rightmost samples.

Generated samples (left panel) and pixel space nearest neighbors (right panel

Generated samples (left panel) and inception feature space nearest neighbors (right panel)

Figure 2.17: Nearest neighbors of generated samples on CelebA-HQ 256.

2.5.5 Sampling efficiency

Tab. 2.9 shows that our method has competitive training and test time sampling efficiency to state-of-the-art EBMs. Although VAEBM typically requires much fewer update steps than our method, its per-step efficiency is much worse (Tab. 2.10), suggesting that its VAE component has considerable computational complexity. We also observe that the quality of our generated samples is not sensitive to the number of sampling steps as long as the overall perturbation (#step $\times$ step-size) remains the same (Tab. 2.11). This allows us to use a much larger step size than the one used during training to speedup test time sampling in real applications.

Table 2.9: The number of update steps in the PGD attack (our method) and Langevin dynamics (other methods). "PCD" refers to using a persistent sampling chain.

	Ours	VAEBM [193]	CF-EBM [216]	JEM [61]
CIFAR-10 (train)	25	6 (PCD)	50	20 (PCD)
CIFAR-10 (test)	32	16	50	100
CelebA-HQ 256 (train)	40	6 (PCD)	90	N/A
CelebA-HQ 256 (test)	20	24	90	N/A

Table 2.10: Number of steps and wall-clock time to generate 50 CIFAR-10 samples. Data of NCSN and VAEBM are from [193].

Model	Steps	Wall-clock time	GPU device
NCSN [162]	1000	107.9 seconds	RTX Titan
VAEBM [193]	16	8.79 seconds	RTX Titan
Ours	32	2.34 seconds	RTX 2080 Ti

Table 2.11: FID scores of samples generated using different combinations of number of steps and step-sizes.

	Number of steps $\times$ step-size	FID
CIFAR-10	64×0.1	13.07
	32×0.2	13.21
	16×0.4	13.49
CelebA-HQ 256	40×4.0	19.19
	20×8.0	18.97
	10×16.0	19.19

2.5.6 Training stability analysis

To gain some insight into the training stability of our approach we investigate whether the PGD attack can be used with the EBMs training objective Eq. (2.2). Specifically, in Algorithm 3, we perform step 5's update on θ using the gradient $\nabla_{\boldsymbol{\theta}}(\frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^*))$. We observe that even under a small learning rate of $1e-6$, $\frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^*)$ quickly increases and eventually overflows. This suggests that the stability of the AT approach can be largely attributed to the log-likelihood objective Eq. (2.11). We argue that the stability is due to the gradient cancelling effect of this objective: when $f_{\boldsymbol{\theta}}$ has a large positive output on a sample $\boldsymbol{x} \sim p_{\text{data}}$, $1 - \sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}))$ approaches 0 and therefore the corresponding scaled gradient in Eq. (2.18) vanishes, and similarly $\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}^*))\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}^*)$ vanishes when $f_{\boldsymbol{\theta}}$ has a large negative output on a sample $\boldsymbol{x}^* \sim p_T^*$. In contrast, the EBMs objective Eq. (2.2) does not have constraints on $f_{\boldsymbol{\theta}}$'s outputs and is therefore prone to divergence.

2.6 Conclusion

We have studied an AT-based approach to learning EBMs. Our analysis shows that binary AT learns a special kind of energy function that models the support of the data distribution, and the training procedure can be viewed as an approximate maximum likelihood learning algorithm. We further propose improved techniques for generative modeling with AT, and demonstrate that this new approach is capable of generating diverse and realistic images. Aside from having competitive image generation performance to explicit EBMs, the studied approach has competitive sampling efficiency, is stable to train, and is well-suited for image restoration and image translation tasks. The main drawback of the proposed AT generative is that its generative modeling performance falls behind state-of-the-art generative models such as GANs and diffusion models. To further improve the model, we consider using data augmentation and self-supervised learning to mitigate overfitting. These strategies have been successfully applied in GANs train-

ing [30, 90, 109, 175, 176, 215, 217]. We have also demonstrated the AT generative model's applications to denoising, inpainting, and image translation. In future work, we plan to do a more comprehensive evaluation of the above applications.

Chapter 3

Applications

In this charter we investigate the AT generative model's applications to detecting adversarial examples, detecting out-of-distribution (OOD) samples, and generative classification. For OOD detection and generative classification, we focus on the *adversarial scenario*, where we assume that the OOD inputs or the samples to be classified are adversarially perturbed. Before discussion these applications, we first provide a review on adversarial machine learning, discussing the existence of adversarial examples, approaches to performing adversarial attacks, and various defense mechanisms.

3.1 Adversarial Machine Learning Background

In machine learning, an *adversarial example* is an input sample that is intentionally modified to cause misclassification of a predictive model. An adversarial examples is created by adding *adversarial perturbation* to a naturally-occurring data sample, where the adversarial perturbation is constrained to have a small L_p norm (typically L_2 or L_∞ norm) such that it is imperceptible to the human eye. Both state-of-the-art deep neural network classifiers and other classical approaches such as logistic regression, SVMs, and nearest neighbor classifiers, are vulnerable to adversarial examples [15]. Beside classification models, semantic segmentation and object de-

tection models, and some generative models such as generative adversarial networks (GANs) and variational autoencoders, have been found to be vulnerable to adversarial attacks [95, 168, 195]. Adversarial examples not only exist in digital world; it is possible to print out 2D images or 3D objects and then use them to perform adversarial attack against camera-based object recognition systems [7, 99].

The vulnerability to adversarial examples have become a major security concern for machine learning models. Designing effective defense mechanisms against adversarial attacks has attracted significant interest of the research community [12, 64, 88, 112, 139, 160]. These efforts have in turn motivated the design of stronger attacks that defeat the proposed defenses [8, 24, 28, 57, 71, 99, 100, 192].

In what follows, we first review the high linearity hypothesis [57] which explains the existence of adversarial examples, then discuss methods for computing adversarial examples, and finally discuss how to defend against adversarial inputs.

3.1.1 On the existence of adversarial examples

Although the discovery of adversarial examples for neural networks dates back to 2013 [166], researchers have not reached a consensus on the reasons of why adversarial examples exist [20, 35, 38, 54, 57, 83, 86, 97, 106, 126, 150, 159, 167, 169, 170, 170, 173, 211]. Here we briefly review one of the first and most popular hypotheses — the high linearity hypothesis [57] which explains why adversarial examples exist for deep neural networks in high-dimensional space.

The high linearity hypothesis attributes the cause of adversarial examples to deep neural networks' linear behavior in high-dimensional space [57]. We can first consider a linear function $f : \mathbb{R}^d \to \mathbb{R}$ defined as

$$f(\boldsymbol{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{3.1}$$

where $\boldsymbol{x}, \mathbf{w} \in \mathbb{R}^d$ are respectively the input and weight vector. We can show that after adding a

L_∞ norm-constrained perturbation $\boldsymbol{\delta} \in \mathbb{R}^d$ to the input, the maximum change of the output is

$$\max_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} |f(\boldsymbol{x} + \boldsymbol{\delta}) - f(\boldsymbol{x})| = \mathbf{w}^\top \boldsymbol{\delta}^* = \epsilon \|\mathbf{w}\|_1, \tag{3.2}$$

where the optimal perturbation is $\boldsymbol{\delta}^* = \epsilon \text{sign}(\mathbf{w})$. The linearity hypothesis [57] states that when the input dimension is high (i.e., d is a large number), $\|\mathbf{w}\|_1$ can add up to cause a large change in the output even when ϵ is a smaller number. In the case of a nonlinear function, we can consider f's local behavior around a given input $\boldsymbol{x}_0$ using its first order Taylor expansion

$$f(\boldsymbol{x}) \approx \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)^\top (\boldsymbol{x} - \boldsymbol{x}_0) + f(\boldsymbol{x}_0), \tag{3.3}$$

and similarly it can be shown that the maximal output change is

$$\max_{\boldsymbol{\delta}: \|\boldsymbol{\delta}\|_\infty \leq \epsilon} |f(\boldsymbol{x} + \boldsymbol{\delta}) - f(\boldsymbol{x})| = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)^\top \boldsymbol{\delta}^* = \epsilon \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)\|_1, \tag{3.4}$$

where the optimal perturbation is given by $\boldsymbol{\delta}^* = \epsilon \text{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0))$. Similar to the linear case, in high-dimensional space, $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)\|_1$ can add up to cause a large change in the output. This approach of computing the optimal adversarial perturbation by $\boldsymbol{\delta}^* = \epsilon \text{sign}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0))$ is known as the *fast gradient sign method* (FGSM). We can apply FGSM to attacking a classifier by defining an appropriate loss function as the optimization objective (see Eq. (3.12)). It has been shown that FSGM is effective against neural network classifiers on MNIST and CIFAR-10 [57].

3.1.2 Adversarial attacks

Notation

In a K class problem, let $\mathcal{X} = [0, 1]^d$ be the input space, $\mathcal{Y} = \{1, ..., K\}$ be the label space, $f : \mathcal{X} \to [0, 1]^K$ be the classification function which outputs the predicted probabilities over the class labels. Let $\boldsymbol{x}$ be an input sample within $\mathcal{X}$, and $y \in \mathcal{Y}$ be the corresponding groundtruth

label. Let $F(\boldsymbol{x}) = \arg\max_{i \in \mathcal{Y}} f_i(\boldsymbol{x})$ be the classifier that outputs the predicted label, where f_i is the output of the i-th class.

Attack with minimum perturbation

Early work on adversarial attacks focus on finding adversarial examples that has the minimum distances (as measured by different vector norms) to the original samples. Adversarial attacks can be *targeted* or *untargeted*. In an *targeted attack*, the adversary aims to find a perturbation that causes the input to be classified to a predefined label $t \in \mathcal{Y}, t \neq y$:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p \text{ subject to } F(\boldsymbol{x} + \boldsymbol{\delta}) = t, \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}. \tag{3.5}$$

The *untargeted attack* instead aims to find the minimum perturbation that only causes misclassification:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p \text{ subject to } F(\boldsymbol{x} + \boldsymbol{\delta}) \neq y, \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}. \tag{3.6}$$

For most classification models, f is a highly nonlinear function of the input, so the above problems do not have closed form solutions. Although existing attacks can only give approximate solutions, in practice they are still very effective against state-of-the-art classifiers. In the following we review several representative attacks.

Box-constrained L-BFGS [166] is one of the early methods for performing targeted attacks against deep neural networks. The attack attempt to solve the following problem

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \lambda \|\boldsymbol{\delta}\|_p + L(f(\boldsymbol{x} + \boldsymbol{\delta}), t) \text{ subject to } \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}, \tag{3.7}$$

where the regularization coefficient λ control the relative importance of the two loss terms and its optimal value is found through line-search. The loss function L measures the discrepancy between the model prediction and the target label (e.g., the cross-entropy loss). The attack solves the above problem using the box-constrained L-BFGS algorithm, a second-order method for

solving optimization problems.

DeepFool [125] performs targeted attack. Consider a linear binary classifier $h(\boldsymbol{x}) = \mathbf{w}^\top \boldsymbol{x} + b$, let $\boldsymbol{x}_0$ be a input sample that satisfies $h(\boldsymbol{x}_0) < 0$, then it can be shown that the minimum L_2 perturbation that changes the prediction to $h(\boldsymbol{x}_0 + \boldsymbol{\delta}) \geq 0$ is

$$\boldsymbol{\delta}^* = -\frac{h(\boldsymbol{x}_0)}{\|\mathbf{w}\|_2^2}\mathbf{w}. \tag{3.8}$$

For a nonlinear binary classifier, h can be approximated by its first order Taylor expansion around $\boldsymbol{x}_0$, and the optimal perturbation can be similarly derived as

$$\boldsymbol{\delta}^* = -\frac{h(\boldsymbol{x}_0)}{\|\nabla_{\boldsymbol{x}} h(\boldsymbol{x}_0)\|_2^2}\nabla_{\boldsymbol{x}} h(\boldsymbol{x}_0). \tag{3.9}$$

Due to the linear approximation, $\boldsymbol{\delta}^*$ is not guaranteed to be a valid adversarial perturbation that causes the sign of the output to change. To increase the attack's effectiveness, DeepFool iteratively update the perturbed sample by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{h(\boldsymbol{x}_t)}{\|\nabla_{\boldsymbol{x}} h(\boldsymbol{x}_t)\|_2^2}\nabla_{\boldsymbol{x}} h(\boldsymbol{x}_t). \tag{3.10}$$

The attack can be extended to multiclass problems by considering a multiclass classifier as a set of binary classifiers, with each one separating samples of a particular class from samples not in that class.

CW [24] performs targeted attack by maximizing the difference between the attacked class's logit output and the largest logit output of the rest classes:

$$\begin{aligned} &\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \lambda\|\boldsymbol{\delta}\|_p + (\max_{k \in \mathcal{Y}, k \neq t} z_k(\boldsymbol{x} + \boldsymbol{\delta}) - z_t(\boldsymbol{x} + \boldsymbol{\delta}))^+, \\ &\text{subject to } \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}, \end{aligned} \tag{3.11}$$

where x^+ is the shorthand for $\max(x, 0)$, $z_k(\cdot)$ is the logit output of class k, and λ balances the

weight of the two loss terms. Unlike some attacks that make use of the cross-entropy loss, CW attack's objective is defined in terms of the logit output of the classifier. This makes the attack less susceptible to the vanishing gradient problem.

JSMA [138] aims to find the adversarial examples with the minimum number of modified pixels (i.e., L_0-based attack). The attack uses an iterative algorithm where in each iteration the pixel that has the greatest impact on the output (as measured by the Jacobian) is modified; the iteration stops when a successful attack is found.

The above attack methods all makes use of the gradient of the input to compute the adversarial perturbation. They assume that the parameters of the attacked model are known, and therefore they are known *white-box* attacks. It is also possible to perform *black-box* attacks by only making use of the final output of the model. *Boundary attack* [18] is such an black-box method. Given an input sample $\boldsymbol{x}$ and its label y, the boundary attack first draws an adversarial example $\boldsymbol{x}'$ from a noise distribution. (Here, "adversarial" means that the noise sample is classified as the attacked class in a targeted attack or a class different from y in an untargeted attack.) The attack then gradually reduce the distance between $\boldsymbol{x}'$ and $\boldsymbol{x}$ by adding random perturbations to $\boldsymbol{x}'$, where the random perturbation $\boldsymbol{\delta}$ is configured in such a way that $\boldsymbol{x}' + \boldsymbol{\delta}$ remains an adversarial example and the distance between $\boldsymbol{x}' + \boldsymbol{\delta}$ and $\boldsymbol{x}$ is reduced. The boundary attack is able to reliably find adversarial examples. In certain cases, the adversarial perturbations found by boundary attack are even smaller than perturbations found by some white-box attack methods.

Attack under perturbation constraint

The attacks discussed in the previous section aim to find adversarial examples that have the minimum distance to the original samples. More recently, researchers consider the problem of finding the *optimal* attack that satisfies a perturbation constraint as measured by some vector norm. For instance, in the case of untargeted attack, these attacks attempt to solve

$$\boldsymbol{\delta}^* = \underset{\|\boldsymbol{\delta}\|_p \leq \epsilon}{\arg\max}\, L(f(\boldsymbol{x} + \boldsymbol{\delta}), y). \tag{3.12}$$

FGSM [57] is one approach to solving the above problem under a L_∞ constraint:

$$\boldsymbol{\delta}^* = \epsilon \text{sign}(\nabla_{\boldsymbol{x}} L(f(\boldsymbol{x}), t)), \tag{3.13}$$

where it satisfies $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$. To increase the attack's effectiveness, a multi-step variant of the FGSM algorithm is proposed [99]:

$$\boldsymbol{x}^{t+1} = \text{Proj}(\boldsymbol{x}^t + \alpha \text{sign}(\nabla_{\boldsymbol{x}} L(f(\boldsymbol{x}^t), y))), \tag{3.14}$$

where α is some step-size, and Proj is the operation of projecting $\boldsymbol{x}^t$ on to the set $\mathbb{B}(\boldsymbol{x}^0, \epsilon) = \{\boldsymbol{x} \in \mathcal{X} : \|\boldsymbol{x} - \boldsymbol{x}^0\|_\infty \leq \epsilon\}$, with $\boldsymbol{x}^0$ being the original input. The corresponding L_2-based attack is

$$\boldsymbol{x}^{t+1} = \text{Proj}(\boldsymbol{x}^t + \alpha \frac{\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^t, y)}{\|\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^t, y)\|_2}), \tag{3.15}$$

with $\mathbb{B}(\boldsymbol{x}^0, \epsilon) = \{\boldsymbol{x}^t \in \mathcal{X} : \|\boldsymbol{x}^t - \boldsymbol{x}^0\|_2 \leq \epsilon\}$. This iterative version of FGSM is also known as the PGD attack [112], and it is currently the most effective norm-constrained attack.

3.1.3 Defense mechanisms

Existing approaches to defending against adversarial inputs can be categorized into three groups: 1) *input preprocessing*, where the input is preprocessed so that the "adversarial noise" is filtered out, 2) *robust classification*, where the classifier is hardened so that its prediction becomes invariant to adversarial perturbations of the input, and 3) *adversarial example detection*, where the input is examined by a detector and subsequently rejected if it is deemed an adversarially modified input.

Input preprocessing

Input preprocessing attempts to eliminate the adversarial noise from the input. Input preprocessing is motivated by the observation that the adversarial noise is not changing the semantics of the original clean input, and therefore it must not changing the clean input along the manifold directions. Hence we should be able to eliminate the adversarial noise by projecting the input onto basis vectors which encode the manifold directions and then reconstructing the input. Approaches based on difference basis are developed, including principle component analysis (PCA) [14], JPG and JPEG compression [39, 48], DCT transform [3], bit compression [204], and nonlinear manifold learning models such as denoising Autoencoders [118]. The approach based on denoising Autoencoders has been shown to be vulnerable to adaptive attacks [23].

Robust classification

Robust classification modifies the classifier to make its prediction invariant to adversarial perturbations of the input. Different approaches to robust classification have been developed, such as making use of alternative loss function [136] or activation function [191], changing network architecture or using denoising filters [66, 85, 104, 196], model weight quantization [157], and model regularization [2, 27, 37]. Most of the above approaches are heuristics and some of them have been shown to be vulnerable to adaptive attacks [174].

Two approaches that are widely considered as reliable defenses are *provable defenses* [122, 143, 186] and *adversarial training* [57, 99, 112]. A provably robust classifier's output is strictly invariant to adversarial perturbation of the input, so long as the perturbation is inside the predefined threat model. This property is achieved by minimizing the loss function's upper bound under the specified threat model when training the classifier. Different approaches to computing the upper bound are proposed, but they all suffer from high computational cost. Adversarial training, in contrast, cannot provide a certificate to its performance, but is much faster to train and therefore more practical. Adversarial training also provides much better clean and robust

accuracies than provable defenses.

The basic idea of adversarial training is to use adversarial examples to train the classifier. Given a dataset $\mathcal{D}$ comprising data samples and their groundtruth labels, the adversarially robust classifier is trained by

$$f^* = \arg\min_f \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\max_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon)} L(f(\boldsymbol{x}'), y)], \tag{3.16}$$

where $\mathbb{B}(\boldsymbol{x}, \epsilon)$, also known as the threat model, is a neighborhood of x: $\mathbb{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{x}' \in \mathcal{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon\}$. Intuitively, the inner maximization attempts to find the adversarial examples that would incur the highest classification loss, and the outer minimization trains the classifier by minimizing the classification loss of these "hard samples". The inner minimization is solved using the PGD attack [99, 112] (see Eq. (3.14) and Eq. (3.15)). The outer maximization is solved using gradient-based optimizer such as the Adam optimizer [92]. In each training iteration, the adversarial examples are first computed using the PGD attack, and then the model parameters are updated to minimize the loss on the adversarial examples.

Adversarial training is widely recognized as the most effective and promising approach to robust classification. Since it was firstly introduced [57, 112], multiple variants and enhancements have been developed [5, 11, 29, 42, 44, 51, 58, 84, 103, 108, 111, 115, 129, 137, 161, 179, 181, 183, 187, 189, 190, 194, 205, 212, 214, 218]. By employing various heuristics, such as early stopping [148], using additional unlabeled or generated data [4, 25, 59, 76, 127, 210], data augmentation [145] (combined with model weight averaging [59]), and using high-capacity model [59, 188], the performance (clean and robust accuracies) of adversarially trained classifier has been significantly improved [34]. The main drawback of adversarial training is that adversarially robust classifiers typically have reduced accuracy on clean data, and it is posit that there is an inherent trade-off between the standard accuracy and adversarially robust accuracy of a model [80, 177].

Detecting adversarial examples

To defend the classifier against adversarial inputs, we can also use an auxiliary detector to detect and thereby reject adversarial inputs. The advantage of using a detector is that we do not need to modify the original classifier, but because some clean samples can also be rejected along with adversarial inputs, this approach also affect the performance of the classifier (on clean samples).

There exists a large body of work on detecting adversarial examples [10, 13, 52, 55, 63, 74, 105, 110, 120, 135, 149, 171, 204, 219]. Most of these methods are based on the following core idea: given a trained K-class classifier, $f : \mathbb{R}^d \to \{1, ..., K\}$, and its corresponding clean training samples, $\mathcal{D} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^N$, generate a set of adversarially attacked samples $\mathcal{D}' = \{\boldsymbol{x}'_j \in \mathbb{R}^d\}_{j=1}^M$, and devise a mechanism to discriminate $\mathcal{D}$ from $\mathcal{D}'$. For instance, Gong et al. [55] use this exact idea and learn a binary classifier to distinguish the clean and adversarially perturbed sets. Similarly, Grosse et al. [63] append a new "attacked" class to the classifier, f, and re-train a secured network that classifies clean images, $\boldsymbol{x} \in \mathcal{D}$, into the K classes and all attacked images, $\boldsymbol{x}' \in \mathcal{D}'$, to the $(K+1)$-th class. These methods make use of precomputed adversarial examples and therefore can only handle *non-adaptive* threats, for which the attacks are not specifically tuned/tailored to bypass the detection mechanism, and the attacker is oblivious to the detection mechanism. Indeed, Carlini and Wagner [21] shows that all the above methods are vulnerable to adaptive attacks.

There are also approaches [120] that explicitly consider *adaptive* attacks and propose to harden the detector against such attacks. For example, Metzen et al. [121] propose to attach a low-capacity detection network to the intermediate layer of the main classification network, and then use the intermediate layer output to train the detector to identify adversarial examples. The adversarial examples are generated by considering both the classifier and detector, using a variant of the L_∞ PGD attack (Eq. (3.14)):

$$\boldsymbol{x}_{i+1} = \text{Proj}(\boldsymbol{x}_i + \alpha[(1-\lambda)\text{sign}(\nabla_x L_c(\boldsymbol{x}_i, y)) + \lambda \text{sign}(\nabla_x L_d(\boldsymbol{x}_i, 1))]), \tag{3.17}$$

where L_c and L_d are respectively the classifier loss and detector loss (adversarial examples are labeled as 1s), α is the step-size, and λ is the convex combination coefficient and is randomly sampled from a $[0, 1]$ uniform distribution during training. Because the adversarial examples depend on the detector, they are recomputed in each iteration whenever the parameters of the detector are updated. Although the detector is hardened against adaptive attacks, it still relies non-robust features from the classifier to identify adversarial examples, and it remains unclear whether the adaptive objective Eq. (3.17) is optimal. Carlini and Wagner [21] show that in order to break this defense, they need to use a larger perturbation budget than the one used to break other detection methods.

To evaluate detection-based approaches under adaptive attacks, Carlini and Wagner [21] propose to perform attacks on a new function g constructed by concatenating the logit outputs of the classifier and detector:

$$g(\boldsymbol{x})_i = \begin{cases} z_f(\boldsymbol{x})_i, & \text{if } i \leq K \\ (z_d(\boldsymbol{x}) + 1) \cdot \max_j z_f(\boldsymbol{x})_j, & \text{if } i = K + 1 \end{cases} \tag{3.18}$$

where $z_f(\boldsymbol{x})$ and $z_d(\boldsymbol{x})$ are respectively the logit outputs of the classifier and detector, and K is the number of classes. It is straightforward to see that whenever x is detected as an adversarial example, $z_d(\boldsymbol{x})$ is greater than zero and therefore $g(\boldsymbol{x})_{K+1}$ has the largest output. Hence in order to create an adversarial example that fools the classifier and at the same time evades detection, we can perform adversarial attack on g, and make sure neither the output corresponding to the groundtruth label nor $K + 1$-th output are the largest.

In summary, the majority of the current detection mechanisms focus on *non-adaptive* threats, for which the attacks are not specifically tuned/tailored to bypass the detection mechanism, and the attacker is oblivious to the detection mechanism. These defenses [13, 52, 55, 63, 74, 105, 110, 120] are significantly less effective under adaptive attacks [8, 22].

3.2 Adversarial Training for Detecting Adversarial Examples

The vulnerabilities of deep neural networks against adversarial examples have become a significant concern for deploying these models in sensitive domains such as health care, finances, autonomous driving, and defense-related applications. Devising a definitive defense against such attacks is proven to be challenging, and the methods relying on detecting adversarial samples are only valid when the attacker is oblivious to the detection mechanism. In this section we propose an adversarial example detection mechanism that can withstand adaptive attacks. Inspired by one-versus-the-rest classification, in a K class classification problem, we train K binary classifiers where the i-th binary classifier is used to distinguish between clean data of class i and adversarially perturbed samples of other classes. At test time, we first use a trained classifier to get the predicted label $\hat{k}$ of the input, and then use the $\hat{k}$-th binary classifier to determine whether the input is a clean sample (of class $\hat{k}$) or an adversarially perturbed example (of other classes). We further devise a generative approach to detecting adversarial examples by interpreting each binary classifier as an unnormalized density model of the class-conditional data. We provide comprehensive evaluation of the above adversarial example detection methods, and demonstrate their interpretability and competitive performances.

3.2.1 Proposed approach

The proposed approach to detecting adversarial examples is based on the following simple idea. Assume there is an input sample $\boldsymbol{x}$, and it is predicted as $\hat{k}$ by the classifier f, then $\boldsymbol{x}$ is either a true sample of class $\hat{k}$ (assuming no misclassification) or an adversarially perturbed sample of other classes. To determine which is the case we can use a binary classifier that is specifically trained to distinguish between clean samples of class $\hat{k}$ and adversarially perturbed samples of other classes. Because $\hat{k}$ can be any one of the K class, we need to train a total of K binary classifier in order to have a complete solution. Fig. 3.1 provides a schematic illustration of the above detection idea. We next provide a mathematical justification of this detection approach.

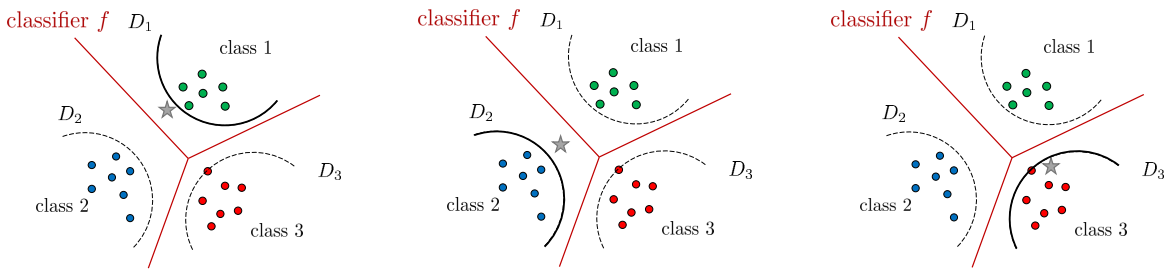


Figure 3.1: A schematic illustration of the proposed method for determining whether an input sample x (represented by the gray star) is an adversarial example. The first figure shows the case where x is predicted by f as class 1 and then x is identified as an adversarial example by D_1. The following two figures shows the other two cases where x is respectively predicted as class 2 and class 3 and then D_2 and D_3 is respectively used to predict whether x is an adversarial example. (Note that there might be some confusion between an outlier and an adversarial example due to space constraints of the 2D drawing. An adversarial example is created by adding adversarial noise to a clean test sample. The adversarial noise changes the decision but does not change the semantic of the clean sample. Therefore in a more realistic setting where the data lie in high dimensional space, the adversarial example would be in the same distribution as the clean samples, even though it is predicted to a different class. But due to space constraints, we are unable to depict such a situation in a 2D drawing.)

In a $K (K \geq 2)$ class classification problem, given a dataset of clean samples $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \in \mathbb{R}^d$, along with labels $\{y_i\}_{i=1}^N, y_i \in \{1, ..., K\}$, let $f : \mathbb{R}^d \to \{1, ..., K\}$ be a classifier on $\mathcal{D}$, and $\mathcal{D}'$ be a set of p-norm bounded adversarial examples computed from $\mathcal{D}$: $\mathcal{D}' = \{\boldsymbol{x}+\boldsymbol{\delta} : f(\boldsymbol{x}+\boldsymbol{\delta}) \neq y, f(\boldsymbol{x}) = y, \boldsymbol{x} \in \mathcal{D}, \boldsymbol{\delta} \in \mathcal{S}\}, \mathcal{S} = \{\boldsymbol{\delta} \in \mathbb{R}^d \,|\, \|\boldsymbol{\delta}\|_p \leq \epsilon\}$. We use $\mathcal{D}_k^f = \{\boldsymbol{x} : f(\boldsymbol{x}) = k, \boldsymbol{x} \in \mathcal{D}\}$ and $\mathcal{D}'^f_k = \{\boldsymbol{x} : f(\boldsymbol{x}) = k, \boldsymbol{x} \in \mathcal{D}'\}$ to respectively denote the clean samples and adversarial examples which are predicted by f as class k. Let $\mathcal{H} = \{d_k\}_{k=1}^K$, where $d_k : \mathbb{R}^d \to [0, 1]$ is a binary classifier trained to distinguish between samples from $\mathcal{D}_k^f$ (assigned as class 1) and samples from $\mathcal{D}'^f_k$ (assigned as class 0). We use $d_k(\boldsymbol{x})$ to model $p(\boldsymbol{x} \in \mathcal{D}_k^f | \boldsymbol{x})$ and predict $\boldsymbol{x} \in \mathcal{D}_k^f$ when $d_k(\boldsymbol{x}) > 0.5$ and $\boldsymbol{x} \in \mathcal{D}'^f_k$ when $d_k(\boldsymbol{x}) \leq 0.5$. Consider the following procedure to determine whether a sample $\boldsymbol{x}$ is an adversarial example (i.e., whether it comes from $\mathcal{D}$ or $\mathcal{D}'$):

> *First obtain the estimated class label $\hat{k} = f(\boldsymbol{x})$, then use the $\hat{k}$-th detector to predict: if $d_{\hat{k}}(\boldsymbol{x}) >= 0.5$ then categorize $\boldsymbol{x}$ as a clean sample, otherwise categorize it as an adversarial example.*

This algorithm can be viewed as a binary classifier, and its accuracy is given by

$$\frac{\sum_{k=1}^{K} |\{\boldsymbol{x} : d_k(\boldsymbol{x}) > 0.5, \boldsymbol{x} \in \mathcal{D}_k^f\}| + |\{\boldsymbol{x} : d_k(\boldsymbol{x}) \leq 0.5, \boldsymbol{x} \in \mathcal{D'}_k^f\}|}{|\mathcal{D}| + |\mathcal{D}'|}. \tag{3.19}$$

Because the errors of individual detectors are independent, maximizing Eq. (3.19) is equivalent to optimizing the performances of individual detectors. d_k solves the binary classification problem of distinguishing between samples from $\mathcal{D}_k^f$ and samples from $\mathcal{D'}_k^f$, and therefore can be trained with a binary classification objective:

$$\boldsymbol{\theta}_k^* = \arg\min_{\boldsymbol{\theta}_k} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D'}_k^f} \Big[L(d_k(\boldsymbol{x}; \boldsymbol{\theta}_k), 0) \Big] + \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_k^f} \Big[L(d_k(\boldsymbol{x}; \boldsymbol{\theta}_k), 1) \Big], \tag{3.20}$$

where L is a loss function that measures the discrepancy between d_k's output and the supplied label (e.g., the negative log likelihood loss). In order to harden d_k against adaptive attacks, we follow [112] and incorporate the adversary into the training objective:

$$\min_{\boldsymbol{\theta}_k} \rho(\boldsymbol{\theta}_k), \quad \text{where} \quad \rho(\boldsymbol{\theta}_k) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\setminus k}^f} \Big[\max_{\boldsymbol{\delta} \in \mathcal{S}, f(\boldsymbol{x}+\boldsymbol{\delta})=k} L(d_k(\boldsymbol{x}+\boldsymbol{\delta}; \boldsymbol{\theta}_k), 0) \Big] + \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_k^f} \Big[L(d_k(\boldsymbol{x}; \boldsymbol{\theta}_k), 1) \Big], \tag{3.21}$$

where $\mathcal{D}_{\setminus k}^f = \{\boldsymbol{x} : f(\boldsymbol{x}) \neq k, y \neq k, \boldsymbol{x} \in \mathcal{D}\}$, and we assume that ϵ is large enough such that $\forall \boldsymbol{x} \in \mathcal{D}_{\setminus k}^f \, , \, \exists \boldsymbol{\delta} \in \mathcal{S} \, , \, s.t. \, f(\boldsymbol{x} + \boldsymbol{\delta}) = k$.

The equality constraint $f(\boldsymbol{x} + \boldsymbol{\delta}) = k$ in Eq. (3.21) complicates the inner maximization. We observe that by dropping this constrain we have the following upper bound of the first loss term:

$$\max_{\boldsymbol{\delta} \in \mathcal{S}, f(\boldsymbol{x}+\boldsymbol{\delta})=k} L(d_k(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}_k), 0) \leq \max_{\boldsymbol{\delta} \in \mathcal{S}} L(d_k(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}_k), 0).$$

Because we are minimizing $L(d_k(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}_k), 0)$, we can instead minimizing this upper bound. This gives us the unconstrained objective

$$\rho(\boldsymbol{\theta}_k) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\setminus k}^f} \Big[\max_{\boldsymbol{\delta} \in \mathcal{S}} L(d_k(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}_k), 0) \Big] + \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_k^f} \Big[L(d_k(\boldsymbol{x}; \boldsymbol{\theta}_k), 1) \Big]. \tag{3.22}$$

We can further simply this objective by using the fact that when $\mathcal{D}$ is the training set, f can overfit on $\mathcal{D}$ such that $\mathcal{D}_{\setminus k} = \{\boldsymbol{x}_i : y_i \neq k\}$ and $\mathcal{D}_k$ are respectively good approximations of $\mathcal{D}^f_{\setminus k}$ and $\mathcal{D}^f_k$:

$$\min_{\boldsymbol{\theta}_k} \rho(\boldsymbol{\theta}_k), \quad \text{where} \quad \rho(\boldsymbol{\theta}_k) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\setminus k}}\Big[\max_{\boldsymbol{\delta}\in\mathcal{S}} L(d_k(\boldsymbol{x}+\boldsymbol{\delta};\boldsymbol{\theta}_k), 0)\Big] + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_k}\Big[L(d_k(\boldsymbol{x};\boldsymbol{\delta}_k), 1)\Big]. \tag{3.23}$$

In words, each detector is trained using clean samples of a particular class and adversarially perturbed samples of other class. The inner maximization is solved using the PGD attack [112]. We use the negative log likelihood loss as L and minimize it using gradient-based optimization.

A generative approach to detecting adversarial examples

The proposed approach to detecting adversarial examples makes use of a trained classifier f to get the predicted label, but f is not strictly necessary: we can use $\mathcal{H} = \{d_k\}_{k=1}^K$ in place of f to perform classification.

We can interpret $\mathcal{H}$ as an *one-versus-the-rest* (OVR) classifier. In a K class classification problem, a OVR classifier consists of K binary classifiers, with each one trained to solve a two-class problem of separating samples in a particular class from samples not in that class. $\mathcal{H}$ differs from a regular OVR classifier in that d_k is trained to distinguish between samples in class k and *adversarially perturbed samples* of other classes, but because the loss on adversarial inputs is an upper bound of the loss on clean samples, the binary classifier should also be able to separate samples of class k from *clean samples* of other classes. When $\mathcal{H}$ is interpreted as an OVR classifier, we can get the prediction by

$$H(\boldsymbol{x}) = \arg\max_k d_k(\boldsymbol{x}). \tag{3.24}$$

We can also interpret $\mathcal{H}$ as a *generative classifier*. Our experiments show that d_k has a strong generative property: performing adversarial attacks on d_k causes high-level features of class k

to appear in the attacked data, and in some cases, the attacked data become valid samples of class k. Although a similar phenomenon is observed in standard adversarial training [50, 154, 177], our model seems to have a much stronger generative property than a softmax adversarially robust classier (Fig. 3.3, Fig. 3.5, and Fig. 3.6). These results motivate us to reinterpret d_k as an unnormalized density model (i.e., an energy-based model [101]) of the class-k data. This interpretation allows us to obtain the class-conditional probability of an input by:

$$p(\boldsymbol{x}|k) = \frac{\exp(-E_{_k}(\boldsymbol{x}))}{Z_k}, \tag{3.25}$$

where $E_{_k}(\boldsymbol{x}) = -z_{d_k}(\boldsymbol{x})$, with z_{d_k} being the logit output of d_k, and

$$Z_k = \int \exp(-E_{_k}(\boldsymbol{x}))dx \tag{3.26}$$

is an intractable normalizing constant known as the partition function. We can then apply the Bayes classification rule to obtain a *generative classifier*:

$$H(\boldsymbol{x}) = \arg\max_k p(k|\boldsymbol{x}) = \arg\max_k \frac{p(\boldsymbol{x}|k)p(k)}{p(\boldsymbol{x})} = \arg\max_k z_{d_k}(\boldsymbol{x}), \tag{3.27}$$

where we have assumed all partition functions $Z_k, k = 1, ..., K$ and class priors $p(k), k = 1, ..., K$ to be equal. Because we explicitly model $p(\boldsymbol{x}, k)$, we can use this quantity to reject low probability inputs which can be any samples that do not belong to class k. In this work we focus on the scenario where low probability inputs are adversarially perturbed samples of other classes and the rejected samples are considered as adversarial examples. Because $d_k(\boldsymbol{x})$ is computed by applying the logistic sigmoid function to $z_{d_k}(\boldsymbol{x})$, and the logistic sigmoid function is a monotonically increasing function of its argument, the generative classifier (Eq. (3.42)) is equivalent to the OVR classifier (Eq. (3.24)).

In the following sections, we will use *integrated detection* to refer to the original detection approach where we make use an extra classifier f, and *generative detection* to refer to this alter-

native approach where we first use the generative classifier Eq. (3.27) to get the predicted label $\hat{k}$ of an input $\boldsymbol{x}$, and then use $d_{\hat{k}}$ to determine whether $\boldsymbol{x}$ is adversarial input.

3.2.2 Evaluation methodology

Binary classifier robustness evaluation

We use AUC (area under the ROC Curve) as the detection performance metric. AUC is an aggregated measurement of detection performance across a range of thresholds, and can be interpreted as the probability that the binary classifier assigns a higher score to a random positive sample than to a random negative example. For a given binary classifier d_k, the AUC is computed on the set in which clean samples from class k are labeled as 1s and adversarially perturbed samples from other classes labeled are as 0s.

Adversarial example detection performance

Having validated the robustness of individual binary classifier, we evaluate the overall performance of the proposed approach to detecting adversarial examples. According to the detection algorithm, we first obtain the predicted label $\hat{k} = f(\boldsymbol{x})$, and then use the $\hat{k}$-th binary classifier's logit output to predict: if $z_{d_{\hat{k}}}(\boldsymbol{x}) \geq T$, then $\boldsymbol{x}$ is a clean sample, otherwise it is an adversarially perturbed sample. For the purpose of this evaluation, we use a universal threshold for all the detectors: $\forall k \in \{1, ..., K\}\, T_k = T$, and report detection performance at a range of different T values. In practice, however, the optimal value of each detector's detection threshold T_k should be determined by optimizing a utility function.

We use $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ to denote the test set that contains clean samples, and $\mathcal{D}' = \{(\boldsymbol{x}_i + \boldsymbol{\delta}_i, y_i)\}_{i=1}^{N}$ to denote the corresponding perturbed test set. For a given T, we compute the true positive rate (TPR) on $\mathcal{D}$ and false positive rate (FPR) on $\mathcal{D}'$ (here, clean samples are in the

positive class). These two metrics are respectively defined as

$$\text{TPR} = \frac{1}{|\mathcal{D}|}|\{\boldsymbol{x} : z_{d_{\hat{k}}}(\boldsymbol{x}) \geq T, k = f(\boldsymbol{x}), (\boldsymbol{x}, y) \in \mathcal{D}\}|, \tag{3.28}$$

and

$$\text{FPR} = \frac{1}{|\mathcal{D}'|}|\{\boldsymbol{x} : z_{d_{\hat{k}}}(\boldsymbol{x}) \geq T, k = f(\boldsymbol{x}), f(\boldsymbol{x}) \neq y, (\boldsymbol{x}, y) \in \mathcal{D}'\}|. \tag{3.29}$$

We observe that for the norm ball constraint we considered in the experiments, not all perturbed samples can cause misclassification on f, so we use $f(\boldsymbol{x}) \neq y$ in the FPR definition to constrain that *only adversarial inputs that actually cause misclassification* are counted as false positives.

Given a clean sample $\boldsymbol{x}$ and its groundtruth label y, we consider three approaches to creating the corresponding adversarial example $\boldsymbol{x}' = \boldsymbol{x}_i + \boldsymbol{\delta}_i$. Here we will focus on untargeted attacks.

Classifier attack. This attack corresponds to the scenario where the adversary is oblivious to the detection mechanism. Inspired by the CW attack [24], the adversarial example $\boldsymbol{x}'$ is computed by minimizing

$$L(\boldsymbol{x}') = z_f(\boldsymbol{x}')_y - \max_{i \neq y} z_f(\boldsymbol{x}')_i, \tag{3.30}$$

where $z_f(\boldsymbol{x}')$ is the classifier's logit outputs.

Detector attack. In this scenario adversarial examples are produced by attacking *only* the detector. We first construct a detection function H by aggregating the logit outputs of individual binary classifiers:

$$z_H(\boldsymbol{x})_i = z_{d_i}(\boldsymbol{x}). \tag{3.31}$$

The adversarial example $\boldsymbol{x}'$ is then computed by minimizing

$$L(\boldsymbol{x}') = -\max_{i \neq y} z_H(\boldsymbol{x}')_i. \tag{3.32}$$

According to our detection rule, a low value of a binary classifier's logit output indicates the detection of an adversarial example, and therefore by minimizing the negative of the logit output

we make the adversarial input harder to detect. H can also be used with the CW loss Eq. (3.30) or the cross-entropy loss, but we find the attack based on Eq. (3.32) to be more effective.

Combined attack. The combined attack is an adaptive method that considers both the classifier and the detector. We consider two loss functions for the combined attack. The first is based on the adaptive attack of [22] which has been shown to be effective against existing detection methods. We first construct a new detection function H with Eq. (3.31) and then use H's largest logit output $\max_{k\neq y} z_H(\boldsymbol{x})_k$ (low value of this quantity indicates detection of an adversarial example) and the classifier logit outputs $z_f(\boldsymbol{x})$ to construct a new classifier g:

$$z_g(\boldsymbol{x})_i = \begin{cases} z_f(\boldsymbol{x})_i & \text{if } i \leq K, \\ (-\max_{j\neq y} z_H(\boldsymbol{x})_j + 1) \cdot \max_j z_f(\boldsymbol{x})_j & \text{if } i = K+1. \end{cases} \tag{3.33}$$

The adversarial example $\boldsymbol{x}'$ is then computed by minimizing the loss function

$$L(\boldsymbol{x}') = \max_i z_g(\boldsymbol{x}')_i - \max_{i\neq y} z_f(\boldsymbol{x}')_i. \tag{3.34}$$

In practice we observe that the optimization of Eq. (3.34) tends to stuck at the point where $\max_{i\neq y} z_f(\boldsymbol{x}')_i$ keeps changing signs while $\max_{j\neq y} z_H(\boldsymbol{x})_j$ staying as a large negative number (which indicates detection). In light of the above issues we derive a more effective attack by combining Eq. (3.30) and Eq. (3.32):

$$L(\boldsymbol{x}') = \begin{cases} z_f(\boldsymbol{x}')_y - \max_{i\neq y} z_f(\boldsymbol{x}')_i & \text{if } z_f(\boldsymbol{x}')_y \geq \max_{i\neq y} z_f(\boldsymbol{x}')_i, \\ -\max_{i\neq y} z_H(\boldsymbol{x}')_i & \text{else.} \end{cases} \tag{3.35}$$

In words, if $\boldsymbol{x}'$ is not yet an adversarial example on f (case 1), optimize it for that goal, otherwise optimize it for evading the detection (case 2).

Adaptive attack for generative detection. We note that the above three attacks are for the original detection approach (i.e., integrated detection). The *generative detection* approach (Sec. 3.2.1)

does not make use of f and we use Eq. (3.32) to create adversarial examples for generative detection.

3.2.3 Experiments

MNIST

Training. We train four detection models (each consists of ten binary classifiers) by using different combinations of p-norm and perturbation limit ϵ (Tab. 3.1). The adversarial examples used for training and validation are computed using PGD attacks of different steps and step-sizes (Tab. 3.1). At each step of the PGD attack we use the Adam optimizer [92] to perform gradient descent, both for L_2-based and L_∞-based attacks. We use 50K samples from the original training set for training and the remaining 10K samples for validation, and report the test performance based on the checkpoint which has the best validation performance. All binary classifiers are trained for 100 epochs, where in each iteration we sample 32 in-class samples as the positive samples, and 32 out-class samples to create adversarial examples as negative samples.

We use the same neural network architecture as in [112] for the binary classifier models. The network consists of two convolutional layers each with 32 and 64 filters, and a fully connected layer with 1024 hidden nodes and 1 output node.

Table 3.1: Training setups for MNIST models.

Training perturbation	Training attack steps, step-size	Validation attack steps, step-size
$L_2, \epsilon = 2.5$	100, 0.1	200, 0.1
$L_2, \epsilon = 5.0$	200, 0.1	200, 0.1
$L_\infty, \epsilon = 0.3$	100, 0.01	200, 0.01
$L_\infty, \epsilon = 0.5$	100, 0.01	200, 0.01

Robustness of binary classifiers. Tab. 3.2 and Tab. 3.3 show that the first two binary classifiers d_1, d_2 are able to withstand PGD attacks with different steps and step-sizes, for both L_2-based and L_∞-based attacks. The binary classifiers also exhibit robustness when the attack uses different

p-norms or perturbation limits than those used for training the model (Tab. 3.4). The models are also robust when the attacks use multiple random restarts (Tab. 3.5).

Table 3.2: AUC scores of the first two binary classifiers d_1, d_2 evaluated under different configurations of the PGD attack. In each step of the PGD attack we use the Adam optimizer to perform gradient descent.

Attack steps, step-size	$L_\infty, \epsilon = 0.3$ model		$L_\infty, \epsilon = 0.5$ model	
	d_1	d_2	d_1	d_2
200, 0.01	0.99959	0.99971	0.99830	0.99869
2000, 0.005	0.99958	0.99971	0.99796	0.99861

Attack steps, step-size	$L_2, \epsilon = 2.5$ model		$L_2, \epsilon = 5.0$ model	
	d_1	d_2	d_1	d_2
200, 0.1	0.99962	0.99968	0.99578	0.99987
2000, 0.05	0.99927	0.99900	0.99529	0.99918

Table 3.3: AUC scores of the first two binary classifiers d_1, d_2 evaluated under different configurations of PGD attacks. In each step of the PGD attack we use normalized gradient [112] (the update rules for L_2-based and L_∞-based attacks are respectively $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \gamma \frac{\nabla f(\boldsymbol{x}^t)}{\|\nabla f(\boldsymbol{x}^t)\|_2}$ and $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \gamma \cdot \text{sign}(\nabla f(\boldsymbol{x}^t))$).

Attack steps, step-size	$L_\infty, \epsilon = 0.3$ model		$L_\infty, \epsilon = 0.5$ model	
	d_1	d_2	d_1	d_2
200, 0.01	0.99962	0.99973	0.99820	0.99901
2000, 0.005	0.99959	0.99971	0.99795	0.99872

Attack steps, step-size	$L_2, \epsilon = 2.5$ model		$L_2, \epsilon = 5.0$ model	
	d_1	d_2	d_1	d_2
200, 0.1	0.99906	0.99916	0.99960	0.99997
2000, 0.05	0.99855	0.99883	0.99237	0.99994

Table 3.4: AUC scores of the first two binary classifiers d_1, d_2 under cross-norm and cross-perturbation attacks. The L_∞-based attack uses steps 200 and step-size 0.01, and The L_2-based attack uses steps 200 and step-size 0.1.

Model	Attack perturbation	Training perturbation			
		$L_\infty, \epsilon = 0.3$	$L_\infty, \epsilon = 0.5$	$L_2, \epsilon = 2.5$	$L_2, \epsilon = 5.0$
d_1	$L_\infty, \epsilon = 0.3$	0.99959	0.99966	0.99927	0.99925
	$L_\infty, \epsilon = 0.5$	0.99436	0.9983	0.99339	0.99767
	$L_2, \epsilon = 2.5$	0.99974	0.99969	0.99962	0.99944
	$L_2, \epsilon = 5.0$	0.96421	0.98816	0.97747	0.99577
d_2	$L_\infty, \epsilon = 0.3$	0.99971	0.99967	0.99949	0.99984
	$L_\infty, \epsilon = 0.5$	0.99778	0.99869	0.99397	0.99961
	$L_2, \epsilon = 2.5$	0.99965	0.99955	0.99968	0.99987
	$L_2, \epsilon = 5.0$	0.98268	0.98687	0.98117	0.99986

Adversarial example detection performance. Figure 3.2 shows the performances of integrated detection and generative detection under different attacks. Combined attack with Eq. (3.35) is the most effective attack against integrated detection, and is much more effective than the combined

Table 3.5: AUC scores of d_1 evaluated under fixed-start and multiple random-restarts attacks. The $L_\infty, \epsilon = 0.5$ model is attacked using $L_\infty, \epsilon = 0.5$ constrained PGD attack of steps 200 and step-size 0.01, and the $L_2, \epsilon = 5.0$ model is attacked using $L_2, \epsilon = 5.0$ constrained PGD attack of steps 200 and step-size 0.1.

	$L_\infty, \epsilon = 0.5$ model	$L_2, \epsilon = 5.0$ model
fixed start	0.99830	0.99578
50 random restarts	0.99776	0.99501

attack with Eq. (3.34). Overall, generative detection outperforms integrated detection when they are evaluated under their respective most effective attack. It is also interesting to see that when the adversarial examples are created by attacking only the classifier (Eq. (3.30)), integrated detection is able to perfectly detect these adversarial examples (see the red curve that overlaps the y-axis).

Given that generative detection is more effective than integrated detection, we compare it with state-of-the-art detection methods [22]. Tab. 3.6 shows that generative detection outperforms the state-of-the-art method by a large margin.

Attack visualization. We can gain some insight into the robustness of the proposed defense by visualizing the adversarial attacks. Since generative detection (Sec. 3.2.1) is our most effective defense, we focus on generating and visualizing attacks against the generative detection. We can generate attacks against the binary classifier d_k to inspect the features d_k has learned. This would be equivalent to performing targeted attacks against the generative classifier Eq. (3.27). Fig. 3.3 shows the clean samples and attacked samples against the generative classifier and the softmax robust classifier [112]. The generative classifier's perturbed samples have visible features of the target class, indicating that individual binary classifiers have learned the high-level features of each classes, and the perturbations have to change the semantics for a successful attack. In contrast, the adversarial perturbations generated by attacking the softmax robust classifier mostly affect the background and are not very interpretable.

Table 3.6: Mean L_2 distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on the perturbed MNIST test set and 0.95 TPR on the clean MNIST test set.

Detection method	Mean L_2 distortion
State-of-the-art [22]	3.68
Ours (with $L_\infty, \epsilon = 0.3$ training perturbation)	4.40
Ours (with $L_\infty, \epsilon = 0.5$ training perturbation)	5.65

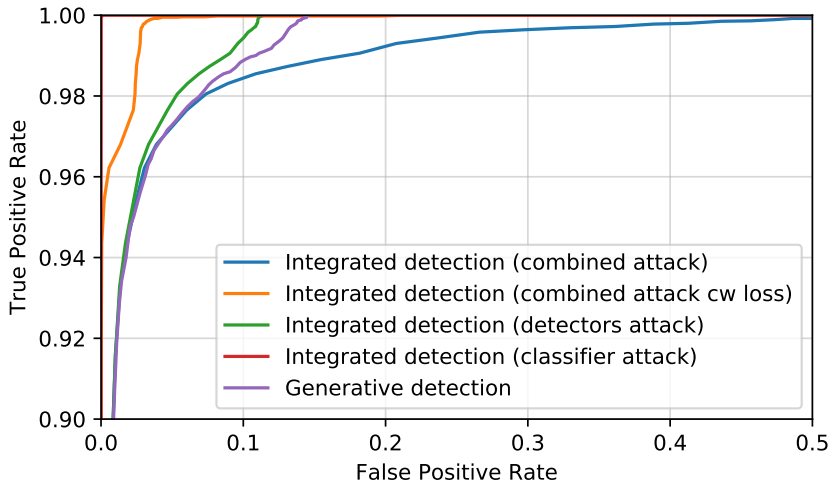


Figure 3.2: Performances of integrated detection and generative detection under $L_\infty, \epsilon = 0.3$ constrained attack.

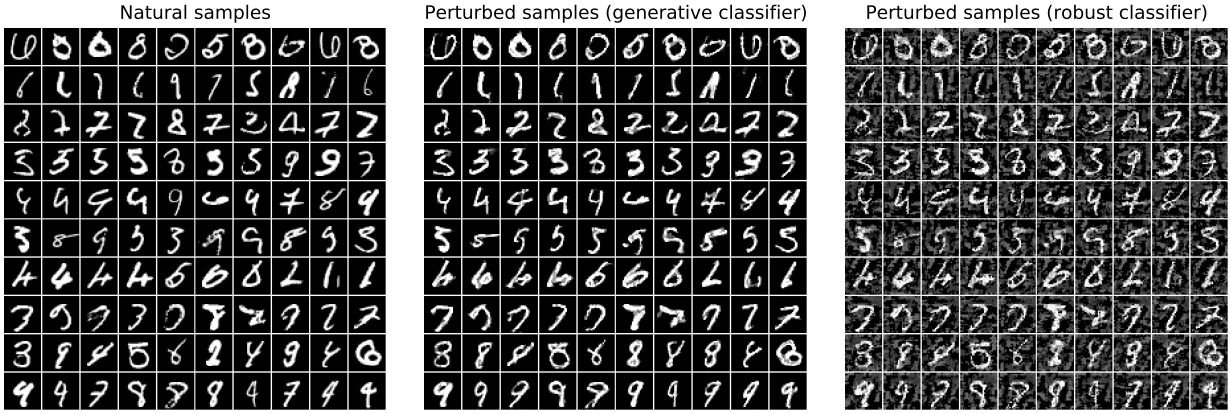


Figure 3.3: Clean samples and corresponding perturbed samples produced by performing targeted attacks against the generative classifier and the softmax robust classifier [112]. Targets from the top row to the bottom row are digit class from 0 to 9. We perform the targeted attack by maximizing the logit output of the targeted class, using $L_\infty, \epsilon = 0.4$ constrained PGD attack of steps 100 and step size 0.01. Both classifiers are trained with $L_\infty, \epsilon = 0.3$ constraint.

CIFAR-10

Training. On CIFAR-10 we train two detection models: one trained with $L_\infty, \epsilon = 8$ constrained PGD attack of steps 40 and step-size 0.5, and the other trained with $L_2, \epsilon = 80$ constrained PGD attack of steps 20 and step-size 10 (note that the scale of ϵ and step-size here is 0-255, as opposed to 0-1 as in MNIST). We train the CIFAR-10 binary classifiers using a ResNet model [112, 113]. To speedup training, we take advantage of pretrained classifier: the subnetwork of f that defines the output logit $z_f(\cdot)_k$ is already a "binary classifier" that would outputs high values for samples of class k, and low values for samples of other classes. The final binary classifier is then trained by finetuning this subnetwork using Eq. (3.23). The pretrained classifier has a test accuracy of 95.01% [113]. At each iteration of training we sample a batch of 300 samples, from which in-class samples are used as positive samples, and an equal number of out-of-class samples are used for creating adversarial examples as negative samples. Adversarial examples for training L_2 and L_∞ models are both optimized using normalized steepest descent based PGD attacks [114].

Robustness of binary classifiers. Tab. 3.7 and Tab. 3.8 show that both the L_∞-based and L_2-based binary classifiers d_1 and d_2 are able to withstand PGD attacks with different steps and step-sizes. Tab. 3.9 shows that the binary classifier d_1 is robust when the attack uses p-norms or perturbation limits that are different than those used for training. The binary classifier is also robust when the attacks use multiple random restarts (Tab. 3.10).

Table 3.7: AUC scores of the first two CIFAR-10 $L_\infty, \epsilon = 8$ binary classifiers d_1, d_2 under $L_\infty, \epsilon = 8$ constrained PGD attacks of different steps and step-sizes.

PGD attack steps, step-size	d_1	d_2
20, 2.0	0.9224	0.9533
40, 0.5	0.9234	0.9553
200, 0.1	0.9231	0.9550
200, 0.5	0.9205	0.9504
500, 0.5	0.9203	0.9500

Adversarial example detection performance. Consistent with the MNIST result, Figure 3.4 shows that combined attack with Eq. (3.35) is the most effective attack against integrated detec-

Table 3.8: AUC scores of the first two CIFAR-10 $L_2, \epsilon = 80$ binary classifiers d_1, d_2 under $L_2, \epsilon = 80$ constrained PGD attacks of different steps and step-sizes.

PGD attack steps, step-size	d_1	d_2
20, 10	0.9839	0.9924
50, 5.0	0.9837	0.9922

Table 3.9: AUC scores of the first CIFAR-10 $L_\infty, \epsilon = 8$ binary classifier d_1 evaluated under PGD attacks of different norms and perturbation limits.

PGD attack	AUC
$L_2, \epsilon = 80$ (steps 20, step-size 10)	0.9814
$L_\infty, \epsilon = 2$ (steps 10, step-size 0.5)	0.9841

Table 3.10: AUC scores of CIFAR-10 d_1 under fixed start and multiple random restarts attacks. The $L_\infty, \epsilon = 2.0$ model is evaluated under PGD attack of steps 10 and step-size 0.5, and the $L_\infty, \epsilon = 8.0$ model is evaluated under PGD attack of steps 40 and step-size 0.5.

	$L_\infty, \epsilon = 2.0$ model	$L_\infty, \epsilon = 8.0$ model
fixed start	0.9866	0.9234
10 random starts	0.9866	0.9233

tion, and generative detection similarly outperforms integrated detection. Table 3.11 shows that generative detection outperforms the state-of-the-art adversarial detection method.

Attack visualization. Same as the MNIST experiment, we visualize adversarial attacks against the proposed defense. Fig. 3.5 shows perturbed samples produced by attacking the generative classifier and the softmax robust classifier [112]. Compared to the softmax robust classifier, the generative classifier's perturbed samples have more visible features of the attacked classes. We also follow [154] and use class-conditional Gaussian noise to perform the attacks. Fig. 3.6 shows that the generated samples of the generative classifier resemble the training samples. In contrast, most of the generated samples of the softmax robust classifier are not recognizable.

Table 3.11: CIFAR-10 mean L_2 distortion (higher is better) of perturbed samples when the detection method has 1.0 FPR on perturbed set and 0.95 TPR on clean set.

Detection method	Mean L_2 distortion (0-1 scale)
State-of-the-art [22]	1.1
Ours (with $L_\infty, \epsilon = 8.0$ training perturbation)	1.5

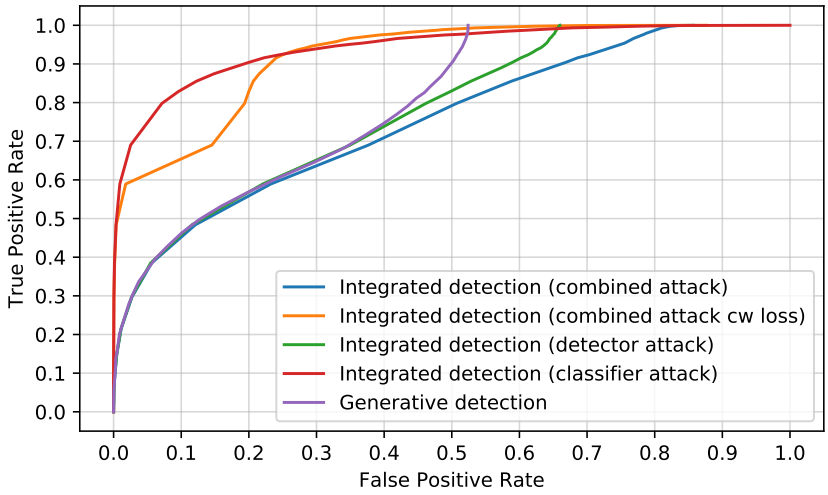


Figure 3.4: Performances of generative detection and integrated detection under $L_\infty, \epsilon = 8$ attack.

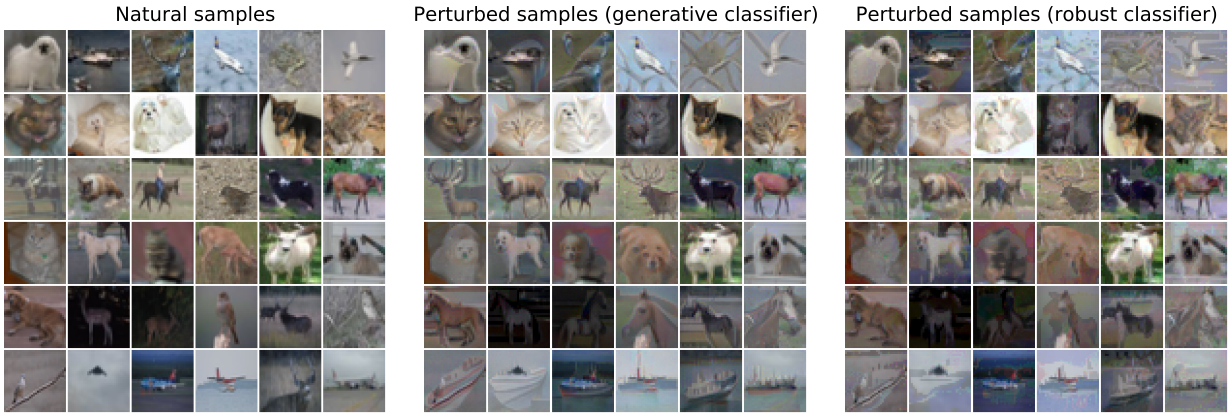
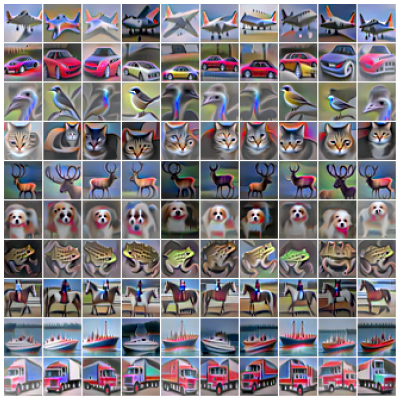


Figure 3.5: Clean samples and corresponding perturbed samples by performing targeted attack against the generative classifier and the robust classifier [112]. The targeted attack is performed by maximizing the logit output of the targeted class. We use $L_\infty, \epsilon = 12$ constrained PGD attack of steps 30 and step-size 2.0 to produce these samples.

(a) Seed images sampled from class-conditional Gaussians



(b) Generated samples of the generative classifier

(c) Generated samples of the softmax robust classifier

Figure 3.6: Seed images and generated images produced by performing targeted adversarial attack on the generative classifier and the softmax robust classier [112]. We use PGD attack of steps 60 and step-size 0.5×255 to perform $L_2, \epsilon = 30 \times 255$ constrained attack (same as in Santurkar et al. [154].

3.2.4 Conclusion

We proposed a principled adversarial example detection method that can withstand adaptive attacks. The idea is to partition the input space into subspaces based on the classifier's decision boundary, train a binary classifier in each subspace to distinguish in-class samples from adversarially perturbed samples of other classes, and then use the binary classifiers to perform clean/adversarial example classification in the subspaces. We provide a comprehensive evaluation of the proposed approach, including a thorough evaluation of the robustness of individual binary classifiers and the overall performance of proposed adversarial example detection approach under different adaptive attacks. Our results show that our approach not only is able to withstand adaptive attacks, but also outperforms existing adversarial example detection approaches by a large margin. Finally, we visualized targeted adversarially attacks against the proposed defense and the attacked data show high-level features of the target classes and this further corroborates the robustness of the proposed defense.

3.3 Worst-Case Out-Of-Distribution Detection

3.3.1 Introduction

Out-of-distribution (OOD) detection, also known as outlier detection, novelty detection, or anomaly detection, deals with the problem of identifying abnormal or unusual observations. Existing approaches to OOD detection can be roughly categorized as density estimation-based approach, reconstruction-based approach, and one-class classification. Density estimation-based approaches first estimate the density function of the in-distribution data, and then use the learned density function to identify OOD inputs which are supposed to lie in low-density areas of the density function. For low-dimensional data, density estimation can be effectively solved using classic approaches such as kernel density estimators (KDE), histogram estimators, and Gaussian mixture models (GMMs). Density estimation in high-dimensional space is much more challenging, and more complex, neural network-based approaches, known as deep generative models, have been developed. Unfortunately, it was observed that certain types of deep generative models (specifically explicit density models), such as Glow [93], PixelCNN [133], PixelCNN++ [152], VAEs [91, 146], and RealNVP flow model [43] tend to assign higher likelihood to OOD inputs than to in-distribution inputs [75, 94, 128, 158]. Reconstruction-based approach aims to learn the manifold structure of the in-distribution data. The model is optimized to well-construct on-manifold data points; a high reconstruction error under the model would then detect off-manifold inputs. Both linear approaches to manifold learning such as PCA and nonlinear approach such as kernel PCA and autoencoders have been adapted for OOD detection. One-class classifier such as Support Vector Data Description (SVDD) aims to learn the smallest hypersphere that encloses the in-distribution data. This can be interpreted as learning a decision boundary that corresponds to a certain density level set of the target distribution. One-class classification is closely connected to binary classification, and can directly incorporates labeled OOD samples to improve the model's performance on OOD detection.

There is also a plethora of OOD detection methods [1, 31, 73, 102, 107, 116, 142, 155] that make use of statistics computed from the predictions or intermediate activations of standard classifiers trained on in-distribution data. For example, Lee et al. [102] fit class conditional Gaussian distributions using intermediate activations of the classifier, and then use Mahalanobis distance to compute confidence scores to identify OOD inputs. Liang et al. [107] improves the effectiveness of a softmax score-based detection approach by using temperature scaling and adding small perturbations to the input. The performance of OOD detection can be significantly improved by making use of large quantities of OOD data to train the model. A notable example is Outlier Exposure (OE) [75], which improves OOD detection by training the model with a large, diverse OOD dataset. OE has been widely adopted as a baseline method for OOD detection.

While some approaches based on generative models and standard classifiers can yield high performances on naturally-occurring OOD inputs, several such methods have been shown [9, 17, 117, 156] to be vulnerable to adversarial manipulation of the OOD inputs. This should come as no surprise as both generative models and standard classifiers themselves are vulnerable to adversarial attacks [95, 141, 166]. Given the above limitation of current approaches, some recent work considers the problem of OOD detection in an adversarial setting [9, 17, 72, 117, 156]. Hein et al. [72] showed that ReLU neural networks produce arbitrarily high confidence predictions far away from the training data, and used adversarial training [112] on out-distribution to enforce low confidence predictions in a neighborhood around OOD inputs. A similar idea is explored by Augustin et al. [9] where the authors also performed adversarial training on in-distribution data. Meinke and Hein [117] uses a density estimator to provide guarantees on the maximal confidence around L_2 ball for uniform noise. Bitterwolf et al. [17] uses interval-bound-propagation (IBP) to certificate worst case guarantees for general OOD inputs under a L_∞ threat model. Methods based on adversarial training on real OOD data [9, 72] are by far the most effective approaches for detecting real adversarial OOD inputs [9, 17].

In this work we apply the AT generative model to worst-case OOD detection. Specifically, following Sec. 2.2.3 and Eq. (2.4), we train the detection model by maximizing the following

objective:

$$J(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_0}[\min_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon)} \log(1 - D(\boldsymbol{x}'))], \tag{3.36}$$

where p_{data} is the normal data distribution, and p_0 is a large and diverse OOD dataset. It can be seen that our approach also performs adversarial training on OOD data, but unlike the above work [9, 17, 72, 117, 156], we focus on training a binary classifier to solve the worst-case OOD detection problem. Compared to the above approaches' multiclass classification objectives which often have some optimization issues [9], our training objective is a simple binary classification objective and is much easier to optimize.

3.3.2 Experiments

Training

We evaluate the proposed OOD detection approach on CIFAR-10 [96], CelebA-HQ 256 [89], AFHQ-CAT 256 [32], and LSUN-Church 256 [208]. The training setups for the later three datasets can be found in Sec. 2.5.1. For CIFAR-10, we use the 80 million tiny images dataset [172] as the p_0 dataset, and train the model with in- and out-distribution adversarial training [9], where in-distribution AT uses a L_2-ball of radius 0.25 and PGD attacks of steps 10 and step-size 0.1, and out-distribution AT uses a L_2-ball of radius 0.5 and PGD attacks of steps 10 and step-size 0.1. Following Augustin et al. [9], we use a batch size of 128 and use the recommended AutoAugment policy from Cubuk et al. [36]. The model is trained for 400 epochs using a SGD optimizer with a fixed learning rate of 0.1.

Results

Tab. 3.12 shows that our model achieves comparable OOD detection performance to the state-of-the-art method of RATIO [9]. We note that OE, RATIO, and JEM are all classifier-based approaches; they perform OOD detection by utilizing a classifier that has low confidence pre-

dictions on the out-distribution data (normal or worst-case). In RATIO, the worst-case out-distribution data is computed by performing PGD attacks on 80 million tiny images [172], whereas in JEM it is computed via Langevin dynamics initialized from uniform random noise. RATIO and our approach's strong worst-case OOD detection performances demonstrates the benefits of using a real and diverse out-distribution dataset to train the model. Our approach does not make use of class labels and therefore can be considered as a unsupervised variant of RATIO.

Tab. 3.13 shows that on CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256 our approach similarly achieves good standard and worst-case OOD detection performances. Although the models are trained with the ImageNet as the OOD dataset, they generalize to unseen OOD datasets even in the worst-case scenario. Note that we have used the same models as in Sec. 2.5.2 for OOD detection evaluation, so the models also have a strong generative property (see Fig. 2.9 and Fig. 2.10). These results suggest that the models have capture the data distributions so that they can be used both for sample generation and for normal and worst-case OOD detection.

Table 3.12: CIFAR-10 standard and worst-case OOD detection results (AUC scores). We use the same settings of AutoAttack [33], number of OOD samples, and perturbation limit as in [9] to compute adversarial OOD samples. Results of OE, JEM, and RATIO are from [9].

OOD dataset	Classifier-based approach			Ours
	OE [75]	JEM [61]	RATIO [9]	
Standard OOD detection				
SVHN	99.4	89.3	96.5	93.5
CIFAR-100	91.4	87.6	91.6	88.7
ImageNet	89.8	86.7	91.3	89.7
Uniform Noise	99.5	11.8	99.9	100
Worst-case OOD detection				
SVHN	0.6	7.3	81.3	83.0
CIFAR-100	2.7	19.2	73.0	70.6
ImageNet	1.5	21.2	73.5	72.5
Uniform Noise	43.1	2.5	99.8	100

Table 3.13: Standard and worst-case OOD detection results (AUC scores) on 256×256 datasets. Adversarial OOD samples are computed by maximizing the model output in a L_2-ball of radius 7.0 (0-1 scale) around OOD samples via Auto-PGD [33] with 100 steps and 5 random restarts. Results are computed using 1024 in-distribution samples and 1024 out-distribution samples.

OOD dataset	In-distribution dataset		
	CelebA-HQ 256	AFHQ-CAT 256	LSUN-Church 256
Standard OOD detection			
Uniform noise	1.0	1.0	0.9476
SVHN	0.9967	0.9944	0.9668
CIFAR-10	0.9978	0.9930	0.9081
ImageNet validation set	0.9986	0.9971	0.9409
AFHQ-CAT 256	0.9984	N/A	0.9691
CelebA-HQ 256	N/A	0.9900	0.9794
LSUN-Church 256	0.9999	0.9997	N/A
Worst-case OOD detection			
Uniform noise	1.0	1.0	0.9330
SVHN	0.9928	0.9880	0.9566
CIFAR-10	0.9952	0.9859	0.8857
ImageNet validation set	0.9973	0.9937	0.9270
AFHQ-CAT 256	0.9958	N/A	0.9587
CelebA-HQ 256	N/A	0.9773	0.9714
LSUN-Church 256	0.9998	0.9991	N/A

3.3.3 Conclusion

In this section we investigate the AT generative model's application to worst-case OOD detection. On CIFAR-10, our approach achieves comparable performance to the state-of-the-art approach of RATIO [9]. Our results on CelebA-HQ 256 and AFHQ-CAT 256 are especially encouraging: the model not only is capable of generating diverse and realistic data samples, but also achieves a good worst-case OOD detection performance on novel OOD datasets. This suggests that the model has correctly captured the data distribution, and demonstrates the benefits of the proposed approach to learning energy-based model by contrasting in-distribution data with adversarially perturbed real OOD data. In terms of applications, the proposed model can be used both for content creation and for detecting adversarially created fake content.

3.4 Generative Robust Classification

In this section we apply the AT generative model introduced in Chapter 2 to generative classification. We first review the general idea behind discriminative classification and generative classification, and then move on to the proposed generative robust classifier. In the experiment section we compare the generative classifier with the discriminative classifier in terms of standard accuracy, robust accuracy, and interpretability. We then provide ablation on how model capacity, weight decay regularization, training perturbation size, data augmentation, and combining in- and out-distribution adversarial training affect the performance of the binary classifiers. We conclude

3.4.1 Introduction

Discriminative classification and *generative classification* are two different approaches to solving classification problems. In discriminative classification, we directly model the posterior distribution of the class labels $p(k|\boldsymbol{x})$. This posterior distribution is typically obtained by applying the *softmax* function to the logit outputs of the classifier:

$$p(k|\boldsymbol{x}) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}, \tag{3.37}$$

where a_k is the model's k-th output. Because the discriminative classifier makes use of the softmax function, it is also known as *softmax classifier*. The discriminative classifier is in fact modeling the decision boundary between different classes. An alternative approach is to first model the class-conditional distributions $p(\boldsymbol{x}|k)$ and then obtain the posterior distribution using Bayes' theorem

$$p(k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|k)p(k)}{p(\boldsymbol{x})}, \tag{3.38}$$

where the prior distribution $p(k)$ can be estimated from the fractions of the training set data points in each of the classes. This is the generative approach to solving classification problems. It seems

that in order to solve the classification problem, the generative classifier needs to solve a more complex problem of modeling the class-conditional distributions. Indeed, the class-conditional distribution may contain a lot of structure that has little effect on $p(k|\boldsymbol{x})$ [16]. But directly modeling $p(\boldsymbol{x}|k)$ can be advantageous when we want to use $p(\boldsymbol{x}|k)$ to detect out-of-distribution inputs.

Deep neural network-based discriminative classifier is the dominant approach to solving classification problems. A widely known issue of this approach is the existence of adversarial examples. Various defense mechanisms have been proposed to address adversarial examples, with *Adversarial training (AT)* being the most successful one. AT improves the classifier's robustness by training the classifier against adversarially perturbed inputs. One issue of AT is that adversarially robust classifiers tend to have a lower accuracy on clean data than standard, non-robust classifiers. Tsipras et al. [177] conjecture that the reduced accuracy on clean data is the consequence of robust classifiers learning fundamentally different features than standard classifiers. Recent work on AT has been focus on using data augmentation (combined with model weight averaging [82]) [145], synthetic training data produced by generative models [60, 145], and high capacity models [4, 59, 194] to improve the standard and robust accuracies. Discriminative classifier also has the issue of producing overconfident predictions on out-of-distribution inputs. Augustin et al. [9] propose to address this problem by combining in- and out-distribution adversarial training to enforce low confidence on adversarial out-of-distribution samples.

Compared to discriminative robust classification, generative robust classification is a less-explored area. Generative robust classification requires the density model $p_{\boldsymbol{\theta}}(\boldsymbol{x}|k)$ to have low likelihood outputs on regular as well as adversarial out-of-distribution samples, which is not achievable with existing density models. In this work we explore modeling $p(\boldsymbol{x}|k)$ with the proposed AT generative model, and then using these conditional models to perform generative robust classification. We demonstrate that this generative robust classifier achieves comparable standard and robust classification accuracies to a state-of-the-art softmax robust classifier, and at the same time is more interpretable. One issue with the proposed generative classifier is that it

does not easily scale to problems with many classes. However, we note there are many practical problems with limited number of classes [47], and our approach may find applications in this kind of problems.

We note that generative classification is also investigated in our previous work [206]. In terms of the formulation of the generative classifier, both [206] and this work use AT to train a binary classifier to model the unnormalized density function of the class-conditional data. The key difference is that this work does not assume that the normalizing constants of the density functions of different classes to be equal, and instead treat the normalizing constants as learnable parameters of the generative classifier. In terms of implementation, our previous work's generative classifier is ten times larger than the softmax robust classifier [112], and yet its performance is much worse [174]. In summary, our specific contributions in this work are:

- We show that when the overall model capacities are similar, the generative classifier achieves comparable standard and robust accuracies to a baseline softmax robust classifier. In particular, the generative classifier is much more robust when the test perturbation is large.
- We study the interpretability of the generative classifier and show that the generative classifier is more interpretable in terms of the qualities of generated samples and counterfactuals.
- We propose to combine in- and out-distribution adversarial training to improve the model's robustness.
- We perform comprehensive ablation to study how calibration, model capacity, weight decay regularization, training perturbation size, data augmentation, and combining in- and out-distribution AT affect the model performance.

3.4.2 Generative robust classification

In a K class classification problem, the proposed generative classifier consists of K binary classifiers, with the k-th binary classifier trained to distinguish clean data of class k from adversarially perturbed data of other classes. Following Yin et al. [207], denote the data distribution of class k

as p_k, the out-of-distribution dataset as p_0, then the k-th binary classifier $D_k : \mathcal{X} \subseteq \mathbb{R}^d \to [0, 1]$ is trained by maximizing the following objective:

$$J(D_k) = \mathbb{E}_{\boldsymbol{x}\sim p_k}[\log D_k(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_0}[\min_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon)} \log(1 - D_k(\boldsymbol{x}'))], \tag{3.39}$$

where $\mathbb{B}(\boldsymbol{x}, \epsilon)$ is a neighborhood of $\boldsymbol{x}$: $\mathbb{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{x}' \in \mathcal{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_2 \leq \epsilon\}$. D_k is defined as $D_k(\boldsymbol{x}) = \sigma(d_k(\boldsymbol{x}))$, where σ is the logistic sigmoid function, and $d_k : \mathbb{R}^d \to \mathbb{R}$ is a neural network with a single output node.

As is discussed in Yin et al. [207], training with a larger perturbation ϵ causes the model to learn to generate more realistic and diverse samples of p_k. Unfortunately, there is typically a trade-off between D_k's discriminative capability and its generative capability. In this work we focus on optimizing the model's (standard and robust) discriminative performance by choosing an approximate ϵ to train the model. Yin et al. [207] also suggests that the p_0 should be diverse in order for D_k to better capture p_k. However, in order to have a fair comparison with the discriminative classifier which does not use additional data, we use the the mixture distribution of other classes as the out-of-distribution dataset: $p_0 = p_{\setminus k} = \frac{1}{K-1}\sum_{i=1,\ldots,K,i\neq k} p_i$.

Following Yin et al. [207], we interpret $d_k(\boldsymbol{x})$ as an unnormalized density model of $p(\boldsymbol{x}|k)$. We can obtain the normalized density function by

$$p(\boldsymbol{x}|k) = \frac{\exp(d_k(\boldsymbol{x}))}{Z_k}, \tag{3.40}$$

with Z_k being the partition function:

$$Z_k = \int \exp(d_k(\boldsymbol{x}))d\boldsymbol{x}. \tag{3.41}$$

We can then apply the Bayes classification rule to obtain a *generative classifier* $g(\boldsymbol{x}) : \mathbb{R}^d \to$

$\{1, ..., K\}$:

$$\begin{aligned} g(\boldsymbol{x}) &= \arg\max_k p(k|\boldsymbol{x}) \\ &= \arg\max_k \frac{p(\boldsymbol{x}|k)p(k)}{p(\boldsymbol{x})} \\ &= \arg\max_k \frac{\exp(d_k(\boldsymbol{x}))p(k)}{Z_k}, \end{aligned} \tag{3.42}$$

Calibration

In general, Z_k is intractable, and Z_k of different classes are not equal, so $g(\boldsymbol{x})$ is also intractable. While Z_k cannot be computed directly, we can treat Z_k as learnable parameters of the generative classifier, and obtain their values by optimizing the generative classifier's performance on a validation set. Absorbing Z_k into the $\exp$ and assuming that $p(k)$ of different classes are equal, the generative classifier can be simplified as

$$g(\boldsymbol{x}) = \arg\max_k d_k(\boldsymbol{x}) + c_k, \tag{3.43}$$

where the calibration constants $\{c_1, ..., c_K\}$ can be learned on a validation set.

Combining in- and out-distribution adversarial training

The robustness of the generative classifier can be further improved by combining in- and out-distribution adversarial training. Consider an adversarially perturbed sample $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$, where $\boldsymbol{x}$ is a clean sample of class k, and $\boldsymbol{\delta}$ is an adversarial perturbation. In order to correctly classify $\boldsymbol{x}'$, the generative classifier Eq. (3.43) needs satisfy

$$\forall i \in \{1, ..., K\} \backslash \{k\} \quad d_k(\boldsymbol{x} + \boldsymbol{\delta}) + c_k > d_i(\boldsymbol{x} + \boldsymbol{\delta}) + c_i. \tag{3.44}$$

However, in Eq. (3.39), d_k is only trained to have high outputs on clean samples of p_k, not perturbed samples of p_k. We can increase d_k's outputs on perturbed samples of p_k by also performing

in-distribution adversarial training:

$$J(D_k) = \mathbb{E}_{\mathrm{x}\sim p_k}[\min_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon_1)} \log D_k(\boldsymbol{x}')] + \mathbb{E}_{\mathrm{x}\sim p_{\setminus k}}[\min_{\boldsymbol{x}'\in\mathbb{B}(\boldsymbol{x},\epsilon_2)} \log(1 - D_k(\boldsymbol{x}'))]. \tag{3.45}$$

By explicitly training d_k to have high outputs on *adversarially perturbed samples* of class k, we make Eq. (3.44) easier to satisfy. In Sec. 3.4.4 we provide an ablation on combining in- and out-distribution adversarial training.

3.4.3 Experiments

Training setup

We evaluate the generative classifier on the CIFAR-10 dataset, a widely used benchmark for robust classification [34]. The proposed generative classifier consists of K binary classifiers in a K class classification problem. In order to have a fair comparison with the discriminative approach which typically uses a single model, we limit the capacity of the binary classifier so that the overall capacities of these two approaches are similar. Specifically, we use a customized model "ResNet18Thinner" (see also in Fig. 3.12) to train the binary classifiers. The ResNet18Thinner model is an ResNet18 architecture with a width multiplier of 0.5, and has 10M parameters. The baseline discriminative robust classifier [49] is based on the ResNet50 architecture which has 90MB parameters.

All the binary classifier are trained with the SGD optimizer using a batch size of 128 and weight decay of 10^{-4} for 2000 epochs. For classes 2, 3, 4, 5, 7, 8, 9 we use a starting learning rate of 0.1, and for classes 0, 1, 6 we find the 0.1 learning rate being too high and instead use a starting learning rate of 0.05. The learning rate is reduced to 0.01 after epoch 1500. The training perturbation size is 0.3 (L_2 norm), in contrast to the perturbation size of 0.5 used in training the softmax robust classifier. Following the common practice in the adversarial machine learning literature [148], we use early stopping on the CIFAR-10 test set to select the model.

Evaluation

We evaluate the generative classifier and softmax robust classifier on the clean test set (clean accuracy) and the adversarially perturbed test set (robust accuracy). The adversarial perturbations are computed by performing untargeted adversarial attacks against the classifiers. Given a test sample $\boldsymbol{x} \in \mathbb{R}^d$ and its label y, for the softmax robust classifier f, the adversarial perturbation $\boldsymbol{\delta}^*$ is computed by solving

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta} \in \mathbb{R}^d, \|\boldsymbol{\delta}\|_p \leq \epsilon}{\arg\max} \; L(f(\boldsymbol{x} + \boldsymbol{\delta}), y), \tag{3.46}$$

where L is the cross-entropy loss. We solve this optimization using the PGD attack [112]. For the generative classifier, it has been shown that directly optimizing the cross-entropy loss is suboptimal [174], so we instead use the attack proposed by Tramer et al. [174]. The attack first computes

$$\boldsymbol{\delta}_k^* = \underset{\boldsymbol{\delta} \in \mathbb{R}^d, \|\boldsymbol{\delta}\|_p \leq \epsilon}{\arg\max} \; d_k(\boldsymbol{x} + \boldsymbol{\delta}) \tag{3.47}$$

for $k = 1, ..., K, k \neq y$ and then computes $\boldsymbol{\delta}^*$ by

$$\boldsymbol{\delta}^* = \underset{k=1,...,K, k \neq y}{\arg\max} \; d_k(\boldsymbol{x} + \boldsymbol{\delta}_k^*). \tag{3.48}$$

Standard accuracy and robust accuracy

Tab. 3.14 shows that the generative classifier has lower standard accuracy but higher robust accuracy compared to the softmax robust classifier. We also evaluate these two classifiers using the test set perturbed with different levels of adversarial noise. Fig. 3.7 shows that when there is no perturbation or when the test perturbation is small, the softmax robust classifier has better performance, and when the test perturbation is of moderate size or large size, the generative classifier outperforms the softmax robust classifier, and the larger the test perturbation is, the larger the performance gap is.

In adversarial machine learning, it has been observed that there is a trade-off between stan-

dard accuracy and robust accuracy [177] — training with adversarially perturbed data allows the classifier to learn more robust and semantically meaningful features to achieve improved robust accuracy at the expense of the standard accuracy. Engstrom et al. [49] further show that larger training perturbations cause the model to perform better under large test perturbations but worse under small or no test perturbations. As Fig. 3.13 suggests, this trade-off between standard performance and robust performance also exists when we use Eq. (3.39) to train the binary classifiers. Note that the training perturbation of the generative classifier is already much smaller than that of the softmax robust classifier (0.3 vs. 0.5). Based on these results, we conjecture that the generative classifier is able to learn more semantically meaningful features by capturing the class-conditional distributions and therefore has a better robust accuracy. Indeed, Tab. 3.15, Fig. 3.8, and Fig. 3.9 show that the generated samples of the generative classifier have better quality than that of the softmax robust classifier, suggesting that the generative classifier has a better captured the class-conditional distributions.

Table 3.14: CIFAR10 standard accuracy and robust accuracy ($\epsilon_{\text{test}} = 0.5$, L_2 norm).

Model	Standard accuracy	Robust accuracy
Generative classifier	88.12%	72.27%
Softmax robust classifier[49]	90.83%	70.17%

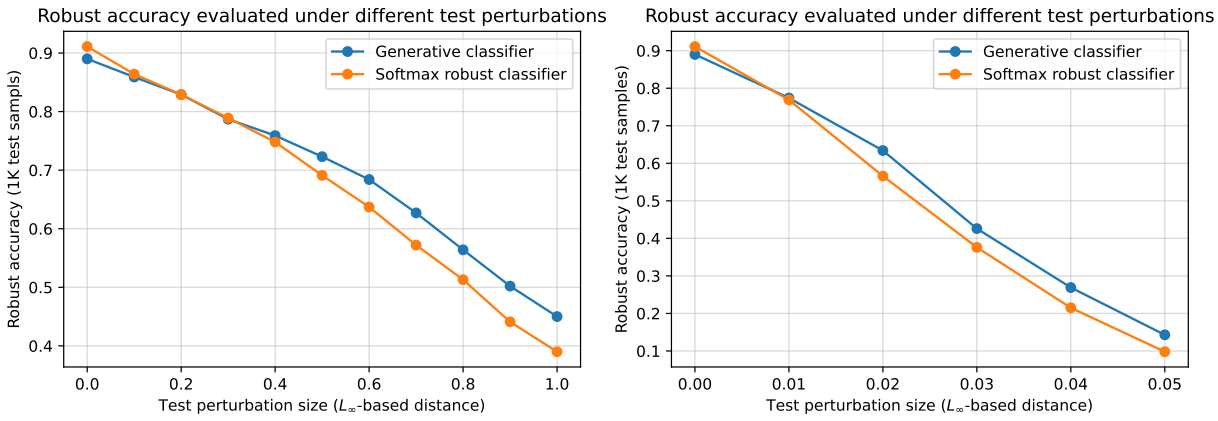


Figure 3.7: The performance of the generative classifier and the softmax robust classifier evaluated under different test perturbation sizes.

Interpretability

We also study the interpretability the generative classifier. We use three approaches to evaluate the classifier's interpretability. First, we visualize the "learned concepts" of the classifier by generating synthetic samples that cause high-confidence predictions. This allows us to understand how the model makes prediction in general and can be considered as a *global interpretability* method [124]. Second, we use integrated gradient (IG) [165] to visualize how the input features affect the prediction. This helps us understand why did the model make a certain prediction for an instance and can be considered as a *local interpretability* method [124]. And lastly, we consider *counterfactual explanation* [180] which is also used by previous work [9] to evaluate the interpretability of adversarially robust classifiers. To access the quality of generated samples (or counterfactuals), we compute the FID [77] between the training samples and the generated samples. FID is a distance metric that measures the similarity between two sets of data samples, and is widely used for evaluating the quality of images produced by generative models.

Generated samples. To generate samples that cause high-confidence predictions, we follow Santurkar et al. [154] and fit a multivariate normal distribution to the class-conditional data and then generate seed images by sampling from the this normal distribution. We then generate samples by performing targeted PGD attacks against the model with the seed images. We use a large perturbation limit so that the generated samples cause high-confidence predictions. Fig. 3.8 shows the seed images and generated images of the generative classifier and the robust discriminative classifier. In both cases, the generated samples resemble the class-conditional data, suggesting the classifiers have captured the high-level features of the target classes. Qualitatively, generated samples of the generative classifier have less artifacts and the foreground objects are more recognizable. To provide a quantitative assessment, we compute the FID of the generated samples. Tab. 3.15 shows that the generated samples of the generative classifier has lower FID than that of the softmax robust classifier under different configurations of the PGD attack, suggesting that the generated samples of the generative classifier more closely resemble the training data. We

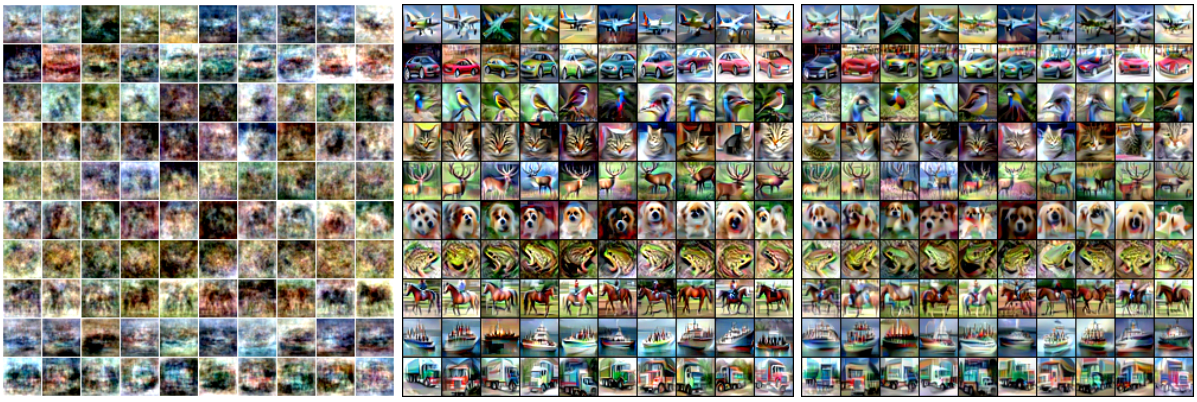
note that samples produced with different PGD attacks have different FIDs, but the samples generated by the generative classifier have consistently lower FIDs than that of the softmax robust classifier.

Counterfactuals. The idea of counterfactual explanation is to find the minimum change to an input sample such that the modified sample is classified to a predefined target class. To generate counterfactuals we take input samples and then perform targeted adversarial attacks against the classifier. For standard, non-robust models, this process would lead to adversarial examples [57] which do not have visually meaningful changes. If on the other hand the classifier has captured the class-specific features, then the classifier's decisions change only when class-specific features appear in the attacked data. Fig. 3.9 shows both the generative classifier and softmax robust classifier are able to generate plausible counterfactual explanations. The counterfactuals generated with the two approaches are qualitative similar in terms of the emergence of class specific features. We again use FID as a quantitative measure and Tab. 3.15 shows that the counterfactuals of the generative classifier has better quality in terms of the FID score.

Table 3.15: FID of generated samples and counterfactuals of the generative classifier and softmax robust classifier under different PGD attacks.

	Generated samples	Counterfactuals
L_2 PGD attack of steps 7 and step-size 1.0		
Generative classifier	50.38	37.92
Softmax robust classifier [49]	61.84	43.65
L_2 PGD attack of steps 10 and step-size 1.0		
Generative classifier	54.78	42.33
Softmax robust classifier [49]	66.15	50.85

Integrated gradient interpretability. Fig. 3.10 shows the integrated gradient (IG) masks generated with different classifiers. The IG mask highlights the features that affect the prediction. The highlighted regions of the IG masks of the generative classifier and softmax robust classifier align with the foreground objects in the input images. The highlighted regions of the IG masks



(a) Seed images (b) Generated samples (generative classifier) (c) Generated samples (softmax robust classifier)

Figure 3.8: Seed images and generated images generated by performing targeted adversarial attack on the generative classifier and the robust discriminative classifier. The PGD attack is L_2-based attack of steps 10 and step-size 1.0.

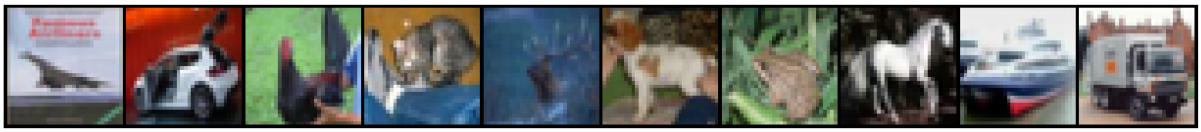
(a) Seed images (b) Counterfactuals for the generative classifier (c) Counterfactuals for the softmax robust classifier

Figure 3.9: Seed images and counterfactuals generated by performing targeted adversarial attack on the generative classifier and the robust discriminative classifier. The PGD attack is L_2-based attack of steps 10 and step-size 1.0.

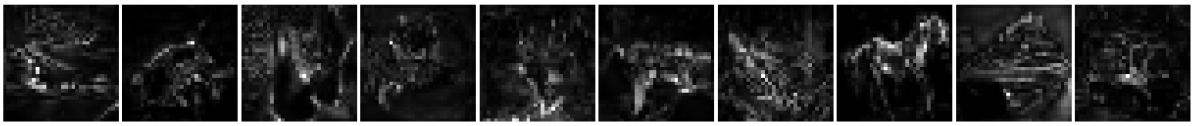
of the regular softmax classifier are scattered and do not align well with the foreground objects.

3.4.4 Ablation study

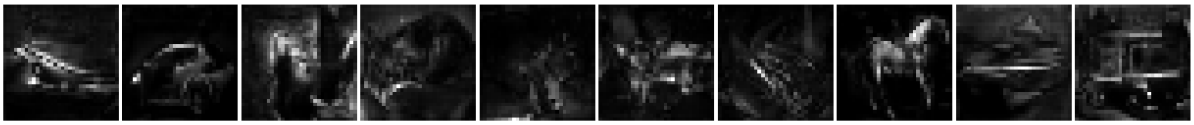
In this section we study how calibration, model capacity, weight decay regularization, training perturbation size, data augmentation, and combining in- and out-distribution adversarial training



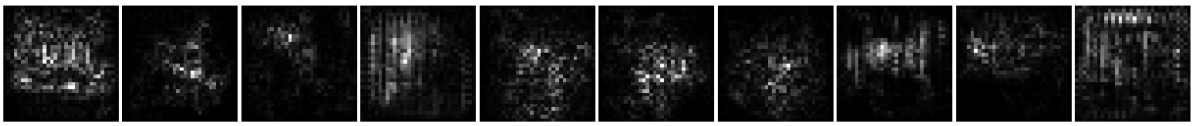
(a) Input images from class 0 to class 9



(b) IG masks generated with the generative classifier



(c) IG masks generated with the softmax robust classifier



(d) IG masks generated with the regular softmax classifier

Figure 3.10: Integrated gradients (IG) masks generated by different classifiers

affect the model performance. To make the analysis more concrete, we study how the above factors affect the performance of individual binary classifiers. We consider both the normal and adversarial scenarios, where in the normal scenario we evaluate the binary classifier on the task of separating clean in-distribution test samples from out-distribution test samples, and in the adversarial scenario we evaluate the binary classifier on the task of separating adversarially perturbed in-distribution and out-distribution test samples. We use the AUC (area under the ROC curve) score as the performance metric.

Calibration

The calibration parameters $\{c_1, ..., c_K\}$ are learned by minimizing the cross-entropy loss of the generative classifier Eq. (3.43) on the training set of CIFAR10. We use early stopping on the test set to select $\{c_1, ..., c_K\}$. The binary classifiers turn out to be well-calibrated, and the calibration

does not have much effect on the generative classifier's performance (Tab. 3.16). Tab. 3.17 shows the values of the learned $\{c_1, ..., c_K\}$ (the initial values of $\{c_1, ..., c_K\}$ are set to zeros). Fig. 3.11 shows the histogram of d_k's outputs on samples of class k and samples of other classes. It can be seen that the learned $\{c_1, ..., c_K\}$ are tiny compared to the outputs of d_k, so they do not have much effect on Eq. (3.43). We note that although the calibration does not have much effect on the evaluated dataset (CIFAR-10), it does not by itself exclude the possibility that for certain datasets the calibration can be helpful.

Table 3.16: Accuracies on the training set and test set before and after calibration.

	Training accuracy	Test accuracy
Generative classifier	95.25%	88.12%
Generative classifier after calibration	95.43%	88.15%

Table 3.17: The values of the learned $\{c_1, ..., c_K\}$.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
-0.068	-0.015	0.066	0.067	-0.037	0.066	-0.068	-0.067	-0.058	0.062

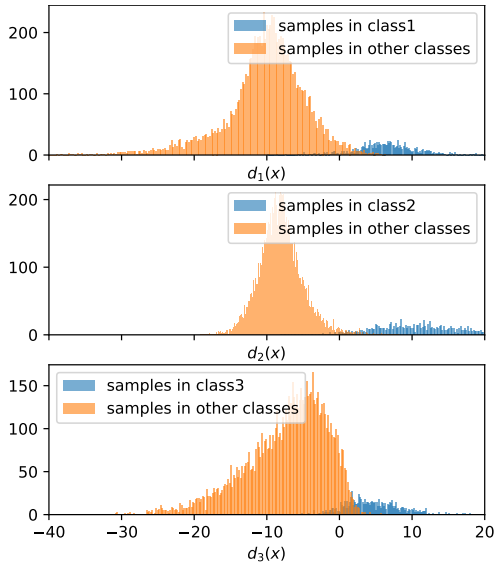


Figure 3.11: Histogram of d_k's outputs on samples of class k and samples of other classes, $k = 1, 2, 3$. (Note that the x axes of the subplots are shared.)

Model capacity and regularization

Consistent with the previous findings [4, 59, 112, 194], we find models with a higher capacity tend to have better robustness (Fig. 3.12). Note that overfitting happened even with ResNet18Thinner, the model with the lowest capacity. To mitigate overfitting, we apply weight decay and Fig. 3.12 shows that the weight decay help the model achieve better robustness.

Figure 3.12: Adversarial AUC score of the class 1 models of different capacities. "ResNet18Thin" and "ResNet18Thinner" are two ResNet18 models respectively with a width multiplier of 0.75 and 0.5.

Training perturbation size.

Similar to Tsipras et al. [177], in Fig. 3.13 we observe a decline in standard performance as the training perturbation size increases. Meanwhile, training with a larger perturbation helps the model achieve better adversarial robustness.

Data augmentation

Data augmentation is a widely used regularization technique for reducing overfitting in learning standard classification models. Unfortunately, the use of data augmentation in training robust classifiers is not as successful. On CIFAR-10, beyond the widely adopted random padding and

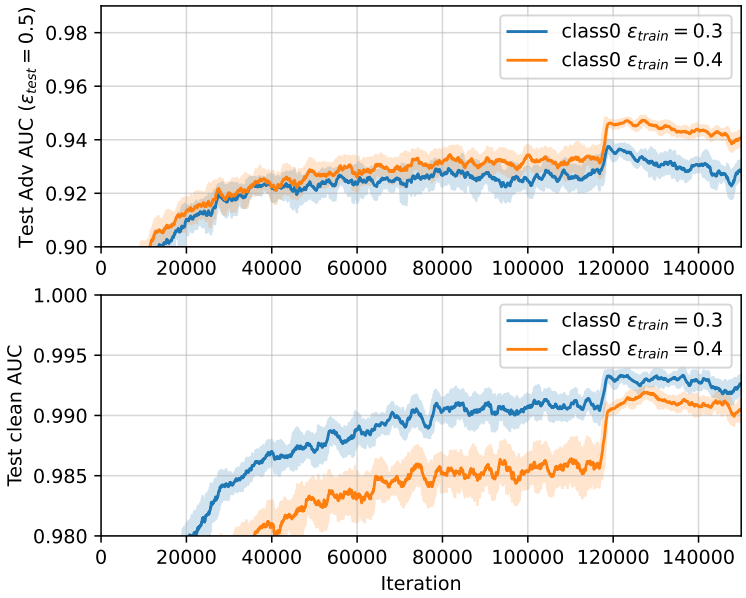


Figure 3.13: Standard and adversarial performances of the class 1 model when trained with different perturbation sizes.

cropping augmentation strategy, none of the more sophisticated augmentation techniques are beneficial for improving model robustness [59, 148]. Rebuffi et al. [145] study this phenomenon and find that when combined with model weight averaging [82], heuristics-driven augmentation techniques such as Cutout [41], CutMix [209] and MixUp [213] can improve robustness. However, data-driven data augmentation approaches such as AutoAugment [36] have not been found to be helpful. In contrast to these work, we find it straightforward to apply AutoAugment in our training to reduce overfitting and obtain better robustness (Fig. 3.14).

Combining in- and out-distribution adversarial training

Fig. 3.15 shows the effect of combining in- and out-distribution adversarial training. It can be seen that combining in- and out-distribution AT helps the model achieve better adversarial performance at the expense of decreased standard performance.

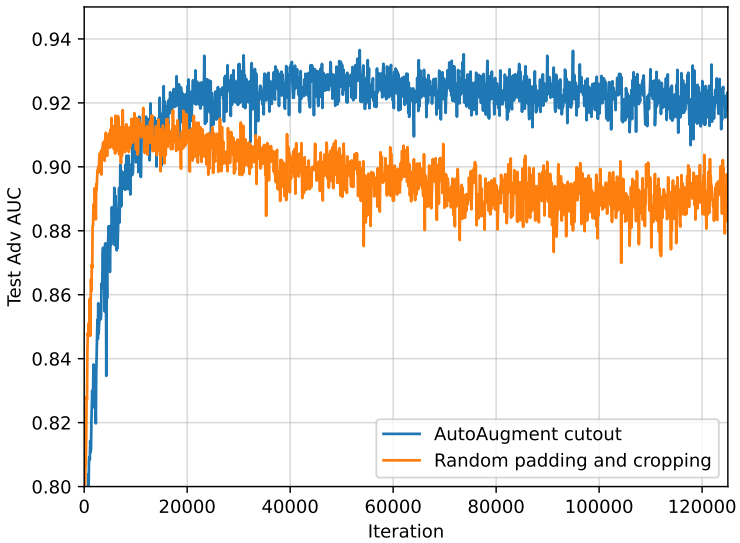


Figure 3.14: Adversarial AUC score of the class 1 model when trained with different data augmentation policies.

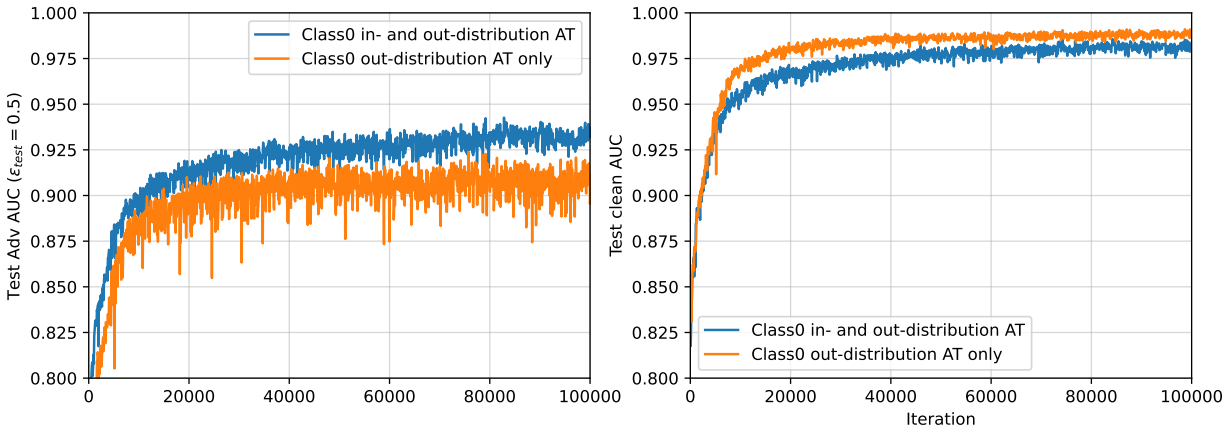


Figure 3.15: Clean and adversarial AUC scores of the class 1 model when trained with combined AT and out-distribution AT only.

3.4.5 Conclusion

We have investigated applying the AT generative model to learning class-conditional distributions and then using the conditional models to perform generative classification. Compared to the baseline softmax robust classifier, the generative classifier achieves comparable performance when there is no adversarial perturbation or when the perturbation is small, and much better performance when the perturbation is of moderate or large sizes. The generated samples and counterfactuals produced by the generative classifier are also more similar to the training data

(as measured by FID) than that of the softmax robust classifier, suggesting that the generative classifier has better captured the class-conditional distributions. We note that the above results are obtained when we use a much smaller model for the binary classifiers so that the overall capacities of the generative classifier and softmax classifier are similar. We further show how model capacity and training perturbation size affect the model performance in a similar way as they would in standard adversarial training. Compared to the softmax robust classifier, we find it much easier to apply advanced augmentation such as AutoAugment in our training. We leave the evaluation of other data augmentation such as CutMix and MixUp to future work. The analysis in [207] also suggests that using a diverse out-distribution dataset improve the generative modeling performance of the binary classifier, but it is unclear whether using auxiliary OOD data helps improve the binary classifier's discriminative capability. Of related work is [9] which show that incorporating auxiliary OOD data helps the robust classifier achieves better standard and robust accuracies on CIFAR-10. In addition, some recent work show that generated in-distribution data produced by some generative models can be leveraged to achieve better robustness [60, 145]. We leave the investigation of using auxiliary OOD data and generated in-distribution data in our approach to future work.

Chapter 4

Conclusions and Future Work

Not being able to correctly estimate the likelihood of out-of-distribution data is a key limitation of existing generative models. In this work we introduce an adversarial training-based approach to learning energy-based models (EBMs) that overcome this limitation. Inspired by recent work that shows adversarially robust classifier has a strong generative property, we investigate training a binary classifier to separate in-distribution and adversarially perturbed out-distribution data. Our analysis shows that in this setup the binary classifier learns a special kind of energy function that models the support of the data distribution, and the learning process is closely related to MCMC-based maximum likelihood learning of EBMs. The training objective of the binary classifier can also be interpreted as a maximin two-player zero-sum game, and is closely related to GANs' minimax game. Based on the above analysis, we propose improved training techniques for generative modeling with adversarial training (AT), and show that this AT generative model is capable of generating realistic and diverse images. Our quantitative evaluation shows that the AT generative model achieves competitive image generation performance to state-of-the-art EBMs, and does not have the training stability issues of standard EBMs. Our evaluation of the model on worst-case OOD detection (Chapter 3) also indicates that the model has the expected behavior on normal and worst-case OOD data. Our results demonstrate the viability of the AT approach to generative modeling and OOD detection, suggesting that AT is a competitive alternative approach

to learning EBMs.

One interesting property of the AT generative model is its capability of transforming out-of-distribution samples into in-distribution samples. This makes the model particularly suitable for image-to-image translation and image restoration tasks such as denoising and inpainting. In Chapter 2 we provide a demonstration of the AT generative model's applications to these tasks. We note that the AT generative model is not specifically trained to perform these tasks, and a single model can be applied to all of these tasks without retraining or finetuning. However, a more systematic evaluation and comparison with existing approaches may be needed in order to fully understand the strengths and weaknesses of the proposed solution to these problems.

In Chapter 3 we investigate applications of the AT generative models to detecting adversarial examples, detecting worst-case OOD inputs, and generative robust classification. For a K class classification problem, the proposed solution to detecting adversarial examples consists of K binary classifiers, with each one trained to distinguish clean samples of a particular class from adversarially perturbed samples of other classes. Based on the analysis in Chapter 2, we can view the binary classifiers as unnormalized density models of the class-conditional distributions, and interpret the process of detecting adversarial examples as using these density models to identify OOD inputs. The proposed defense not only outperforms existing approaches by a large margin, but also is robust to adaptive attacks, as demonstrated by our mathmatical justification, systematic evaluation, and independent verification by other researchers. Adversarial training is widely considered reliable defense against adversarial attacks, and has become the de facto approach to training adversarially robust classifiers. Our novel approach to detecting adversarial examples allows us to employ adversarial training to train a robust defense, and unifies robust classification and detecting adversarial examples under the same framework of robust optimization.

Our result on worst-case OOD detection shows that the AT generative model not only is able to generate diverse and realistic images, but also has low probability outputs on normal and worst-case OOD inputs. This suggests that the AT generative model can be applied to both content generation and verification. This is in stark contrast with existing generative models which

can generate realistic data samples but are unable to tell whether the input belongs to the training distributions. However, this generative property on in-distribution data and discriminative property on OOD data is achieved by using a large and diverse OOD dataset to train the model. For a given task, such a dataset may not be readily available.

We explored generative classification by using the AT generative model to learn the class-conditional distributions. Our result shows that under the condition of similar model capacity, the generative classifier achieves comparable performance to a baseline softmax robust classifier when there are no adversarial perturbation or when the test perturbation is small, and much better performance when the test perturbation size exceeds the training perturbation size. The generative classifier is also more interpretable than the softmax robust classifier in the sense that it can generate synthetic samples or visual counterfactuals that more closely resemble the training data. As the generative classifier needs to train separate models for each classes, this approach may not scale well to problems with a large number of classes.

The main drawbacks of the AT generative model are that it requires a large, diverse OOD dataset to train, and its image generation performance falls behind state-of-the-art generative models such as GANs and diffusion models. As discussed in Chapter 2, the diversity of the OOD dataset has a large impact on the AT generative model's performance, so the OOD dataset should be chosen carefully. One possible issue that can affect the model' generative modeling performance is overfitting, a problem that also plagues other related generative models such as GANs. In future work, we plan to improve the AT generative model by mitigating overfitting using transfer learning, (differentiable) data augmentation, and self-supervised training. In addition to continuing improving the AT generative model, we are also interested in applying the AT generative model to sequential data.

Bibliography

[1] Vahdat Abdelzad, Krzysztof Czarnecki, Rick Salay, Taylor Denounden, Sachin Vernekar, and Buu Phan. Detecting out-of-distribution inputs in deep neural networks using an early-layer output. *arXiv preprint arXiv:1910.10307*, 2019. 3.3.1

[2] Sravanti Addepalli, Vivek BS, Arya Baburaj, Gaurang Sriramanan, and R Venkatesh Babu. Towards achieving adversarial robustness by enforcing feature consistency across bit planes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1020–1029, 2020. 3.1.3

[3] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3389–3398, 2018. 3.1.3

[4] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems*, 32, 2019. 3.1.3, 3.4.1, 3.4.4

[5] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020. 3.1.3

[6] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=0PtUPB9z6qK`. 2.3

[7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017. 3.1

[8] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. URL `https://arxiv.org/abs/1802.00420`. 3.1, 3.1.3

[9] Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in-and out-distribution improves explainability. In *European Conference on Computer Vision*, pages 228–245. Springer, 2020. 1.1, 3.3.1, 3.3.1, 3.3.2, 3.3.2, 3.12, 3.3.3, 3.4.1, 3.4.3, 3.4.5

[10] Yuval Bahat, Michal Irani, and Gregory Shakhnarovich. Natural and adversarial error detection using invariance to image transformations. *arXiv preprint arXiv:1902.00236*, 2019. 3.1.3

[11] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2019. 3.1.3

[12] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in neural information processing systems*, pages 2613–2621, 2016. 3.1

[13] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2017. 3.1.3, 3.1.3

[14] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2018. 3.1.3

[15] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial

machine learning. *Pattern Recognition*, 84:317–331, 2018. 3.1

[16] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 3.4.1

[17] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Certifiably adversarially robust detection of out-of-distribution data. *Advances in Neural Information Processing Systems*, 33, 2020. 3.3.1, 3.3.1

[18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017. 3.1.2

[19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1xsqj09Fm`. 2.5.2, 2.5

[20] Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. In *International Conference on Machine Learning*, pages 831–840. PMLR, 2019. 3.1.1

[21] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017. 3.1.3, 3.1.3

[22] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017. URL `https://arxiv.org/abs/1705.07263`. 3.1.3, 3.2.2, 3.2.3, 3.6, 3.11

[23] Nicholas Carlini and David Wagner. Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017. 3.1.3

[24] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural net-

works. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017. 3.1, 3.1.2, 3.2.2

[25] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019. 3.1.3

[26] Ciwan Ceylan and Michael U Gutmann. Conditional noise-contrastive estimation of unnormalised models. In *International Conference on Machine Learning*, pages 726–734. PMLR, 2018. 2.3

[27] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. *arXiv preprint arXiv:1912.10185*, 2019. 3.1.3

[28] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 3.1

[29] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 699–708, 2020. 3.1.3

[30] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12154–12163, 2019. 2.6

[31] Wenhu Chen, Yilin Shen, Hongxia Jin, and William Wang. A variational dirichlet framework for out-of-distribution detection. *arXiv preprint arXiv:1811.07308*, 2018. 3.3.1

[32] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8188–8197, 2020. (document), 2.5.1, 2.6, 2.5.3,

3.3.2

[33] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020. 3.12, 3.13

[34] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. 3.1.3, 3.4.3

[35] Ekin D Cubuk, Barret Zoph, Samuel S Schoenholz, and Quoc V Le. Intriguing properties of adversarial examples. *arXiv preprint arXiv:1711.02846*, 2017. 3.1.1

[36] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 3.3.2, 3.4.4

[37] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, Jeremy Dawson, and Nasser M Nasrabadi. Exploiting joint robustness to adversarial perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1122–1131, 2020. 3.1.3

[38] Amit Daniely and Hadas Shacham. Most relu networks suffer from l2 adversarial perturbations. *Advances in Neural Information Processing Systems*, 33:6629–6636, 2020. 3.1.1

[39] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017. 3.1.3

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and*

pattern recognition, pages 248–255. Ieee, 2009. 2.4

[41] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 3.4.4

[42] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018. 3.1.3

[43] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1.1, 3.3.1

[44] Yinpeng Dong, Zhijie Deng, Tianyu Pang, Jun Zhu, and Hang Su. Adversarial distributional training for robust deep learning. *Advances in Neural Information Processing Systems*, 33:8270–8283, 2020. 3.1.3

[45] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL `https://proceedings.neurips.cc/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf`. 2.1, 2.2.2, 2.2.2, 2.3, 2.5

[46] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. In *ICML*, 2021. 2.1, 2.3, 2.5

[47] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 3.4.1

[48] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016. 3.1.3

[49] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`. 3.4.3, 3.4.3, 3.14, 3.15

[50] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019. 2.1, 3.2.1

[51] Farzan Farnia, Jesse M Zhang, and David Tse. Generalizable adversarial training via spectral normalization. *arXiv preprint arXiv:1811.07457*, 2018. 3.1.3

[52] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. URL `https://arxiv.org/abs/1703.00410`. 3.1.3, 3.1.3

[53] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=v_1Soh8QUNc`. 2.5.2, 2.5

[54] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018. 3.1.1

[55] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017. URL `https://arxiv.org/abs/1704.04960`. 3.1.3, 3.1.3

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1.1, 2.2.1, 2.4.1, 2.4.4, 2.4.4

[57] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. URL `https://arxiv.org/abs/1412.6572`. 3.1, 3.1.1, 3.1.1, 3.1.1, 3.1.2, 3.1.3, 3.1.3, 3.4.3

[58] Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1211–1220, 2020. 3.1.3

[59] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020. 3.1.3, 3.4.1, 3.4.4, 3.4.4

[60] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34:4218–4233, 2021. 3.4.1, 3.4.5

[61] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkxzx0NtDB`. 2.1, 2.2.2, 2.3, 2.5, 2.9, 3.12

[62] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No mcmc for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=ixpSxO9flk3`. 2.1, 2.3, 2.5

[63] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, 2017. 3.1.3, 3.1.3

[64] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014. 3.1

[65] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C

Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf. 2.5

[66] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020. 3.1.3

[67] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010. 2.3

[68] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8670–8679, 2019. 2.3

[69] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Joint training of variational auto-encoder and latent energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7978–7987, 2020. 2.3

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2.5.1

[71] Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkpiPMbA-. 3.1

[72] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019. 3.3.1, 3.3.1

[73] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 3.3.1

[74] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *ICLR*, 2017. URL `https://arxiv.org/abs/1608.00530`. 3.1.3, 3.1.3

[75] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018. 1.1, 3.3.1, 3.12

[76] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, pages 15663–15674. PMLR, 2019. 3.1.3

[77] 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. In *Advances in neural information processing systems*, pages 6626–6637, 2017. 2.5.1, 2.5.2, 3.4.3

[78] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`. 2.3, 2.5, 2.6

[79] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 2.3

[80] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint*

arXiv:1905.02175, 2019. 2.1, 3.1.3

[81] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2.5.3

[82] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 3.4.1, 3.4.4

[83] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018. 3.1.1

[84] Yunseok Jang, Tianchen Zhao, Seunghoon Hong, and Honglak Lee. Adversarial defense via learning to generate diverse attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2740–2749, 2019. 3.1.3

[85] Ahmadreza Jeddi, Mohammad Javad Shafiee, Michelle Karg, Christian Scharfenberger, and Alexander Wong. Learn2perturb: an end-to-end feature perturbation learning to improve adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1241–1250, 2020. 3.1.3

[86] Saumya Jetley, Nicholas Lord, and Philip Torr. With friends like these, who needs adversaries? *Advances in neural information processing systems*, 31, 2018. 3.1.1

[87] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. *Advances in Neural Information Processing Systems*, 33: 16199–16210, 2020. 2.3

[88] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306*, 2015. 3.1

[89] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

2.5.1, 2.5.2, 2.6, 3.3.2

[90] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems*, volume 33, pages 12104–12114, 2020. URL https://proceedings.neurips.cc/paper/2020/file/8d30aa96e72440759f74bd2306c1fa3d-Paper.pdf. 2.5.1, 2.5, 2.6, 2.6

[91] Diederik P Kingma. Fast gradient-based inference with continuous latent variable models in auxiliary form. *arXiv preprint arXiv:1306.0733*, 2013. 1.1, 3.3.1

[92] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3.1.3, 3.2.3

[93] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018. 1.1, 2.6, 3.3.1

[94] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33, 2020. 1.1, 3.3.1

[95] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *2018 ieee security and privacy workshops (spw)*, pages 36–42. IEEE, 2018. 3.1, 3.3.1

[96] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2.5.1, 2.5.2, 3.3.2

[97] Dmitry Krotov and John Hopfield. Dense associative memory is robust to adversarial inputs. *Neural computation*, 30(12):3151–3167, 2018. 3.1.1

[98] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*,

2019. 2.3

[99] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 2.2.3, 3.1, 3.1.2, 3.1.3, 3.1.3

[100] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. URL `https://arxiv.org/abs/1607.02533`. 3.1

[101] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 2.1, 2.2.2, 3.2.1

[102] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018. 3.3.1

[103] Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 272–281, 2020. 3.1.3

[104] Guanlin Li, Shuya Ding, Jun Luo, and Chang Liu. Enhancing intrinsic adversarial robustness via feature pyramid decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 800–808, 2020. 3.1.3

[105] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5775–5783, 2017. URL `https://arxiv.org/abs/1612.07767`. 3.1.3, 3.1.3

[106] Yueru Li, Shuyu Cheng, Hang Su, and Jun Zhu. Defense against adversarial attacks via controlling gradient leaking on embedded manifolds. In *European Conference on Computer Vision*, pages 753–769. Springer, 2020. 3.1.1

[107] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

3.3.1

[108] Wei-An Lin, Chun Pong Lau, Alexander Levine, Rama Chellappa, and Soheil Feizi. Dual manifold adversarial robustness: Defense against lp and non-lp adversarial attacks. *Advances in Neural Information Processing Systems*, 33:3487–3498, 2020. 3.1.3

[109] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2020. 2.6

[110] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018. URL `https://arxiv.org/abs/1801.02613`. 3.1.3, 3.1.3

[111] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Adversarial neural pruning with latent vulnerability suppression. In *International Conference on Machine Learning*, pages 6575–6585. PMLR, 2020. 3.1.3

[112] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 2.2.3, 2.3, 2.4.5, 3.1, 3.1.2, 3.1.3, 3.1.3, 3.2.1, 3.2.1, 3.2.3, 3.3, 3.2.3, 3.3, 3.2.3, 3.2.3, 3.5, 3.6, 3.3.1, 3.4.1, 3.4.3, 3.4.4

[113] MadryLab. CIFAR10 Adversarial Examples Challenge. `https://github.com/MadryLab/cifar10_challenge`, . 3.2.3

[114] MadryLab. MNIST Adversarial Examples Challenge. `https://github.com/MadryLab/mnist_challenge`, . 3.2.3

[115] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR, 2020. 3.1.3

[116] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058, 2018. 3.3.1

[117] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. *arXiv preprint arXiv:1909.12180*, 2019. 3.3.1, 3.3.1

[118] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017. 3.1.3

[119] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 2.4.5

[120] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *CoRR*, abs/1702.04267, 2017. URL `https://arxiv.org/abs/1702.04267`. 3.1.3, 3.1.3

[121] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017. 3.1.3

[122] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3578–3586. PMLR, 2018. 3.1.3

[123] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1QRgziT-`. 2.5

[124] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020. 3.4.3

[125] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE*

conference on computer vision and pattern recognition, pages 2574–2582, 2016. 3.1.2

[126] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. 3.1.1

[127] Amir Najafi, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. Robustness to adversarial perturbations in learning from incomplete data. *Advances in Neural Information Processing Systems*, 32, 2019. 3.1.3

[128] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018. 1.1, 3.3.1

[129] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 262–271, 2020. 3.1.3

[130] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *NeurIPS*, 2019. 2.1, 2.2.2, 2.2.2, 2.3, 2.5

[131] 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, pages 5272–5280, 2020. 2.1, 2.2.2, 2.2.2, 2.3, 2.4.3

[132] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. MCMC should mix: Learning energy-based model with flow-based backbone. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=4C93Qvn-tz`. 2.3

[133] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 1.1, 3.3.1

[134] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/file/fa3060edb66e6ff4507886f9912e1ab9-Paper.pdf. 2.3

[135] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, pages 4579–4589, 2018. URL https://arxiv.org/abs/1706.00633. 3.1.3

[136] Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *arXiv preprint arXiv:1905.10626*, 2019. 3.1.3

[137] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019. 3.1.3

[138] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016. 3.1.2

[139] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016. 3.1

[140] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 2.6

[141] Phillip Pope, Yogesh Balaji, and Soheil Feizi. Adversarial robustness of flow-based gen-

erative models. In *International Conference on Artificial Intelligence and Statistics*, pages 3795–3805. PMLR, 2020. 3.3.1

[142] Igor M Quintanilha, Roberto de ME Filho, José Lezama, Mauricio Delbracio, and Leonardo O Nunes. Detecting out-of-distribution samples using low-order deep features statistics. 2018. 3.3.1

[143] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018. 3.1.3

[144] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7:1, 2017. 2.3

[145] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021. 3.1.3, 3.4.1, 3.4.4, 3.4.5

[146] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, pages 1278–1286, 2014. 1.1, 3.3.1

[147] Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *arXiv preprint arXiv:2006.12204*, 2020. 2.3

[148] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020. 3.1.3, 3.4.3, 3.4.4

[149] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*, 2019. 3.1.3

[150] Andras Rozsa, Manuel Gunther, and Terrance E Boult. Towards robust deep neural networks with bang. *arXiv preprint arXiv:1612.00138*, 2016. 3.1.1

[151] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization

to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909, 2016. 2.3

[152] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture. *ICLR*. 3.3.1

[153] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016. 2.5.1, 2.5.2

[154] Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. In *Advances in Neural Information Processing Systems*, pages 1260–1271, 2019. 2.1, 2.5, 3.2.1, 3.2.3, 3.6, 3.4.3

[155] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with in-distribution examples and gram matrices. *arXiv preprint arXiv:1912.12510*, 2019. 3.3.1

[156] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Better the devil you know: An analysis of evasion attacks using out-of-distribution adversarial examples. *arXiv preprint arXiv:1905.01726*, 2019. 3.3.1, 3.3.1

[157] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. *arXiv preprint arXiv:2004.10162*, 2020. 3.1.3

[158] Alireza Shafaei, Mark Schmidt, and James J Little. Does your model know the digit 6 is not a cat? a less biased evaluation of" outlier" detectors. 2018. 1.1, 3.3.1

[159] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018. 3.1.1

[160] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness

with principled adversarial training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk6kPgZA-. 3.1

[161] Chuanbiao Song, Kun He, Jiadong Lin, Liwei Wang, and John E Hopcroft. Robust local features for improving the generalization of adversarial training. *arXiv preprint arXiv:1909.10147*, 2019. 3.1.3

[162] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf. 2.3, 2.5, 2.10

[163] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020. 2.3, 2.5

[164] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS. 2.3, 2.5

[165] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017. 3.4.3

[166] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 3.1.1, 3.1.2, 3.3.1

[167] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 international joint conference on neural networks (IJCNN)*, pages 426–433. IEEE, 2016. 3.1.1

[168] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational au-

toencoders. *arXiv preprint arXiv:1612.00155*, 2016. 3.1

[169] Saeid Asgari Taghanaki, Kumar Abhishek, Shekoofeh Azizi, and Ghassan Hamarneh. A kernelized manifold mapping to diminish the effect of adversarial perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11340–11349, 2019. 3.1.1

[170] Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016. 3.1.1

[171] Shixin Tian, Guolei Yang, and Ying Cai. Detecting adversarial examples through image transformation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3.1.3

[172] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008. 2.4.5, 2.4, 3.3.2, 3.3.2

[173] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017. 3.1.1

[174] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020. 3.1.3, 3.4.1, 3.4.3

[175] Ngoc-Trung Tran, Viet-Hung Tran, Bao-Ngoc Nguyen, Linxiao Yang, and Ngai-Man Man Cheung. Self-supervised gan: Analysis and improvement with multi-class minimax game. *Advances in neural information processing systems*, 32, 2019. 2.6

[176] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021. 2.6

[177] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander

Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018. 2.1, 3.1.3, 3.2.1, 3.4.1, 3.4.3, 3.4.4

[178] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/file/e3b21256183cf7c2c7a66be163579d37-Paper.pdf. 2.5.2, 2.6

[179] BS Vivek and R Venkatesh Babu. Single-step adversarial training with dropout scheduling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 947–956. IEEE, 2020. 3.1.3

[180] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017. 3.4.3

[181] Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6629–6638, 2019. 3.1.3

[182] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. A unified contrastive energy-based model for understanding the generative ability of adversarial training. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XhF2VOMRHS. 2.3, 2.5

[183] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019. 3.1.3

[184] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. (document), 2.7, 2.8

[185] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. 2.2.2, 2.3

[186] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018. 3.1.3

[187] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 3.1.3

[188] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *Advances in Neural Information Processing Systems*, 34, 2021. 3.1.3

[189] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020. 3.1.3

[190] Chang Xiao and Changxi Zheng. One man's trash is another man's treasure: Resisting adversarial examples by adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 412–421, 2020. 3.1.3

[191] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. *arXiv preprint arXiv:1905.10510*, 2019. 3.1.3

[192] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3905–3911. AAAI Press, 2018. 3.1

[193] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on*

Learning Representations, 2021. URL https://openreview.net/forum?id=5m3SEczOV8L. (document), 2.1, 2.3, 2.5.2, 2.5, 2.6, 2.9, 2.10

[194] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *arXiv preprint arXiv:1906.03787*, 2019. 3.1.3, 3.4.1, 3.4.4

[195] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1369–1378, 2017. 3.1

[196] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019. 3.1.3

[197] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR, 2016. 2.2.2, 2.2.2, 2.3

[198] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):27–45, 2018. 2.3

[199] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8629–8638, 2018. 2.3

[200] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2.3

[201] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 2.3

[202] Jianwen Xie, Zilong Zheng, and Ping Li. Learning energybased model with variational auto-encoder as amortized sampler. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, volume 2, 2021. 2.3, 2.5

[203] Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. A tale of two flows: Cooperative learning of langevin flow and normalizing flow toward energy-based model. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=31d5RLCUuXC`. 2.3, 2.5

[204] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017. 3.1.3, 3.1.3

[205] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120, 2019. 3.1.3

[206] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Gat: Generative adversarial training for adversarial example detection and robust classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJeQEp4YDH`. 2.1, 2.2.3, 3.4.1

[207] Xuwang Yin, Shiying Li, and Gustavo K Rohde. Learning energy-based models with adversarial training. In *European Conference on Computer Vision*, pages 209–226. Springer, 2022. 3.4.2, 3.4.2, 3.4.5

[208] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 2.5.1, 2.5.2, 3.3.2

[209] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and

Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3.4.4

[210] Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*, 2019. 3.1.3

[211] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14521–14530, 2020. 3.1.1

[212] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019. 3.1.3

[213] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3.4.4

[214] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. *arXiv preprint arXiv:2005.00060*, 2020. 3.1.3

[215] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020. 2.6

[216] Yang Zhao, Jianwen Xie, and Ping Li. Learning energy-based generative models via coarse-to-fine expanding and sampling. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=aD1_5zowqV`. 2.1, 2.3, 2.5, 2.6, 2.9

[217] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmen-

tations for gan training. *arXiv preprint arXiv:2006.02595*, 2020. 2.6

[218] Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1181–1190, 2020. 3.1.3

[219] Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 7913–7922, 2018. 3.1.3

[220] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2.5.3