Generative Models as a Robust Alternative for Image Classification: Progress and Challenges

An Ju

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-47
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-47.html

May 11, 2021

Acknowledgement

I want to thank Professor David Wagner for advising me. This thesis will not be possible without you. I also want to thank my committee members. Thank Professor Trevor Darrell for your feedback and kindly supporting me with computing resources. Thank Professor Jiantao Jiao and Doctor Nicholas Carlini for your time and effort.

The analysis on GANs is a joint work with Doctor Samaneh Azadi at UC Berkeley. She is an outstanding researcher who always has insightful feedback.

Generative Models as a Robust Alternative for Image Classification: Progress and Challenges

by

An Ju

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Wagner, Chair
Professor Trevor Darrell
Professor Jiantao Jiao
Doctor Nicholas Carlini

Spring 2021

Generative Models as a Robust Alternative for Image Classification: Progress and Challenges

Abstract

Generative Models as a Robust Alternative for Image Classification: Progress and Challenges

by

An Ju

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor David Wagner, Chair

The tremendous success of neural networks is clouded by the existence of adversarial examples: maliciously engineered inputs can cause neural networks to perform abnormally, causing security and trustworthiness concerns. This thesis will present some progress on an alternative approach for robustly classifying images. Generative classifiers use generative models for image classification, showing better robustness than discriminative classifiers. However, generative classifiers face some unique challenges when images are complex. This thesis will present an analysis of these challenges and remedies.

Generative classifiers suffer from out-of-domain reconstructions: overpowered generators can generate images out of the training distribution. This thesis demonstrates a method to address out-of-domain reconstructions in generative classifiers. Combined with other extensions, our method has successfully extended generative classifiers from robustly recognizing simple digits to classifying structured colored images. Besides, this thesis conducts a systematic analysis of out-of-domain reconstructions on CIFAR10 and ImageNet and presents a method to address this problem on these realistic images.

Another challenge of generative classifiers is measuring image similarity. This thesis will demonstrate that similarity measurements are critical for complex images such as CIFAR10 and ImageNet. It also presents a metric that has significantly improved generative classifiers on complex images.

The challenges of generative classifiers come from modeling image distributions. Discriminative models do not have such challenges because they model label distribution instead of image distribution. Therefore, the last part of this thesis is dedicated to a method that connects the two worlds. With the help of randomized smoothing, the new generative method leverages discriminative models and model image distribution under noises. Experiments showed that such a method improves the robustness of unprotected models, suggesting a promising direction for connecting the world of generative models and discriminative models.

To my parents

and my fiancee, Hongyu Zhang

Contents

List of Figures

List of Tables

Acknowledgments

I want to thank Professor David Wagner for advising me. This thesis will not be possible without you. I also want to thank my committee members. Thank Professor Trevor Darrell for your feedback and kindly supporting me with computing resources. Thank Professor Jiantao Jiao and Doctor Nicholas Carlini for your time and effort.

The analysis on GANs is a joint work with Doctor Samaneh Azadi at UC Berkeley. She is an outstanding researcher who always has insightful feedback.

I have also received great advice from my colleagues, especially Alan Rosenthal and Chawin Sitawarin. Alan and I have been working on some exciting projects about generative classifiers, such as detecting adversarial examples and reducing the running time of generative classifiers. His work gives me many inspirations. If you are interested, please read Alan's master report to learn about his amazing work.

Chapter 1

Introduction

1.1 Adversarial examples: a twenty-first-century cloud over artificial neural networks

Artificial neural networks have scored huge success in many tasks and have become fundamental in various areas. However, they are black-box models whose risks and shortcomings are not apparent to users [16]. Lack of transparency causes serious concerns, especially when neural networks are widely applied [117]. For example, AI helper devices, such as Amazon Echo and Google Home, could be controlled silently to execute some actions [90], and AI autonomous driving systems will make mistakes when certain maliciously constructed objects are detected [34, 113]. Both examples are caused by *adversarial examples* of neural networks.

Deep neural networks are susceptible to adversarial examples: a deep model's accuracy drops significantly under adversarially chosen perturbations, even though these perturbations

Figure 1.1: Neural networks that recognize traffic signs are susceptible to adversarial attacks. In the first example, the neural network misclassifies a perturbed priority road sign as a stop sign. In the second example, the neural network misclassifies a non-traffic sign as a stop sign. In both cases, the perturbation cannot change human perception, but has caused the neural network to misclassify with a high confidence. Images are adapted from [97].

do not change human perception [101, 14]. They are a "cloud" [1] over artificial neural networks, indicating a fundamental difference between artificial neural networks and human's biological neural systems. They also pose real threats as neural models have been deployed as critical components in various domains, such as transportation, malware detection, and financial sectors. Figure 1.1 shows how adversarial examples may cause serious security concerns to neural models in real world.

We are interested in bounded adversarial examples in particular. Specifically, a bounded adversarial example is constrained by its distance to the corresponding natural data point. For example, constructing an adversarial example bounded by its L_2 norm to the natural data point x with the goal of causing a neural network f to misclassify can be framed as finding an x' such that

$$f(x) \neq f(x'), \text{and}$$
$$\|x' - x\|_2 \leq \epsilon$$

where ϵ is a hyperparameter. L_p bounded robustness provides a well-defined problem and thus has attracted many studies [117, 109]. Furthermore, L_p robustness is correlated with transfer learning [92], representation learning [33], and improved training stability for generative classifiers [123]. Therefore, understanding bounded adversarial examples and building a robust model against such attack is an important research question and may provide us with a novel understanding of how neural network works [26, 49].

Many defenses have been proposed since Szegedy et al. found that deep neural networks are susceptible to carefully engineered adversarial inputs [101]. However, later studies showed that many of these defenses are breakable [14]. The state-of-the-art robustness is achieved by a method called adversarial training [70, 25, 41]. Adversarial training focuses on methods that use carefully designed training examples to improve model robustness. However, adversarial training seems to only provide robustness against the specific attack used at training time: if adversarially trained with L_∞ bounded attacks, the neural network's robustness against L_2 bounded attacks is weaker than its robustness against L_∞ attacks or sometimes no robustness against other types of attacks at all [87, 95]. Therefore, it is interesting to find alternatives to adversarial training that may achieve "intrinsic" robustness against any adversarial examples. Generative classifiers, as we will discuss next, is a promising alternative method.

1.2 Generative classifiers: a robust alternative

Given a data distribution $(X, y) \sim \mathcal{D}$, a classifier models $p(y|X)$. *Discriminative classifiers* directly model $p(y|X)$. For example, a ResNet model [43] is trained with cross-entropy

[1] Towards the end of 19th century, Physicist Lord Kelvin gave a speech entitled "Nineteenth-Century Clouds over the Dynamical Theory of Heat and Light". Removal of the two clouds, two unexplainable observations by the Physics theory back then, led to the theory of relativity and quantum mechanics: not only a more insightful understanding of basic laws, but also a change of paradigm in Physics.

loss, which estimates the distance between real label and predicted $p(y|X)$. In comparison, *generative classifiers* model the conditional distribution of inputs $p(X|y)$ and use the Bayes rule for classification:

$$p(y|X) = \frac{p(X|y)p(y)}{p(X)} \propto p(X|y)p(y)$$

Therefore, the goal of generative classifiers is to learn a conditional data distribution, the distribution of samples from a specific class. Sample distribution is more complex than label distribution, so generative classifiers face more challenges than discriminative classifiers. However, some recent progress in generative models leads us to investigate generative classifiers again: can we use state-of-the-art generative models to build a generative classifier that is comparable discriminative classifiers?

Generative classifiers rely on generative models, which map the data distribution to a distribution of latent representations. For example, variational autoencoders[58] encode the input X to a latent space $\mathcal{Z}$ and generate a X' from an encoded latent code z. Therefore, X has a latent representation z under this model. Furthermore, the distance (L_2 distance) between X and X', together with the likelihood of z, is used to approximate for $p(X|y)$; if X is represented by a representation z with a higher likelihood with respect to the prior distribution of z and X' is close to X, this sample X will have a high likelihood under the generative model.

Generative classifiers are more interpretable than discriminative classifiers because they give a way of measuring the uncertainty of a classifier's prediction. For example, a cat-dog discriminative classifier will always classify an image as either cat or dog; in comparison, generative classifiers can detect an "out-of-domain" image, such as an image of a laundry machine, because generative classifiers have low likelihood $p(X|y)$ for both $y = \text{dog}$ and $y = \text{cat}$. Figure 1.2 demonstrates this example with some fictitious numbers.

Furthermore, generative classifiers are more robust against adversarial attacks [69, 68, 95]. Schott et al. showed that generative models can achieve state-of-the-art robustness on MNIST [95]. Furthermore, adversarial examples that fool generative classifiers are more likely to fool human subjects [37], suggesting that generative classifiers are more consistent with human perception. Because generative classifiers model the distribution of images X instead of labels y, they do not use models that reduce the dimensionality of inputs, whereas discriminative classifiers reduces X, a very high-dimension point, to y, a low-dimension vector. In Chapter 2, we will present more details about Schott et al.'s generative classifiers where we will also provide more intuitions why generative classifiers provide more adversarial robustness than discriminative models.

It worth noting that generative models have been used to defend adversarial examples in other ways as well. For example, Defense GAN [93] uses generative models to map an adversarially perturbed input onto natural data manifolds and expects a discriminative classifier to classify samples on natural data manifolds correctly; unfortunately, this defense was later broken by an adaptive attack [51]. Xie et al. uses denoisers, a special kind of generative model, to improve robustness [114]; their motivation is to map noisy features to a clean manifold. Although these methods rely on generative models to model a natural data

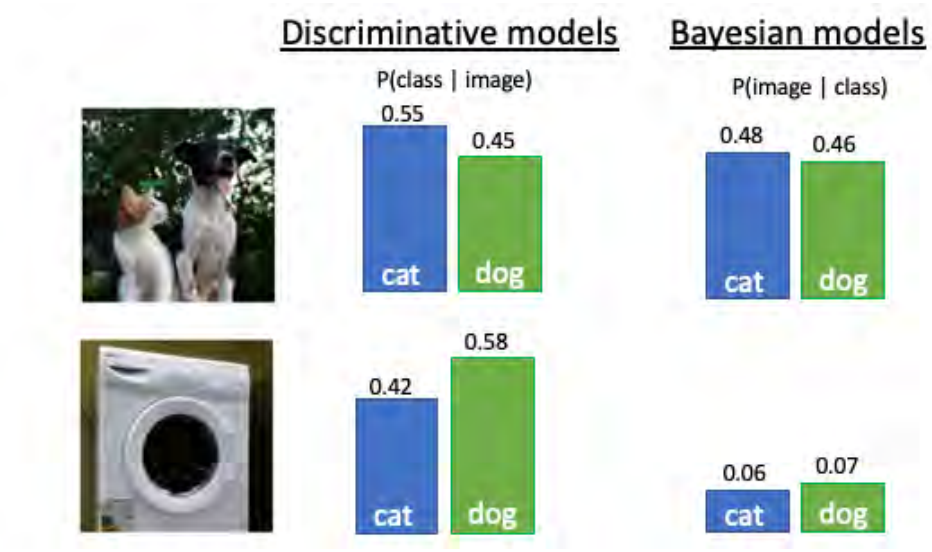


Figure 1.2: Discriminative models always classify an input image into one of the candidate classes thus performing counterintuitively on some images (the bottom row). Bayesian models estimate conditional likelihoods of the image and thus can model uncertainty more accurately. This example is adapted from [69]. Numbers shown above are illustrative.

manifold, they are fundamentally different from generative classifiers because these methods directly model $p(y|X)$ for classification instead of using the Bayes Rule.

There are several choices of generative models when building generative classifiers. Schott et al. uses variational autoencoders [58, 86] in their ABS model. In Chapter 2, we follow their direction and use an improved VAE model (adversarial autoencoder [72]) as the generative building block in generative classifiers. Meanwhile, Fetaya et al. [36] and Mackowiak et al. [69] use normalizing flows to build generative classifiers. Normalizing flows [31, 59] could compute the exact likelihood of an input, an advantage that generative classifiers may benefit from. However, compared with VAEs, normalizing flows are computationally expensive and susceptible to adversarial attacks [83].

Generative adversarial networks (GANs) [39] are strong generative models, achieving better reconstructions than VAEs in many domains. Recent studies showed that GANs are energy-based models with an implicit sample distribution [17, 4]. Nonetheless, better reconstructions does not necessarily imply better likelihood estimation [103]. In Chapter 3, we examine the challenges of GANs in generative classifiers more systematically.

1.3 Challenges: where is the limit of generative classifiers

Generative classifiers try to simultaneously achieve two objectives:

- Reconstruct in-domain images well: given an $(X, y) \sim \mathcal{D}$, the generator for class y should have a latent representation for X.
- Avoid reconstructing out-of-domain images: given an $(X, y) \sim \mathcal{D}$, any latent representation from the generator for class $y' \neq y$ should stay far away from X.

However, there are three problems that prohibit generative classifiers from achieving the two objectives.

Finding the best reconstruct of in-domain images is difficult. Some generative models suffer from mode loss [73], meaning that not all samples have corresponding latent representations. Even when such a latent representation exists, finding it could be difficult. We will present some challenges of finding an optimal latent representation in Chapter 3.

Generative models on Out-of-Domain samples Generative models are supposed to "reject" out-of-domain samples, but in practice, they can model out-of-domain samples equally well as in-domain samples [79, 78, 20]. Several studies found that generative models cover a wider range of data points than the data distribution they are trained on. Kos et al. demonstrated that a generative model trained to output digits have latent representations for shoes and shirts images [61]; furthermore, these out-of-domain latent representations have a distribution indistinguishable from natural latent representations. Several remedies have been proposed. Hendrycks et al. [45] expose the generative model to proxy OoD samples. Their scheme is simple and applicable to generative classifiers. Chapter 2 will present how we use outlier exposure to improve generative classifiers. Meanwhile, Nalisnick et al. [78] propose a typicality test for detecting OoD samples. Their method works well when the model can access multiple samples from the same distribution. Another method uses Watanabe Akaike Information Criterion (WAIC) to address this issue [20]. Although empirically effective, the authors acknowledge that this method does not prevent the issue in theory; furthermore, this method requires multiple models, which is costly at inference time.

Chapter 3 presents an analysis of this phenomenon on more complex images and more powerful generative models.

Similarity metrics used for measuring likelihood are unreliable. How close is a generated image $G(z)$ to the input X? This is a critical and fundamental question for generative classifiers. Simple metrics such as L_2 distance are inconsistent with human perception [124] and thus will cause the likelihood estimation of generative classifiers to be inaccurate. More recent studies rely on perceptual distance, which measure the similarity of

two images with features extracted by neural networks. Chapter 3 compares several choices of metrics to demonstrate that distance metrics are critical for generative models.

In summary, this thesis presents three studies on generative classifiers. In Chapter 2, we extended Schott et al.'s simple generative classifier to more complex image domains; the goal of this study is to show that we can improve the performance of generative classifiers by addressing some of the issues mentioned above. In Chapter 3, we focus on realistic images and state-of-the-art generative models; our goal is to analyze the three issues systematically with realistic images, showing the gap between generative classifiers and discriminative classifiers. Finally, Chapter 4 presents a study where we connect generative classifiers with discriminative classifiers; our method borrows ideas from randomized smoothing and formulates it as a generative approach, allowing us to borrow strength from both sides.

Chapter 2

E-ABS: Extending ABS to more complex image domains.

2.1 Introduction

Analysis-by-Synthesis (ABS) [95] is a generative classifier that has achieved state-of-the-art robustness on MNIST [64] against L_p bounded perturbations. Furthermore, studies indicated that ABS is more consistent with human perception: compared to other defenses, adversarial examples generated by ABS are more likely to fool human subjects [37]. Therefore, ABS opens a promising research direction on defending adversarial examples.

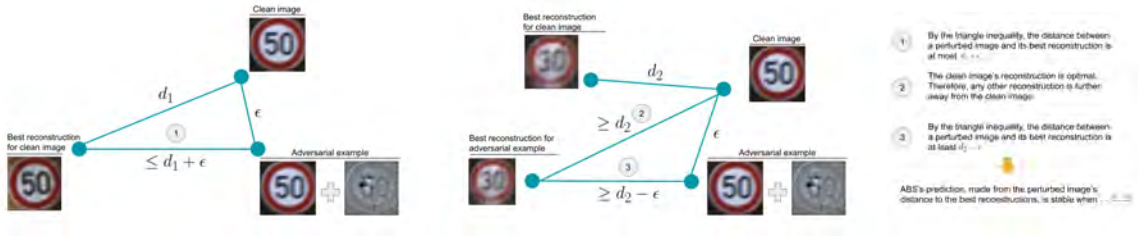


Figure 2.1: A simplified explanation of why ABS's predictions are stable under perturbations. Left: suppose a clean 50 image is distance d_1 away from its reconstruction using the model for class 50; then a perturbed image (of l_2 norm ϵ from the original) will be at most $d_1 + \epsilon$ away from its optimal reconstruction. Right: suppose the clean image is distance d_2 away from its reconstruction using the model for class 30; then the perturbed image will be at least $d_2 - \epsilon$ away from its optimal reconstruction. Therefore, the classification is stable when $\epsilon < (d_1 - d_2)/2$.

What is ABS?

ABS classifies with class-conditional likelihood estimates. Given a datum (x, y) where $x \in \mathcal{X} \subset \mathbb{R}^N, y \in \mathcal{Y} = \{1, \ldots, K\}$, [1]

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \propto P(x|y)P(y).$$

Accordingly, ABS esimates $P(x|k), k \in \mathcal{Y}$ and chooses the class with highest likelihood as its prediction.

Schott et al.'s ABS uses variational autoencoders (VAEs) [58, 86] for class-conditional likelihood estimation. VAEs use variational inference to estimate $P(X)$ [2]. Given a variational distribution $Q(Z)$ where $Z \in \mathcal{Z} = \mathbb{R}^M$ is a latent representation, we have a lower bound of $P(x)$ from

$$\log P(X) - \mathrm{D}_{\mathrm{KL}}[Q(Z)\|P(Z|X)] = E_{Z\sim Q(Z)}[\log P(X|Z)] - \mathrm{D}_{\mathrm{KL}}[Q(Z)\|P(Z)] \tag{2.1}$$

where $\mathrm{D}_{\mathrm{KL}}[\cdot\|\cdot]$ is KL-divergence. Since KL-divergence is non-negative, the right side is a lower bound for $P(X)$ known as the Evidence Lower Bound (ELBO).

The choice of $Q(Z)$ is arbitrary, but better $Q(Z)$ gives a tighter bound. VAEs use an encoder to propose a variational distribution $Q(Z|X)$ and a decoder to estimate $\log P(X|Z)$. The encoder maps an image $x \in \mathcal{X}$ to the parameters of $Q(Z|x)$; typically $Q(Z|x)$ is a multivariate Gaussian, and thus the encoder outputs a mean vector and a variance vector. The decoder maps a latent vector $z \in \mathcal{Z}$ back to the input space $\mathcal{R}^N$; the output is viewed as the mean of a Gaussian distribution, so $\log P(x|z)$ becomes $\|x - G(z)\|_2^2$ where $G(z)$ is the reconstructed image.

Given a K-class classification task, ABS trains K class-specific VAEs; the VAE for class $k \in \mathcal{Y}$ maximizes in-distribution sample likelihood by optimizing the ELBO objective (2.1) on $\{(x, y)|y = k\}$. At test time, as encoders are deep neural networks that are susceptible to attack, ABS replaces the encoder with an optimization step and estimates the class-conditional likelihood $\log P(x|k)$ as

$$\max_{z\in\mathcal{Z}} \left[\|x - G_k(z)\|_2^2 - \beta\, \mathrm{D}_{\mathrm{KL}}[\mathcal{N}(z, \mathbf{1})\|\, \mathcal{N}(\mathbf{0}, \mathbf{1})]\right] \tag{2.2}$$

where $\mathcal{N}(\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and variance Σ, and β is a hyperparameter [47]. Denote the optimal z in (2.2) with z^*. Inference uses the variational distribution $\mathcal{N}(z^*, \mathbf{1})$ instead of the distribution $Q(Z|x)$ given by the encoder. It avoids encoders and thus is more robust to adversarial perturbations.

Conceptually, ABS learns K class-specific data manifolds and classifies an input x by its distance to these manifolds. Figure 2.1 shows intuition on why ABS's predictions are stable under small perturbations. When the learned manifolds are good representations of the real data distribution, ABS pushes adversarial examples towards the human-perceptual decision boundary [95].

[1] In this chapter, we use X, Y to represent random variables, x, y to represent data, and $\mathcal{X}, \mathcal{Y}$ to represent sets.

[2] In ABS, $P(X)$ becomes $P(X|Y)$ for a class-specific VAE.

ABS's limitations

Despite state-of-the-art robustness on MNIST, ABS fails on more challenging datasets such as SVHN [80] and CIFAR10 [62]. Schott et al. pointed out that ABS has low clean accuracy on CIFAR10. Fetaya et al. found that behaviors of ABS-like models are different from MNIST on CIFAR10 and these undesired behaviors cause underperformance on CIFAR10. Based on their observations, Fetaya et al. claimed that ABS-like models are ineffective classifiers on complex images [36].

ABS's limitations are limitations of generative models. Generative models are the building block of ABS. Given a K-class classification task, ABS learns K class-conditional data distributions with generative models. At inference time, ABS estimates the input's conditional likelihood of each class and classifies with Bayes' rule. Therefore, ABS needs high-quality conditional likelihood estimates. However, several studies suggest that generative models may give unreliable estimates to out-of-distribution samples on complex datasets [79, 45, 20]. This explains Fetaya et al.'s observation, where ABS-like models give a high likelihood to an interpolation of two images. Besides, variational autoencoders [58], the generative model used by Schott et al., could fail to learn a distribution of latent vectors that matches the prior [89, 27]; this also undermines ABS's performance.

How E-ABS addresses the limitations

E-ABS introduces three extensions to address these issues. First, we use adversarial autoencoders [72] to improve estimates for in-distribution samples. Second, we optimize a variational distribution at inference time. Third, we introduce a discriminative loss that uses outlier exposure [45] to improve the model's estimates for out-of-distribution samples. These extensions improve E-ABS's clean accuracy and robust accuracy on datasets that are more complex than MNIST while retaining ABS's certified robustness.

Empirically, we show that E-ABS outperforms adversarial training [70] and ABS [95] on several real-world datasets. We run extensive experiments on two simple datasets (MNIST [64] and Fashion MNIST [112]) and two more challenging real-world datasets (SVHN [80] and a dataset of European traffic signs). We measure robustness against a wide range of attacks with carefully chosen parameters. Results suggest that E-ABS preserves ABS's performance on simple datasets and sets a new state-of-the-art on SVHN and traffic signs superior to prior work.

The rest of this chapter is organized as follows: Section 2.2 explains E-ABS's extensions; Section 2.3 and Section 2.4 present E-ABS's implementation and experiments where we compare E-ABS with other baseline defenses on four datasets. Section 2.5 is a discussion on some future research directions to improve E-ABS.

2.2 E-ABS Description

Generative Models

We use adversarial autoencoders (AAE) [72] to estimate class-conditional probabilities, because VAEs may fail to match the prior $P(z)$ and give unreliable likelihood estimates [27, 89], AAE uses a discriminator D to distinguish latent vectors encoded from input images from vectors sampled from the prior. AAE trains the encoder and discriminator like a generative adversarial network (GAN) [39], pushing the encoder's marginal distribution $Q(Z)$ to match the prior $P(Z)$ [72].

We denote the discriminator's output with $D(z)$. $D(z)$ is the probability that an input z is sampled from the prior so $0 \leq D(z) \leq 1$. Accordingly, the objective for training AAE's encoder and decoder is to minimize

$$\mathop{\mathbb{E}}_{(x,y)\sim P(X,Y)} \mathop{\mathbb{E}}_{z\sim Q_\phi(Z|x)} [c(x, G_\theta(z)) - \beta \log D_\eta(z)] \tag{2.3}$$

where θ, ϕ, η denote model parameters, $c(\cdot, \cdot)$ is a cost function such as squared error, and β is a hyperparameter. Similar to GANs, the objective for training discriminators is to minimize

$$\mathop{\mathbb{E}}_{x\sim P(X)} \mathop{\mathbb{E}}_{\substack{z\sim Q_\phi(Z|x) \\ \tilde{z}\sim P(Z)}} -(\log D_\eta(\tilde{z}) + \log(1 - D_\eta(z)))$$

Although AAEs do not give an explicit probability estimation like VAEs, they implicitly minimize the Wasserstein distance [2, 10] between the learned $P_\theta(X)$ and the prior $P(X)$. Tolstikhin et al. showed that AAE's objective (2.3) is a relaxed version of the optimization

$$\inf_{Q:Q_Z=P_Z} \mathop{\mathbb{E}}_{x\sim P(X)} \mathop{\mathbb{E}}_{z\sim Q(Z|x)} [c(x, G(z))]$$

which is the Wasserstein distance between $P_\theta(X)$ and $P(X)$ under a cost c [105]. Therefore, (2.3) measures the distance of the input image to the learned manifold under a cost, and we use (2.3) for classification in E-ABS. In our experiments, we choose L_2 distance as our cost function, which means $c(x, y) = \|x - y\|_2^2$.

Discriminative Loss

A discriminative loss is used at training time to expose conditional generative models to out-of-distribution samples. Hendrycks et al. showed that outlier exposure fixes generative models' undesired behavior on out-of-distribution samples [79, 20, 45], improving the quality of likelihood estimates [45]. Specifically, we minimize E-ABS's cross-entropy with respect to class-conditional likelihood estimates. This loss facilitates each conditional AAE to recognize out-of-distribution samples and avoid giving high likelihood estimates to these samples.

The discriminative loss necessitates a structural change where class-specific encoders are replaced by a shared encoder. With class-specific encoders, a discriminative loss hinders each

encoder to match the marginal distribution of latent vectors because the loss encourages a higher discriminator loss $-\log D(z)$ for OoD samples. To address this, we use the same encoder for all classes; because all samples are in-distribution for the encoder, this issue does not arise. Formally, given a datum (x, y) and $l_k(x)$ denotes the model's estimate (2.3) for a class $k \in \mathcal{Y}$, the discriminative loss is defined as

$$\begin{aligned} & -\log \frac{e^{-l_y(x)}}{\sum_{k\in\mathcal{Y}} e^{-l_k(x)}} \\ = & - \mathop{\mathbb{E}}_{z\sim Q_(Z|x)} \log \frac{e^{-c(x,G_y(z))+\beta\log D(z)}}{\sum_k e^{-c(x,G_k(z))+\beta\log D(z)}} \\ = & - \mathop{\mathbb{E}}_{z\sim Q_(Z|x)} \log \frac{e^{-c(x,G_y(z))}}{\sum_k e^{-c(x,G_k(z))}} \end{aligned} \tag{2.4}$$

Because of the shared encoder, the effect of the discriminator $D(z)$ has cancelled out, so with this change to the architecture, the discriminative loss no longer encourages the encoder to produce latent vectors far away from the prior for OoD samples.

Combining (2.3) and (2.4), the training objective for encoders and decoders in E-ABS is

$$\mathop{\mathbb{E}}_{(x,y)\sim P(X,Y)} \mathop{\mathbb{E}}_{z\sim Q_\phi(Z|x)} c(x, G_{\theta_y}(z)) - \beta \log D_\eta(z) - \gamma \log \frac{e^{-c(x,G_{\theta_y}(z))}}{\sum_{k\in\mathcal{Y}} e^{-c(x,G_{\theta_k}(z))}} \tag{2.5}$$

where β and γ are hyperparameters. Algorithm 1 summarizes the training method. In practice, we update discriminators and encoders/decoders in an interleaved fashion; we choose $\beta = 1$ and $\gamma = 10$ for all datasets.

Variational Inference

ABS-like models estimate the likelihood for each class through an optimization process in the latent space that maximizes the likelihood estimate. Schott et al. fix the variance of the variational distribution $Q(Z)$ during this optimization and optimize its mean. Therefore, the KL divergence term drives latent vectors towards the origin, moving away from the Gaussian prior's typical set [78]. For AAE, such an inference method leads to outlier latent vectors that significantly deviate from the prior because AAE's discriminator is a non-smooth neural network. These are undesired behaviors that undermine the model's performance.

To address this issue, we optimize both mean and variance of the variational distribution $Q^*(Z)$ at test time. Formally, we use gradient methods to find

$$\min_{\mu,\Sigma} \mathop{\mathbb{E}}_{z\sim\mathcal{N}(\mu,\Sigma)} [c(x, G(z)) - \beta \log D(z)] \tag{2.6}$$

The reparameterization trick [86] allows us to optimize the variational distribution's parameters μ and Σ directly.

Algorithm 1 Training E-ABS.

Inputs: Hyperparameters $\beta > 0$, $\gamma > 0$.
Initialize parameters of the encoder Q_ϕ, the discriminator D_η, and K decoders $G_{\theta_1}, \ldots, G_{\theta_K}$.

repeat
Sample $(x_1, y_1), \ldots, (x_n, y_n)$ from the training set.
Sample $\tilde{z}_1, \ldots, \tilde{z}_n$ from the prior $P(Z)$.
Sample z_i from $Q_\phi(Z|x_i)$ for $i = 1, \ldots, n$.
Update ϕ and θ_i for $i = 1, \ldots, K$ by descending

$$\frac{1}{n}\sum_{j=1}^{n} c(x_i, G_{\theta_{y_i}}(z_i)) - \beta \log D_\eta(z_i) - \gamma \log \frac{e^{-c(x_i, G_{\theta_{y_i}}(z_i))}}{\sum_{k=1}^{K} e^{-c(x_i, G_{\theta_k}(z_i))}}$$

Update η by descending

$$-\frac{\beta}{n}\sum_{i=1}^{n} \log D_\eta(\tilde{z}_i) + \log\left(1 - D_\eta(z_i)\right)$$

until convergence.

To avoid local minima for the optimization process, Schott et al. start the optimization from the best point out of 8000 random vectors. Similarly, we sample 8000 variational distributions that are parameterized by random means and unit variance. We also include $Q_\eta(Z|x)$, the encoder's output, as a candidate. Therefore, we choose the best from $8000 + 1$ variational distributions as the starting point of the optimization.

When optimizing (2.6), we use one sample to compute the expectation. Importance weighted sampling gives better estimations with more samples [13]. In practice, we find that a one-sample approximation is sufficient for reliable estimation and is more efficient than importance weighted sampling.

Lower Bounds for the Robustness of E-ABS

Using the same technique from Schott et al., we can deduce a lower bound for the distance to the nearest adversarial examples for E-ABS. For simplicity, we analyze L_2 bounded perturbations and use an exact z instead of a variational distribution $Q(z)$. We analyze the case where $c(x, y) = \|x - y\|_2^2$.

Given an input $x \in \mathcal{X}$ and a class $k \in \mathcal{Y}$, our estimate of $-\log P(x|k)$ is given by

$$l_k^*(x) = \min_{z \in \mathcal{Z}} \left[\|x - G_k(z)\|_2^2 - \beta \log D(z)\right].$$

Let z_k^* denote the optimal z for class k. Given a perturbation δ where $\|\delta\|_2 \leq \epsilon$, under certain conditions, we have

$$(d_k(x) - \epsilon)^2 \leq l_k^*(x+\delta) \leq l_k^*(x) + 2\epsilon\|x - G_k(z_k^*)\|_2^2 + \epsilon^2$$

where $d_k(x) = \min_{z\in\mathcal{Z}} \|x - G_k(x)\|_2$. As ABS-like models make predictions with $\arg\min_k l_k^*(x)$, adversarial perturbations increase $l_y^*(x)$ while decreasing $l_k^*(x)$, and the optimal perturbation is achieved when

$$l_y^*(x) + 2\epsilon\|x - G_y(z_y^*)\|_2^2 + \epsilon^2 = (d_k(x) - \epsilon)^2$$

for some k. Therefore, we have the following lower bound on ϵ:

$$\epsilon^* = \min_{k\in\mathcal{Y}} \frac{d_k^2(x) - l_y^*(x)}{2d_k(x) + 2\|x - G_y(z_y^*)\|_2^2} \tag{2.7}$$

(2.7) suggests that robustness improves when $l_y^*(x)$ decreases and $d_k(x)$ increases, which has a direct connection to the goodness of generative models. On the one hand, $l_y^*(x)$ is lower when generative models can model in-distribution samples well. On the other hand, increasing $d_k(x)$ means that the learned manifold $\{G_k(z)|z \in \mathcal{Z}\}$ for class k is away from samples from other classes.

Proving E-ABS's robustness lower bound

Our analysis is similar to Schott et al.'s analysis under L_2 attacks [95]. The main difference is that the discriminator loss $-\beta \log D(z)$ is non-negative, which simplifies our conclusions.

Assumptions and Notations For simplicity, we study exact inference and L_2 attacks. Exact inference means that we find

$$z^* = \arg\min \|x - G(z)\|_2^2 - \beta \log D(z)$$

where $D(z)$ is discriminator's output and $0 \leq D(z) \leq 1$; β is a hyperparameter and $\beta > 0$.

A Bound of the Distance of Adversarial Examples for E-ABS

Given an input x and any class k, E-ABS estimates the class-conditional negative log-likelihood with

$$l_k^*(x) = \min_z \|x - G_k(z)\|_2^2 - \log D(z) \tag{2.8}$$

Given a perturbation δ where $\|\delta\|_2 = \epsilon$, we want to find a lower bound of ϵ for δ to change E-ABS's prediction from y, the ground truth, to k, a specific class $k \neq y$.

First, we show that

Claim 1.

$$l_y^*(x+\delta) \leq l_y^*(x) + 2\epsilon\|x - G_y(z^*)\|_2 + \epsilon^2$$

where $z^* = \arg\min_z \|x - G_y(z)\|_2^2 - \log D(z)$.

Proof.

$$\begin{aligned}
l_y^*(x+\delta) =& \min_z \|x+\delta - G_y(z)\|_2^2 - \beta \log D(z) \\
\leq& \|x+\delta - G_y(z^*)\|_2^2 - \beta \log D(z^*) \\
=& \|x - G_y(z^*)\|_2^2 + 2\delta^T(x - G_y(z^*)) + \epsilon^2 - \beta \log D(z^*) \\
\leq& \|x - G_y(z^*)\|_2^2 + 2\epsilon\|x - G_y(z^*)\|_2 + \epsilon^2 - \beta \log D(z^*) \\
=& l_y^*(x) + 2\epsilon\|x - G_y(z^*)\|_2 + \epsilon^2
\end{aligned}$$

□

Second, we show that

Claim 2. *Let* $d_k(x) = \min_z \|x - G_k(z)\|_2^2$ *and assume* $\epsilon < d_k(x)$,

$$l_k^*(x+\delta) \geq (d_k(x) - \epsilon)^2$$

Proof.

$$l_k^*(x+\delta) = \min_z \|x+\delta - G_k(z)\|_2^2 - \log D(z)$$

Since $0 \leq D(z) \leq 1$ and $\beta > 0$, we know

$$\begin{aligned}
l_k^*(x+\delta) \geq& \min_z \|x+\delta - G_k(z)\|_2^2 \\
=& \min_z \|x - G_k(z)\|_2^2 + 2\delta^T(x - G_k(z)) + \epsilon^2 \\
\geq& \min_z \|x - G_k(z)\|_2^2 - 2\epsilon\|x - G_k(z)\|_2 + \epsilon^2
\end{aligned}$$

By the definition of $d_k(x)$, we know

$$\|x - G_k(z)\| \geq d_k(x).$$

Therefore, when $d_k(x) > \epsilon$,

$$\begin{aligned}
l_k^*(x+\delta) \geq& d_k^2(x) - 2\epsilon d_k(x) + \epsilon^2 \\
=& (d_k(x) - \epsilon)^2
\end{aligned}$$

□

We know from the proof that when $\epsilon > d_k(x)$, $l_k^*(x+\delta) \geq 0$.

With Claim 1 and Claim 2, we can find a lower bound for ϵ from

$$l_y^*(x) + 2\epsilon\|x - G_y(z^*)\|_2 + \epsilon^2 = (d_k(x) - \epsilon)^2$$

and the lower bound for an untargeted attack is the minimal of all bounds, which gives

$$\epsilon^* = \arg\min_k \frac{d_k^2(x) - l_y^*(x)}{2d_k(x) + 2\|x - G_y(z^*)\|_2}$$

This bound holds when $\epsilon < d_k(x)$, that is, adversarial perturbations are not large enough to move x to its best reconstruction $G_k(z^*)$ for some class $k \neq y$.

An interesting observation is that Claim 1 reaches the optimal value when δ has the opposite direction as $x - G_y(x)$ and Claim 2 reaches the optimal value when δ has the same direction as $x - G_k(x)$. This suggests that our gradient-based attacks that use

$$\frac{\partial l(x, z^*)}{\partial x_i} = 2 * (x_i - G_{ij}(z^*))$$

to compute gradients are consistent with the theoretical analysis presented in this section. Schott et al.'s LDA attack is closely related to this observation as well, as the LDA attack searches adversarial examples along the direction $G_k(x) - x$.

2.3 Experiments

Datasets

We evaluate our models on four datasets. At training time, we augment datasets with additive Gaussian noise, except for adversarially trained models. We use a random 10% of each dataset's training set as a validation set.

MNIST is a dataset of handwritten digits [64]. MNIST has a clean background and binarized values, making it naturally robust against some adversarial perturbations. Schott et al. showed that binarized CNN, a simple extension to CNN models that exploits the binarized value distribution, can achieve robustness comparable with Madry et al.'s adversarially trained models [95].

Fashion MNIST is a dataset proposed by Xiao et al. as an MNIST alternative [112]. Previous studies suggest that Fashion MNIST is more challenging than MNIST. Therefore, we use Fashion MNIST to complement our comparison on simple datasets.

SuperTraffic-10 is a European traffic sign dataset composed of three datasets: the German Traffic Sign (GTS) dataset [100], the DFG traffic sign dataset [102], and the Belgium Traffic Sign (BTS) dataset [104]. We merge the three datasets because they all contain European traffic signs. We filter out small images (images that are smaller than 32×32 pixels) and choose the top 10 classes with the most images. All images in SuperTraffic-10 are 32×32 RGB images, making it a more complex dataset than MNIST or Fashion MNIST.

The Street View House Numbers (SVHN) dataset is a real-world dataset of digits [80]. SVHN is more challenging than MNIST because its images are 32×32 RGB images, and its images are taken from the real world.

Attacks

We evaluate model robustness against gradient-based attacks. ABS-like models use optimization at inference time, which makes it difficult for attacks to computing gradients. We propose an adaptive method to compute gradients. This method is an extension of Schott et al.'s Latent Descent Attack customized for ABS [95]. Furthermore, we confirm on SVHN with gradient-free attacks that E-ABS does not have obfuscated gradients [3].

All attacks are implemented with Foolbox [85].

An adaptive method to compute gradients. Adaptive attacks are necessary when evaluating model robustness [15, 106]. ABS-like models have convoluted gradients because they run several iterations of optimization at inference time. Therefore, adaptive methods to compute gradients are necessary to evaluate the robustness of these models properly.

We compute gradients from the optimal variational distribution $Q^*(Z)$ and exclude the optimization process from gradients. Specifically, given z^* sampled from the optimal $Q^*(Z)$, the reconstruction loss is the L_2 distance between x and $G(z^*)$. Therefore, its gradient is given by

$$\frac{\partial l(x, z^*)}{\partial x_i} = 2\left(x_i - G_i(z^*)\right) \tag{2.9}$$

where i indexes the location of a pixel in the image.

This gradient method is efficient and effective. Projected gradient descent [63, 70] using this gradient method extends and improves Schott et al.'s Latent Descent Attack [95] on ABS-like models. LDA searches for adversarial examples in the direction of the closest out-of-distribution reconstruction. A PGD attack with gradients given by (2.9) also pushes the adversarial example towards the best OoD reconstruction.

Unlike many other models, E-ABS's classification decision is randomized thanks to its use of the reparameterization trick at inference time. We use expectation over transformation [3], with 5 samples per batch, to deal with this randomness. Section 2.4's experiments suggest that 5 samples are sufficient to stabilize gradient-based attacks.

Gradient-based attacks. We use two gradient-based L_∞ attacks:

- (**PGD**) Projected gradient descent attack [63, 70] with 80 steps. We use 5 random starts for MNIST and Fashion MNIST, and 20 random starts for SuperTraffic-10 and SVHN.
- (**DeepFool**) DeepFool attack [76] with 100 steps.

We use L_∞ PGD to choose attack parameters. In Section 2.4, supplementary experiments suggest that 20 random starts and 80 steps are sufficient.

We use four gradient-based L_2 attacks:

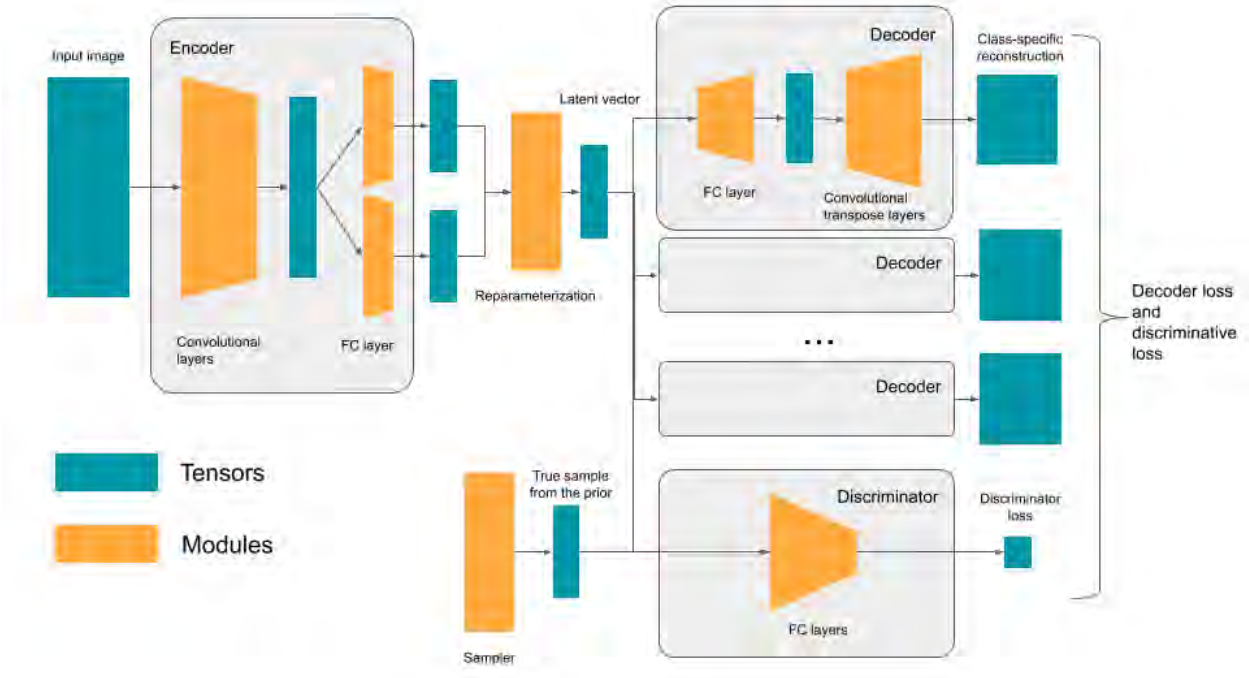


Figure 2.2: E-ABS's structure. E-ABS has one shared encoder with multiple decoders. The discriminator distinguishes vectors sampled from $Q(Z|X)$ from vectors sampled from $P(Z)$.

- (**PGD**) Projected gradient descent attack with the same parameters as L_∞ PGD.
- (**DDN**) Decoupled direction and norm attack [88] with 80 steps. DDN is an extension of the PGD attack.
- (**CW**) Carlini-Wagner attack [14] with 5 binary searches and 100 steps.
- (**DeepFool**) DeepFool attack with the same parameters as L_∞ DeepFool.

Furthermore, we include an attack that adds Gaussian noise with increasingly large standard deviation. Previous studies suggest that a model's performance under additive Gaussian noises is correlated with a model's robustness against L_2 attacks [35].

Gradient-free attacks. We use two gradient-free attacks: a Boundary attack [11] and a PGD attack with gradient estimation. We use 100000 iterations for the Boundary attacks and 50 steps for the PGD attack with gradient estimation. We initialize the Boundary attack with a random sample that is classified as a different class from the ground truth.

Models

We run attacks against E-ABS and two baseline defenses: Schott et al.'s ABS model and an adversarially trained model. Our E-ABS model uses a convolutional encoder and decoder; the discriminator is a two-layer feed-forward network. Figure 2.2 shows E-ABS's structure. The other models share E-ABS's modules. ABS uses the same model structure as E-ABS, but does not have discriminators. CNNs add a linear layer after E-ABS's encoder for classification. Adversarially trained CNNs are tuned to have comparable clean data accuracy with E-ABS. Because adversarially trained CNNs under L_∞ attacks do not generalize to L_2 attacks [95], we include both Adv-L_∞, an adversarially trained model with L_∞ attacks, and Adv-L_2, an adversarially trained model with L_2 attacks.

In our experiments, the E-ABS encoder uses 4 convolutional layers followed by two parallel fully-connected layers that compute the variational distribution's parameters. All convolutional layers use batch normalization [50] and Leaky ReLU activation [115]. We use dropout [99] after the last convolutional layer. The decoder uses a fully-connected layer followed by convolution transpose layers. The fully-connected layer's output has the same size as the first convolution transpose layer's output depth. We use dropout [99] after the fully-connected layer. All convolution transpose layers, except the last layer, use batch normalization [50] and ReLU activation [77]; the last layer uses sigmoid. Table 2.1 shows more details of the architecture we use in our experiments.

We use Adam [57] and batch size 512 to train all models. We use an initial learning rate 0.001 for CNN, ABS, and E-ABS, and halve the learning rate every 200 epochs. We train CNN, ABS, and E-ABS for 500 epochs, except for ABS and E-ABS on SVHN, where we train 800 epochs. For the adversarially trained model, we use the pre-trained CNN model and retrain the model for 200 epochs with a learning rate 0.0001 and no learning rate decay.

At training time, we augment data with additive Gaussian noise. We use a standard deviation 0.2 for MNIST and Fashion MNIST, and 0.01 for SuperTraffic-10 and SVHN.

Ablation Study

E-ABS consists of three separate extensions to ABS. We present an ablation study to demonstrate that all three extensions are necessary. We denote the three extensions with A (for AAE-based models), D (for models with a discriminative loss and shared encoder), and V (for models that use variational inference). This ablation study examines all combinations of the three extensions on SVHN with L_∞ and L_2 PGD attacks.

When training A-ABS and AV-ABS models, we find that training discriminators with both in-distribution samples and out-of-distribution samples improves the model's performance. This way, class-specific discriminators see latent vectors encoded from not only in-distribution images but also out-of-distribution images.

Metrics

We report each model's clean accuracy and accuracy under bounded perturbations. Unless otherwise specified, we choose 1000 random test samples when reporting robustness results. We run McNemar's test [30] on every pair of models to test whether the difference in their performance is statistically significant; McNemar's test is a pairwise test with a null hypothesis that none of the models performs better than the other. We choose $\alpha = 0.01$. Besides results under each attack, we report the model's accuracy under the combination of all attacks of the same type.

Model parameters

Table 2.1 shows parameters for E-ABS. The other models (CNN, Adv-L_∞, Adv-L_2, and ABS) share the same modules.

2.4 Results

E-ABS extends ABS to complex image domains.

On SuperTraffic-10 and SVHN, E-ABS outperforms ABS, as shown in Table 2.4 and Table 2.5. Specifically, ABS is only 45% accurate on SVHN, and E-ABS increases clean data accuracy to 88%, which is comparable with an unprotected CNN (90%). These results suggest that E-ABS can classify complex images and maintain high robustness.

E-ABS outperforms adversarially trained CNNs on SuperTraffic-10 and SVHN as well, achieving state-of-the-art robustness on these datasets.[3] Also, E-ABS is robust against both L_∞ and L_2 attacks, while adversarially trained CNNs are robust only against one type of attack.

On MNIST and Fashion MNIST, E-ABS's accuracy is comparable with unprotected CNNs, and its robustness is comparable with baseline models, as shown in Table 2.2 and Table 2.3. E-ABS outperforms ABS on MNIST, but the two models are not directly comparable on Fashion MNIST: E-ABS has better clean data accuracy, comparable L_2 robustness, and worse L_∞ robustness than ABS.

On MNIST, our results are mostly consistent with [95]. Schott et al.'s ABS has better clean data accuracy than ours. In comparison, our ABS baseline uses the same structure as E-ABS, which gives it more capacity and dropout layers. These structural differences might explain the differences between our numbers and numbers reported by Schott et al. on MNIST. Another difference is that we augment data with Gaussian noise; this could explain why our unprotected CNN has better robustness than Schott et al. on MNIST.

[3]The best-published result on SVHN that we know of is 55.59% accuracy under PGD-L_∞ attack with $\epsilon = 8/255$ [42]; E-ABS achieves 58%. SuperTraffic-10 is created by us, but GTS [100], one of the three datasets included in SuperTraffic-10's, has the best known result of 67.9% under PGD-L_2 attack with $\epsilon = 0.2$ [22], according to robust-ml.org. E-ABS's accuracy is 85% under the same attack.

Table 2.1: The model parameters for each dataset.

MNIST and Fashion MNIST								Latent dimensions: 10
Encoder				Decoder				Discriminator
Channels	Kernel	Stride	Padding	Channels	Kernel	Stride	Padding	Neurons
16	5	2	2	32	4	1	0	256
32	5	2	2	32	5	2	0	128
64	5	2	2	16	5	2	0	
64	4	1	1	8	4	1	0	
				1	1	1		

SuperTraffic-10								Latent dimensions: 16
Encoder				Decoder				Discriminator
Channels	Kernel	Stride	Padding	Channels	Kernel	Stride	Padding	Neurons
16	5	2	2	64	4	1	0	256
32	5	2	2	64	4	2	1	128
64	5	2	2	32	4	2	1	
128	4	1	0	16	4	2	1	
				3	1	1	0	

SVHN								Latent dimensions: 40
Encoder				Decoder				Discriminator
Channels	Kernel	Stride	Padding	Channels	Kernel	Stride	Padding	Neurons
32	5	2	2	64	4	1	0	512
64	5	2	2	64	4	2	1	256
128	5	2	2	32	4	2	1	
128	4	1	0	16	4	2	1	
				3	1	1	0	

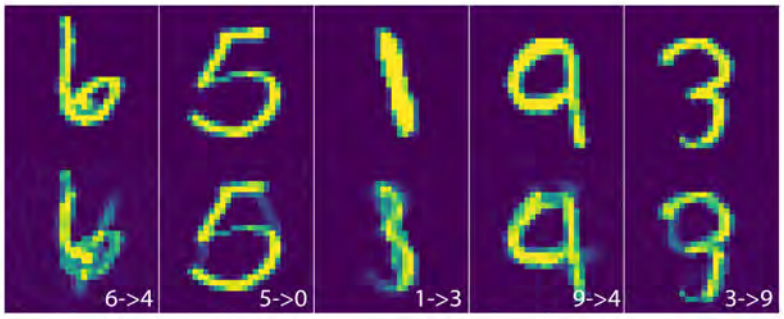


Figure 2.3: E-ABS's adversarial examples on MNIST under PGD-L_2 attack. Top: Clean images. Bottom: Adversarial examples.

Figure 2.4: E-ABS's adversarial examples on SVHN under PGD-L_2 attack. Top: Clean images. Bottom: Adversarial examples.

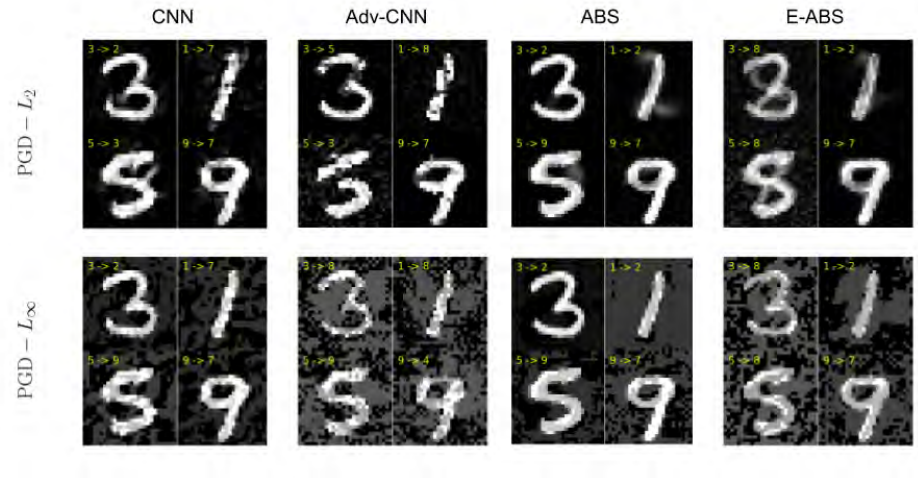


Figure 2.5: Adversarial examples on MNIST.

On MNIST, E-ABS preserves the desired property of ABS that adversarial examples are semantically consistent with human perception. Figure 2.3 shows some adversarial examples of E-ABS on MNIST under a PGD-L_2 attack. However, this property is less clear on complex datasets, such as SVHN, as shown in Figure 2.4.

Figure 2.5 to Figure 2.8 show adversarial examples on MNIST, Fashion MNIST, SuperTraffic-10, and SVHN. We generate examples with PGD-L_2 and PGD-L_∞. We compare four models: CNN, Adv-L_∞, ABS, and E-ABS. For each dataset, we choose four images to present.

In summary, results from Table 2.2 to Table 2.5 suggest that

- E-ABS has better clean data accuracy than ABS on both complex and simple datasets.
- Compared with ABS, E-ABS has comparable robustness on simple datasets and better robustness on complex datasets.
- E-ABS outperforms all baselines on SuperTraffic-10 and SVHN, providing a new state-of-the-art on these datasets.
- E-ABS's clean data accuracy is comparable with an unprotected CNN on all datasets except for SuperTraffic-10.

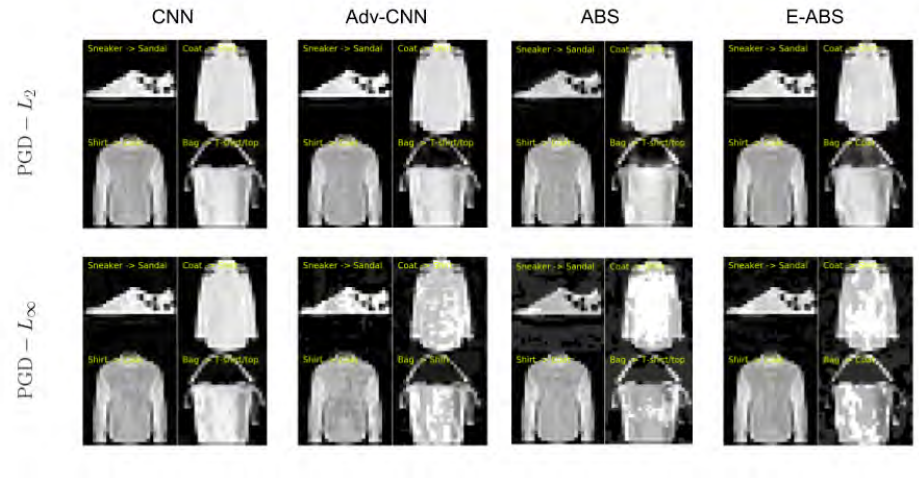


Figure 2.6: Adversarial examples on FMNIST.

All three extensions are necessary for E-ABS.

Table 2.6 shows results of the ablation study of three improvements proposed in Section 2.2. For efficiency, we run experiments on SVHN and compare PGD-L_∞ and PGD-L_2 attacks. Table 2.6 presents these results.

Table 2.6 suggests that no single technique can significantly improve ABS's performance. AV-ABS and DV-ABS outperform ABS in terms of clean data accuracy, but cannot compete with E-ABS. In particular, we see that AAE-based models have a much worse accuracy on clean data with Schott et al.'s inference (A-ABS and AD-ABS). We hypothesize that Schott et al.'s inference leads to latent vectors away from the Gaussian prior's typical set [78].

Our gradient-based attacks are sufficient for evaluating E-ABS.

In this section, we present some supplementary expeirments to justify our choice of parameters for gradient-based attacks. All experiments use E-ABS model.

Expectation-over-transformations

We test a range of runs for EOT. We evaluate the model's accuracy under a L_∞ PGD attack with 5 random starts on SVHN. Table 2.7 presents experiment results.

Table 2.7 suggests that 5 runs are enough to stabilize model outputs. Specifically, with fewer runs, the model has seemingly better robustness because it may misclassify due to

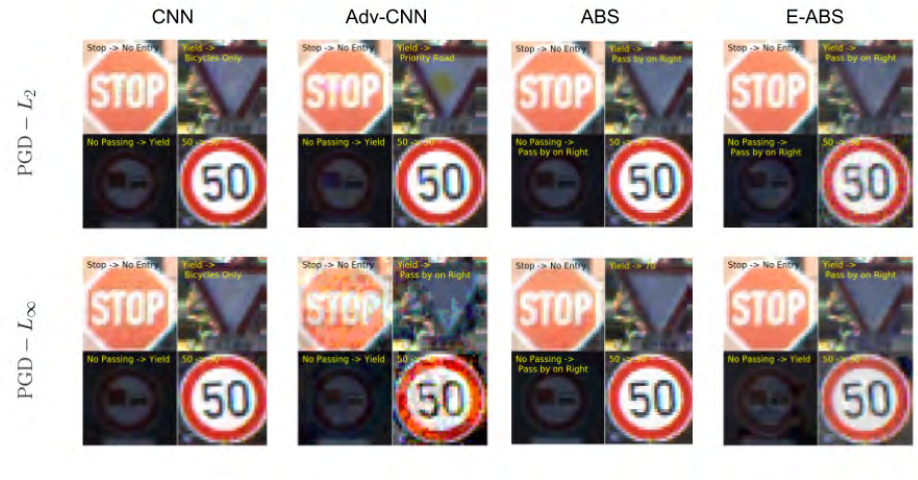


Figure 2.7: Adversarial examples on SuperTraffic-10.

inference-time randomness, which leads an attack to believe that it has succeeded falsely.

Number of PGD random starts

We run experiments to decide the number of random starts necessary for PGD attacks. We evaluate the model's accuracy under a L_∞ PGD attack with 100 steps. We use 3 runs for EOT for efficiency. Table 2.8 presents these experiments.

Based on Table 2.8, we choose 20 random starts for our PGD attacks on SVHN and SuperTraffic-10.

PGD steps

We evaluate the model's robustness under L_∞ PGD attacks to choose a proper number of PGD steps. We use 20 random starts for all PGD attacks, 3 runs for EOT, and 200 random samples from SVHN. We use less EOT runs and random samples for efficiency concerns. Table 2.9 shows the results.

Table 2.9 suggests that PGD attacks achieve the best results with 50 steps. We choose 80 steps in our experiments.

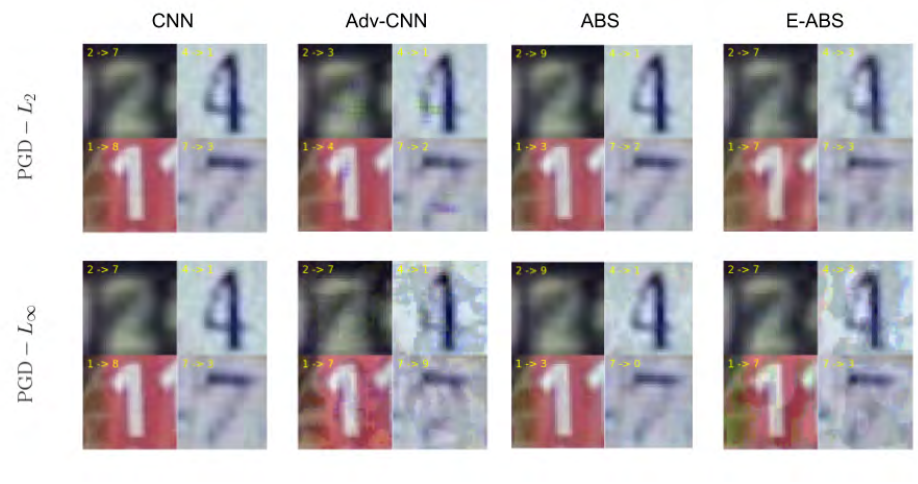


Figure 2.8: Adversarial examples on SVHN.

2.5 Discussion

This paper presents E-ABS, an extension to Schott et al.'s ABS model. E-ABS has state-of-the-art robustness on SVHN and a traffic sign dataset, beating Madry et al.'s adversarial training [70]. Compared with ABS, E-ABS achieves significantly better clean accuracy, comparable with unprotected CNN models on the two real-world datasets. Therefore, E-ABS has successfully addressed the shortcomings of ABS observed by Fetaya et al. [36], suggesting that robust classification with conditional generative models is a promising research direction.

Despite the improvements, E-ABS still has limitations on more complex datasets. On CIFAR10 [62], E-ABS achieves 60% clean accuracy, leaving a significant gap to state-of-the-art convolutional models. Therefore, there are still obstacles to use conditional generative models in datasets such as CIFAR and ImageNet [28]. Two obstacles are particularly critical.

First, generative models are sensitive to image similarity measures. We use L_2 distance to measure the image similarity. However, this metric is known to perform poorly in high-dimensional spaces: on the one hand, a one-pixel translation could lead to a large L_2 distance; on the other hand, two objects sharing the same background could be very close in terms of L_2 distance. Therefore, finding a better distance metric that captures semantic similarity while staying robust against small perturbations might further improve the performance of ABS-like models.

Second, ABS-like models have two efficiency bottlenecks. The time for inference scales linearly with the number of classes, making inference inefficient on large datasets such as

CIFAR100 and ImageNet. Also, running time grows approximately linearly with the number of iterations of optimization used during inference. With more iterations, the model is more stable and has better accuracy and robustness. Better sampling or optimization methods may allow more efficient inference.

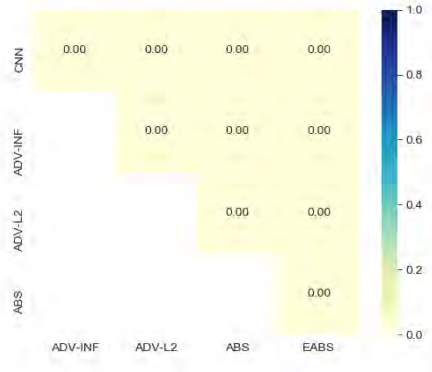
Our work may be of independent interest as an application of generative models that give explicit likelihood estimates. Traditionally, these models yield lower-quality reconstructions than GANs, which do not give likelihood estimates. Therefore, the success of E-ABS on complex image domains motivates research into better generative models for distribution matching: progress on such models may lead to more robust models.

Table 2.2: Results of different models on MNIST. Reported numbers are accuracy under bounded perturbations. Results are based on 1000 samples. McNemar's test shows that all differences are significant ($p < 0.01$).

	CNN	Adv-L_∞	Adv-L_2	ABS	E-ABS
Clean	99.5%	98.5%	98.6%	96.1%	99.4%
L_∞ attack ($\epsilon = 0.3$)					
PGD	0%	90.4%	0%	3.4%	17.1%
DeepFool	17.4%	92.9%	43.8%	6.2%	31%
Noise	99.3%	98.4%	98.5%	95.8%	99.1%
All L_∞	0%	90.3%	0%	3.3%	16.5%
L_2 attack ($\epsilon = 1.5$)					
PGD	72.8%	88.4%	89.3%	75.2%	90.6%
DDN	62.3%	81.7%	87.5%	74.1%	91.4%
CW	86.8%	91.2%	94.7%	82.8%	94.8%
DeepFool	83.8%	93.1%	92.2%	95.9%	91.1
Noise	99.3%	98.5%	98.6%	74.4%	99.4%
All L_2	61.8%	81.5%	87.5%	73.7%	90.4%

McNemar's test p values

L_∞ attacks



L_2 attacks

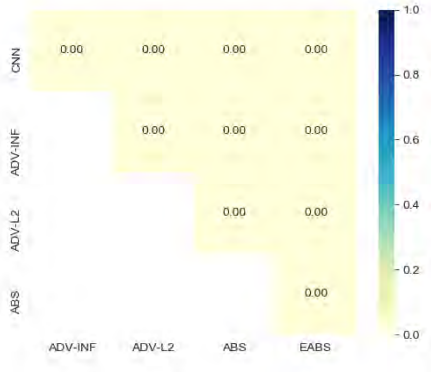
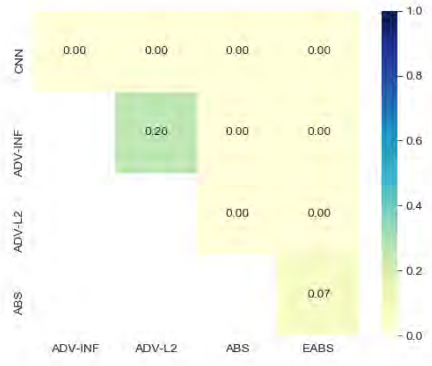


Table 2.3: Results of different models on Fashion MNIST. Reported numbers are accuracy under bounded perturbations. Results are based on 1000 samples. McNemar's test shows that all differences are significant ($p < 0.01$) except for: Adv-L_∞ and Adv-L_2 under L_∞ attacks, ABS and E-ABS under L_∞ attacks, Adv-L_∞ and ABS under L_2 attacks, and ABS and E-ABS under L_2 attacks.

	CNN	Adv-L_∞	Adv-L_2	ABS	E-ABS
Clean	91.0%	88.9%	87.4%	81.6%	90.1%
L_∞ attack ($\epsilon = 0.1$)					
PGD	9.4%	57%	56%	46.2%	43.5%
DeepFool	25.9%	64.9%	63.2%	47.7%	45.7%
Noise	90.5%	88.9%	86.5%	81.4%	0.9%
All L_∞	9.4%	57%	55.7%	45.8%	43.5%
L_2 attack ($\epsilon = 1.5$)					
PGD	19.1%	51.9%	59.2%	46.6%	43.3%
DDN	13.4%	46.6%	57.2%	45.4%	49.9%
CW	30.7%	61%	63.6%	50.7%	54.1%
DeepFool	28.9%	56.4%	60.4%	46.4%	44.3%
Noise	90.9%	88.8%	87.1%	81.2%	89.6%
All L_2	13.2%	45.9%	55.1%	45.2%	41.7%

McNemar's test p values

L_∞ attacks



L_2 attacks

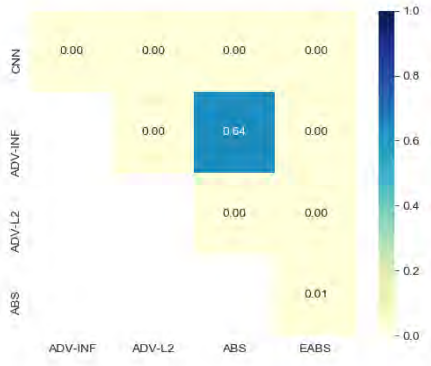
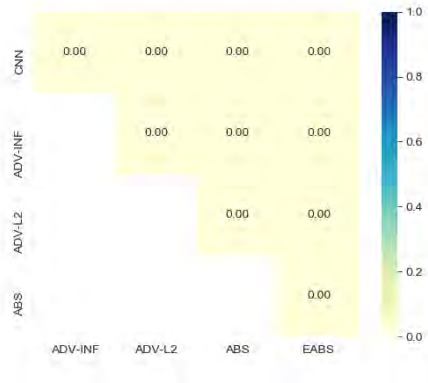


Table 2.4: Results of different models on SuperTraffic-10. Reported numbers are accuracy under bounded perturbations. Results are based on 1000 samples. McNemar's test shows that all differences are significant ($p < 0.01$) except for Adv-L_∞ and ABS under L_2 attacks.

	CNN	Adv-L_∞	Adv-L_2	ABS	E-ABS
Clean	99%	91.4%	91.6%	84.9%	92.7%
L_∞ attack ($\epsilon = 8/255$)					
PGD	29.8%	74.3%	60.2%	53.4%	70.9%
DeepFool	48.1%	74.9%	64.6%	54.5%	82.1%
Noise	99.1%	91.7%	91.6%	84.8%	91.7%
All L_∞	29.8%	73.7%	59.9%	53.2%	69.8%
L_2 attack ($\epsilon = 1.5$)					
PGD	14.8%	46.2%	53.7%	48%	66.1%
DDN	9.3%	44.6%	52.4%	47.1%	73%
CW	14.5%	51%	54.5%	47%	73.8%
DeepFool	33%	49%	56.6%	48.7%	93.3%
Noise	98.6%	91.6%	91.4%	86.3%	73.8%
All L_2	8.7%	44.3%	52.2%	46.3%	59.9%

McNemar's test p values

L_∞ attacks



L_2 attacks

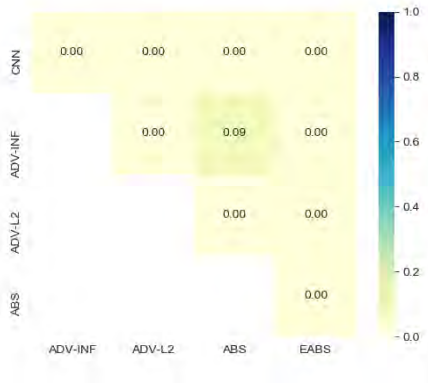
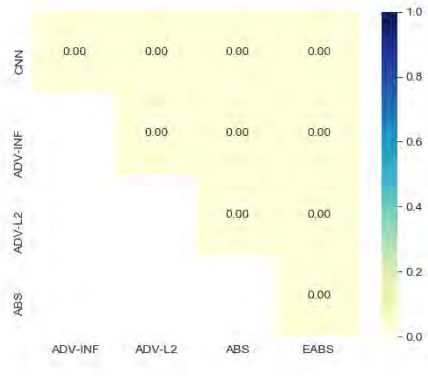


Table 2.5: Results of different models on SVHN. Reported numbers are accuracy under bounded perturbations. Results are based on 1000 samples. McNemar's test shows that all differences are significant ($p < 0.01$) except for Adv-L_∞ and ABS under L_∞ attacks.

	CNN	Adv-L_∞	Adv-L_2	ABS	E-ABS
Clean	91.3%	89.0%	87.1%	45.8%	89.2%
L_∞ attack ($\epsilon = 8/255$)					
PGD	0%	37.2%	26.7%	6.6%	57%
DeepFool	0.1%	50.4%	36.6%	8.4%	64.1%
Noise	90.7%	89%	87%	45.7%	89.3%
All L_∞	0%	37.2%	26.7%	6.6%	55.7%
L_2 attack ($\epsilon = 1.5$)					
PGD	0%	5.1%	13.4%	4.4%	40.9%
DDN	0%	3.7%	11.6%	4.3%	57.8%
CW	0%	5.5%	13.7%	4.5%	47.6%
DeepFool	0%	13.4%	19.1%	4.7%	51.6%
Noise	90%	89.4%	87.4%	43.5%	88.8%
All L_2	0%	3.6%	10.9%	4.3%	36%

McNemar's test p values

L_∞ attacks



L_2 attacks

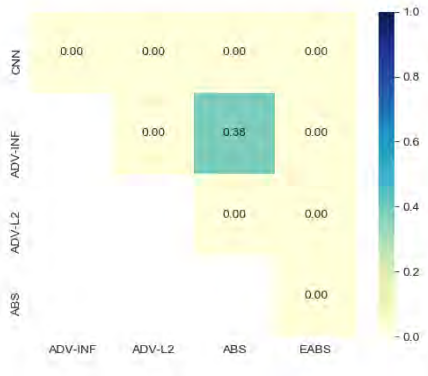


Table 2.6: An ablation study on SVHN. We report the mean/median distance of the closest adversarial examples.

Model	Natural Accuracy	PGD-L_∞	PGD-L_2
A-ABS	21.89%	0.020 / 0.013	0.98 / 0.75
D-ABS	43.82%	0.023 / 0.025	0.063 / 0.056
V-ABS	42.01%	0.008 / 0.006	0.10 / 0.08
AD-ABS	30.59%	0.013 / 0.013	0.66 / 0.56
AV-ABS	53.46%	0.025 / 0.022	0.81 / 0.75
DV-ABS	62.42%	0.016 / 0.006	0.38 / 0.09
ADV-ABS	88.30%	0.046 / 0.041	1.78 / 1.56

Table 2.7: Experiments on the number of runs for EOT, based on an L_∞ PGD attack 5 random starts on SVHN.

Number of runs	1	2	3	5	10
Accuracy	57.5	58.4	57.8	58.7	58.6

Table 2.8: Experiments on the number of random starts for PGD, based on an L_∞ PGD attack on SVHN.

Number of random starts	5	10	15	20	50
Accuracy	57.9	57.3	56.8	56	56

Table 2.9: Experiments on the number of PGD steps, based on L_∞ PGD attacks on SVHN.

Number of steps	0	10	30	50	100	200	500
Accuracy (%)	87.5	78	63.5	51.5	51.5	52	51.5

Chapter 3

Analyzing Analysis-By-Synthesis model with GANs

3.1 Introduction

The success of generative classifiers relies on three parts, as shown in Figure 3.1: the modelling of conditional data manifolds, an effective inference method that recovers the latent representation of an image, and a reliable method for estimating the likelihood of an input give its recovered representation. In E-ABS [53], we use AAE [72] with a shared encoder to model conditional data manifolds, variational inference to recover a latent representation of the input image, and a combination of L_2 distance and AAE's discriminator loss to estimate the likelihood of an image. In this chapter, we explore the challenges of generative classifiers more systematically.

Learning a generative model Generative adversarial networks (GANs) [39] are powerful generative models that have outperformed variational autoencoders [58] at modelling complex data manifolds. Therefore, it is natural to question if GANs provide an improvement to ABS. This chapter examines conditional GAN models including Auxiliary-classifier GANs (AC-GAN) [81, 38] and BigGAN [12, 122]. GANs have state-of-the art performance on modelling complex images and thus allowing us to examine ABS on more realistic datasets such as CIFAR10 [62] and ImageNet [28].

Finding a latent representation for the sample Given a generative model G and a prior of latent distribution $p(z)$, the likelihood of a sample X is given by

$$\log p(X) \propto \mathbb{E}_{z\sim p(z)}\mathcal{D}(G(z), X) \tag{3.1}$$

where $D(\cdot,\cdot)$ is a measurement of $\log p(X|z)$ under G. Equation 3.1 does not require any latent representation of X. However, the expectation in Equation 3.1 is intractable because of $G(z)$. Therefore, we approximate the expectation with some methods.

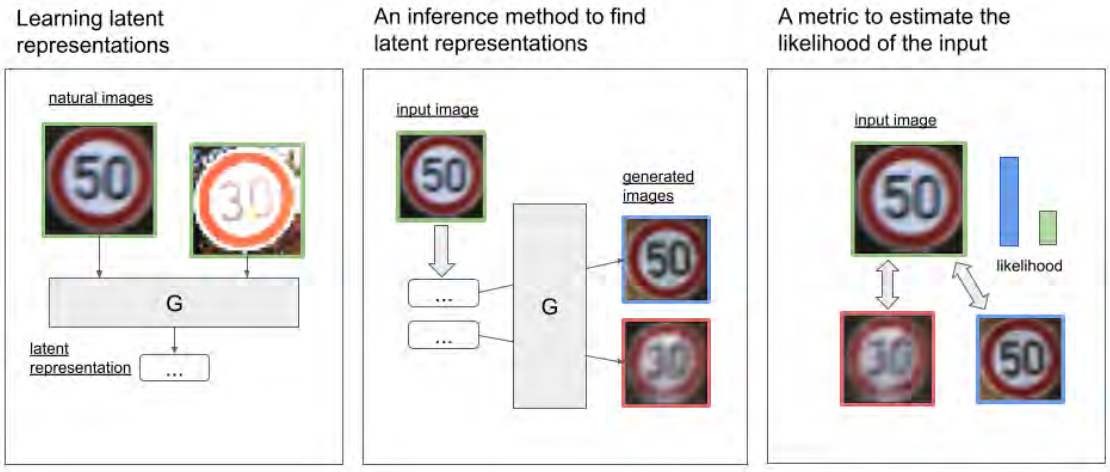


Figure 3.1: Generative classifiers relies on three steps: learning a generative model, finding the latent representation of an input, and estimating the likelihood.

Schott et al.'s ABS and our E-ABS approximate the expectation with a variational distribution $q_y(z)$. Therefore, given a sample point X and a class y (because we estimate a class-conditional likelihood), the latent representation of X is $q_y^*(z)$. In GANs, it is more common to use maximum-likelihood estimate (MLE) where we find a single latent code z such that

$$z^*_{X,y} := \arg\max_z \left(\log p(z) + \mathcal{D}(G(z, y), X)\right) \tag{3.2}$$

Compared to variational inference, Equation 3.2 yields one latent code that represents X.

The MLE approach is commonly referred to as GAN inversion [111]. GAN inversion yields an image's latent representation which encodes semantic information and thus is useful for tasks such as image editing [125, 108, 32] and style transfer [127]. Apart from finding a latent code, some inversion methods use overparameterization to achieve more accurate reconstruction. For example, Bau et al. adapts the generative model when inverting an image [6] and Zhu et al. optimizes latent codes at multiple hierarchies [125]. Besides overparameterization, state-of-the-art inversion methods use an encoder to provide a good starting point for the inversion optimization [125, 126, 6].

In this chapter, we present studies on various GAN inversion methods for ABS. We found that most GAN inversion methods, customized for recovering an in-domain image precisely, do not fit for ABS. Following Zhu et al.'s method on in-domain inversion [125], we propose a method to address out-of-domain samples. Experiments suggest that our method outperforms other methods, achieving a better classification accuracy on natural data.

Measuring image similarity

Variational autoencoders, including VAE [58] and AAE [72], measure the likelihood of X given a reconstructed image X' with squared error $\|X - X'\|_2^2$. Using L_2 norm as a distance measure between X and X' has a simple statistical explanation:

$$X = X' + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma\mathbb{I})$. This model naturally yields

$$\log p(X|X') = \log p(X - X') \propto \|X - X'\|_2^2$$

which is the L_2 distance metric.

However, L_2 distance fails to capture image similarity when images are complex. L_2 assumes that all pixels are equally important and independent, lacking a knowledge of structural similarity between two images, which is why more advanced image similarity metrics, such as structural similarity index measure (SSIM) [124] have been proposed to replace L_2 measurements.

When neural networks have dominated image recognition tasks, researchers found that deep features of neural networks convey high-level information consistent with human's visual perception [82]. Accordingly, distance metrics defined by deep features have been proposed for various tasks. Specifically, perceptual loss [52] is commonly used for inverting generative models [126, 32]. Following perceptual loss, Zhang et al. [120] proposed Learned Perceptual Image Patch Similarity (LPIPS) as a distance metric optimized for visual similarity and Kettunen et al. [54] proposed Ensembled LPIPS (E-LPIPS) to measure image similarity more robustly.

In this chapter, we examine various distance metrics. Our experiments indicate that robust distance metrics are critical for generative classifiers. The inversion process may overfit a distance metric, especially metrics defined by deep features, and thus fail to reject out-of-domain samples. Meanwhile, a robust distance metric can significantly improve the classification accuracy of generative classifiers.

3.2 Method

To understand the challenges of generative classifiers, we focused on GANs and designed experiments to analyze various inversion methods and distance metrics. Specifically, we analyze three aspects:

- How is the optimization of inversion initialized?
- What variables are optimized? What is the latent representation?
- What distance metrics are used?

Algorithm 2 The general framework of ABS with GANs.

Input: An image X.
Given a GAN model G, an initializer I, a similarity metric D, a regularization function R, an optimizer O, and a hyperparameter λ.
for each class y **do**
  Initialize the optimization $z_{\text{init}} \leftarrow I(X, y, G)$
  $z \leftarrow z_{\text{init}}$
  for each optimization step **do**
    Reconstruct $X' \leftarrow G(z, y)$
    Compute loss $l = D(X, X') + \lambda R(z, z_0)$
    $z \leftarrow O(z, \nabla z)$
  end for
  Compute the conditional likelihood for class y: $l_y \leftarrow D(X, X') + \lambda R(z, z_{\text{init}})$
end for
return l_y as the **negative** log-likelihood of sample X for class y.

Algorithm 2 presents the general framework that we use for classification with conditional GANs. This algorithm presents three steps to compute a sample's conditional likelihood l_y:

- First, we initialize the search with $z_0 = I(X, y, G)$.
- Second, we use optimization to find a latent code z^* that minimizes $l(X, G(z, y), z, z_0)$.
- Third, we use the optimized objective $l(X, G(z^*, y), z^*, z_0)$ as l_y.

Specifically, the hyperparameter λ represents the strength of regularization: stronger regularization means searching a smaller subset of the latent space and may prevent out-of-domain reconstructions, as our experiments will demonstrate. The regularization function R constraints the change of z; a simple regularization function is $R(z, z_{\text{init}}) = \|z\|_2^2$, but when initialized by an encoder, the regularization function becomes $R(z, z_{\text{init}}) = \|z = z_{\text{init}}\|_2^2$. The algorithm does not require the same function $l(X, y, G(z^*, y), z_0)$ when estimating conditional likelihoods at the third step. In our experiments, using a different function, such as a different distance metric or a transformed loss, might improve performance. However, to narrow down our scope of analysis, we choose to keep the two loss functions consistent in our experiments.

Initialization

We compared two initialization methods: initializing with random codes and initializing with an encoder. Initializing with random codes is used by Schott et al.'s ABS model. It is a simple approach readily usable with any generator. However, starting at random points may lead to suboptimal results because of local optimal and saddle points, which are common when images are complex [95]. Therefore, training an encoder for initialization is more commonly used when images are complex [125, 126].

Algorithm 3 Initializing the optimization with random points.

Input: An image X and a class y.
Given a generator G.
Sample N random codes $\mathcal{Z} = \{z_i | z_i \sim p(z)\}$.
Choose $z_i \in \mathcal{Z}$ that minimizes the distance between X and $G(z_i, y)$:

$$z_i^* \leftarrow \arg\min_{z_i \in \mathcal{Z}} \|X - G(z_i, y)\|_2^2$$

return z_i^*

To mitigate the issue of local minima, for random initialization, we sample more random codes and choose the best one as the initialization point. The initialization method is summarized in Algorithm 3. In practice, $\mathcal{Z}$ and the set of generated images $\{G(z_i, y) | z_i \in \mathcal{Z}\}$ are cached to save computing time. Therefore, we choose L_2 distance because caching deep features requires more space and more computation: deep features require more space than the raw image and feature distance typically requires deep features from more than one layers. When X is batched, L_2 distance also allows us to compute pairwise distance as a matrix operation, making it an efficient initialization method.

Initializing with an encoder is straightforward: the encoded latent code is used as the starting point, as Algorithm 4 describes. However, there are several ways to train the encoder, and we compared two methods in this study.

As a basis, Zhu et al. [126, 127] train the encoder to minimize the distance between reconstructed image and the input image. However, the inverted latent representation might not be the best representation: the encoder may provide atypical latent codes that do not behave like typical latent codes [125]. Therefore, Zhu et al. [125] presented in-domain encoder where the encoder jointly minimizes the distance of reconstructed image to the input and a discriminator loss. With this objective, the encoder provides an image that is not only close to the input, but also classified by the discriminator as a real image. We refereed to this method as **Enc**. Specifically, the training objective of **Enc** is Equation 3.3:

$$l_{\text{enc}}(X, y, z) = D(X, G(E(X), y)) + D_{\text{image}}(G(E(X), y)) \tag{3.3}$$

where the first term in Equation 3.3 is the distance between an input X and reconstructed image $G(E(X), y)$ (conditionally generated with the correct label y) and the second term is the discriminator loss given by an image discriminator D_{image}; D_{image} is jointly trained with the encoder E, as GAN training. At optimization time, the optimization is constrained to stay close to the encoded latent code with a regularization term. Therefore, with **Enc**, the objective of inference in Algorithm 2 becomes

$$l(X, X', z, z_{\text{init}}) = D(X, X') + \lambda \|z - z_{\text{init}}\|_2^2 \tag{3.4}$$

In our experiments, we found that Zhu et al.'s encoder may provide latent codes with a large L_2 norm; as the prior of latent codes is a multivariate Gaussian distribution, having

Algorithm 4 Initializing the optimization with an encoder.

Input: An image X and a class y.
Given a GAN model G and an encoder E.
$z \leftarrow E(X)$
return z

a large L_2 norm means the latent code has a low likelihood and away from latent codes sampled directly from the prior. Therefore, we propose a new method where we jointly train a discriminator on latent codes; the discriminator tries to distinguishes encoded latent codes from codes sampled from the prior. We refer to this method as **Enc-D**. Specifically, the training objective of **Enc-D** is Equation 3.5.

$$l_{\text{enc-d}}(X, y, z) = D(X, G(E(X), y)) + D_{\text{latent}}(E(X)) \tag{3.5}$$

which differs from Equation 3.3 by the second term. The discriminator D_{latent} distinguishes encoded latent codes $E(X)$ from latent codes sampled from the prior; it is jointly trained with the encoder. At optimization time, the regularization is the same as **Enc**, which is presented as Equation 3.4: we constraint z to stay close to the initialization point.

Optimized variables

We compared two choices of optimized variables. One method optimizes the latent code only. Meanwhile, studies show that overparameterization improves the quality of reconstructions [6, 125]. Therefore, we also compared a method where several copies of a latent code are optimized independently at various levels.

Figure 3.2 illustrates an overparameterized inference method. Our method relies on self-modulated GANs [19] where the latent code is used as an input to all blocks in the generator. At optimization time, we create independent copies of the latent code and optimize these copies independently. Specifically, given an original latent code z as the input to the generator, we create $z_1, z_2, \ldots, z_k$ for k blocks. The input to block i, being z at training time, becomes $z + z_i$ at inference time, and z_i is regularized by its L_2 norm. We indicate our overparameterized methods with $+$ in the method name.

Distance metrics

We compared several perceptual distance models. These models use features extracted by a neural network (base model) for measuring perceptual similarities. A simple perceptual distance compares the L_2 distance the deep features of images, which we refer to as feature distance. Given two images X_1 and X_2, a feature distance metric extracts features from multiple layers $F_1(X_1), F_2(X_1), \ldots, F_K(X_1)$ and $F_1(X_2), F_2(X_2), \ldots, F_K(X_2)$. Then the distance of X_1 and X_2 is measured by the pairwise distance of these features. In this study, we use

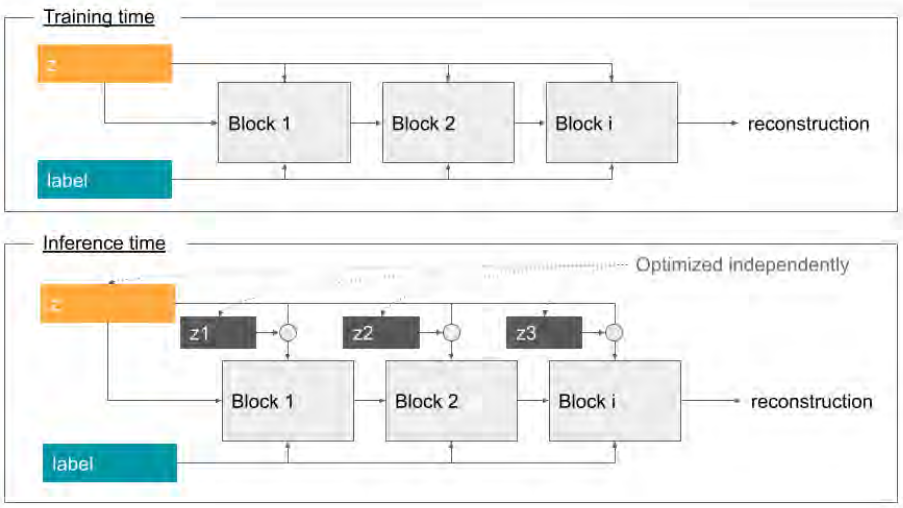


Figure 3.2: The overparameterization for GAN inversion. Our method is built for self-modulated GANs [19]. At training time, the same latent code z is used as input for all blocks. At inference time, each block has its own copy of a latent code z and they are optimized independently.

Equation 3.6 to compute distances from these features:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{c_k}\left\|\text{normalize}(F_k(X_1)) - \text{normalize}(F_k(X_2))\right\|_2^2 \tag{3.6}$$

where normalize is a normalization function and c_k is the number of channels of feature $F_k(X)$. Specifically, it normalizes the L_2 norm of each channel's feature to $\sqrt{w_k * h_k}$ where w_k and h_k is the spacial dimension of feature $F_k(X)$. Normalizing each feature channel's L_2 norm to $\sqrt{w_k * h_k}$ makes the numerical scale of two images' distance at each layer k comparable; the normalizing factor $\sqrt{w_k * h_k}$ is the expected L_2 norm of a random normal vector with length $w_k * h_k$. This normalization is not used on ImageNet in order to keep the implementation details consistent with the other perceptual distance.

Besides feature distance, LPIPS [54] optimizes the weights of deep features on a dataset customized to measure visual similarity. LPIPS is thus more consistent with human perception on realistic images.

We compared four choices of perceptual distance on CIFAR10 and two on ImageNet. On CIFAR10, we compared the standard feature distance [52, 54] that uses a VGG net [96] to extract deep features; the VGG net is pretrained on CIFAR10 as well. Besides, we compared feature distances with a base model that is adversarially robust and a base model that is robust to image corruptions on CIFAR10. We also evaluated LPIPS on CIFAR10. On

ImageNet, we evaluated the standard feature distance and LPIPS. Both perceptual distances use a VGG net pretrained on ImageNet to extract features.

Apart from perceptual distances, the $D(\cdot, \cdot)$ used in Algorithm 2 also has L_2 distance on raw pixels. For example, when using a normalized feature distance with VGG, the distance metric is

$$D(X_1, X_2) = \|X_1 - X_2\|_2^2 + \lambda_{\text{perceptual}} \frac{1}{K} \sum_{k=1}^{K} \frac{1}{c_k} \|\text{normalize}(F_k(X_1)) - \text{normalize}(F_k(X_2))\|_2^2 \tag{3.7}$$

We set $\lambda_{\text{perceptual}} = 1$ by default.

Summary

We summary the meaning of inference methods in this part to emphasize their differences.

- **Direct**. Optimizing a latent code z initialized by the point out of random latent codes sampled from $p(z)$ that yields the best L_1 distance to the input.
- **Enc**. Optimizing a latent code z initialized by an encoder. The encoder is jointly trained with an image-level discriminator, as proposed by Zhu et al. [125].
- **Enc-D**. Optimizing a latent code z initialized by an encoder. The encoder is jointly trained with an latent-space discriminator.
- **Direct+**, **Enc+**, and **Enc-D+**. The overparameterized version of optimization methods. Besides optimizing one latent code z, it optimizes $\{z_i\}$ as the input for generator blocks.

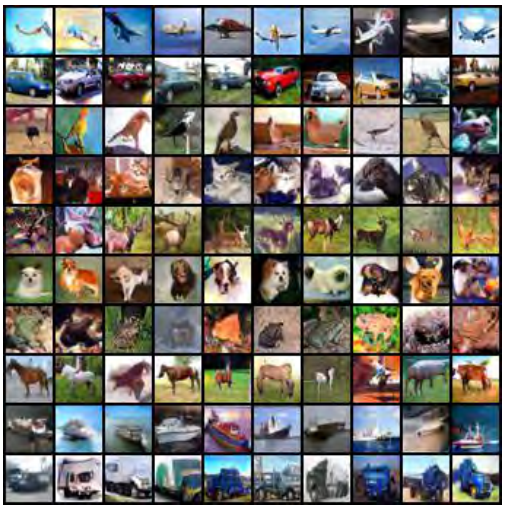
3.3 Experiments

To demonstrate the limitations of GANs, we experiment with several GAN models, inversion methods, similarity metrics, and datasets.

Datasets

We run experiments on CIFAR10 [62] and Imagenette [48], a ten-class subset of ImageNet [29]. CIFAR10 has ten classes with 32×32 RGB images. However, CIFAR10 has a low resolution and many images are indistinguishable even for human eyes. Therefore, we also experiment with ImageNet, where images have a higher resolution. ABS's inference time scales linearly with the number of classes. Due to limited computational resources, we choose to experiment with Imagenette, a ten-class subset of the original ImageNet.

(a) Samples from the cGAN model.



(b) Samples from the FQ-BigGAN model.

Figure 3.3: Random samples generated by GAN models on CIFAR10.

Models

For CIFAR10 experiments, we experimented two generative models: a conditional GAN with projection discriminator (cGAN) [74] and a feature-quantized BigGAN (FQ-BigGAN) [122].

cGAN with a projection discriminator [38] is a conditional generative model built upon the auxiliary-classifier GAN [81]. Besides the projection head for better class conditional modelling, we use spectral normalization to regularize the discriminator [75] and self-modulation [19] to use latent codes at various hierarchies. The cGAN model's structure is similar to Miyato et al.'s official implementation [1].

The cGAN is trained on CIFAR10 dataset where we randomly split 20% of the training data for validation. Furthermore, we use differentiable data augmentation at training time to more efficiently use data [121]. The trained cGAN has an FID score [46] (lower is better) of 14.57 and a Inception score [91, 5] of 6.58 (higher is better). Figure 3.3a shows sampled images from this model.

FQ-BigGAN [122] is a conditional generative model that has state-of-the-art performance on CIFAR10. It extends the BigGAN [12] model with feature-quantization for discriminators. We used Zhao et al.'s [122] official implementation for CIFAR10 [2]. The model's FID (lower is better) is 5.717 and its Inception score is 8.354 (higher is better). Figure 3.3b shows sampled images from this model.

For ImageNet experiments, we used a pretrained BigGAN [12]. We obtained the pretrained weights from Brock et al.'s [12] official repository [3]. Figure 3.4 shows sampled images from

[1] `https://github.com/crcrpar/pytorch.sngan_projection`

[2] `https://github.com/YangNaruto/FQ-GAN`

[3] `https://github.com/ajbrock/BigGAN-PyTorch`

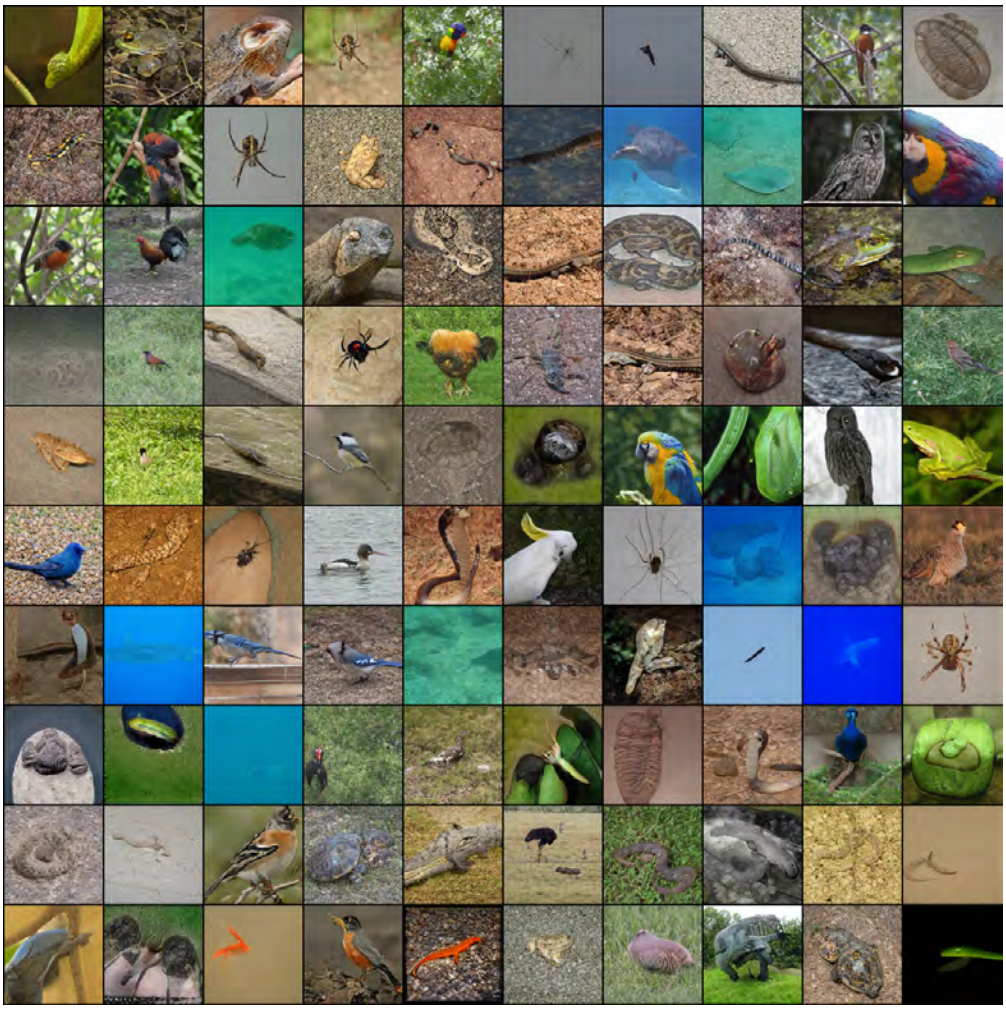


Figure 3.4: Samples from the BigGAN model on ImageNet.

this model.

Inference methods and similarity metrics

Initialization We compared two initialization methods: initializing with random points and initializing with an encoder. Section 3.2 explains the two methods.

In our experiments, we use 300 random initialization points, sampled from the prior $p(z)$. The 300 random points are cached and used for all samples.

The encoder used in both CIFAR10 and ImageNet is adapted from a ResNet18 model [43]. When training **Enc** on ImageNet, we started with 100 epochs where only the encoder was trained. The image discriminator used by **Enc** is pretrained when training the generator so jointly training the discriminator and encoder could lead to mode collapse. After the

first 100 epochs with the image discriminator frozen, we jointly trained the encoder and the image discriminator for another 200 epochs. There was no frozen epochs when training **Enc-D** because both the encoder and the latent-space discriminator are trained from scratch. Some recent studies use U-shaped encoder-generator pairs [94]. We use encoders to provide a starting point for optimization so we sacrificed precision for efficiency and simplicity. In practice we found a simple ResNet encoder is sufficient.

Inference We use Adam optimizer [57] with a cyclical learning rate [98] for optimization. The cyclical learning rate schedule starts the optimization with 1/10 the maximum learning rate, it gradually increases the learning rate to the maximum learning rate in the first 30% of the optimization, and it decreases the learning rate to 1/100 of the maximum learning rate in the second 60% of the optimization. The maximum learning rate is 0.1 by default.

Distance metrics We compared several perceptual distances:

- **VGG**: A feature distance with VGG network. With experiments on CIFAR, the network is trained for classifying CIFAR images. We use a public implementation of VGG and their pretrained model [4]. With experiments on Imagenette, the network is trained for classifying ImageNet images. We use the official implementation of LPIPS [120] without the LPIPS layer.
- **LPIPS**: The standard LPIPS distance. We use the same LPIPS model on CIFAR10 and Imagenette. When used on CIFAR10 images, we rescale images to 64×64.
- **Robust-Corruptions**: A ResNet18 model [60] trained for robustness against general image corruptions [44]. We use the pretrained model from robustbench library.
- **Robust-L_∞**: A ResNet18 model [110] trained for robustness against L_∞ bounded adversarial attacks ($\epsilon = 8/255$ on CIFAR10). We use the pretrained model from robustbench library.

On CIFAR10, we normalized features as described in Section 3.2 for feature distances. On ImageNet, we use the official LPIPS's implementation of feature distance, which does not normalize deep features.

3.4 Results

Trade-off between reconstruction quality and classification accuracy.

Comparing various inference methods on CIFAR10, the model's classification accuracy decreases as the quality of inversion improves. Table 3.1 shows that for all inference methods

[4] `https://github.com/kuangliu/pytorch-cifar`

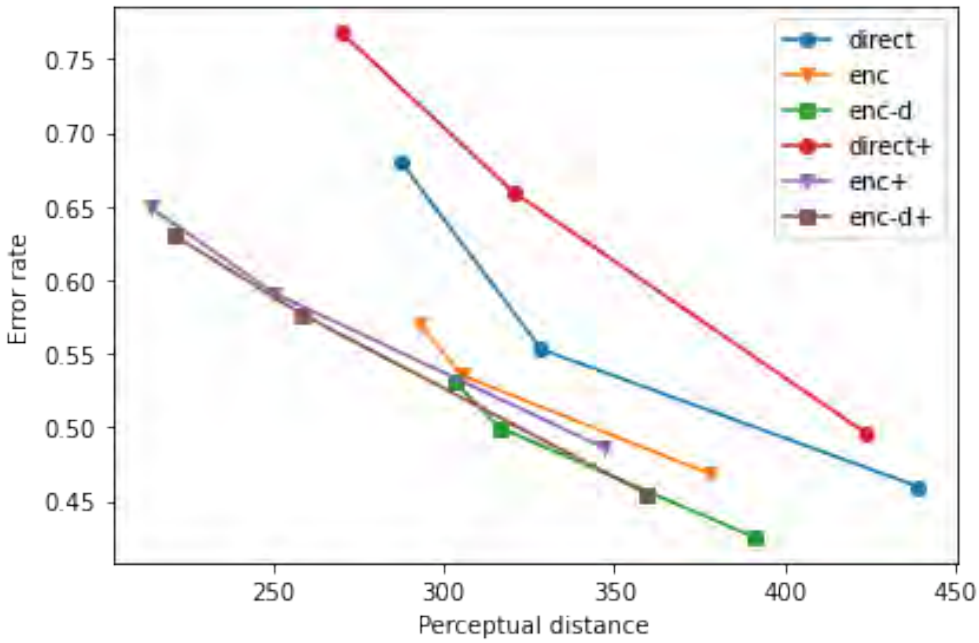


Figure 3.5: Error rate versus perceptual distance between in-domain reconstructions and the input. All inference methods have a trade-off between error rates and reconstruction quality: lower error rates correspond to higher perceptual distance. Encoder-guided inference methods (**Enc** and **Enc-D**) perform better than **Direct**.

(with 100 optimization steps), improved quality of inversion, indicated by smaller perceptual distance between in-domain reconstructions and the input, corresponds to higher classification error rates. Figure 3.5 visualizes this trade-off. For each method, the regularization hyperparameter λ controls the strength of optimization: stronger regularization leads to higher perceptual distance and lower error rates.

As another evidence of the trade-off, Table 3.2 shows that **Enc-D**'s error rate increases as the number of optimization steps increases. As increasing the number of optimization steps leads to better reconstructions, this observation also supports the trade-off.

Our observations indicate that classification errors of generative classifiers are due to overpowered generators. When in-domain reconstructions are better reconstructed, the error rate increases, suggesting that generators reconstruct out-of-domain samples better with more powerful optimization methods, causing the classification to fail. There are two hypotheses regarding why generator reconstruct out-of-domain samples:

- Stronger inference methods cause generators to reconstruct out-of-domain samples that are similar to the input.
- Perceptual distance is unstable and stronger inference methods may overfit the percep-

tual distance: generators may construct samples that are perceptually distant from the input but are close to the input with respect to the perceptual distance.

The first hypothesis attributes classification errors to overpowered GANs: GANs can reconstruct out-of-domain samples naturally. The second hypothesis attributes errors to the perceptual model: the perceptual model relies on deep features and therefore could be overfit by the optimization process.

Preventing OoD reconstructions on CIFAR10

Figure 3.6 shows some examples of out-of-domain reconstructions. With sufficient number of optimization steps, the input image can be reconstructed conditionally with any class. This indicates a shortcoming of the generative model: it can generate samples that are out of the domain of training samples. Figure 3.7 demonstrate examples from all inference methods, suggesting that all methods suffer from out-of-domain reconstructions.

CIFAR10 images are blurry making it difficult to tell whether out-of-domain reconstructions exist in many cases. Therefore, we measure the FID score of conditionally generated images and real images from the same class. Specifically, given a class y, we denote all training images from class y with $\mathcal{X}_y$, all training images from class that is not y with $\mathcal{X}_{\neg y}$, and use $\text{Inf}(X, y)$ to refer to an inference method that returns a latent representation z given an input X and y. We collect two sets of generated images with Inf:

$$\mathcal{D}_{+y} = \{G(z, y) | z \leftarrow \text{Inf}(X, y), X \in \mathcal{X}_y\} \mathcal{D}_{-y} \quad = \{G(z, y) | z \leftarrow \text{Inf}(X, y), X \in \mathcal{X}_{\neg y}\} \quad (3.8)$$

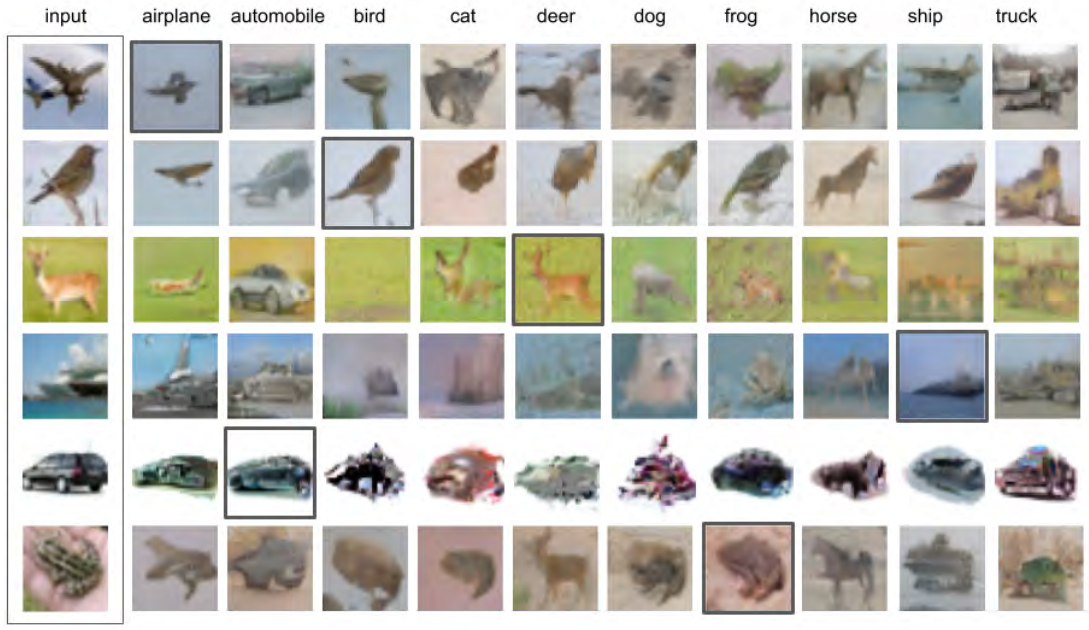
Then we compute two FID scores

$$\text{FID}_{\text{in-domain}} = \text{FID}(\mathcal{D}_{+y}, \mathcal{X}_y) \quad (3.9)$$

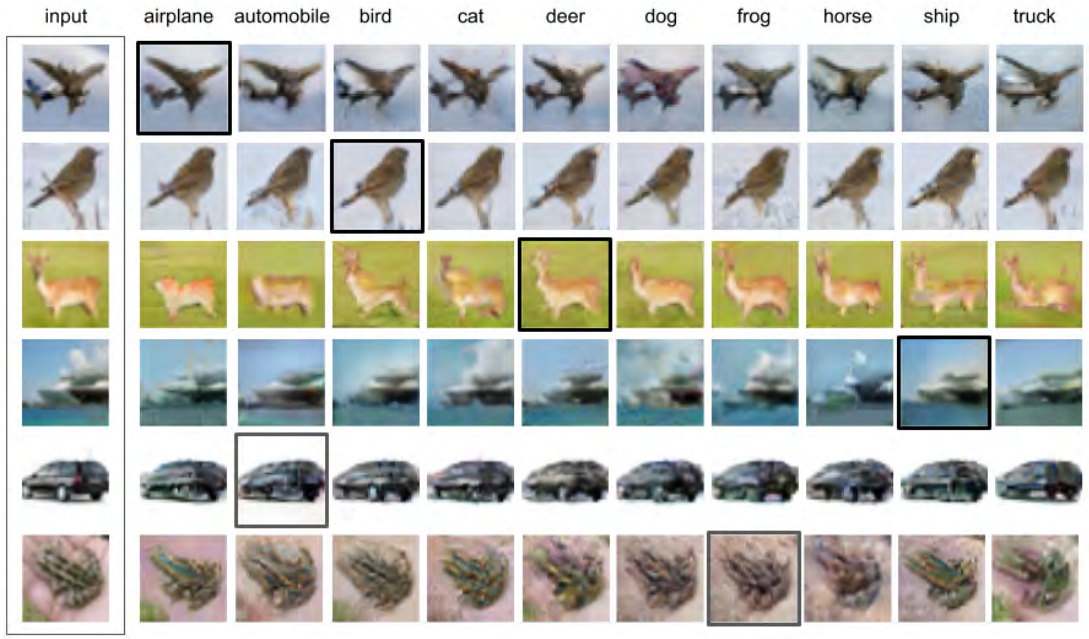
$$\text{FID}_{\text{out-of-domain}} = \text{FID}(\mathcal{D}_{-y}, \mathcal{X}_y) \quad (3.10)$$

FID score is used to measure the distance of two distributions. Lower FID score indicates that the two image distributions are close perceptually. Therefore, $\text{FID}_{\text{in-domain}} \approx \text{FID}_{\text{out-of-domain}}$ would indicate no out-of-domain samples exist, while $\text{FID}_{\text{in-domain}} < \text{FID}_{\text{out-of-domain}}$ would indicate the existence of out-of-domain samples.

Table 3.3 shows these FID scores and the difference between $\text{FID}_{\text{out-of-domain}}$ and $\text{FID}_{\text{in-domain}}$ Results indicate that most methods suffer from out-of-domain reconstructions: reconstructed images deviate from real in-domain images when inverting an out-of-domain image. Specifically, a model's error rate is positively correlated with the gap between $\text{FID}_{\text{out-of-domain}}$ and $\text{FID}_{\text{in-domain}}$, suggesting that methods can achieve better accuracy by staying in domain. **Enc-D** with $\lambda_2 = 10$, the best model in terms of error rates (Table 3.1), achieves the smallest gap between $\text{FID}_{\text{out-of-domain}}$ and $\text{FID}_{\text{in-domain}}$; it confirms that **Enc-D** is better at avoiding out-of-domain samples than other methods.

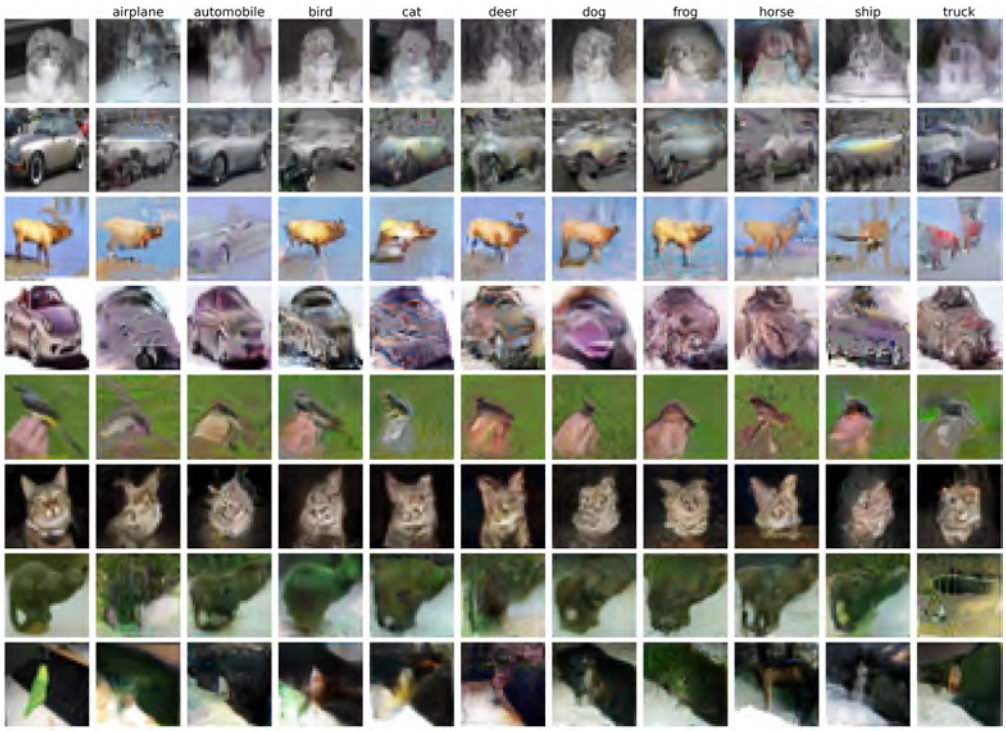


(a) Examples with 10 optimization steps.

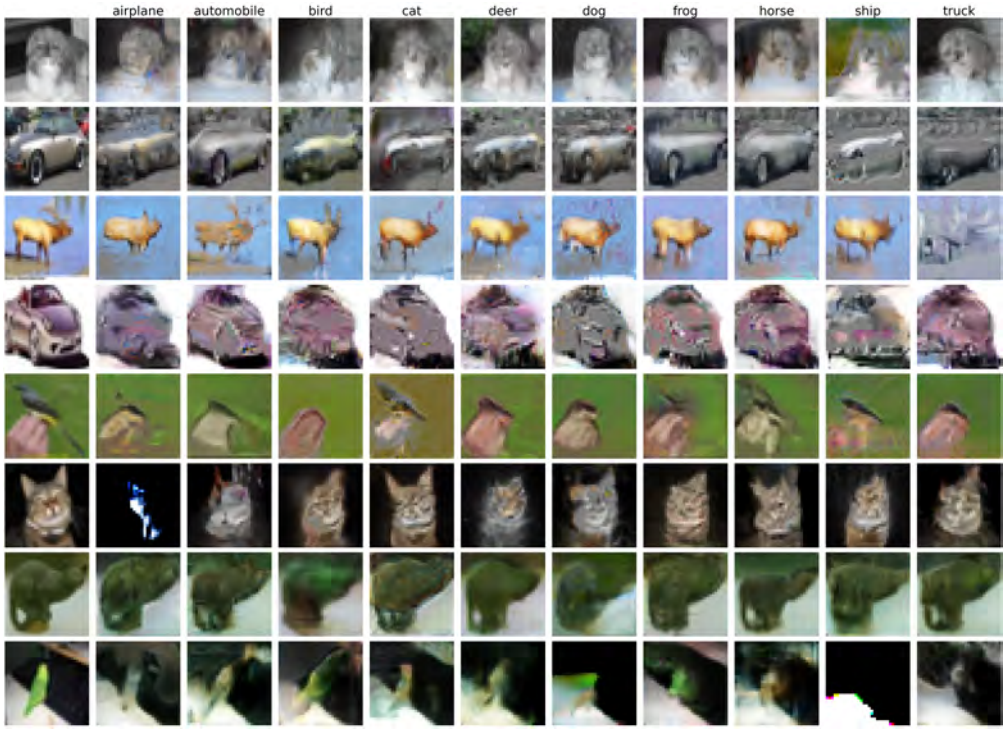


(b) Examples with 200 optimization steps.

Figure 3.6: Some examples of the input image and conditionally reconstructed images. The in-domain reconstructions are highlighted. With 200 optimization steps, any class can conditionally reconstruct the input image. These examples are generated by **Enc-D+** method with $\lambda_2 = 0.1$.

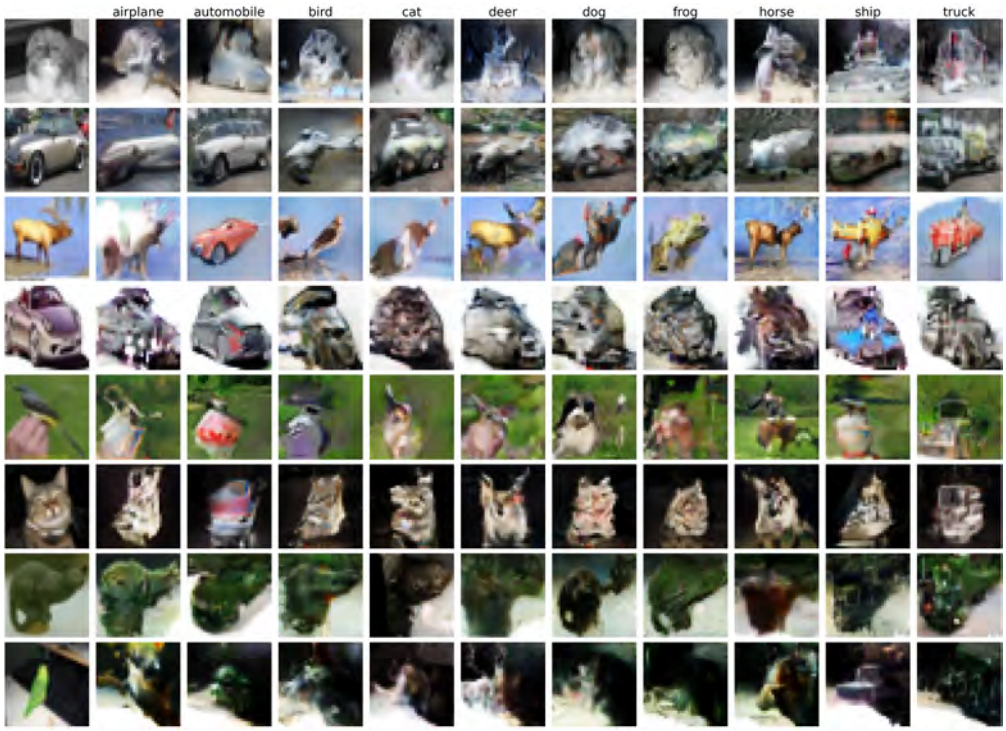


(a) Direct

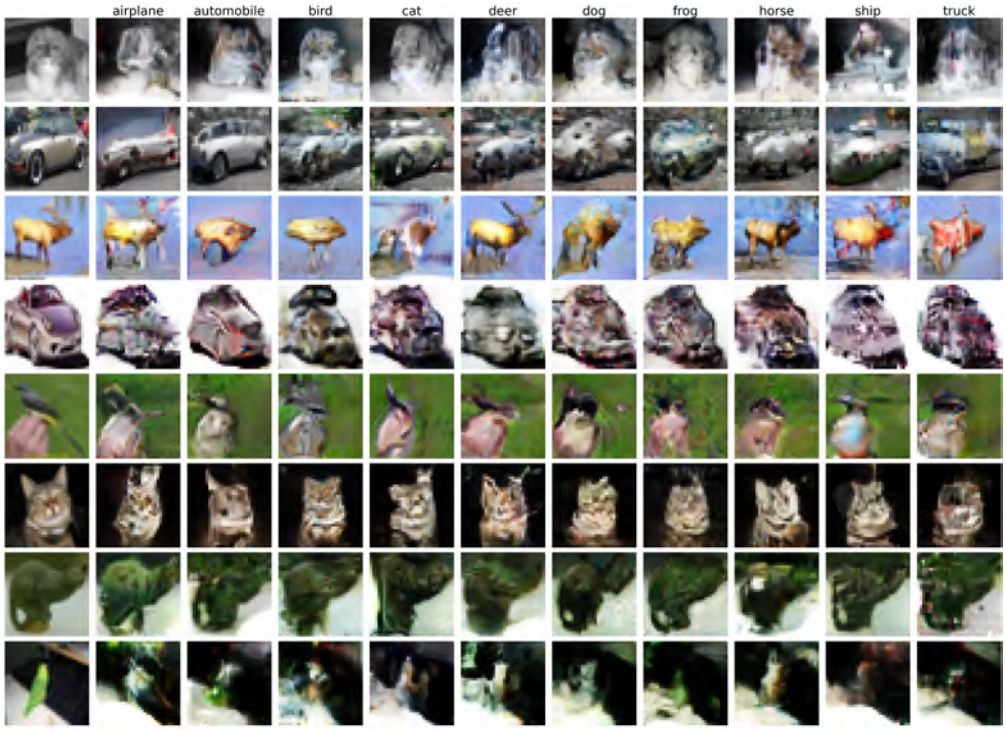


(b) Direct+

Figure 3.7: Some examples from various inference methods. All examples are generated with $\lambda = 0.1$ under 100 optimization steps.

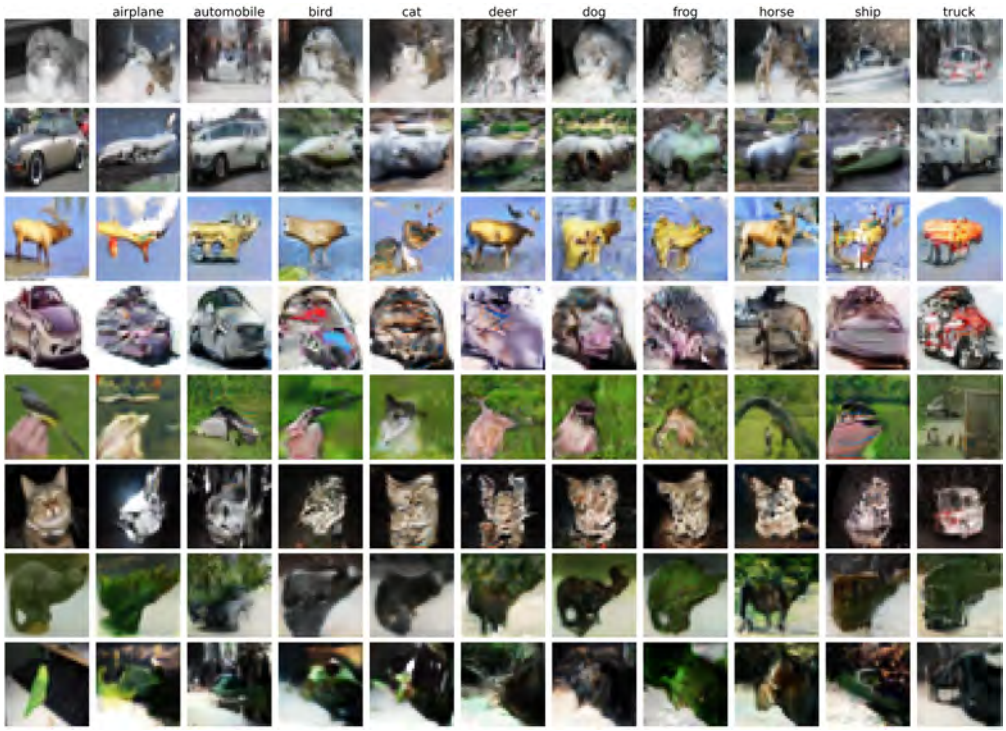


(c) Enc

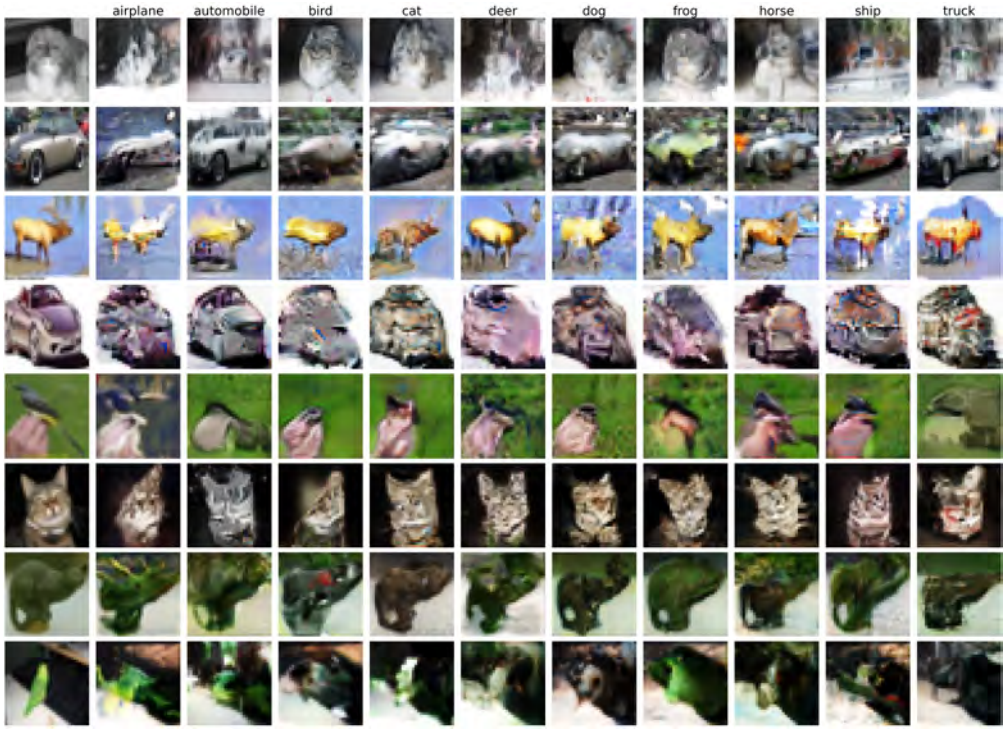


(d) Enc+

Figure 3.7: Some examples from various inference methods. All examples are generated with $\lambda = 0.1$ under 100 optimization steps.

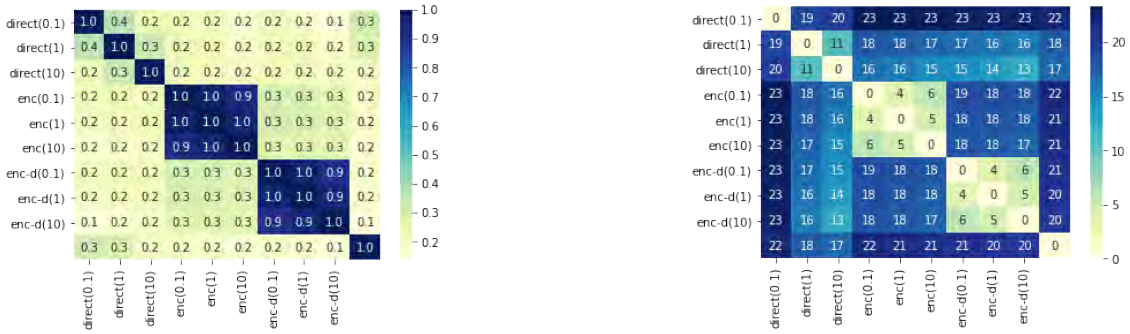


(e) Enc-D



(f) Enc-D+

Figure 3.7: Some examples from various inference methods. All examples are generated with $\lambda = 0.1$ under 100 optimization steps.



(a) Average cosine similarity of optimal latent codes.

(b) Average L_2 distance of optimal latent codes.

Figure 3.8: The average cosine similarity (left) and L_2 distance (right) of optimal latent codes. Different optimization methods yield latent codes that are distant from each other. This indicates that inverting GANs may suffer from local minima, and more importantly, the perceptual distance should be stable over images constructed from a wide range of latent codes.

Seeking a better perceptual distance

When inverting GANs, the optimal latent code is not unique. Our experiments showed that various inference methods yield latent codes that are distant from each other, as presented in Figure 3.8. Therefore, a reliable perceptual distance needs to handle images reconstructed from a wide range of latent codes, being able to tell the subtle difference of images that are all perceptually similar to the input (see Figure 3.9).

An ideal perceptual distance could tell images reconstructed from "normal" latent codes from images reconstructed from "abnormal" latent codes, inducing a smooth loss surface that is easier to optimize over and eventually providing a more accurate estimate of conditional likelihood. Unstable perceptual distance, on the other hand, would cause the optimization to yield latent codes that overfit the perceptual distance: the perceptual distance may measure an image with some imperceptible patterns as closer to the input image than other images that would be perceived similar by humans; meanwhile, the perceptual distance may measure a blurry image as close to the input while human would determine them as far apart. Essentially, unstable perceptual distance may fail to reflect the actual perceptual similarity between the reconstructed image and the input.

The standard feature distance **VGG** suffers from instability. We demonstrate **VGG**'s instability by interpolating optimal latent codes. Given optimal latent codes z_1 and z_2 of the same image and from the same class y obtained by two separate runs, we find an interpolated latent code $z = a * z_1 + (1 - a) * z_2, a \in [0, 1]$ and an interpolated image is $G(z, y)$. Specifically in Figure 3.9 and Table 3.4, the order of interpolation $k \in \{0, 1, \ldots, 10\}$ means $a = k/10$. Therefore, $k = 0$ gives z_1 and $k = 10$ gives z_2.

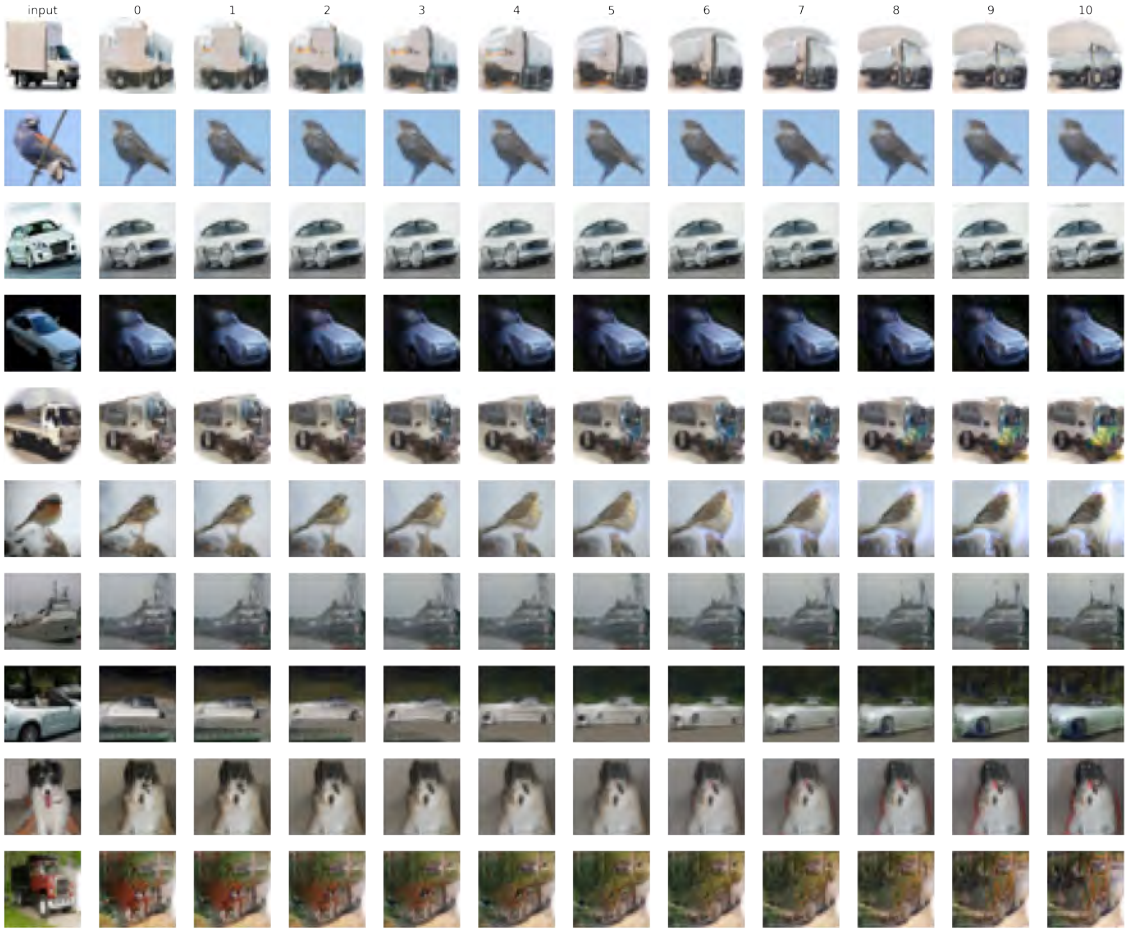
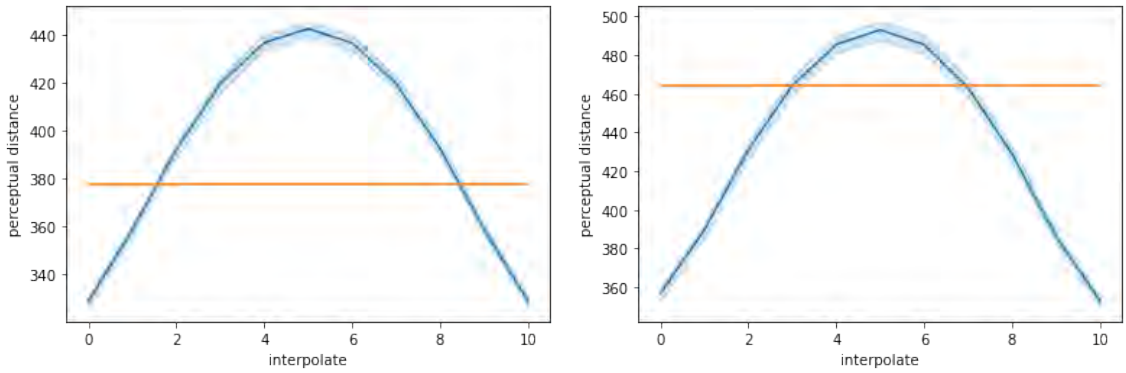


Figure 3.9: Interpolating the in-domain latent presentations of two **Direct** methods. Two endpoint images are the most similar to the input, as they are obtained by the optimization process. Compared to endpoint images, interpolated images are more blurry, but are still similar to the input.

Comparing interpolated images in Figure 3.9 and their perceptual distances to the input image in Figure 3.10a, the **VGG** feature distance may suffer from overfitting. Table 3.4 and Figure 3.10 show that the average perceptual distance is much lower on the two endpoints than images reconstructed from interpolated latent codes under **VGG**. Yet, Figure 3.9 indicates that interpolated images are perceptually similar to endpoint images. We thus hypothesize that the standard **VGG** feature distance could be overfitted by the optimization, in a way similar to an adversarial attack on the perceptual model. In case of overfitting, the perceptual distance gives inaccurate conditional likelihoods and overestimates the likelihood of the input image conditioned on a wrong class, causing generative classifiers to misclassify.

In pursuit of a better perceptual distance for generative classifiers, we compared the stan-



(a) Inference with **VGG** (b) Inference with **Robust-Corruptions**

Figure 3.10: The average perceptual distance of reconstructed images from interpolated latent codes. The horizontal line denotes the average perceptual distance between out-of-domain reconstructions and the input; informally, surpassing this line indicates more than 50% classification error rates. Two runs with the **Direct** method and $\lambda = 1$ are compared in both figures. More information can be found in Table 3.4

dard feature distance **VGG** to feature distance with more robust models in Table 3.5. Results indicated that feature distance with a base model robust to corruptions [44] outperforms other perceptual metrics. Table 3.6 compares **Robust-Corruptions** to **VGG** on various inference methods, showing that **Robust-Corruptions** perceptual distance outperforms the standard **VGG** perceptual distance in all cases.

Furthermore, our analysis on interpolations also suggests that **Robust-Corruptions** suffers less from overfitting. Comparing Figure 3.10b to Figure 3.10a, **Robust-Corruptions**'s peak value is closer to the reference line than **VGG**. The reference line represents the average perceptual distance of out-of-domain reconstructions to the input at two endpoints. Informally, surpassing this reference line means that the model has an error rate over 50%. Therefore, Figure 3.10 suggests that although **Robust-Corruptions** has a similar trend on interpolations with **VGG**, its classification accuracy is less severely influenced by the potential overfitting problem.

We hypothesize that **Robust-Corruptions** perceptual distance outperforms standard **VGG** feature distance because the base model is trained on images similar to generated images when inverting the GAN. Generative images may be more similar to corrupted images as they are blurry and sometimes contain abnormal patterns during the optimization. Therefore, **Robust-Corruptions** may be better at capturing meaningful and class-specific features from images, especially generated images, and thus estimating conditional likelihoods more reliably than the standard **VGG**. This hypothesis is also supported by Table 3.5 where **LPIPS** performed badly on CIFAR10, although it captures perceptual similarity well on real images. Compared to other models, **LPIPS** is not trained on CIFAR10 images, which could

explain that this perceptual distance performed the worst.

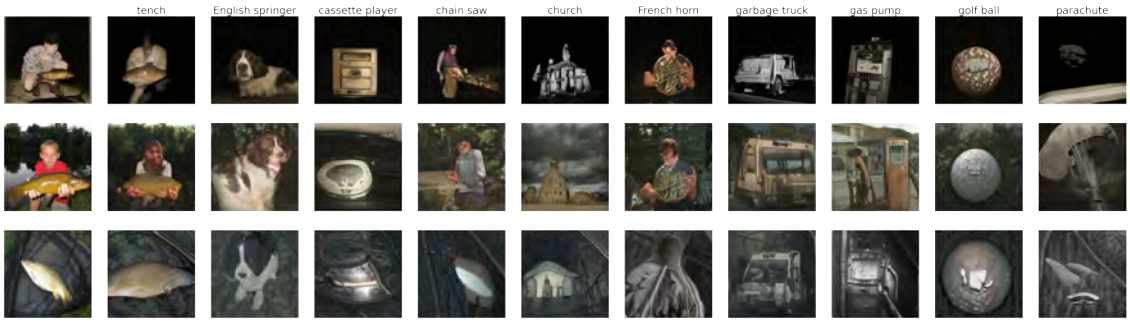
Summary On CIFAR10, we have compared three inference methods: **Direct**, **Enc**, and **Enc-D**. We found that all methods have a trade-off between classification accuracy and reconstruction quality: reconstructing input image better decreases the model's accuracy. There are two reasons for such a trade-off: 1) Overpowered generators can reconstruct an out-of-domain samples with the same quality as in-domain samples. 2) Perceptual distance is unstable over reconstructed images, causing unreliable likelihood estimations. Our experiments suggest that both problems exist on CIFAR10. Meanwhile, our proposed **Enc-D** method and a perceptual distance with corruption-robust model can address the two issues better than other methods.

Experiments on Imagenette

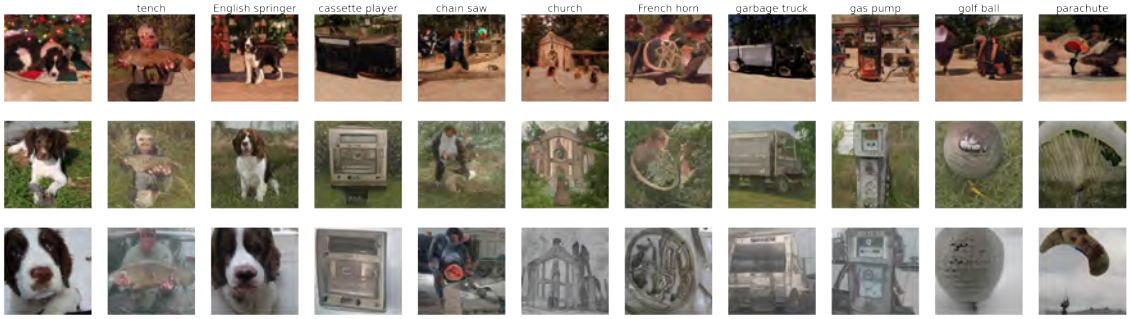
CIFAR10 images are small and blurry, meaning that out-of-domain reconstructions are more likely to exist and measuring perceptual similarity is more difficult. Therefore, we conducted similar experiments on Imagenette to study the behavior of generative classifiers on more realistic images. We used BigGAN pretrained on the full ImageNet dataset for experiments on Imagenette. We note that training the generative model with images from more classes may benefit generative classifiers with more constraints on in-domain and out-of-domain reconstructions: images from other classes may help generators from generating out-of-domain samples on this smaller subset of classes.

Our experiments showed that our **Enc-D** method still outperforms other inference methods on Imagenette. Table 3.7 presents the model's accuracy under various configurations. Our **Enc-D** method has better classification accuracy than other methods, suggesting that **Enc-D** can avoid out-of-domain samples on Imagenette as well. Furthermore, in Table 3.7, encoder-guided inference methods outperform direct optimization methods, suggesting that reconstructing ImageNet-level images relies more heavily on encoders. Table 3.7 also shows that, unlike CIFAR10, using LPIPS as a distance metric yields better accuracy than perceptual loss. This is encouraging because LPIPS is tuned to match human visual perceptual similarity.

Our model still has the trade-off between accuracy and reconstruction quality on Imagenette. Table 3.8 shows that the model's accuracy decreases as the number of optimization steps increase. Meanwhile, we do not suffer as much from out-of-domain reconstructions on Imagenette as on CIFAR10. Figure 3.11 presents some examples and generators are better at staying in-domain on Imagenette compared to Figure 3.6. It is possible that training on images from more classes can prevent out-of-domain reconstructions, as we discussed above. It is also possible that reconstructing out-of-domain images is more difficult when image resolution is high, because each pixel contains less information and the generator thus covers a narrower range of values per pixel.

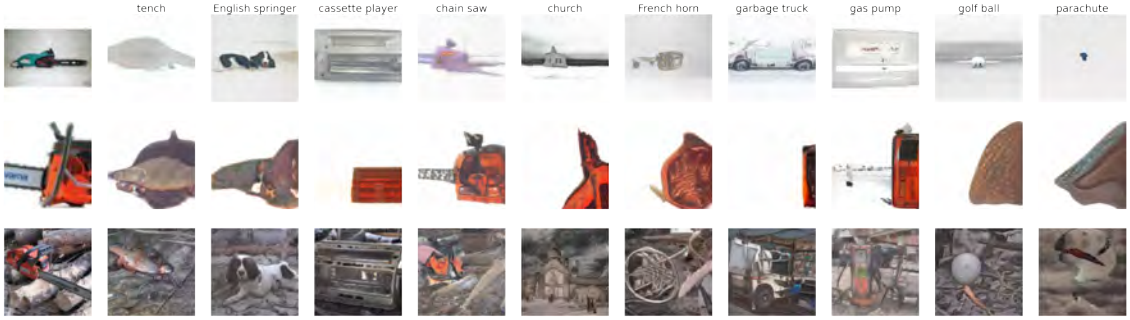


(a) Examples from the **tench** class



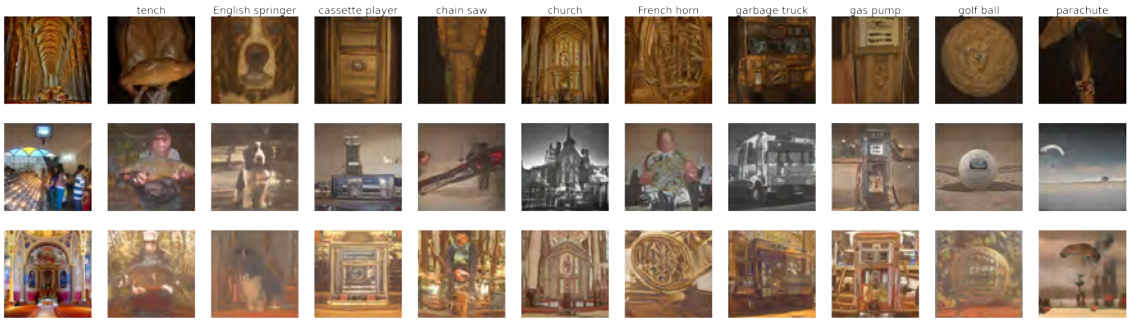
(b) Examples from the **English springer** class

(c) Examples from the **cassette player** class

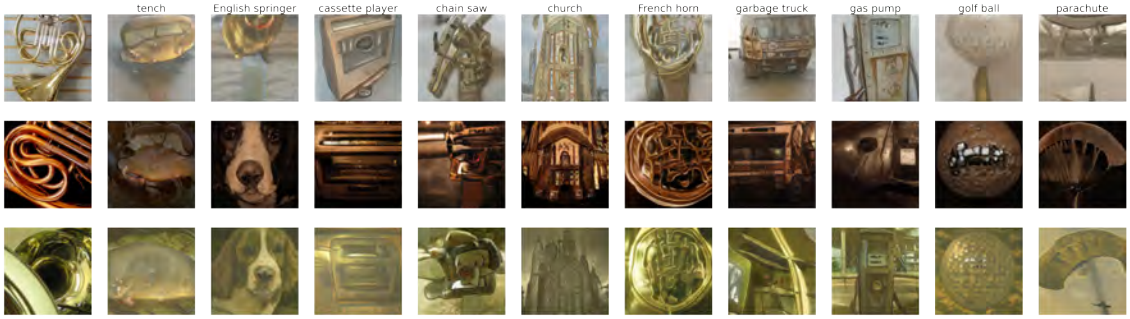


(d) Examples from the **chain saw** class

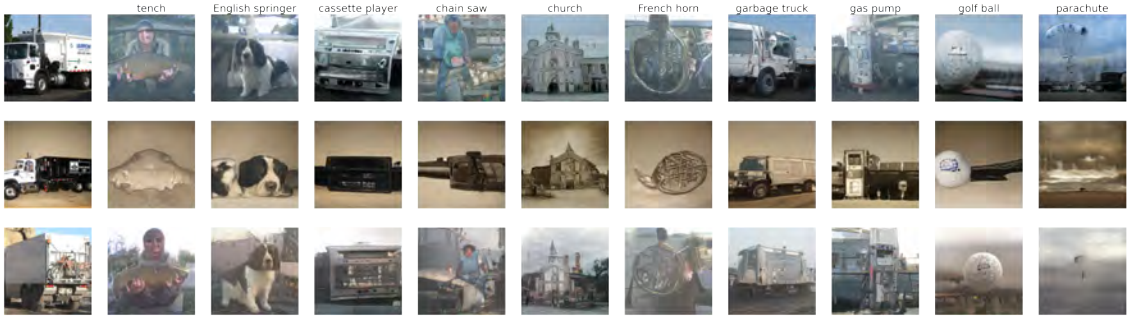
Figure 3.11: Some examples of **Enc-D** on Imagenette. The first column is the input image. The ten classes are: tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute.



(e) Examples from the **church** class



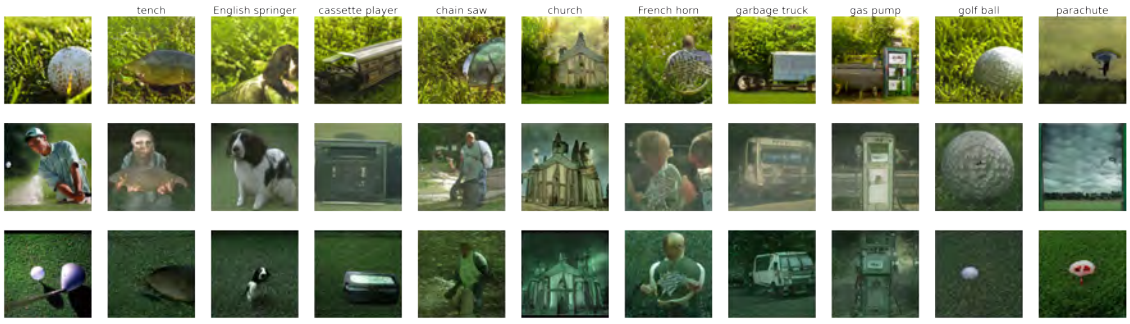
(f) Examples from the **French horn** class



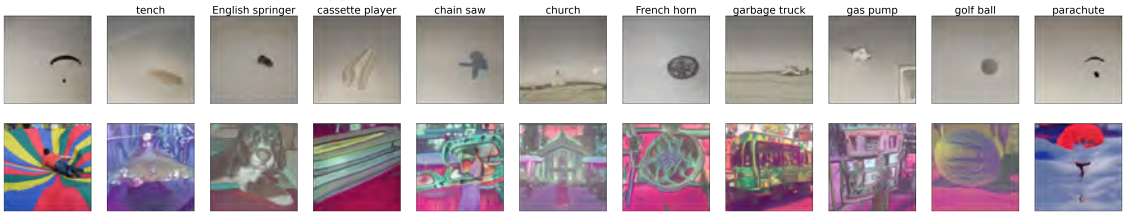
(g) Examples from the **garbage truck** class

(h) Examples from the **Gas pump** class

Figure 3.11: Some examples of **Enc-D** on Imagenette. The first column is the input image. The ten classes are: tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute.



(i) Examples from the **golf ball** class



(j) Examples from the **parachute** class

Figure 3.11: Some examples of **Enc-D** on Imagenette. The first column is the input image. The ten classes are: tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute.

3.5 Summary and Future Work

This chapter analyzes the inversion of GANs for building generative classifiers. Our analysis shows two challenges of generative classifiers when images are complex.

First, generative models suffer from out-of-domain reconstructions with complex images. Our analysis compared various inference methods and found that classification accuracy decreases as the inference method reconstructs an input better. Qualitatively, with sufficient strength, generators can reconstruct the input image condition on any class, suggesting that an inversion method that can prevent out-of-domain reconstructions is needed. In comparison, our inference method **Enc-D** is better at preventing out-of-domain reconstructions than other methods. Our analysis on in-domain and out-of-domain FID scores indicate that **Enc-D** is better at keeping reconstructions in-domain; besides, **Enc-D** also performs better on the trade-off and has achieved the best classification accuracy in our experiments.

Second, perceptual distance's stability is critical for generative classifiers when images are complex. Our analysis shows that perceptual distance needs to be stable over reconstructed images from a wide range of latent codes. Yet, standard feature distance models and LPIPS are unstable and may lead to overfitting. Comparing various distance metrics, we found that feature distance with a corruption-robust model outperforms the rest. Experiments confirmed that this **Robust-Corruptions** perceptual distance achieves better classification accuracy

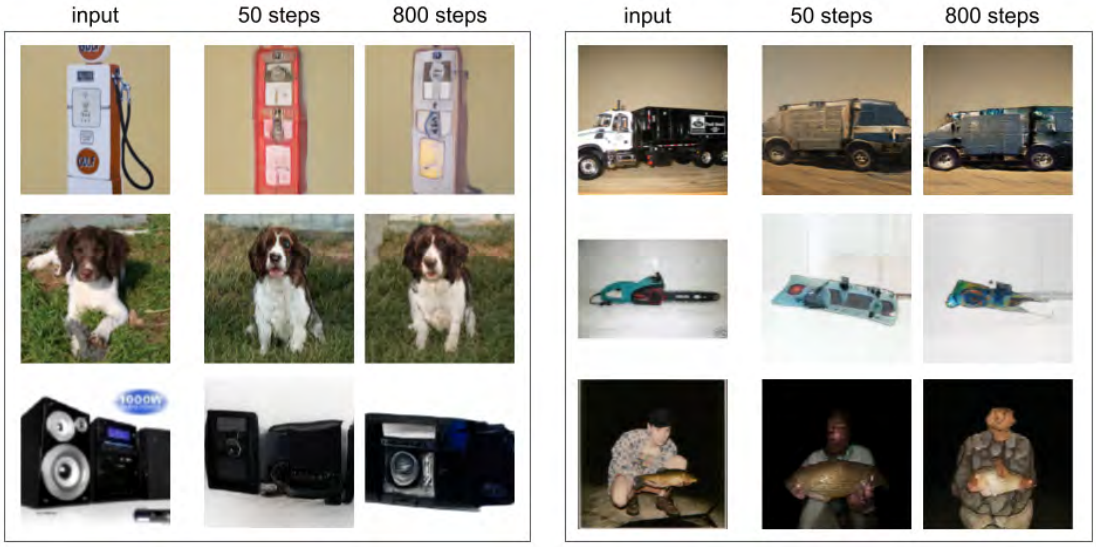


Figure 3.12: Comparing in-domain reconstructions on Imagenette. In this case, more optimization steps can reconstruct the input image better. These examples are generated by **Enc** with 50 and 800 steps respectively.

than the standard **VGG** distance on all inference methods.

Our study indicates several future research directions. First, our analysis on ImageNet indicates that out-of-domain reconstructions do not exist when images are high-resolution. We hypothesize that CIFAR models suffer more because they are overpowered: the latent space and the generator's capacity have more redundancy when images are small and blurry. It is interesting to investigate such a hypothesis and propose a training method or design principles so that we can avoid out-of-domain reconstructions.

Second, investigation into perceptual distance for generative classifiers is important and interesting. Perceptual distance relies on deep features, which are unstable under adversarial attacks. On the one hand, an unstable perceptual distance may undermine the robustness of generative classifiers; on the other hand, an inference method, which optimizes an objective in a way similar to an adversarial attack, may overfit the perceptual distance and cause inaccurate estimation of conditional likelihoods. Thus, a perceptual distance, such E-LPIPS [54], that is robust under adversarial attacks, may further improve the performance of generative classifiers.

We may also consider perceptual distance customized for GAN inversion or generative classifiers because most perceptual models thus far are trained to compare real images. Borrowing ideas from adversarial training [70], we can train a perceptual distance to measure reconstructed in-domain images as closer to the original input image and measure out-of-

domain reconstructions and the input as further away. Meanwhile, contrastive learning [55, 18, 56] can create stable representations for a range of downstream tasks; methods such as SimCLR [18] forces an image to have stable representations under random augmentation, which may also provide a stable perceptual distance metric. Therefore, contrastive learning, especially supervised contrastive learning where class information also influences representations [56], may prove to be a good perceptual distance for generative classifiers.

Beyond the two issues, there are other interesting research questions around generative classifiers as well. For example, the running time of generative classifiers reported in Algorithm 2 depends linearly on the number of classes K. Such a dependency would become enormous on large datasets such as ImageNet [28]. Hierarchical generative classifiers may reduce the dependency to $O(\log K)$ by modelling multiple classes with the same condition. However, such a method may require generative models to model overlapped classes [38] and pose some new challenges regarding inference methods.

Table 3.1: Various inference methods (100 optimization steps) on CIFAR10 under **VGG** feature distance. Reported numbers are the generative classifier's classification error rate, the L_2 distance between the input image and the image reconstructed from the ground-truth class (in-domain images), and the perceptual distance between the input image and the image reconstructed from the ground-truth class. The table shows a trade-off between error rates and perceptual distance. Figure 3.5 demonstrates this trade-off with as a scatter plot.

(a) Results with regular inference methods.

Inference Method	λ	Error rate	In-domain L_2 distance	In-domain perceptual distance
Direct	0.1	67.98	183.80	287.47
	1	55.28	201.41	328.58
	10	45.96	262.18	439.20
Enc	0.1	56.98	183.64	292.92
	1	53.52	190.14	305.21
	10	46.82	233.22	378.04
Enc-D	0.1	53.08	181.61	303.12
	1	49.96	187.33	316.41
	10	42.50	225.74	391.22

(b) Results with overparameterized inference methods.

Inference Method	λ	Error rate	In-domain L_2 distance	In-domain Perceptual distance
Direct+	0.1	76.8	184.86	269.83
	1	65.96	203.34	320.42
	10	49.62	257.44	423.57
Enc+	0.1	64.9	138.43	214.00
	1	59.1	159.65	250.16
	10	48.62	215.65	346.80
Enc-D+	0.1	63.04	138.75	220.72
	1	57.62	158.74	258.68
	10	45.44	210.70	359.62

Table 3.2: Classification error rate versus the number of optimization steps on CIFAR10. In-domain perceptual distance measures the average perceptual distance (**Corruption-Robust**) between inverted in-domain images and the input. Out-of-domain perceptual distance measures the average perceptual distance between inverted out-of-domain images and the input. More optimization steps improves both in-domain perceptual distance and out-of-domain perceptual distance, suggesting that the quality of reconstructions improve with more optimization steps. However, generative classifiers have a higher error rate under more optimization steps. All experiments use **Enc-D** method with $\lambda = 0.1$.

Steps	10	50	100	500
Error rate	11.82	19.96	25.16	45.28
In-domain perceptual distance	454.85	348.25	300.92	212.75
Out-of-domain perceptual distance	882.69	538.19	438.69	273.52

Table 3.3: Comparing FID scores on CIFAR10. $\text{FID}_{\text{in-domain}}$ represents the average distribution distance between class-conditional real images and reconstructed images conditionally generated to invert in-domain images. $\text{FID}_{\text{out-of-domain}}$ represents the average distribution distance between class-conditional real images and reconstructed images conditionally generated to invert out-of-domain images. Smaller FID scores indicate the two distributions, generated images and real images, are closer.

(a) Regular inference methods.

Inference Method	λ	$\text{FID}_{\text{in-domain}}$	$\text{FID}_{\text{out-of-domain}}$	Difference
Direct	0.1	84.90	119.35	34.45
	1	75.83	102.68	26.85
	10	76.35	80.68	4.33
Enc	0.1	77.61	102.06	24.45
	1	76.50	98.15	21.65
	10	73.97	86.68	12.71
Enc-D	0.1	72.62	95.64	23.01
	1	72.07	92.47	20.40
	10	70.56	78.89	8.33

(b) Overparameterized inference methods.

Inference Method	λ	$\text{FID}_{\text{in-domain}}$	$\text{FID}_{\text{out-of-domain}}$	Difference
Direct+	0.1	96.45	132.23	35.80
	1	84.32	116.21	31.89
	10	75.28	84.94	9.66
Enc+	0.1	80.26	116.35	36.08
	1	77.69	108.69	31.00
	10	73.33	91.61	18.29
Enc-D+	0.1	75.84	112.28	36.45
	1	73.41	103.23	29.81
	10	70.40	83.85	13.44

Table 3.4: Perceptual distance between reconstructed images from interpolated latent codes and the input. The two endpoints are obtained by two separate runs with **Direct** method and $\lambda = 1$. Each column reports the mean of perceptual distance between reconstructed images and the input across all test points. Reported numbers are mean perceptual distance, the 95% confidence interval of the mean, and the standard deviation. As a reference, the average perceptual distance of out-of-domain reconstructions to the input is 378.9. for **VGG** and 464.6 for **Robust-Corruptions**.

Inference with **VGG**

Order	0	1	2	3	4	5	6	7	8	9	10
Mean	328.7	358.6	392.3	419.7	436.8	442.8	436.8	419.7	392.6	358.9	328.8
CI	2.4	2.6	3.0	3.3	3.5	3.6	3.5	3.3	3.0	2.6	2.4
Std	85.6	94.4	107.3	119.4	127.4	130.2	127.2	119.1	107.5	94.4	85.3

Inference with **Robust-Corruptions**

Order	0	1	2	3	4	5	6	7	8	9	10
Mean	356.7	389.6	431.0	464.3	485.6	493.1	485.5	463.1	428.9	386.2	352.9
CI	3.4	3.7	4.3	4.8	5.1	5.2	5.1	4.8	4.3	3.7	3.4
Std	121.6	134.6	155.3	172.6	184.0	187.9	183.7	172.3	155.0	133.7	120.8

Table 3.5: Comparing perceptual models on CIFAR10. All experiments used **Enc-D** method with 200 optimization steps, $\lambda = 10$. Furthermore, all methods use $\lambda_{\text{percept}} = 10$ to emphasize perceptual distance.

Perceptual model	Error rate
LPIPS	56.48
VGG	47.62
Robust-L_∞ [110]	39.76
Robust-Corruptions [60]	29.28

Table 3.6: The error rate of various inference methods under **Robust-Corruptions** perceptual distance. Compared to Table 3.1, **Robust-Corruptions** outperforms **VGG** for all methods.

(a) Results with regular inference methods.

Inference Method	λ	Error rate with **Robust-Corruptions**	Error rate with **VGG**
Direct	0.1	51.08	67.98
	1	28.84	55.28
	10	12.20	45.96
Enc	0.1	27.00	56.98
	1	27.60	53.52
	10	22.74	46.82
Enc-D	0.1	24.44	53.08
	1	24.14	49.96
	10	20.08	42.50

(b) Results with overparameterized inference methods.

Inference Method	λ	Error rate with **Robust-Corruptions**	Error rate with **VGG**
Direct+	0.1	70.01	76.8
	1	47.32	64.96
	10	17.24	49.62
Enc+	0.1	40.26	64.90
	1	36.98	59.10
	10	27.98	48.62
Enc-D+	0.1	37.53	63.04
	1	34.00	57.62
	10	23.94	45.44

Table 3.7: Various inference methods on Imagenette.

Inference Method	Distance metric	Accuracy
Direct	LPIPS	53.8
Enc	LPIPS	74.4
Enc	VGG	71.9
Enc-D	LPIPS	76.1

Table 3.8: Classification error rate versus the number of optimization steps on Imagenette. In-domain perceptual distance measures the average perceptual distance (LPIPS) between inverted in-domain images and the input. Out-of-domain perceptual distance measures the average perceptual distance between inverted out-of-domain images and the input. The trade-off between classification accuracy and reconstruction quality still exists on Imagenette. All experiments use **Enc** inference method.

Optimization steps	1	10	50	100	300	800
Accuracy	76.97	76.62	76.35	74.4	69.3	67.26
In-domain perceptual distance	5.10	4.72	4.19	3.95	3.69	3.57
Out-of-domain perceptual distance	6.56	6.02	5.38	5.05	4.76	4.43

Chapter 4

Variational Inference Applied to Smoothing

4.1 Introduction

Discriminative models, such as ResNet [43], are good at modeling complex functions but lack interpretability; Bayesian models, such as Schott et al.'s ABS [95], are more interpretable but underperform when images are complex. Therefore, we ask: can we combine the two worlds? Randomized smoothing, an effective defense against adversarial examples [21, 67], could be the bridge that connects them.

Randomized smoothing estimates the input's expected class likelihoods under random noise. It uses a discriminative model to directly estimate the input's likelihood for each class and estimates the expectation over a bundle of randomly augmented inputs. Given the input X, and a class y, the estimation of $p(y|X)$ is given by

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left[F(X+\epsilon_i)=y\right] \tag{4.1}$$

where ϵ is the noise and $\mathbb{I}$ is an indicator function. F in Equation 4.1 represents the predicted class of a neural network f; in most cases, let $f_k(X)$ be the kth element of the neural network's output: $F(X)$ is $\arg\max_k f_k(X)$. Randomized smoothing provides certified robustness [67, 21] and can achieve good empirical robustness as well when replacing Gaussian noise with a set of random transformations [84].

Direct sampling, used by randomized smoothing, is inefficient and inaccurate when the input space is large. For sampling noise from a isotropic Gaussian distribution, the variance is proportional to the dimension of ϵ, which is the same as the dimension of input X. Studies shoed that the number of samples required to achieve the same accurate estimate increases exponentially as the variance of noise increases [67, 21]. Therefore previous studies usually rely on Gaussian noise with a small variance, which prohibits the model to thoroughly examine its neighboring space and limits the model's robustness to attacks with small bounds. We

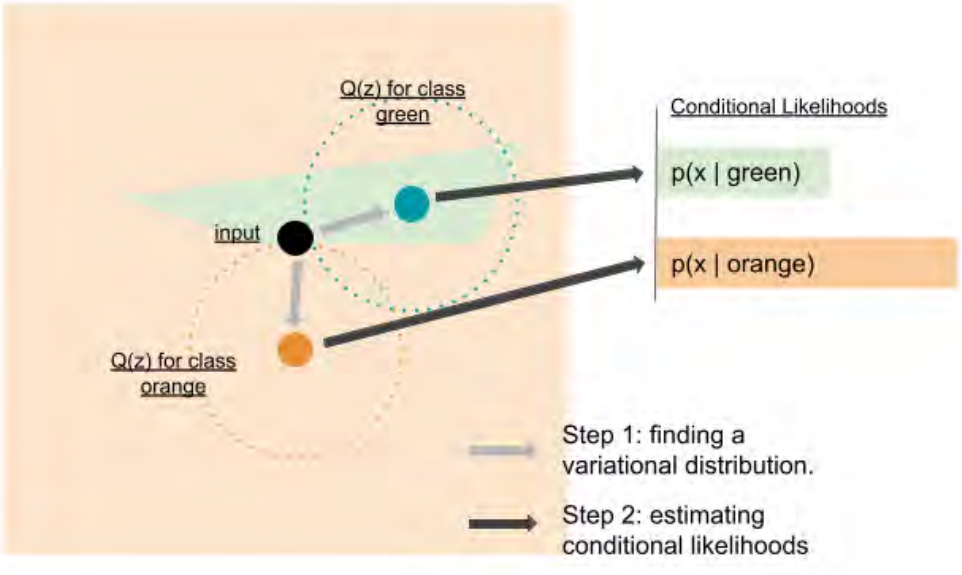


Figure 4.1: An overview of our variational method. We want to classify the green class from the orange class. Although this sample (black) is supposed to be classified as orange, the neural model f may have an adversarial subspace as illustrated by the green region. Our variational method classifies by exploiting the neighboring loss surface: it finds an optimal variational distribution for the green and orange class respectively, noticing the difference in classification uncertainty and the entropy of variational distributions, and thus correctly classifies this sample as orange.

propose to use a variational approach [8] to estimate the input's conditional likelihood. Our variational approach uses alternative distributions to sample noise more effectively, providing better efficiency and uncertainty measurements.

We use variational methods to find an alternative distribution of noise for each class, thus improving the efficiency and accuracy of likelihood estimates. For each class y, we find an alternative distribution of noise that, when added to the input image, yield a higher likelihood for class y. This yields a Bayesian method for classification where we estimate the likelihood of the input conditional on each class separately. Intuitively, variational methods find the region of noise for each class where the model is give higher likelihood to that class. Such methods exploit the neural model's spatial information around the input sample, concentrating samples on regions that are most significant for classification. Figure 4.1 illustrates this idea.

We want to note an important difference between the randomized smoothing method illustrated in Figure 4.1 and the original smoothing method. Randomized smoothing converts model predictions into hard labels and classifies with a majority vote. Therefore, Equation 4.1 uses $\mathbb{I}\left[F(x+\epsilon)=y\right]$ in the expectation. Meanwhile, in Figure 4.1, the expectation is taken directly over $f(x+\epsilon)$, the neural network's output. Using hard labels is critical for the

certifying the robustness of randomized smoothing. However, as we focus on empirical robustness, this difference is not critical. Our variational method uses the expectation of $f(x+\epsilon)$, the neural network's output, instead of hard labels. Replacing network outputs with hard labels may improve robustness, as hard-labels contain less information about the loss surface.

4.2 Method

In this section, we first introduce how variational inference can improve randomized smoothing. Then we present our variational method.

Notations

In this section, we assume that each datum (x, y) is sampled from a real data distribution p_d. A neural network is represented by f and its kth output is f_k. When a neural network is trained with cross-entropy loss, its f_k can be viewed as an estimation of $\log p(x|k)$, which leads to

$$\log p(y|x) = \mathcal{L}_y(f(x)) = \log \frac{e^{f_y(x)}}{\sum_k e^{f_k(x)}} \tag{4.2}$$

Equation 4.2 is the commonly used log-softmax function for converting neural network outputs to a log likleihood.

Besides, randomized smoothing samples noise from a prior distribution, which we refer to as p_ϵ. Typically, this is a isotropic Gaussian distribution $\mathcal{N}(0, \sigma I)$. Therefore, when our variational distribution $q(\epsilon)$ is also a Gaussian distribution, their KL divergence $\mathcal{D}_{\mathrm{KL}}(q(\epsilon)\|p_\epsilon)$ can be computed directly.

Randomized smoothing and variational inference

Randomized smoothing is a method to estimate sample likelihoods [21]. Given a datum $(x, y) \sim p_d$, we train a classifier f such that

$$\mathbb{E}_{\epsilon \sim p_\epsilon} e^{\mathcal{L}_y(f(x+\epsilon))} \approx p_d(y|x). \tag{4.3}$$

Equation 4.3 is an approximation that ignores a procedure of randomized smoothing where neural network outputs are converted to hard labels. However, as we have explained in Section 4.1, this procedure provides certified robustness while we focus on empirical robustness. Thus, our approximation does not influence our analysis.

A direct estimation of Equation 4.3 is through sampling independent ϵ from p_ϵ. THis method is used by randomized smoothing but is inefficient. We propose an alternative method where we rely on variational methods [8]. Given a neural network f, we estimate $\log p_d(y|x)$

with a variational distribution $q(\epsilon)$ of noise. $\log p_d(y|x)$ can be estimated by the expectation of $p(y|x,\epsilon)$ under $q(\epsilon)$ and the divergence between $q(\epsilon)$ and p_ϵ:

$$\log p(y|x) - \mathcal{D}_{\mathrm{KL}}\left(q(\epsilon)\|p(\epsilon|y,x)\right) = \mathbb{E}_{\epsilon\sim q}\log p(y|\epsilon,x) - \mathcal{D}_{\mathrm{KL}}\left(q(\epsilon)\|p_\epsilon\right) \tag{4.4}$$

Equation 4.4 is the foundation of variational inference [8] and its right side is called the Evidence Lower Bound (ELBO). Specifically, as $\mathcal{D}_{\mathrm{KL}}\left(q(\epsilon)\|p(\epsilon|y,x)\right)$ is non-negative, The ELBO is a lower bound of the sample's likelihood. Thus, with Equation 4.4, we can obtain a lower bound of $p(y|x)$ for each $(x,y)\sim p_d$:

$$\log p(y|x) \geq \mathrm{ELBO}_y \tag{4.5}$$

$$= \mathbb{E}_{\epsilon\sim q}\log f_\theta(x+\epsilon)_y - \mathcal{D}_{KL}\left(q(\epsilon)\|p_\epsilon\right) \tag{4.6}$$

Equation 4.6 inspires our variational method. Given an input x, for each class y,

- Find a variational $q(\epsilon)$ that maximizes ELBO_y.
- Use ELBO_y as the estimate of $\log p(y|x)$.

An overview of our variational method Our variational method estimates a sample's conditional likelihood separately for each class. For each class y, we find a variational distribution of noise $q_y^*(\epsilon)$ that minimizes the ELBO:

$$q_y^* := \arg\min_q \mathbb{E}_{\epsilon\sim q}\log p(y|\epsilon,y) - \mathcal{D}_{\mathrm{KL}}(q\|p_\epsilon) \tag{4.7}$$

A sample's conditional likelihood $p(y|x)$ is thus estimated by q_y^* with

$$p(y|x) := \mathbb{E}_{\epsilon\sim q_y^*}\mathcal{L}_y(f(x+\epsilon)) - \mathcal{D}_{\mathrm{KL}}(q_y^*\|p(\epsilon)) \tag{4.8}$$

where $\log p(x|\epsilon,y)$ is given by the log-softmax function $\mathcal{L}_y(f(x+\epsilon))$ in Equation 4.8, as explained in Equation 4.2.

A simplified method The reparameterization trick [58] allows us to find q_y^* through optimization, but this process is time-consuming. Specifically, optimizing a distribution's covariance matrix is challenging: large covariance values lead to slower convergence while small covariance values lead to numerical instability. Therefore, we propose an alternative simplified optimization method where the covariance matrix is fixed.

The optimization has two steps. First, we find a maximum likelihood estimation of the noise. Second, we use this estimated noise as the mean and use a fixed covariance matrix to construct q_y^*. With this simplified method, the optimization is similar to finding an adversarial perturbation in a targeted attack; thus, the method first tries to find

$$\mu_y^* := \arg\min_{\|\epsilon\|_p\leq d} \mathcal{L}_y(f(x+\epsilon)) \tag{4.9}$$

Algorithm 5 Variational Randomized Smoothing: a simplified method.

Inputs: Hyperparameters $d > 0$, $\sigma > 0$.
Given an input x and a base model f.
Initialize $logits = []$
for each class y **do**
  $\mu_y^* \leftarrow \arg\min_{\|\epsilon\|_p \leq d} \mathcal{L}_y(f(x+\epsilon))$
  Sample ϵ from $N(\mu_y^*, \sigma\mathbb{I})$.
  Append $\mathcal{L}_y(f(x+\epsilon))$ to $logits$ as an estimate of $p(y|x)$.
end for
return Return $logits$.

where $\mathcal{L}$ is the log-softmax function explained in Equation 4.2 and d is a hyperparameter. Then, the variational distribution is given by

$$q_y^* = N(\mu_y^*, \Sigma)$$

where Σ is a fixed covariance matrix and in our experiments we use $\Sigma = \sigma I$, a diagonal matrix with variance σ. The simplified method is summarized as Algorithm 5.

To obtain stable gradients when attacking the model, we use the expectation-over-transformation trick [14] sampling more ϵ from the optimized variational distribution and using the average $\mathcal{L}_y(f(x+\epsilon))$ as the estimate of $p(y|x)$.

4.3 Experiments

We evaluated our methods on CIFAR10 [62] with various base models.

Models We evaluate the variational method with both bare classifiers and robust classifiers.

- **Bare pretrained** models are pretrained CIFAR10 classifiers.
- **Bare Gaussian** models are CIFAR10 classifiers trained with Gaussian noises. We used the variational method's prior of Gaussian noise to data augmentation. By default, the Gaussian noise's standard deviation is 0.05.
- **Robust** models are CIFAR10 classifiers adversarially trained for L_∞ robustness. As we evaluated with L_∞ bounded attacks, we chose a model trained with Fast Adversarial Training [110] to be robust against L_∞ attacks. The bound of attacks, used at both training and evaluation time, is $8/255$.

We use ResNet18 [43] as the basic model. The **Bare pretrained** and **Bare Gaussian** implementation is obtained from a public repository [1]. The **Robust** is given by the robustbench [22] library; we use the model pretrained with fast adversarial training [110].

[1] `https://github.com/kuangliu/pytorch-cifar`

We use a projected optimization to find optimal variational distributions. At each step, the mean vector is projected to the same L_p norm ball used for attacks. Choosing the same norm ball guarantees that the optimization can recover the natural input. After projection, we choose a fixed standard deviation of 0.05.

We used one-step optimization in our experiments by default. We also initialized the search with random points with in the L_p norm ball. Studies showed that one-step optimization with random starting points is already a good method for finding adversarial perturbations [110]. Our experiment results showed that increasing the number of steps leads to an increase of robustness, but improvements led by increased optimization steps are not as significant as adding our variational method to a base model. Meanwhile, increasing the number of optimization steps would significantly increase the inference time. For efficiency, we thus use one-step optimization as the default choice.

Threat model We evaluate our variational method with white-box attacks. For standard evaluation, we use the AutoAttack library [24]. AutoAttack library has both gradient-based attacks and gradient-free attacks; its standard version comprises of four attacks: APGD-CE (untargeted gradient-based attack), APGD-DLR (targeted, gradient-based attack), FAB (targeted, gradient-based attack) [23], and Square attack (untargeted, score-based attack) [1]. AutoAttack is designed as a standard benchmark for adversarial robustness.

To adaptively attack our method, we apply the expectation-over-transformation (EOT) trick [14] to estimate gradients under random noises. For each variational distribution Q_y^*, we use the mean of 5 random samples to estimate the conditional likelihood.

Table 4.2 presents the change of model robustness as we increase the strength of attacks. It confirms that with sufficiently large strength, our attack can eventually break the variational optimization defense. Therefore, our attack is a valid evaluation for the robustness of our variational defense.

Datasets We evaluate our method on CIFAR10 [62]. CIFAR10 is a standard benchmark for adversarial robustness. Another advantage of experimenting with CIFAR10 is that many pretrained models are available on CIFAR10, so we can evaluate our method with a rich variety of base models.

4.4 Results

Our variational method improves the robustness of bare models. We observe an improved robustness on bare models with our variational method. As Table 4.1 shows, adversarial accuracy improves by 8-10 absolute percentage when our variational method is applied. Furthermore, the variational method does not hurt natural accuracy. In both cases, the model's natural clean accuracy stays unchanged when the variational method is applied. This is an important property as many defenses sacrifice natural accuracy for better adversarial robustness [70, 118].

Table 4.1: The variational method improves the robustness of bare models. Reported numbers are natural accuracy and adversarial accuracy. Adversarial accuracy is measured by AutoAttack [24] with $\epsilon_\infty = 2/255$. We used one-step optimization in our variational methods.

Model	With inference?	Natural	Adversarial
Bare pretrained	N	92.5	20.3
Bare pretrained	Y (ours)	92.8	30.6
Bare Gaussian	N	92.7	64.2
Bare Gaussian	Y (ours)	92.7	72.3

Table 4.2: The adversarial accuracy of **Bare Gaussian** with our variational inference method as attack strength improves. Adversarial accuracy is measured by AutoAttack [24]. We used one-step optimization in our variational methods.

ϵ_∞	2/255	4/255	8/255
Adversarial accuracy	72.3	36.8	1.4

Our variational method has weaker improvements on robust models. Applied to robust models, our variational method shows smaller improvements than bare models. As Table 4.3 shows, adversarial accuracy slightly improves when our variational method is applied. Although the improved adversarial accuracy is not as significant as for bare models, we obtain this improvement with a modest loss of natural accuracy (83.3% $\rightarrow$ 82.6%). Thus, our variational method is still useful when the base model is a robust model.

Table 4.3: Natural and adversarial accuracy of our variational methods with a robust base model. The adversarial accuracy is measured by AutoAttack with $\epsilon_\infty = 8/255$. The base model is a ResNet18 trained with $\epsilon_\infty = 8/255$ adversarial attacks as well.

Model	With inference?	Natural	Adversarial
Wong et al. [110]	N	83.3	43.2
Wong et al. [110]	Y (ours)	82.6	45.1

Increasing the number of optimization steps improves our variational method. Our variational method's robustness improves as the number of optimization steps increase. When **Bare-Gaussian** uses 10 optimization steps, the variational method achieves 74.4% adversarial accuracy with L_∞ bounded attack ($\epsilon = 2/255$), which is a 2.1 percentage points improvement over 1-step optimization while maintaining the same natural accuracy.

Although increasing the number of optimization steps improves adversarial robustness, the cost is more inference time. As the variational method finds an optimal variational distribution for each class, adding one extra optimization step roughly causes 10 extra forward and backward passes at inference time on CIFAR10. Considering the base model only has one forward pass, the cost of adding optimization steps is significant. Therefore, one-step optimization with random starting points could be a good balance in practice: it increase the adversarial accuracy of base models at a minimal (but still significant compared to bare models) computational overhead.

4.5 Related Works

Our method is based on randomized smoothing. Randomized smoothing uses smoothed classifiers to improve robustness against adversarial attacks. It can achieve certified robustness [21, 67] and it is much simpler compared to other approaches for certified robustness [119, 40]. Recent studies have extended randomized smoothing to more attacks [65, 66] and demonstrated that randomized smoothing can achieve empirical robustness as well [84]. Our study focuses on empirical robustness investigating whether we can obtain better empirical robustness with smoothed classifiers through variational inference.

Our method dynamically finds an alternative noise distribution at test time. Test-time adaptation has been used to defend against adversarial attacks [71] and general corruptions [107]. Compared to previous methods for test-ime adaptation, our variational method is derived from a theoretical model and its class-conditional adaptation is unique.

Our method is a Bayesian classifier that leverages the power of discriminative classifiers. Previous studies have examined the combination of generative and discriminative models as well [9, 7, 116], but their investigation does not focus on adversarial robustness nor do they examine neural networks. In comparison, our work focuses specifically on adversarial examples and uses randomized smoothing to connect discriminative and generative models.

4.6 Summary and Future Work

This chapter presents a variational method that can improve the robustness of regular discriminative models. Inspired by our generative approach, this variational method estimates conditional likelihoods with variational sampling. For each class, our method finds an optimal variational distribution where the sample's conditional likelihood is maximized. In practice, we use targeted attacks to find the optimal variational distribution and add Gaussian noise

to smooth these distributions. The standard AutoAttack evaluation with EOT confirms that our method can improve the robustness of bare models and robust models.

Our method brings significant improved robustness of unprotected models when used jointly with Gaussian-augmented training against small perturbations. Furthermore, unlike adversarial training [70] or TRADES [118], our method does not impact natural accuracy, which is a desired property considering adversarial examples are rare in real world. However, our method brings minimal improvements over robust base models. We hypothesize that robust models are trained for maximal performance under bounded perturbations, making them unstable under Gaussian noise. Therefore, a combination of adversarial training and Gaussian noise augmentation may further improve the effect of our variational method.

Our method adapts model inputs dynamically at test-time. Recent studies showed that test-time adaptation can improve model robustness against adversarial attacks and general corruptions [107]. Therefore, it is interesting to investigate a combination of input perturbation and model perturbation, where both input noises and model weights are modelled as a posterior distribution and we can use variational methods for efficient estimation of the expectation.

Chapter 5

Conclusion

Generative classifiers are a promising alternative for robustly classifying images. They model the distribution of images, instead of labels, which makes them fundamentally different from discriminative models. Studies on MNIST [64] demonstrated that generative classifiers are more robust than discriminative models [95] and more consistent with human perception [37]. However, generative classifiers face several challenges, especially on complex images.

There are two major obstacles for generative classifiers to become robust classifiers on complex images. First, generative classifiers' classification accuracy on natural images is worse than discriminative models. Our studies have further demonstrated several challenges for generative classifiers to accurately classify complex images. Second, generative classifiers rely on perceptual distances and encoders to improve classification accuracy. However, as both perceptual distances and encoders rely on neural networks, they may undermine the robustness of generative classifiers while improving classification accuracy. Our study presents some solutions to the first obstacle.

To accurately classify natural images, generative classifiers want to simultaneously achieve two objectives: accepting an in-domain sample and rejecting all out-of-domain samples. State-of-the-art generative models, such as BigGAN [12, 122] can model in-domain samples well; combined with powerful inversion methods [125], we can find a precise latent representation of an in-domain sample. However, our study shows that rejecting out-of-domain samples could be challenging. Specifically, in Chapter 3, we show that most methods that invert in-domain samples well would fail on rejecting out-of-domain samples. Therefore, generative classifiers require us to view generative models from a more balanced perspective.

Chapter 3 analyzes the trade-off between in-domain and out-of-domain samples in details. We examined two reasons that generative classifiers fail on out-of-domain samples: overpowered generative models and overfitted perceptual distance. We compared various inference methods and two datasets. Our results confirm that both problems exist on CIFAR10 and are responsible for generative classifiers' classification errors. To address overpowered generative models, we present a **Enc-D** method such that an inversion process will yield in-domain latent representations. To address overfitted perceptual distance, we found that using corruption-robust models for perceptual distance can improve the performance of generative classifiers.

When images become more complex, such as ImageNet, our experiments suggested that overfitted perceptual distance is the main culprit for classification errors.

Chapter 2 presents a more direct response to the shortcomings of generative classifiers. In this work, we relied on outlier exposure [45] to reject out-of-domain samples and adversarial autoencoders [72] to model in-domain samples. We also presented an inference method that works better with generative classifiers. Our model, E-ABS, has successfully extended a simple generative classifier, ABS [95], to more challenging image domains. Experiments confirmed that E-ABS has better adversarial robustness than discriminative models on several datasets. Therefore, our study shows that addressing the problems of in-domain and out-of-domain samples is the key to extend generative classifiers.

The challenges of generative classifiers come from modelling complex image distributions. Discriminative models are more successful because they model the label distribution. Chapter 4 tries to connect the generative world and the discriminative world through randomized smoothing [21]. Instead of modelling image distributions, we model the label distribution under expectation. Such a method allows us to use discriminative classifiers under a Bayesian formulation. Our experiments indicated that our method can improve adversarial robustness of unprotected models, showing some promising progress to bring the two worlds together.

Efficiency is another challenge of generative classifiers. Compared to discriminative models, generative classifiers rely on an optimization process to estimate conditional likelihoods, making it much slower at inference time. This is beyond the scope of this thesis, but we have been working on improving the efficiency of generative classifiers as well. For example, hierarchically structuring class-conditional generative models could reduce the dependency of inference time on the number of classes from $O(K)$ to $O(\log K)$. More efficient inversion methods would also improve the inference time of generative classifiers.

As neural networks have become fundamental in our life, the pursue of reliable and trustworthy building blocks becomes more urgent than ever. Generative classifiers are more robust and more interpretable than discriminative models, so they could be a critical component in a robust neural network system. Our studies have showed several challenges of generative classifiers and some encouraging progress. Future studies may provide better answers to these challenges, making neural networks understandable, reliable, and trustworthy.

Bibliography

[1] Maksym Andriushchenko et al. *Square Attack: a query-efficient black-box adversarial attack via random search*. 2020. arXiv: 1912.00049 [cs.LG].

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *arXiv preprint arXiv:1701.07875* (2017).

[3] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples". In: *arXiv preprint arXiv:1802.00420* (2018).

[4] Samaneh Azadi et al. *Discriminator Rejection Sampling*. 2019. arXiv: 1810.06758 [stat.ML].

[5] Shane Barratt and Rishi Sharma. *A Note on the Inception Score*. 2018. arXiv: 1801.01973 [stat.ML].

[6] David Bau et al. "Semantic photo manipulation with a generative image prior". In: *ACM Transactions on Graphics* 38.4 (July 2019), pp. 1–11. ISSN: 1557-7368. DOI: 10.1145/3306346.3323023. URL: http://dx.doi.org/10.1145/3306346.3323023.

[7] JM Bernardo et al. "Generative or discriminative? getting the best of both worlds". In: *Bayesian statistics* 8.3 (2007), pp. 3–24.

[8] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773. eprint: https://doi.org/10.1080/01621459.2017.1285773. URL: https://doi.org/10.1080/01621459.2017.1285773.

[9] Guillaume Bouchard and Bill Triggs. "The Tradeoff Between Generative and Discriminative Classifiers". In: *16th IASC International Symposium on Computational Statistics (COMPSTAT '04)*. Prague, Czech Republic, Aug. 2004, pp. 721–728. URL: https://hal.inria.fr/inria-00548546.

[10] Olivier Bousquet et al. "From optimal transport to generative modeling: the VEGAN cookbook". In: *arXiv preprint arXiv:1705.07642* (2017).

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

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: 1809.11096 [cs.LG].

[13] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders". In: *arXiv preprint arXiv:1509.00519* (2015).

[14] N. Carlini and D. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. May 2017, pp. 39–57.

[15] Nicholas Carlini et al. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).

[16] Davide Castelvecchi. "Can we open the black box of AI?" In: *Nature News* 538.7623 (2016), p. 20.

[17] Tong Che et al. *Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling*. 2020. arXiv: 2003.06060 [cs.LG].

[18] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[19] Ting Chen et al. *On Self Modulation for Generative Adversarial Networks*. 2019. arXiv: 1810.01365 [cs.LG].

[20] Hyunsun Choi, Eric Jang, and Alexander A Alemi. "WAIC, but why? Generative ensembles for robust anomaly detection". In: *arXiv preprint arXiv:1810.01392* (2018).

[21] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *arXiv preprint arXiv:1902.02918* (2019).

[22] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. "Provable robustness of ReLu networks via maximization of linear regions". In: *arXiv preprint arXiv:1810.07481* (2018).

[23] Francesco Croce and Matthias Hein. *Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack*. 2020. arXiv: 1907.02044 [cs.LG].

[24] Francesco Croce and Matthias Hein. *Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks*. 2020. arXiv: 2003.01690 [cs.LG].

[25] Francesco Croce et al. "RobustBench: a standardized adversarial robustness benchmark". In: *arXiv preprint arXiv:2010.09670* (2020).

[26] Ekin D. Cubuk et al. *Intriguing Properties of Adversarial Examples*. 2017. arXiv: 1711.02846 [stat.ML].

[27] Bin Dai and David Wipf. "Diagnosing and enhancing VAE models". In: *arXiv preprint arXiv:1903.05789* (2019).

[28] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[29] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[30] Thomas G Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[31] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *arXiv preprint arXiv:1605.08803* (2016).

[32] Alexey Dosovitskiy and Thomas Brox. *Generating Images with Perceptual Similarity Metrics based on Deep Networks*. 2016. arXiv: 1602.02644 [cs.LG].

[33] Logan Engstrom et al. *Adversarial Robustness as a Prior for Learned Representations*. 2019. arXiv: 1906.00945 [stat.ML].

[34] Kevin Eykholt et al. *Robust Physical-World Attacks on Deep Learning Models*. 2017. arXiv: 1707.08945 [cs.CR].

[35] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1632–1640.

[36] Ethan Fetaya et al. "Understanding the limitations of conditional generative models". In: *arXiv preprint arXiv:1906.01171* (2019).

[37] Tal Golan, Prashant C Raju, and Nikolaus Kriegeskorte. "Controversial stimuli: pitting neural networks against each other as models of human recognition". In: *arXiv preprint arXiv:1911.09288* (2019).

[38] Mingming Gong et al. "Twin auxiliary classifiers gan". In: *Advances in neural information processing systems* 32 (2019), p. 1328.

[39] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[40] Sven Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models*. 2019. arXiv: 1810.12715 [cs.LG].

[41] Sven Gowal et al. *Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples*. 2021. arXiv: 2010.03593 [stat.ML].

[42] Minghao Guo et al. "When NAS Meets Robustness: In Search of Robust Architectures against Adversarial Attacks". In: *arXiv preprint arXiv:1911.10695* (2019).

[43] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[44] Dan Hendrycks and Thomas Dietterich. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*. 2019. arXiv: 1903.12261 [cs.LG].

[45] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. "Deep anomaly detection with outlier exposure". In: *arXiv preprint arXiv:1812.04606* (2018).

[46] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG].

[47] Irina Higgins et al. "beta-VAE: Learning basic visual concepts with a constrained variational framework." In: *ICLR* 2.5 (2017), p. 6.

[48] Jeremy Howard. *Imagenette*. URL: https://github.com/fastai/imagenette/.

[49] Andrew Ilyas et al. *Adversarial Examples Are Not Bugs, They Are Features*. 2019. arXiv: 1905.02175 [stat.ML].

[50] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[51] Ajil Jalal et al. "The Robust Manifold Defense: Adversarial Training using Generative Models". In: (2019). arXiv: 1712.09196 [cs.CV].

[52] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. 2016. arXiv: 1603.08155 [cs.CV].

[53] An Ju and David Wagner. "E-ABS: Extending the Analysis-By-Synthesis Robust Classification Model to More Complex Image Domains". In: *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*. AISec'20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 25–36. ISBN: 9781450380942. DOI: 10.1145/3411508.3421382. URL: https://doi.org/10.1145/3411508.3421382.

[54] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. *E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles*. 2019. arXiv: 1906.03973 [cs.CV].

[55] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. "Contrastive Representation Learning: A Framework and Review". In: *IEEE Access* 8 (2020), pp. 193907–193934. ISSN: 2169-3536. DOI: 10.1109/access.2020.3031549. URL: http://dx.doi.org/10.1109/ACCESS.2020.3031549.

[56] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021. arXiv: 2004.11362 [cs.LG].

[57] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[58] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[59] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10215–10224.

[60] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. *On the effectiveness of adversarial training against common corruptions*. 2021. arXiv: 2103.02325 [cs.LG].

[61] Jernej Kos, Ian Fischer, and Dawn Song. "Adversarial Examples for Generative Models". In: *2018 IEEE Security and Privacy Workshops (SPW)*. 2018, pp. 36–42. DOI: 10.1109/SPW.2018.00014.

[62] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[63] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533* (2016).

[64] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

[65] Alexander Levine and Soheil Feizi. *(De)Randomized Smoothing for Certifiable Defense against Patch Attacks*. 2021. arXiv: `2002.10733 [cs.LG]`.

[66] Alexander Levine and Soheil Feizi. *Improved, Deterministic Smoothing for L1 Certified Robustness*. 2021. arXiv: `2103.10834 [cs.LG]`.

[67] Bai Li et al. *Certified Adversarial Robustness with Additive Noise*. 2019. arXiv: `1809.03113 [cs.LG]`.

[68] Yingzhen Li, John Bradshaw, and Yash Sharma. "Are Generative Classifiers More Robust to Adversarial Attacks?" In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 3804–3814. URL: `http://proceedings.mlr.press/v97/li19a.html`.

[69] Radek Mackowiak et al. *Generative Classifiers as a Basis for Trustworthy Image Classification*. 2020. arXiv: `2007.15036 [cs.CV]`.

[70] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[71] Kaleel Mahmood et al. *BUZz: BUffer Zones for defending adversarial examples in image classification*. 2020. arXiv: `1910.02785 [cs.LG]`.

[72] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015).

[73] Qi Mao et al. "Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[74] Takeru Miyato and Masanori Koyama. *cGANs with Projection Discriminator*. 2018. arXiv: `1802.05637 [cs.LG]`.

[75] Takeru Miyato et al. *Spectral Normalization for Generative Adversarial Networks*. 2018. arXiv: `1802.05957 [cs.LG]`.

[76] 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*. 2016, pp. 2574–2582.

[77] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[78] Eric Nalisnick et al. "Detecting out-of-distribution inputs to deep generative models using a test for typicality". In: *arXiv preprint arXiv:1906.02994* (2019).

[79] Eric Nalisnick et al. "Do deep generative models know what they don't know?" In: *arXiv preprint arXiv:1810.09136* (2018).

[80] Yuval Netzer et al. "Reading digits in natural images with unsupervised feature learning". In: (2011).

[81] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans". In: *International conference on machine learning*. PMLR. 2017, pp. 2642–2651.

[82] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[83] Phillip Pope, Yogesh Balaji, and Soheil Feizi. "Adversarial robustness of flow-based generative models". In: *arXiv preprint arXiv:1911.08654* (2019).

[84] Edward Raff et al. "Barrage of Random Transforms for Adversarially Robust Defense". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[85] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox: A Python toolbox to benchmark the robustness of machine learning models". In: *arXiv preprint arXiv:1707.04131* (2017). arXiv: 1707.04131. URL: http://arxiv.org/abs/1707.04131.

[86] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

[87] Leslie Rice, Eric Wong, and Zico Kolter. "Overfitting in adversarially robust deep learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 8093–8104. URL: http://proceedings.mlr.press/v119/rice20a.html.

[88] Jérôme Rony et al. "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4322–4330.

[89] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. "Distribution matching in variational inference". In: *arXiv preprint arXiv:1802.06847* (2018).

[90] Nirupam Roy et al. "Inaudible Voice Commands: The Long-Range Attack and Defense". In: *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 547–560. ISBN: 978-1-939133-01-4. URL: https://www.usenix.org/conference/nsdi18/presentation/roy.

[91] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[92] Hadi Salman et al. *Do Adversarially Robust ImageNet Models Transfer Better?* 2020. arXiv: 2007.08489 [cs.CV].

[93] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models". In: (2018). arXiv: 1805.06605 [cs.CV].

[94] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. "A U-Net Based Discriminator for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[95] Lukas Schott et al. "Towards the first adversarially robust neural network model on MNIST". In: *arXiv preprint arXiv:1805.09190* (2018).

[96] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[97] Chawin Sitawarin et al. *DARTS: Deceiving Autonomous Cars with Toxic Signs*. 2018. arXiv: 1802.06430 [cs.CR].

[98] Leslie N. Smith. *Cyclical Learning Rates for Training Neural Networks*. 2017. arXiv: 1506.01186 [cs.CV].

[99] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[100] J. Stallkamp et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition". In: *Neural Networks* 32 (2012), pp. 323–332.

[101] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[102] Domen Tabernik and Danijel Skočaj. "Deep Learning for Large-Scale Traffic-Sign Detection and Recognition". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[103] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *arXiv preprint arXiv:1511.01844* (2015).

[104] Radu Timofte, Karel Zimmermann, and Luc Van Gool. "Multi-view traffic sign detection, recognition, and 3D localisation". In: *Machine vision and applications* 25.3 (2014), pp. 633–647.

[105] Ilya Tolstikhin et al. "Wasserstein auto-encoders". In: *arXiv preprint arXiv:1711.01558* (2017).

[106] Florian Tramer et al. "On adaptive attacks to adversarial example defenses". In: *arXiv preprint arXiv:2002.08347* (2020).

[107] Dequan Wang et al. *Tent: Fully Test-time Adaptation by Entropy Minimization*. 2021. arXiv: 2006.10726 [cs.LG].

[108] Ting-Chun Wang et al. *High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs*. 2018. arXiv: 1711.11585 [cs.CV].

[109] Rey Reza Wiyatno et al. *Adversarial Examples in Modern Machine Learning: A Review*. 2019. arXiv: 1911.05268 [cs.LG].

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

[111] Weihao Xia et al. *GAN Inversion: A Survey*. 2021. arXiv: 2101.05278 [cs.CV].

[112] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[113] Cihang Xie et al. "Adversarial Examples for Semantic Segmentation and Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). DOI: 10.1109/iccv.2017.153. URL: http://dx.doi.org/10.1109/ICCV.2017.153.

[114] Cihang Xie et al. "Feature Denoising for Improving Adversarial Robustness". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[115] Bing Xu et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[116] Jing-Hao Xue and D Michael Titterington. "On the generative–discriminative tradeoff approach: Interpretation, asymptotic efficiency and classification performance". In: *Computational statistics & data analysis* 54.2 (2010), pp. 438–451.

[117] X. Yuan et al. "Adversarial Examples: Attacks and Defenses for Deep Learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.9 (2019), pp. 2805–2824. DOI: 10.1109/TNNLS.2018.2886017.

[118] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *arXiv preprint arXiv:1901.08573* (2019).

[119] Huan Zhang et al. *Towards Stable and Efficient Training of Verifiably Robust Neural Networks*. 2019. arXiv: 1906.06316 [cs.LG].

[120] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV].

[121] Shengyu Zhao et al. *Differentiable Augmentation for Data-Efficient GAN Training*. 2020. arXiv: 2006.10738 [cs.CV].

[122] Yang Zhao et al. *Feature Quantization Improves GAN Training*. 2020. arXiv: 2004.02088 [cs.LG].

[123] Jiachen Zhong, Xuanqing Liu, and Cho-Jui Hsieh. *Improving the Speed and Quality of GAN by Adversarial Training*. 2020. arXiv: 2008.03364 [cs.LG].

[124] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[125] Jiapeng Zhu et al. "In-Domain GAN Inversion for Real Image Editing". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 592–608. ISBN: 978-3-030-58520-4.

[126] Jun-Yan Zhu et al. *Generative Visual Manipulation on the Natural Image Manifold*. 2018. arXiv: 1609.03552 [cs.CV].

[127] Jun-Yan Zhu et al. *Toward Multimodal Image-to-Image Translation*. 2018. arXiv: 1711.11586 [cs.CV].