Université de Montréal

Contributions to Generative Models and Their Applications

par

Tong (Gerry) Che

Department of Computer Science and Operation Research
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Discipline

October 25, 2022

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Contributions to Generative Models and Their Applications

présentée par

Tong (Gerry) Che

a été évaluée par un jury composé des personnes suivantes :

Liam Paull
(président-rapporteur)

Yoshua Bengio
(directeur de recherche)

Aaron Courville
(membre du jury)

Honglak Lee
(examinateur externe)

(représentant du doyen de la FESP)

Résumé

Les modèles génératifs sont une grande classe de modèles d'apprentissage automatique pour l'apprentissage non supervisé. Ils ont diverses applications dans l'apprentissage automatique et l'intelligence artificielle. Dans cette thèse, nous discutons de nombreux aspects des modèles génératifs et de leurs applications à d'autres problèmes d'apprentissage automatique. En particulier, nous discutons de plusieurs sujets importants dans les modèles génératifs, y compris comment stabiliser la formation GAN discrète avec un échantillonnage d'importance, comment faire un meilleur échantillonnage à partir de GAN en utilisant une connexion avec des modèles basés sur l'énergie, comment mieux former des modèles auto-régressifs avec l'aide d'une formulation de modèle basée sur l'énergie, ainsi que deux applications de modèles génératifs à d'autres problèmes d'apprentissage automatique, l'une sur les réseaux résiduels, l'autre sur la vérification de la sécurité.

Mots-clés : apprentissage automatique, Deep Learning, Intelligence Artificielle, Modèles Génératifs

Abstract

Generative models are a large class of machine learning models for unsupervised learning. They have various applications in machine learning and artificial intelligence. In this thesis, we discuss many aspects of generative models and their applications to other machine learning problems. In particular, we discuss several important topics in generative models, including how to stabilize discrete GAN training with importance sampling, how to do better sampling from GANs using a connection with energy-based models, how to better train auto-regressive models with the help of an energy-based model formulation, as well as two applications of generative models to other machine learning problems, one about residual networks, the other about safety verification.

Keywords: Machine Learning, Deep Learning, Artificial Intelligence, Generative Models

Contents

Résumé 5

Abstract 7

List of tables 13

List of figures 15

Liste des sigles et des abréviations 17

Chapter 1. Background and Introduction 19

1.1. Motivations and Overview 19

1.2. Restricted Boltzmann Machines 21

1.3. Generative Adversarial Networks 22

1.3.1. Original GANs 23

1.3.2. f-GANs 23

1.3.3. IPM and W-GANs 24

1.4. Autoregressive Models 26

1.4.1. Exposure Bias Problem 27

1.5. Variational Auto-encoders 28

1.6. Deep Energy-based Models 29

1.7. Main Contributions 30

Chapter 2. Stabilizing Training of Discrete GANs with Importance Sampling 33

2.1. Preface 33

2.1.1. My Role and Contributions 33

2.1.2. Motivation and Context 33

2.1.3. Main Contribution 34

2.1.4. Follow-up works and Impacts 35
2.2. Problem Formulation And Analysis 35
2.3. Maximum-Likelihood Augmented Discrete Generative Adversarial Networks .. 37
2.3.1. Basic Model of MaliGAN 37
2.3.2. Analysis 38
2.3.3. Variance Reduction Techniques in MaliGAN 40
2.3.3.1. **Monte Carlo Tree Search** 40
2.3.3.2. **Mixed MLE-Mali Training** 41
2.3.3.3. Single real data based renormalization 42
2.4. Experiments 43
2.4.1. Discrete MNIST 43
2.4.2. Poem Generation 43
2.4.3. Sentence-Level Language Modeling 44
2.4.4. Conditional Dialog Generation 45
2.5. Related Work 46
2.6. Discussions and Future Work 47
Chapter 3. Better Auto-Regressive Model Training with Energy-Based Models 49
3.1. Preface 49
3.1.1. My Role and Contributions In This Project 49
3.1.2. Motivation and Context 49
3.1.3. Main Contribution 50
3.1.4. Follow-up Works and Impacts 51
3.2. Exposure bias and incoherence problems in Auto-regressive models 51
3.3. Integrate EBMs into auto-regressive models seamlessly 52
3.4. Optimization 53
3.5. Training and Inference 56
3.6. Experiments 57
3.6.1. Application to Neural Machine Translation 57
3.6.2. Application to Language Modeling 59

3.6.3. Application to Image Generation 60
3.7. Related Works 61
3.7.1. auto-regressive generative models 61
3.7.2. Energy-based models 62
3.8. Conclusion 62
Chapter 4. Improved GAN sampling with Energy-Based Models 63
4.1. Preface 63
4.1.1. My Role and Contributions In This Project 63
4.1.2. Motivation and Context 63
4.1.3. Main Contribution 64
4.1.4. Follow-up Works and Impacts 64
4.2. Methodology 65
4.2.1. GANs as an Energy-Based Model 65
4.2.2. Rejection Sampling and MCMC in Latent Space 66
4.2.3. Main Theorem 66
4.2.4. Sampling Wasserstein GANs with Langevin Dynamics 68
4.2.5. DDLS Algorithm 69
4.2.6. Practical Issues and the Mode Dropping Problem 70
4.3. Related Work 71
4.4. Experimental results 72
4.4.1. Synthetic dataset 72
4.4.2. CIFAR-10 and CelebA 74
4.4.3. ImageNet 76
Chapter 5. Application: Verifying Deep Discriminative Models with Deep Generative Models 79
5.1. Preface 79
5.1.1. My Role and Contributions In This Project 79
5.1.2. Motivation and Context 79
5.1.3. Main Contribution 80
5.1.4. Follow-up Works and Impacts 81
5.2. Related Work 81

5.3. Methodology 83
5.3.1. Basic Model 84
5.3.2. Disentanglement constraints for anomaly detection 85
5.3.3. Measuring the likelihood as anomaly score 85
5.3.4. Theoretical Justification 86
5.3.5. Intuitive Justifications 87
5.3.6. Can We Replace VAEs with Other Density Estimators? 88
5.4. Experimental results 88
5.4.1. Detecting OOD samples for classification 91
5.4.2. Detecting adversarial examples 94
5.4.3. OOD for image captioning 95
5.5. Conclusions 96
Chapter 6. Future Works and Summary of Contributions 97
6.1. Overall Conclusions 97
6.2. Future Developments of Generative Models 97
6.2.1. Better Density Estimation 97
6.2.2. Generating High Quality Samples and Modeling the Distribution Well 98
6.2.3. Learning Disentangled/Causal Representations 98
6.2.4. Learning Interpretable Hierarchical Models 99
6.2.5. Large-scale Generative Modelling for Foundations in AI 99
References 101

List of tables

1 Experimental results on Poetry Generation task. The result of SeqGAN is from [Yu et al., 2017]. 44

2 Experimental results on PTB. Note that we evaluate the models in sentence-level. 45

3 Human evaluation results on dialog response generation. 46

1 Comparison of BLEU4 scores between our approach E-ARM and the base ARGM trained just with cross-entropy loss on six translation pairs of IWSLT14 datasets. We use "-" to denote that the training trick (either label smoothing or scheduled sampling) is not used while "✔" indicates we use it. "**5 B**" means that we use beam searching with 5 beams. 57

2 Translation performance of our E-ARM on WMT16 English→German, evaluated by BLEU4. We uniformly use 5 beams to apply beams searching. "**L.S.**" denotes Label Smoothing and "**S.S.**" denotes Scheduled Sampling. 59

3 Test set performance of different models on WikiText103. Evaluation is conducted using perplexity. 60

4 Ablation of different λ and the start epoch where we introduce the E-ARM into the training on WikiText103. Evaluation is conducted using perplexity. 60

5 Test set performance of different models on MNIST and CIFAR-10 in bits/dim (lower is better), training performance in brackets. 61

1 DDLS suffers less from mode dropping when modeling the 2d synthetic distribution in Figure 3. Table shows number of recovered modes, and fraction of "high quality" (less than four standard deviations from mode center) recovered modes. 72

2 DDLS has lower Earth Mover's Distance (EMD) to the true distribution for the 2d synthetic distribution in Figure 3, and matches the performance of DOT on the Swiss Roll distribution. 73

3 Inception and FID scores on CIFAR-10 and CelebA (grouped by corresponding baseline modles), showing a substantial quantitative advantage from DDLS,

compared to MH-GAN [Turner et al., 2019], DRS [Azadi et al., 2018] and DOT [Tanaka, 2019] using the same architecture. 75

4 Inception score for ImageNet, showing the quantitative advantage of DDLS. 76

1 Summary comparison of the characteristics of the recent related methods. 81

2 OOD verification results of image classification under different validation setups. All metrics are percentages and the best results are in bold. The backbone classifier in SUF [Lee et al., 2018b] and our DVN is ResNet34 [He et al., 2016], while ODIN [Liang et al., 2018] use more powerful wide ResNet 40 with width 4 [Zagoruyko and Komodakis, 2016]. 89

3 Comparison of AUROC (%) under different validation setups. The best results are in bold. We also compared the use of negative samples for training and input image pre-processing. 91

4 Test error rate of classification on CIFAR-10/100 using DenseNet as backbone. Our DVN does not re-train or modify the structure of the original trained classifier. 92

5 The performance of DVN w/o disentanglement of y from z with ResNet backbone, and using CIFAR-10/SVHN as in-distribution/OOD, respectively. 93

6 The performance of DVN w/o replace $p(z)$ with $q(z)$. We use ResNet backbone, and choose CIFAR-10/SVHN as in-distribution/OOD. 94

7 The performance of DVN use $q(z|x)$ and $q(z|x,y)$ encoder. We use ResNet backbone, and choose CIFAR-10/SVHN as in-distribution/OOD. 94

8 OOD verification results of image caption under different validation setups. We use CUB-200, LSUN and COCO as the OOD of Oxford-102, while using Oxford-102, LSUN and COCO as OOD of CUB-200. 95

List of figures

1 The training loss of the generator (a) and the discriminator (b) of MaliGAN on Discrete MNIST task; Samples generated by REINFORCE-like model (c) and by MaliGAN (d). 44

1 Samples of CIFAR-10 from Gated Pixel-CNN (w/E-ARM). 61

1 Progression of Inception Score with more Langevin dynamics sampling steps. 69

2 CIFAR-10 Langevin dynamics visualization 77

3 DDLS, generator alone, and generator + DOT, on 2d mixture of Gaussians distribution 78

4 DDLS, the generator alone, and generator + DOT, on the swiss roll dataset. 78

5 CIFAR-10 Langevin dynamics visualization, initialized at a sample from the generator (left column). The latent space Markov chain appears to mix quickly, as evidenced by the diverse samples generated by a short chain. Additionally, the visual quality of the samples improves over the course of sampling, providing evidence that DDLS improves sample quality. 78

6 Top-5 nearest neighbor images (right columns) of generated CIFAR-10 samples (left column). 78

1 A network trained on CIFAR-10 with ten-class softmax output will predict the resized 32x32x3 view of the AAAI logo (OOD sample w.r.t. CIFAR-10) as a plane with high confidence. 82

2 The architecture of our proposed Deep Verifier Network (DVN). We use ground-truth label y of training example x during training while using the trained model prediction y' of testing image during testing. 84

3 FPR (for OOD) and TPR (for ID) under different δ when using CIFAR-10 as the in-distribution dataset, and use Tiny-ImageNet(resize), LSUN and Gaussian/Uniform noise as OOD. CIFAR-10 only applicable to the TPR which use the dashed red

line and indicated by the right axis while the other OOD datasets use the left FPR axis. 90
4 Comparison with baseline MSP [Hendrycks and Gimpel, 2017] using DenseNet, with Tiny-ImageNet as in-distribution and LSUN as OOD. 90

Liste des sigles et des abréviations

KQ-Methode	Méthode des moindres carrés, de l'allemand *Methode der kleinsten Quadrate*
MCMC	Monte Carlo par chaînes de Markov, de l'anglais *Markov Chain Monte Carlo*
MSE	Erreur quadratique moyenne, de l'anglais *Mean Square Error*
NDR	Retract d'un voisinage, de l'anglais *Neighbourhood Deformation Retract*
OLS	Moindres carrés ordinaires, de l'anglais *Ordinary Least Square*
ZFC	Théorie des ensembles de Zermelo-Fraenkel avec l'axiome du choix

Chapter 1

Background and Introduction

In this chapter, I will first provide a brief introduction of several popular deep generative models and introduce some notations that will be used throughout this thesis. Furthermore, I will then discuss several contributions of this thesis made surrounding this topic.

1.1. Motivations and Overview

Deep learning aims at designing learning architectures and algorithms which are able to discover deep hierarchical representations of complex data. Supervised learning, unsupervised learning and semi-supervised learning are three important learning settings which have attracted significant attention for many decades.

In supervised learning, one is given a labelled dataset $D = \{x_i,y_i\}_{i=1}^N$, where the task is usually to predict y_i using x_i. We have made significant progress on supervised deep learning in the last few years. The task is clear: we need to design a neural network architecture to predict y given x, and optimize the trainable weights in the learning architecture with optimization algorithms such as stochastic gradient descent, using backpropagation to compute the gradients.

Although the recent success in supervised learning has already resulted in a large number of applications, it is not enough for achieving human-level artificial intelligence (AI). One of the most important goals for AI is to develop algorithms and techniques that endow computers with the capability of learning and understanding our world on their own, with minimal human intervention. This is why a lot of recent research efforts have been devoted to unsupervised learning. The main goal for unsupervised learning is to learn useful information from the dataset without any additional label or supervision from humans. In this setting, one is given an unlabelled dataset $D = \{X_i\}_{i=1}^N$. The main goal is usually to discover the internal properties of D, for example, the principal components, the low dimensional structure and good representations of D. Unsupervised learning is generally considered to be a significant part of human learning, as well as an important step toward artificial intelligence.

Unsupervised learning models are sometimes designed to be useful for downstream tasks, such as supervised learning and reinforcement learning. However, it is now widely believed that unsupervised learning itself is a standalone learning problem, even without any downstream tasks. A typical unsupervised learning model problems is to build a trainable probabilistic model $p_\theta(x)$ and then fit the model on D. Then the trained parameters θ contains most essential information of the data. For example, clustering can be done by fitting a Gaussian mixture model on training data and the model parameters can determine which cluster a data point should belong to.

An important class of unsupervised learning is generative modeling. The motivation for inventing generative models as a way of unsupervised learning is simple: since the performance of unsupervised learning algorithms without downstream tasks are sometimes hard to evaluate, a natural way to test these unsupervised learning methods is to see whether they are able to generate good and novel samples from the same distribution. For instance, if we want to know whether a student understands English, a good way is to test if they can write good articles in English.

Generative models have deep impacts on artificial intelligence in several ways and thus have a wide range of applications:

- Generative models made it easy for us to generate novel samples from a specific distribution or conditioned on a specific input. These novel samples can be of very high dimension and have rich internal structure. In machine learning these kind of learning problems are also called structured prediction.
- Given a specific dataset, one can devise a generative model endowed with proper inductive biases based on the insights of the structure of the data. After training on the dataset, the components of the generative model can capture the structure of the data generating distribution and can thus be used to complete downstream tasks. As an example, AIR [Eslami et al., 2016] is a deep generative model with object-centric inductive biases, so that after training it can identify both the objects and the background, in other words it can learn to understand the main elements of a scene. This understanding can be achieved without any human supervision or labelling. The learned modules of the generative model can be made useful to downstream tasks, such as object detection and image segmentation.

Generative modeling has a very long history and is deeply rooted in statistics and statistical learning. A first course in statistics in high school is likely to cover the basic techniques of computing the mean and variance for a given dataset. The motivation behind these techniques is the Gaussian assumption: if we assume the dataset is sampled from a Gaussian

distribution, then the sufficient statistics of the data are its mean and variance. Here the Gaussian assumption is the "inductive bias" we are using, and we are learning a generative model of the data by fitting the two parameters.

In this thesis, we are focusing on so-called deep generative models. The word "deep" has several implications:

- We are focusing on generative models that utilize deep multi-layer transformations, the transformations can take the forms of a deep neural network or deep graphical models.
- We are focusing on modelling data with rich internal structure and complex generating distributions, such as natural images and natural language text.

In this chapter, we will briefly review some of the most popular generative models, including restricted Boltzmann machines[Hinton, 2002a], variational auto-encoders[Kingma and Welling, 2013], auto-regressive models, generative adversarial networks [Goodfellow et al., 2014b] and energy-based models.

1.2. Restricted Boltzmann Machines

The Restricted Boltzmann Machine (RBMs) [Rumelhart and McClelland, 1987] is one of the earliest deep generative models. It is simple because it has only two layers of random variables, but the idea of the RBM inspired many successful generative models. Let $\{0,1\}^d$ is the set of binary vectors for any $d \in \mathbb{N}_+$. In this model we have two binary random vectors: the observed vector $\mathbf{x}$ taking value in $\{0,1\}^d$ and the hidden vector $\mathbf{h}$ taking value in $\{0,1\}^f$. In order to define the joint probability, the model first defines an energy function:

$$E(\mathbf{x},\mathbf{h},\mathbf{w}) = -\mathbf{x} \cdot \mathbf{W}\mathbf{h} - \mathbf{w}_1 \cdot \mathbf{x} - \mathbf{w}_2 \cdot \mathbf{h} \tag{1.2.1}$$

in which $\mathbf{w} = (\mathbf{W} \in \mathbb{R}^{d \times f}, \mathbf{w}_1 \in \mathbb{R}^d, \mathbf{w}_2 \in \mathbb{R}^f)$ is the set of trainable weights, and "$\cdot$" stands for dot product between vectors. It is further assumed that W is a symmetric matrix. Given this energy function, the model probability distribution is defined as:

$$p_{\mathbf{w}}(\mathbf{x},\mathbf{h}) = e^{-E(\mathbf{x},\mathbf{h},\mathbf{w})} / Z(\mathbf{w}) \tag{1.2.2}$$

where $Z(\mathbf{w}) = \sum_{\mathbf{x},\mathbf{h}} e^{-E(\mathbf{x},\mathbf{h},\mathbf{w})}$ is the normalization constant. The marginal probability of an observed vector x can be written as:

$$p_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{h} \in \{0,1\}^f} e^{-E(\mathbf{x},\mathbf{h},\mathbf{w})} / Z(\mathbf{w}). \tag{1.2.3}$$

Usually, given a dataset $D = \{x_i\}_{i=1}^N$, we want to train the parameters by maximizing the empirical log-likelihood $l(\mathbf{w}) = \sum_i \log p_{\mathbf{w}}(x_i)$ as an approximation for $L(\mathbf{w}) = \mathbb{E}_{x\sim p_d}[\log p_{\mathbf{w}}(x)]$. However, directly optimizing the objective is intractable. A common way of training RBM is through Contrastive Divergence (CD) algorithm [Hinton, 2002a], which provides a tractable biased estimator of the log-likelihood gradient. First we can write the gradient of the log-likelihood objective as:

$$\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}) &= \mathbb{E}_{x\sim p_d}[\nabla_{\mathbf{w}} p_{\mathbf{w}}(x)/p_{\mathbf{w}}(x)] \\
&= \mathbb{E}_{x\sim p_d}[\textstyle\sum_{h\in\{0,1\}^f} \nabla_{\mathbf{w}} p_{\mathbf{w}}(x,h)/p_{\mathbf{w}}(x)] \\
&= \mathbb{E}_{x\sim p_d, h\sim p_{\mathbf{w}}(h|x)}[\nabla_{\mathbf{w}} p_{\mathbf{w}}(x,h)/p_{\mathbf{w}}(x,h)] \\
&= \mathbb{E}_{x\sim p_d, h\sim p_{\mathbf{w}}(h|x)}[-\nabla_{\mathbf{w}} E(x,h;\mathbf{w}) - \nabla_{\mathbf{w}} \log Z(\mathbf{w})] \\
&= \mathbb{E}_{x,h\sim p_{\mathbf{w}}(h,x)}[\nabla_{\mathbf{w}} E(x,h;\mathbf{w})] - \mathbb{E}_{x\sim p_d, h\sim p_{\mathbf{w}}(h|x)}[\nabla_{\mathbf{w}} E(x,h;\mathbf{w})].
\end{aligned}$$

The intractability in the above gradient estimator is due to the hardness of sampling. The second term $\mathbb{E}_{x\sim p_d, h\sim p_{\mathbf{w}}(h|x)}[\nabla_{\mathbf{w}} E(x,h;\mathbf{w})]$ is tractable since the conditional distribution is much easier to sample. The CD algorithm replaces the sampling distribution $x,h \sim p_{\mathbf{w}}(h,x)$ with the resulting distribution after one step of Gibbs sampling. More precisely, one first samples an $x' \sim p_d$ and gets $h' \sim p_{\mathbf{w}}(h'|x')$, then samples $x \sim p_{\mathbf{w}}(x|h')$, finally getting $h \sim p_{\mathbf{w}}(h|x)$. We then use (x,h) to provide a biased sampling approximation of the first term $\mathbb{E}_{x,h}[\nabla_{\mathbf{w}} E(x,h;\mathbf{w})]$.

The binary RBMs is a simple learning machine which is inspired by statistical physics (Ising model) that could be generalized in multiple ways. For example, by incorporating Gaussian distributions RBMs have been extended to model continuous variables such as images [Hinton et al., 2006]. It could also be used as building blocks for much deeper learning machines such as deep Boltzmann machines [Salakhutdinov and Hinton, 2009a] and deep belief networks [Hinton et al., 2006].

1.3. Generative Adversarial Networks

Generative adversarial networks (GAN) [Goodfellow et al., 2014c] have demonstrated their potential on various tasks, such as image generation, image super-resolution, 3D object generation, and video prediction [Radford et al., 2015, Ledig et al., 2016, Sønderby et al., 2016, Nguyen et al., 2016, Wu et al., 2016a, Mathieu et al., 2015]. The objective is to train a parametrized function (the generator) which maps noise samples (e.g., uniform or Gaussian) to samples whose distribution is close to that of the data generating distribution. In order to train the generator, the GAN method introduces an additional parametrized

function, the discriminator, to measure some kind of discrepancy between the generator distribution p_g and the data generating distribution p_d. Specifically, the generator and the discriminator compete in a two-player minimax game, where the generator is trained to map samples from an simple prior distribution to data points. The goal of the discriminator is to distinguish between real and generated samples, while the goal of the generator is to "fool" the discriminator by producing samples that are close to real data points.

GANs have a large number of different formulations and variants. We will briefly review some of the most popular formulations here.

1.3.1. Original GANs

The basic scheme of the GAN training procedure is to train a discriminator which assigns higher probabilities to real data samples and lower probabilities to generated data samples, while simultaneously trying to improve a generator so as to move the generated samples towards the real data manifold using the gradient information provided by the discriminator. In a typical setting, the generator $G_\theta(\mathbf{z})$ (with parameter θ) and the discriminator $D_\phi(\mathbf{x})$ (with parameter ϕ) are represented by deep neural networks. The objective for the original GAN training is:

$$L(\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim p_d}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D_\phi(G_\theta(\mathbf{z})))] \tag{1.3.1}$$

where p_g is the distribution of the generator, and p_d the distribution of data. The GAN algorithm is listed in Alg 1.

We want to solve the Min-max problem $\min_\theta \max_\phi L(\theta,\phi)$. We have the following theorem, whose proof is included in [Goodfellow et al., 2014c].

Theorem 1.3.1. *If p_d and p_g have the same support, then for any fixed θ, write D^* to be the optimal discriminator that maximizes L. Then $D^* = \frac{p_g}{p_g+p_d}$. Thus $L(\theta,D^*) = JSD(p_g,p_d)$. Where $JSD(\cdot,\cdot)$ denotes the Jensen-Shannon divergence.*

1.3.2. f-GANs

A generalization for original GAN is the f-GAN. First, assume $f : \mathbb{R}_+ \to \mathbb{R}$ is a lower semi-continuous convex function and satisfies $f(1) = 0$. Its convex conjugate is f^* defined as:

$$f^*(x) = \sup_{t\in \mathrm{dom}(f)} xt - f(t).$$

The f-divergence is defined as $D_f(p||q) = \mathbb{E}_{q(x)}[f(p(x)/q(x))]$. There is a duality between f and f^*, in the sense that for any lower semi-continuous convex function, $f = f^{**}$. So we

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

• Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

• Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

• Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

• Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

• Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule.

have:

$$D_f(p||q) = \mathbb{E}_{q(x)}[\sup_{t \in \text{dom}(f^*)} tp(x)/q(x) - f^*(t)].$$

Let $T : X \to \mathbb{R}$ be an arbitrary function, we have the variational lower bound:

$$D_f(p||q) \geq \mathbb{E}_{q(x)}[T(x)p(x)/q(x) - f^*(T(x))] = \mathbb{E}_{p(x)}[T(x)] - \mathbb{E}_{q(x)}[f^*(T(x))].$$

For any fixed convex function f, we are optimizing the loss

$$L(\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim p_d}[T_\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[f^*(T_\phi(G_\theta(\mathbf{z})))]. \tag{1.3.2}$$

1.3.3. IPM and W-GANs

Besides original GANs, an important class of GAN models are defined by Integral Probability Metrics (IPMs). Write $\mathcal{F}$ a family of functions, P,Q are two probability distributions. Define the integral probability metrics as:

$$d_{\mathcal{F}}(P,Q) = \max_{f\in\mathcal{F}} d_f(P,Q) = \max_{f\in\mathcal{F}} |\mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{y\sim Q}[f(y)]| \tag{1.3.3}$$

This measure of discrepancy corresponds to a GAN formulation:

$$L(\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim p_d}[f_\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[f_\phi(G_\theta(\mathbf{z}))]. \tag{1.3.4}$$

A popular choice of $\mathcal{F}$ is the class of K-Lipschitz functions, where the GAN formulation corresponds to Wasserstein GANs. Some other choices are available, for example, mean and variance matching GANs and Fisher GANs. We have the following theorem:

Theorem 1.3.2. *Let $\mathcal{F}$ be the class of K-Lipschitz functions with positive constant $K > 0$. Consider the GAN formulation corresponding to $\min_\theta \max_\phi L(\theta,\phi)$. For an fixed θ, the optimal discriminator $D^* \in \mathcal{F}$ is $D^* = \arg\max_{f\in\mathcal{F}} d_f(p_g,p_d)$. We have $L(\theta,D^*) = \mathbb{E}_{\mathbf{x}\sim p_d}[D^*(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[D^*(G_\theta(\mathbf{z}))] = K \cdot D_W(p_g,p_d)$. $D_W(p_g,p_d)$ is the Wasserstein distance between the generator distribution and the data distribution.*

The proof can be found on [Arjovsky et al., 2017a].

The major challenge for training WGANs is how to impose the K-Lipschitz constraint. As the discriminator is parameterized as a neural network, it is difficult to find the accurate constraint in the weight space which corresponds exactly to the K-Lipschitz constraint in function space. Hence we use approximations instead. K-Lipschitz functions can be realized approximately using weight clipping [Arjovsky et al., 2017a], gradient penalty [Gulrajani et al., 2017] and spectral normalization [Miyato et al., 2018a].

Weight clipping directly clips the weights to $[-c,c]$ with a constant $c > 0$, thus bounding the L^∞ norm of the weights in the discriminator. The resulting weights forms a compact set and the value $\max_{x,y} \frac{|f(x)-f(y)|}{|x-y|}$ has a maximum K on this set. Because the weight bound is a sufficient but not necessary condition of the K-Lipschitz property, the discriminator has to be K-Lipschitz during optimization, but it can be much more restrictive than the Lipschitz condition.

The Gradient Penalty [Gulrajani et al., 2017] method randomly samples some points from the space and imposes the K-Lipschitz condition as a constraint on the gradients at these points. Note that this is not a sufficient condition for the discriminator to be K-Lipschitz.

It is obvious that the composition of 1-Lipschitz functions is 1-Lipschitz. So if we can ensure that every transformation layer of a neural network is 1-Lipschitz, then the entire neural network function is 1-Lipstchiz. The Spectral Normalization method [Miyato et al., 2018a] tries to ensure each layer of the discriminator is 1-Lipschitz. To do so it only has to make each linear layer in the discriminator 1-Lipschitz, by dividing the weights by the absolute value of the spectral value. Similar to weight clipping, this results in a sufficient condition for the discriminator to be K-Lipschitz, it can also be more restrictive than the K-Lipschitz condition, but it is much more flexible than weight clipping.

1.4. Autoregressive Models

Auto-regressive generative models[1] is a large class of generative models which rely on decomposing any joint distribution $p(\mathbf{x})$ into a product of conditional distributions using the product rule of probability by ordering those random variables within the joint distribution and characterizing each random variable dependent on all variables prior to it. Formally, we use $\mathbf{x}_{<k}$ denote the vector variable covering all random variables before the timestep k and use x_k denote the random variable at timestep k. Then we have the auto-regressive decomposition of distribution p:

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathbf{x}_{<k}). \tag{1.4.1}$$

In general, modeling distributions with the auto-regressive decomposition has achieved remarkable accomplishments in numerous areas such as NLP and image generation [Vaswani et al., 2017, Radford et al., 2019, van den Oord et al., 2016b,c, Salimans et al., 2017] thanks to its ability to avoid the challenge of modeling joint high-dimensional distributions directly without necessarily having to face the curse of dimensionality, as originaly highlighted by [Bengio and Bengio, 2000, Bengio et al., 2000].

A basic example of auto-regressive models is the family of Recurrent Neural Networks (RNNs). An RNN is a neural architecture to process variable-length sequential data and it can be naturally used as a learning machine to model distributions with the auto-regressive distribution decomposition, where the same parameters are used to predict $p(x_k|\mathbf{x}_{<k})$ for all k. Below we briefly discuss RNNs.

Consider an RNN processing an input sequence $x = (x_1,x_2 \cdots x_T)$. At each timestep t, the RNN computes a hidden state h_t using the hidden state in the last timestep h_{t-1} and input x_t. The hidden states are computed step-by-step in an inductive manner: $h_t = f_\theta(h_{t-1},x_t)$, where f_θ is the transition function of the RNN and θ is the trainable parameters of the RNN. The state h_t is also called the "belief state", in the sense that it represents all the memory of the sequence before timestep $t+1$:

$$p_\theta(\mathbf{x}) = \prod_{k=1}^{K} p_\theta(x_k|\mathbf{x}_{<k}), \qquad p_\theta(x_k|\mathbf{x}_{<k}) = g_\phi(x_k,h_{k-1})$$

where p_θ is the model distributions and g_ϕ maps the current belief state to the predicted distribution of the next token. Different design choices in f_θ can lead to different RNN

[1] In this chapter, the term "auto-regressive model" is generally used to denote the auto-regressive generative model for convenience, though strictly speaking, they are different.

variants, such as LSTMs[Hochreiter and Schmidhuber, 1997] or GRUs[Cho et al., 2014]. Below we briefly explain LSTMs.

When trained on long sequences, a common problem of RNNs is gradient vanishing [Bengio et al., 1994]. When doing backpropagation through many timesteps, the gradient can shrink exponentially with the number of timesteps and the model can fail to capture long term time dependencies in this case. The LSTM architecture [Hochreiter and Schmidhuber, 1997] was designed to combat the gradient vanishing problem. The core idea behind the LSTM is to control updates in memory states through sigmoid gates. Besides hidden states h_t as common RNNs, LSTMs represent memory states using cell states, denoted as c_t. There are three kinds of gates in LSTMs, the input gate i_t, the forget gate f_t and the output gate o_t. The update rules are:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{1.4.2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{1.4.3}$$

$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot \tilde{c}_{t-1}, h_t = o_t \odot \tanh(c_t) \tag{1.4.4}$$

where $\odot$ stands for elementwise multiplication. The gating mechanism of LSTMs controls the update of the cell states and makes the gradient flow smoothly when $f_t \approx 1$ and the gradient vanishing problem becomes much less severe.

1.4.1. Exposure Bias Problem

When training autoregressive generative models, the exposure bias problem [Bengio et al., 2015a, Ranzato et al., 2016a] is an important issue, which greatly affects the model's deployment performance. During the training stage, the autoregressive model is always conditioned on ground truth token sequences. During the generation stage, however, the model has to rely on its own previously generated tokens to predict the next token, when the model. If an incorrect token is selected, this error can be amplified in the following steps because the next prediction will be made using the incorrect input (one unlike those in the training set). Namely, the probability distribution of the generated sequence can diverge from the real data distribution because errors can accumulate with the number of timesteps.

The diverge between training and generating sequence distributions is called the exposure bias problem. It is the main challenge when training autoregressive models to generate long sequences.

1.5. Variational Auto-encoders

Like RBMs and GANs, the variational auto-encoder is a latent variable generative model. Different from RBMs, the correlation between hidden and observed variables is not given by a predefined simple energy function, instead it can be highly non-linear and is parameterized with deep neural networks. Different from GANs, VAEs are trained to approximately but more directly maximize the likelihood of training data.

Suppose we want to train a latent variable model $p_\theta(x) = \mathbb{E}_{z\sim p(z)} p_\theta(x|z)$ on dataset $D = \{x_i\}_{i=1}^N$, where $p(z)$ is a simple prior distribution such as isotropic Gaussian. In practice, the distributions $p_\theta(x|z)$ are also assumed to be simple distributions such as isotropic Gaussians. The main complexity of the model lies in the mapping $z \to p_\theta(x|z)$, which can be highly nonlinear and is parameterized with a deep neural network called the decoder in VAEs.

Usually one want to optimize the log-likelihood of data $L(\theta) = \sum_i \log p_\theta(x_i)$, however the log-likelihood $\log p_\theta(x)$ is intractable. The only tractable likelihood quantities are $p_\theta(x,z) = p(z)p_\theta(x|z)$, $p(z)$ and $p_\theta(x|z)$. Thus one wants to represent $\log p_\theta(x)$ in terms of these quantities. Let $p_\theta(z|x)$ be the conditional distribution defined by $p_\theta(x,z)$, and $q_\phi(z|x)$ be a trained approximation of $p_\theta(z|x)$, which is usually parameterized by another neural network called the encoder.

Variational auto-encoders [Kingma and Welling, 2013] provide a way for training the generative model by using a variational lower bound:

$$\log p_\theta(x_i) \geq \mathbb{E}_{z\sim q(\cdot|x_i)}[\log p_\theta(x,z) - \log q(z|x_i)] = L(\theta,\phi;x_i) \tag{1.5.1}$$

where $q(z|x) = q_\phi(z|x)$ is a trained inference model. With VAEs we train both q and p to maximize the lower bound. The training objective of a VAE is:

$$L_{\text{VAE}}(\theta,\phi;D) = -\sum_i^N L(\theta,\phi;x_i). \tag{1.5.2}$$

The error term for the lower bound is:

$$\log p_\theta(x_i) - L(\theta,\phi;x_i) = \text{KL}(q(\cdot|x_i)||p_\theta(\cdot|x_i)). \tag{1.5.3}$$

Thus, minimizing L_{VAE} is equivalent to maximizing $\mathbb{E}_{x\sim p_d}[\log p_\theta(x)]$ and at the same time minimizing $\mathbb{E}_{x\sim p_d}[\text{KL}(q(\cdot|x)||p_\theta(\cdot|x))]$. The lower bound is tight when the model q_ϕ is expressive enough to represent real $p_\theta(z|x)$. The main optimization trick of L_{VAE} is to backpropagate the objective into q. Consider the more general problem of optimizing $\mathbb{E}_{z\sim q_\phi(z|x)}[f(z,x)]$, where f is a smooth function and $q_\phi(z|x) = \mathcal{N}(z;\mu_\phi(x),\sigma_\phi^2(x))$. The

reparameterization trick relies on the following transform:

$$\mathbb{E}_{z\sim q_\phi(z|x)}[f(z,x)] = \mathbb{E}_{\epsilon\sim N(0,1)}[f(\epsilon\cdot\sigma_\phi(x)+\mu_\phi(x),x)].$$

We can backprop through f into μ_ϕ and σ_ϕ. After training, the resulting generative model $p_\theta(x) = \mathbb{E}_z[p_\theta(x|z)]$ is an approximation to real data distribution. One can sample from $p_\theta(x)$ by first drawing samples from $z \sim p(z)$, and then mapping z to pixel space $x \sim p_\theta(x|z)$.

1.6. Deep Energy-based Models

Energy-based models [LeCun et al., 2006] can express any probability density $p(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^D$ as

$$p_\theta(\mathbf{x}) = \frac{\exp(-\mathbf{E}_\theta(\mathbf{x}))}{\mathbf{Z}_\theta}, \tag{1.6.1}$$

where $E_\theta : \mathbb{R}^D \to \mathbb{R}$ denotes an energy function which aims to map a D-dimensional datapoint to a scalar, and $\mathbf{Z}(\theta) = \sum_{\mathbf{x}} \exp(-\mathbf{E}_\theta(\mathbf{x}))$ denotes the normalizing constant, also known as the partition function. Any function can be used as an energy function to represent an EBM as long as it can generate a single scalar given some input $\mathbf{x}$ and the normalization constant $\mathbf{Z}(\theta)$ exists (is not infinite). EBMs can be viewed as a mathematical model for associative memory, observed originally by Hopfield [Hopfield, 1982]. After the model is trained to fit a dataset, the model should have low energies on data points that satisfy the same statistical structure in the training data and captured by the model, and high energies elsewhere.

Some energy-based models use shallow predefined energy functions, such as RBMs. In this section we are talking about deep energy-based models. Namely the energy functions are not predefined, they can be highly nonlinear and very complex. In other words, they are parameterized by deep neural networks.

Several algorithms have been devised to optimize EBMs [Hinton, 2002b, Kim and Bengio, 2016a, Grathwohl et al., 2020, Xiao et al., 2021], in particular via gradient-based maximum likelihood. Specifically, the gradient of the log-likelihood with respect to θ can be expressed as

$$\mathbb{E}_{p_d(\mathbf{x})}\left[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x})\right] = \mathbb{E}_{p_\theta(\mathbf{x})}\left[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(\mathbf{x})\right] - \mathbb{E}_{p_d(\mathbf{x})}\left[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(\mathbf{x})\right]. \tag{1.6.2}$$

We call the first term in the right part of Eq. 1.6.2 the negative phase term while call the second term the positive phase term. For the positive phase, we can very easily derive a gradient estimator by sampling from p_d, which is the distribution where the training samples come from. For the negative phase, the story is more involved because p_θ is implicitly defined

and there is no easy way to sample from this distribution. People usually adopt MCMC methods [Hinton, 2002b, Welling and Teh, 2011a] to sample data from $p_\theta(\mathbf{x})$ for estimating the expectation $\mathbb{E}_{p_\theta(\mathbf{x})}$.

1.7. Main Contributions

The main contributions of this thesis are about discovering new connections between different classes of generative models. First, we discuss several contributions and progress made by the author and the collaborators on generative models, including GANs, auto-regressive models and energy-based models.

A core problem of training auto-regressive models is the exposure bias problem which we discussed above. In chapter 2 and 3, we bridge the gap between auto-regressive models, GANs and energy-based models to solve the exposure bias problem. In this thesis we propose two principled ways to address the exposure bias problem. In Chapter 2, we show that exposure bias can be solved with GANs, since GAN discriminator can evaluate the quality of very long sequences as a whole. However, training GAN models on discrete data is very challenging due to the high variances of gradient estimators. In order to address this challenge, we discuss how to stabilize training of discrete GANs with importance sampling. This chapter can be viewed as a contribution of bridging the gap between auto-regressive models and GANs.

In Chapter 3, we also discuss another way of solving the exposure bias problem in auto-regressive models with the help of energy-based models. In order to train energy-based models, one has to consider the contributions of both the negative phase and the positive phase to the gradient estimator. In the positive phase, the models are conditioned on the training data in the same way as teacher forcing. While in the negative phase, the models are conditioned on self-generated data. In this way, the models can learn how to deal with self-generated data during training. This chapter can be viewed as a contribution of bridging the gap between auto-regressive models and energy-based models.

In Chapter 4, we discuss a new theoretical connection between GANs and Energy-based models which yield a novel method to sample from trained GAN models. We start from an implicit energy-based model defined jointly by a GAN generator and discriminator. We find that the implicit energy-based model takes on a simpler, tractable form when it is written as an energy-based model over the generator's latent space. In this way, we propose a theoretically grounded way of generating high quality samples from GANs through what we call Discriminator Driven Latent Sampling (DDLS). DDLS leverages the information contained in the discriminator to re-weight and correct the biases and errors in the generator.

This chapter can be viewed as a contribution of bridging the gap between energy-based models and GANs.

We also discuss an application of generative models. Specifically, in Chapter 6, we consider the problem of verifying the predictions of deep discriminative models with deep generative models. This verification problem is very important for safety-critical applications of deep learning. We propose to perform such verification of deep discriminative models by using deep generative models that try to generate the input conditioned on the label selected by the discriminative model. We call this concept "deep verifier". The high-level idea is simple: we train an inverse verification model $p(x|y)$ on the training data pairs (x,y). Intuitively speaking, for an input-output pair (x,y) with y picked by the predictive model, we verify whether the input x is consistent with y, by estimating if $p(x|y)$ is larger than a threshold. We design a density estimator of $p(x|y)$ using modified conditional VAEs. To ensure that the class code y is not ignored as a conditioning variable, we impose a disentanglement constraint based on minimizing mutual information between latent variable z and the label y. Although many different kinds of density estimators can be used in theory, we argue that the design of our model is robust to OOD samples and adversarial attacks, due to the use of latent variables with explicit and accurate density estimation.

These contributions have been published in the following papers:

- R Devon Hjelm, Athul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary seeking generative adversarial networks. ICLR 2018 [Hjelm et al., 2017]
- Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983, 2017. [Che et al., 2017]
- Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, [Che et al., 2020b]
- Tong Che, Xiaofeng Liu, Site Li, Yubin Ge, Ruixiang Zhang, Caiming Xiong, and Yoshua Bengio. Deep verifier networks: Verification of deep discriminative models with deep generative models, AAAI 2021 [Che et al., 2021]
- Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In International Conference on Learning Representations, 2018. [Jastrzebski et al., 2018]

The following chapters explain each of these contributions and put them in context.

Chapter 2

Stabilizing Training of Discrete GANs with Importance Sampling

2.1. Preface

2.1.1. My Role and Contributions

This paper is an unpublished work, its preprint version was announced on Arxiv[Che et al., 2017], however a concurrent work [Hjelm et al., 2017] was published at ICLR 18. Thus we decide to not to publish this paper and let the paper remain on Arxiv.

I proposed this idea independently with the first author of [Hjelm et al., 2017]. These two projects were completed mostly independently with each other in the early stage. There were some communications and collaborations between the two teams in the late stage of the projects. My contributions to [Che et al., 2017] include proposing the idea, formulating the mathematical details, designing the experiments, and contributing about 20% of the code for the experiments.

2.1.2. Motivation and Context

Generative adversarial networks play a pivotal role in many natural language processing tasks such as language modeling, machine translation, and dialogue generation. However, the generated sentences are often unsatisfactory [Sordoni et al., 2015, Bowman et al., 2015, Serban et al., 2017, Wiseman and Rush, 2016]. For example, they often lack of consistency in long-term semantics [Bowman et al., 2015, Zhang et al., 2016a].

This is largely attributed to the defect of teacher forcing [Williams and Zipser, 1989], which trains auto-regressive models to maximize the conditional probabilities of next tokens based on the ground-truth histories. It prohibits the trained model to take advantage of learning in the the context of its previous generated words to make the next prediction, the so-called *exposure bias* problem [Ranzato et al., 2016b]. Hence, it is difficult to approach

the true underlying distribution [Ranzato et al., 2016b, Bengio et al., 2015b]. Another limitation is that teacher forcing is inapplicable to those auto-regressive models with latent random variables, which have performed better than vanilla auto-regressive (deterministic state) recurrent neural networks on multiple tasks [Serban et al., 2017, Zhang et al., 2016a, Miao et al., 2016].

An attractive solution is using generative adversarial networks (GAN) [Goodfellow et al., 2014a]. The exposure bias problem could be prevented if the generative model was able to visit its own predictions during training and had an overall view on the generated sequences. This can be achieved with an additional discriminator trained to separate real versus generated sequences. The generative model is then able to exploit signals from the discriminator to improve itself. Since the discriminator is trained on the entire sequence, it can provide the training signal to avoid the exposure bias problem.

However, it is nontrivial to apply GANs to discrete data. It is difficult to optimize the generator using the signal provided by the discriminator. In fact, it is usually very hard to push the generated distribution to the real data distribution, if not impossible, by moving the generated sequence (e.g., a faulty sentence) towards a "true" one (e.g., a correct sentence) in a high-dimensional discrete state space. As standard back-propagation fails in discrete settings, the generator can be optimized using the discriminator's output as a reward via reinforcement learning. Unfortunately, even with careful pre-training, the policy could hardly get positive and stable reward signals from the discriminator.

2.1.3. Main Contribution

We aim at borrowing the benefit from GAN to avoid exposure bias meanwhile improving the GAN's training stability. We achieve this by proposing **Ma**ximum-**Li**kelihood Augmented Discrete **G**enerative **A**dversarial **N**etworks (**MaliGAN**). At the core of this model is the novel GAN training objective inspired by [Norouzi et al., 2016]. We use importance sampling and several variance reduction techniques in order to successfully optimize this objective. The procedure was discovered independently from us by [Hjelm et al., 2017] in the context of image generation.

The new target brings several attractive properties in the proposed MaliGAN. First, it is theoretically consistent and easier to optimize than vanilla sequential GANs (Section 2.3.2). Second, it allows the model not only to maximize the likelihood of good behaviors, but also to minimize the likelihood of bad behaviors, with the help of a GAN discriminator. Equipped with these strengths, the model focuses more on improving itself by gaining beneficial knowledge that is not yet well acquired, and excluding the most probable and harmful

behaviors. Combined with several proposed variance reduction techniques, the proposed MaliGAN successfully and stably models discrete data sequences (Section 2.4).

2.1.4. Follow-up works and Impacts

This project, together with its concurrent work [Hjelm et al., 2017] have attracted significant amount of attention from the research community. Utilizing GANs to help training discrete sequential models has been an active area of research, due to the fact that it could help fighting the exposure bias problem. Below we briefly review several lines of follow-up works.

To overcome the above shortcomings of MLE training, many new sequential GAN formulations for discrete outputs have been proposed in the literature [Guo et al., 2018, Lamprier et al., 2022]. To further combat the instability and high variance of training these models due to the non-stationarity of their reward distribution, recently some advances on smoother sampling techniques and the use of control variates [Scialom et al., 2020a] was proposed. GAN approaches usually under-performed MLE training in most real-world tasks [Caccia et al., 2020], suffering from mode dropping problems [Che et al., 2016] that often sacrifice diversity for quality. Some recent works based on cooperative decoding [Scialom et al., 2021, 2020b] try to use the discriminator information not only during training but also during test time, by relying on the discriminator not only as reward, but also for sampling.

It is worth mentioning that [Lamprier et al., 2022] is a direct follow-up of our work. They extend our formulation and achieved SOTA results on language GAN models. This shows the power of our proposed framework.

Another line of followup works focus on training language GANs purely from scratch [de Masson d'Autume et al., 2019], without the MLE pretraining. This can be a challenging task because discrete GANs can drop modes very easily. Different from our work where we use only the sequence level discriminator signal, these works rely on the discriminators providing information (dense reward) at every time-step, to stabilize the training.

In summary, this project has made significant impacts to the research community, evidenced by its number of citations (218) and direct followup works.

2.2. Problem Formulation And Analysis

In this work, we aim to fit discrete sequential data under the GAN setting [Goodfellow et al., 2014a]. GANs defines a framework for training generative models by posing it as a minimax game against a discriminative model. The goal of the generator G is to match its

distribution p_g to the real data distribution p_d. To achieve this, the generator transforms noise z sampled from $p(z)$ to a data sample $G(z)$. Following this, the discriminator D is trained to distinguish between the samples coming from p_d and p_g, and can be used to provide a training signal to the generator.

When applying the GAN framework to discrete data, the discontinuity prohibits the update of the generator parameters via standard back-propagation. To tackle this, one way is to employ a typical reinforcement learning (RL) strategy that directly uses the GAN discriminator's output, D or $\log D$ as a reward. In practice, the problem is usually solved by REINFORCE-like algorithms [Williams, 1992], perhaps with some variance reduction techniques.

In general, we train a generator $G(\mathbf{x})$ together with a discriminator $D(\mathbf{x})$. In its original form, the discriminator is trained to distinguish between the generating distribution p_θ and the real data distribution p_d. The generator is then trained to maximize $\mathbb{E}_{\mathbf{x}\sim p_\theta}[\log D(\mathbf{x})]$.[1]

To better analyze the loss, we slightly modify it by putting on a maximum-entropy regularizer $\tau\mathbb{H}(p_\theta)$ to encourage diversity. Now, define the normalized probability distribution $q'(\mathbf{x}) = \frac{1}{Z(D)}D(\mathbf{x})^{1/\tau}$. We can safely assume that $D(\mathbf{x})$ is of compact support, so $Z(D)$ is finite. We obtain a regularized loss$\mathcal{L}'_{GAN}(\theta) = -\mathbb{E}_{\mathbf{x}\sim p_\theta}[\log D(\mathbf{x})] - \tau\mathbb{H}(p_\theta) = \tau\mathrm{KL}(p_\theta||q') + c(D)$, where *c(D)* is a constant depending only on D. Hence, optimizing the traditional GAN is almost equivalent to optimizing the KL-divergence $\mathrm{KL}(p_\theta||q')$. Yet using this regularized objective to train discrete GANs with REINFORCE, there are still two major problems. The first problem, called the moving target problem, is that the target distribution q' always moves with G during training. It makes the training procedure unstable and harder to converge to a Nash equilibrium. The second problem is that the training signal is very weak when the generator is not able to produce plausible sequences. Having some samples $\mathbf{x}_i \sim p_\theta$, we want to change θ a bit in order to adjust the likelihood of samples $\mathbf{x}_i$ to improve the generator. However, since initially p has a poor policy, it can hardly generate realistic sequences to get positive rewards, and thus the training progress is hampered. Though the dedicated pre-training and variance reduction mechanisms help [Yu et al., 2017], the RL algorithms applied to the moving target distribution still seems very unstable and unscalable.

To address these two problems, we propose an alternative way to utilize the information of the discriminator. The discriminator's output is used to construct a novel objective as a proxy for the log-likelihood objective (Section 2.3.1). Different from the original objective, the

[1]One can also minimize $\mathbb{E}_{\mathbf{x}\sim p_\theta}[\log(1 - D(\mathbf{x}))]$), but in practice we optimize the $\log D$ version as it provides more stable gradients [Goodfellow et al., 2014a].

target distribution in our objective is the data generating distribution and thus is fixed. This addresses the first problem. We then employ importance sampling to make the objective trainable. The second problem is addressed by using minibatch renormalization, which reinforces each training sample not based on the absolute value of D, but based on the relative discriminator output in a mini-batch. Compared with those reinforcement learning approaches that directly adopt D or $\log D$ as reward signals, the novel training objective has much less variance. The analysis and discussions will be presented in Section 2.3.2.

2.3. Maximum-Likelihood Augmented Discrete Generative Adversarial Networks

2.3.1. Basic Model of MaliGAN

We propose Maximum-Likelihood Augmented Discrete Generative Adversarial Networks (MaliGAN) to generate discrete data. With MaliGAN, we train a discriminator $D(\mathbf{x})$ as standard way in vanilla GAN. The difference comes from a novel objective for the generator to optimize.

To derive the objective, we keep a fixed copy $p'(\mathbf{x})$ of the current generator p_θ. From the basic property of GANs, we know that an optimal $D^* = \frac{p_d}{p_d+p'}$ [Goodfellow et al., 2014a]. In this case, we have $p_d = \frac{D^*}{1-D^*}p'$. Motivated by this observation, we set the target distribution q for maximum likelihood training to be $q = \frac{D(\mathbf{x})}{1-D(\mathbf{x})}p'$. Let $r_D(\mathbf{x}) = \frac{D(\mathbf{x})}{1-D(\mathbf{x})}$, we define the augmented target distribution as:

$$q(\mathbf{x}) = \frac{1}{Z}\frac{D(\mathbf{x})}{1-D(\mathbf{x})}p'(\mathbf{x}) = \frac{r_D(\mathbf{x})}{Z}p'(\mathbf{x})$$

where Z is the normalization constant. Note that q is a fixed probability distribution with respect to θ, the objective to optimize is $L_G(\theta) = \mathrm{KL}(q(\mathbf{x})||p_\theta(\mathbf{x}))$. This objective has an attractive property that q is always approximately the data generating distribution p_d when D is sufficiently trained. Hence, q can be viewed as a "fixed" distribution during the entire training process. By defining the gradient as $\nabla L_G = \mathbb{E}_q[\nabla_\theta \log p_\theta(\mathbf{x})]$, we have the following importance sampling formula:

$$\nabla L_G = \mathbb{E}_{p'}[\frac{q(\mathbf{x})}{p'(\mathbf{x})}\nabla_\theta \log p_\theta(\mathbf{x})] = \frac{1}{Z}\mathbb{E}_{p_\theta}[r_D(\mathbf{x})\nabla_\theta \log p_\theta(\mathbf{x})]$$

This importance sampling procedure was discovered independently from us by [Hjelm et al., 2017]. We propose to optimize the generator using the following novel gradient estimator:

$$\nabla L_G(\theta) \approx \sum_{i=1}^{m} \left(\frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} - b/m\right) \nabla \log p_\theta(\mathbf{x}_i) = E(\{\mathbf{x}_i\}_1^m) \tag{2.3.1}$$

where b is a baseline adopted widely in RL approaches, which helps reduce variances by decreasing the probabilities of generating bad samples. In practice, we let b increase very slowly from 0 to 1, since the coefficients $\sum_i \frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} = 1$. Combined with the objective of the discriminator in an ordinary GAN, the proposed MaliGAN algorithm is shown in Algorithm 2. This algorithm provides a new way of passing the information of the discriminator to the generator. Generally speaking, it applies to arbitrary architectures of the generator with or without latent variables.

2.3.2. Analysis

We analyze the proposed objective in Eq. 2.3.1 from both theoretical and practical perspectives. In the following theorem, we show that the novel objective optimizes the KL divergence $\mathrm{KL}(q(\mathbf{x})||p_\theta(\mathbf{x}))$ when D is optimal. The proof for the theorem is given in supplementary material.

Theorem 2.3.1. *We have the following two theoretical guarantees for the objective in Eq. 2.3.1:*

(i) *If discriminator $D(\mathbf{x})$ is optimal, we have:*

$$\mathbb{E}_{p_d}[\log p_\theta(\mathbf{x})] = \frac{1}{Z}\mathbb{E}_{p'}[r_D(\mathbf{x}) \log p_\theta(\mathbf{x})], \text{ where } Z = \mathbb{E}_{p'}[r_D(\mathbf{x})] = 1$$

(ii) *If $D(x)$ is optimal, we also have that the estimator in Eq. 2.3.1 is a consistent estimator of $\nabla L_G(\theta)$.*

Proof. When we are not doing optimization, note that $p_\theta = p'$.

For (i), see the arguments in Section 1.3.1. For (ii), we can see that

$$\begin{aligned} \lim_{m\to\infty} E(\{\mathbf{x}_i\}_1^m) = & \lim_{m\to\infty} \sum_{i=1}^m \frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} \nabla \log p_\theta(\mathbf{x}_i) - \sum_{i=1}^m b/m \nabla \log p_\theta(\mathbf{x}_i) \\ = & \quad T_1 - T_2 \end{aligned}$$

Where $T_1 = \lim_{m\to\infty} \sum_{i=1}^{m} \frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} \nabla \log p_\theta(\mathbf{x}_i)$, $T_2 = \lim_{m\to\infty} \sum_{i=1}^{m} b/m \nabla \log p_\theta(\mathbf{x}_i)$. From central limit theorem,

$$\begin{aligned} \lim_{m\to\infty} \sum_{i=1}^{m} b/m \nabla \log p_\theta(\mathbf{x}_i) = &\quad b\mathbb{E}_{x\sim p_\theta}[\nabla \log p_\theta(x)] \\ = &\quad b \int_x \nabla p_\theta(x) dx = 0 \end{aligned}$$

and

$$\lim_{m\to\infty} \sum_{i=1}^{m} 1/m r_D(\mathbf{x}_i) = \quad \mathbb{E}_{x\sim p_\theta}[r_D(x)] = 1$$

we also have:

$$\lim_{m\to\infty} \sum_{i=1}^{m} 1/m r_D(\mathbf{x}_i) \nabla \log p_\theta(\mathbf{x}_i) = \mathbb{E}_{p_\theta}[r_D(\mathbf{x}) \nabla \log p_\theta(\mathbf{x})] = \mathbb{E}_{p_d}[\nabla \log p_\theta(\mathbf{x})]$$

$$\begin{aligned} T_1 = &\quad \lim_{m\to\infty} \sum_{i=1}^{m} \frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} \nabla \log p_\theta(\mathbf{x}_i) \\ = &\quad \lim_{m\to\infty} \frac{m}{\sum_j r_D(x_j)} \sum_{i=1}^{m} 1/m r_D(x_i) \nabla \log p_\theta(\mathbf{x}_i) \\ = &\quad \lim_{m\to\infty} \frac{m}{\sum_j r_D(x_j)} \cdot \lim_{m\to\infty} \sum_{i=1}^{m} 1/m r_D(x_i) \nabla \log p_\theta(\mathbf{x}_i) \\ = &\quad 1 \cdot \mathbb{E}_{p_d}[\nabla \log p_\theta(\mathbf{x})] \end{aligned}$$

So we have $\lim_{m\to\infty} E(\{\mathbf{x}_i\}_1^m) = \mathbb{E}_{p_d}[\nabla \log p_\theta(\mathbf{x})]$.

□

In addition to its attractiveness in theory, we now demonstrate why the gradient estimator in Eq. 2.3.1 of $\nabla L_G(\theta)$ practically produces better training signal for the generator than the original GAN objective. Similar discussions can be found in [Norouzi et al., 2016, Bornschein and Bengio, 2015].

In the original GAN model (optimizing D or $\log D$) applied to sequence generation, REINFORCE based gradient estimators would be drastically inefficient when almost all generated sequences had a very small D value. Unfortunately, this is very common in the early stage of GAN training.

In the MaliGAN objective, however, the partition function Z is estimated using the samples from the mini-batch, which helps dealing with the above dilemma. In our approach, the probability of each sequence is adjusted not according to the absolute value of the discriminator output, but its relative quality in that mini-batch. This ensures that the

model can always learn something as long as there exist some generations better than others in that mini-batch.

From a theoretical point of view, this normalization procedure also helps. Although at the first glance, when D is optimal, one can prove that $Z = 1$, so estimating Z seems to only introduce additional variance to the model. However, using this estimator in fact reduces the variance due to the following reason: $r_D(\mathbf{x})$ is actually a function with singularity when $\mathbf{x}$ is in a region Ω in the data space on which $D(\mathbf{x}) \approx 1$. On such a region Ω, $r_D \gg 0$ and $p'(\Omega) \approx 0$, making the ratio r_D blow up. In our target $\frac{1}{Z(\theta')}\mathbb{E}_{p'}[r_D(\mathbf{x}) \log p_\theta(\mathbf{x})]$, since it is almost impossible to get samples from Ω with p' in a reasonable size mini-batch, the actual distribution we are sampling from is a "regularized" distribution $p_{\setminus\Omega}$ where $p_{\setminus\Omega}(\Omega) = 0$ and $p_{\setminus\Omega} \approx p'$. So when doing importance sampling to estimate our training objective $\nabla L_G = \mathbb{E}_{p_d}[\nabla_\theta \log p_\theta(\mathbf{x})]$ with small mini-batches, we are actually doing normalized-weights importance sampling based on $p_{\setminus\Omega}$: $\nabla L_G \approx \mathbb{E}_{p_{\setminus\Omega}}[r_D(\mathbf{x})\nabla_\theta \log p_\theta(\mathbf{x})]/\mathbb{E}_{p_{\setminus\Omega}}[r_D(\mathbf{x})]$. Since the Monte Carlo estimator has much more variance to estimate $\mathbb{E}_{p'}[r_D(\mathbf{x})\nabla_\theta \log p_\theta(\mathbf{x})]$ than $\mathbb{E}_{p_{\setminus\Omega}}[r_D(\mathbf{x})\nabla_\theta \log p_\theta(\mathbf{x})]$, in practical mini-batch training settings, we can view that we are doing importance sampling with the distribution $p_{\setminus\Omega}$, and this objective has much less variance compared to importance sampling with p' on r_D which has an infinite singularity. This is why estimating $Z = \mathbb{E}_{p_{\setminus\Omega}}[r_D(\mathbf{x})]$ is important in order to reduce the variance in the mini-batch training setting.

2.3.3. Variance Reduction Techniques in MaliGAN

2.3.3.1. **Monte Carlo Tree Search**. Instead of using the same weight for all time steps in one sample, we use the following formula:

$$\mathbb{E}_{p_\theta}[r_D(\mathbf{x})\nabla p(\mathbf{x})] = \mathbb{E}_{p_\theta}[\sum_{t=1}^{L} Q(a_t,\mathbf{s}_t)\nabla p_\theta(a_t|\mathbf{s}_t)]$$

where $Q(a,\mathbf{s})$ stands for the "expected total reward" given by $r_D = \frac{D}{1-D}$ of generating token a given previous generation $\mathbf{s}$, which can be estimated with, e.g., Monte Carlo tree search (MCTS, [Silver et al., 2016]).

Thus, following the gradient estimator presented in Theorem 2.3.1, we derive another gradient estimator:

$$\nabla L_G(\theta) \approx \frac{\sum_i L_i}{m \sum Q(a_t^i,\mathbf{s}_t^i)} \sum_{i,t}^{m,L_i} Q(a_t^i,\mathbf{s}_t^i)\nabla \log p_\theta(a_t^i|\mathbf{s}_t^i)$$

where m is the minibatch size. Using MCTS brings in two benefits: it allows different steps of the generated sample to be adjusted with different weights, and it gives us a more stable estimator of the partition function Z. These two properties dramatically reduce the proposed estimator's variance.

2.3.3.2. **Mixed MLE-Mali Training**. When dealing with long sequences, the above model may result in accumulated variance. In the context of auto-regressive models where MLE training is possible, to alleviate the issue, we significantly reduce the variance by clamping the input using the training data for N time steps, and switch to a free running mode for the remaining $T - N$ time steps. Then during our training procedure, inspired from [Ranzato et al., 2016b], we slowly move N from T towards 0.

We first assume D is trained to discriminate generating distribution p_d and p_f^N, where the distribution p_f^N is defined as $p_f^N(x_0, \cdots x_L) = p_d(x_0, \cdots x_N)p_\theta(x_{N+1}, \cdots x_L|x_0, \cdots x_N)$. For any sequence $\mathbf{x} = \{x_i\}_{i=1}^L$, let $\mathbf{x}_{\leq N} = (x_0,x_1, \cdots x_N)$,$\mathbf{x}_{>N} = (x_{N+1}, \cdots x_L)$, $r_D = \frac{D}{1-D}$. For each sample $\mathbf{x}_i$ from the real data batch, if it has length larger than N, we fix the first N words of $\mathbf{x}_i$, and then sample n times from our model till the end of the sequence, and get n samples $\{\mathbf{x}_{i,j}\}_{j=1}^n$.

We then have the following series of mini-batch estimators for each $0 \leq N \leq T$:

$$\nabla L_G^N \approx \frac{1}{m}[\sum_{i=1,j=1}^{m,n}(\frac{r_D(\mathbf{x}_{i,j})}{\sum_k r_D(\mathbf{x}_{i,k})} - b/n)\nabla \log p_\theta(\mathbf{x}_{i,j}^{>N}|\mathbf{x}_i^{\leq N}) + \sum_{i=1}^{m}\sum_{t=0}^{N}\nabla \log p_\theta(a_t^i|\mathbf{s}_t^i)] = E_N(\{\mathbf{x}_{i,j}\}) \tag{2.3.2}$$

One difference is that here we normalize the coefficients $r_D(\mathbf{x}_{i,j})$ based only on samples generated from a single real sample $\mathbf{x}_i$. The reason of using this trick will be explained in next sub-section.

Algorithm 2 MaliGAN

f

Require: A generator p with parameters θ.
A discriminator $D(x)$ with parameters θ_d.
A baseline b.

for number of training iterations **do**
  for k steps **do**
    Sample a minibatch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from p_θ.
    Sample a minibatch of samples $\{\mathbf{y}_i\}_{i=1}^m$ from p_d.
    Update the parameter of discriminator by taking gradient descend of discriminator loss

$$\sum_i [\nabla_{\theta_d} \log D(\mathbf{y}_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(\mathbf{x}_i))]$$

  end for
  (Optional) Sample a mini-batch of latent variables $\{\mathbf{z}_i\}_{i=1}^m$ from prior distribution.
  Sample a mini-batch of samples $\{\mathbf{x}_i\}_{i=1}^m$ from $p_\theta(\cdot)$ (without latent variable) or $p_\theta(\cdot|\mathbf{z}_i)$.
  Update the generator by applying gradient $\sum_{i=1}^m (\frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} - b/m) \nabla \log p_\theta(\mathbf{x}_i)$ or $\sum_{i=1}^m (\frac{r_D(\mathbf{x}_i)}{\sum_j r_D(\mathbf{x}_j)} - b/m) \nabla \log p_\theta(\mathbf{x}_i|\mathbf{z}_i)$
end for

Algorithm 3 Sequential MaliGAN with Mixed MLE Training

Require: A generator p with parameters θ.
A discriminator $D(x)$ with parameters θ_d.
Maximum sequence length T, step size K.
A baseline b, sampling multiplicity m.

$N = T$
Optional: Pretrain model using pure MLE.
for number of training iterations **do**
  $N = N$ - K
  for k steps **do**
    Sample a minibatch of training data $\{\mathbf{y}_i\}_{i=1}^m$.
    While keeping the first N words the same as in $\{\mathbf{y}_i\}_{i=1}^m$, sample a minibatch of sequences $\{\mathbf{x}_i\}_{i=1}^m$ using p_θ starting from time step N.
    Update the discriminator by taking gradient descend of discriminator loss, same as Alg 2.
  end for
  Sample a minibatch of training data $\{\mathbf{x}_i\}_{i=1}^m$.
  For each sample $\mathbf{x}_i$ with length larger than N in the mini-batch, clamp the generator to the first N words of s, and freely run the model to generate m samples $\mathbf{x}_{i,j}, j = 1, \cdots m$ till the end of the sequence.
  Update the generator by applying the mixed MLE-Mali gradient update following Eq. 2.3.2.
end for

2.3.3.3. Single real data based renormalization. Many generative models have multiple layers of randomnesses. In these models, high-level random variables are usually responsible for modeling high-level decisions or "modes" of the probability distribution. Changing them can result in much larger effects than those from changing low-level variables. Motivated by this observation, in each mini-batch we first draw a mini-batch of high-level latent variables (e.g. 32), and then for each high-level value we draw a number of low-level data samples (e.g. 32). Then we re-estimate the partition function Z from the low-level samples that are generated by each high-level samples. Because lower-level sampling has a much smaller variance, the model can receive better gradient signals from the weights provided by the discriminator.

This sampling principle corresponds to applying the mixed MLE-Mali training discussed above in the auto-regressive settings. In this case we first sample a few data samples, then fix the first N words and let the network generate a lot of samples after the first N to form the next mini-batch. We refer to this full algorithm as sequential MaliGAN with Mixed MLE Training, which is summarized in Algorithm 3.

The benefits of this single real sample based renormalization are two-folds. First, consider S is a sample from the training set and the first N words $S_{\leq N}$ are completed by our model.

The conditional distribution $p_d(S'_{>N}|S_{\leq N})$ should be much simpler than the full distribution p_d. Namely, $p_d(S'_{>N}|S_{\leq N})$ consists of only one or a few "modes". So this renormalization technique can be viewed as training the model on these simpler conditional distributions, which gives more stable gradients.

Second, this normalization scheme makes our model robust to mode missing, a common failure pattern during GAN training [Che et al., 2016]. Single sample based renormalization ensures that for every real sample S, the model can receive a moderately strong training signal for how to perform better on generating $S_{>N}$ conditioned on $S_{\leq N}$. However, in batch-wise renormalization as in the basic MaliGAN, this is not possible because there might be some completions S' with $r_D(S')$ very large, so other training samples in that mini-batch receives very little gradient signals.

2.4. Experiments

We conduct experiments on four discrete sequence generation tasks.

2.4.1. Discrete MNIST

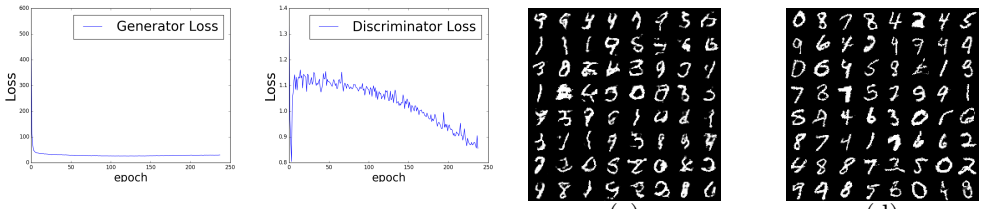
We first evaluate MaliGAN on the binarized image generation task for the MNIST hand-written digits dataset, similar with [Hjelm et al., 2017]. To generate the discrete samples, we sample from the generator's output binomial distribution. We compare our MaliGAN with the models trained using the discriminator's output as a direct reward, and denote it as the REINFORCE-like model.

The comparison results are shown in Figure 1. The two figures in the left are training losses of the generator and discriminator from the proposed MaliGAN. We can see the training process of MaliGAN is stable and the loss curve is meaningful. The right two figures are samples generated by the REINFORCE-like model and by MaliGAN. Clearly, the samples generated by MaliGAN have much better visual quality and resemble closely the training data.

2.4.2. Poem Generation

The second experiment is a Chinese poem generation task [Zhang and Lapata, 2014].[2]. We refer with *Poem-5* and *Poem-7* to those consisting of 5 or 7 Chinese characters each in a short sentence, respectively.

[2] `http://homepages.inf.ed.ac.uk/mlap/Data/EMNLP14/`



(a) (b) (c) (d)

Fig. 1. The training loss of the generator (a) and the discriminator (b) of MaliGAN on Discrete MNIST task; Samples generated by REINFORCE-like model (c) and by MaliGAN (d).

We compare four models: (1) The auto-regressive model with same architecture but trained with maximum likelihood (MLE); (2) MaliGAN-basic trained with Algorithm 2 without MCTS; (3) MaliGAN-full trained by Algorithm 3 with all the variance reduction techniques included; and (4) SeqGAN [Yu et al., 2017]. Following [Yu et al., 2017], we report the BLEU-2 scores in Table 1 [Papineni et al., 2002].

On both tasks, MaliGAN-full obtained the best BLEU-2 scores on par, and MaliGAN-basic was the next best. Clearly, MLE lagged far behind despite the same architecture, which should be attributed to the inherent defect in the MLE teacher-forcing training framework. As pointed by previous researchers [Wiseman and Rush, 2016], BLEU might not be a proper evaluation metric, we also calculate the Perplexity of these four models, obtaining similar results.

Table 1. Experimental results on Poetry Generation task. The result of SeqGAN is from [Yu et al., 2017].

Model	Poem-5		Poem-7	
	BLEU-2	PPL	BLEU-2	PPL
MLE	0.6934	564.1	0.3186	192.7
SeqGAN	0.7389	-	-	-
MaliGAN-basic	0.7406	548.6	0.4892	182.2
MaliGAN-full	**0.7735**	**538.4**	**0.5712**	**179.1**

2.4.3. Sentence-Level Language Modeling

We also examine the proposed algorithm on a more challenging task, sentence-level language modeling. We examine three types of models, MLE, MaliGAN without MCTS and MaliGAN with MCTS. We also examine 1-layer and 2-layer architectures. For evaluation we

report sentence-level perplexity, which is the averaged perplexity on all sentences in the test set. We also report BLEU-2 scores. The results are shown in Table 2.

Table 2. Experimental results on PTB. Note that we evaluate the models in sentence-level.

	MLE-1-layer	MaliGAN (w/o)	MaliGAN (full)-1-layer	**MaliGAN (full)-2-layer**
BLEU-2	0.5328	0.7105	0.7464	**0.7631**
Valid-Perplexity	141.9	131.6	128.0	**119.2**
Test-Perplexity	138.2	125.3	123.8	**118.1**

From Table 2 we can see, both MaliGAN-basic and MaliGAN-full obtained a notably lower perplexity compared with the MLE model. Although the PTB dataset is much more difficult, we obtain results consistent with Table 1. considering the small size of the PTB data, it is encouraging to see that our model is more robust to overfitting. These results strengthen our belief to realize our algorithm on even larger datasets, which we leave as a future work.

Besides, we can see that with advanced techniques provided in Section 2.3.3, MaliGAN-full performed in a more stable way during training and can to some extent achieve lower perplexity scores than MaliGAN-basic. We believe these fruitful techniques will be beneficial in other similar problem settings. Additionally, the BLEU-2 scores indicate the sentences generated by MaliGAN are more perceptually natural. We provide some generated sentences in the supplementary material.

2.4.4. Conditional Dialog Generation

In addition to unconditional discrete data generation, we also validate the proposed algorithm in a conditional setting. We conduct experiments to generate responses based on the input dialog utterances. More precisely, both the generator $p_\theta(\mathbf{x}|\mathbf{z},C)$ (or $p_\theta(\mathbf{x}|C)$) and the discriminator $D(\mathbf{x},C)$ are now conditioned on a dialog context C. The discriminator in this case has to not only examine the quality of generated response, but also has to take into consideration the consistency with the given context.

To augment expressiveness, we experiment with two kinds of models, one with latent variables using the MCTS version of MaliGAN and one without latent variables using mixed MLE-Mali Training. The compared models include a conditional VAE model, a vanilla sequence-to-sequence model trained with MLE, a GAN model with vanilla REINFORCE and a GAN model with Gumbel-Softmax [Maddison et al., 2016, Jang et al., 2016]. The latter two GAN models failed to generate any legal sentences, so we remove them from evaluation pool.

We perform detailed human evaluations on the generated responses.[3] We distribute 50 randomly sampled responses generated from 10 dialog contexts to five evaluators. Detailed results are given in Table 3.

From the results, we can see that MaliGAN training procedure can significantly improve the coherence and relevance of the generated response than pure MLE based training. This validates with the intuition that GAN discriminators is able to detect coherence problems in dialog responses.

Table 3. Human evaluation results on dialog response generation.

	Seq2Seq	CVAE	Mixed MLE-Mali	MaliGAN+latent
Coherence	0.66	0.76	**0.98**	0.90
Relevance	0.60	0.82	1.04	**1.16**
Grammar	1.20	1.34	**1.42**	1.32

2.5. Related Work

To improve the performance of discrete auto-regressive models, some researchers aim to tackle the *exposure bias* problem [Wiseman and Rush, 2016, Ranzato et al., 2016b, Serban et al., 2016]. The second issue is the discrepancy between the objective during training, i.e., to maximize the word-level probabilities, and the evaluation metric during testing, i.e., sequence-level metrics such as BLEU. This discrepancy is analyzed in [Ranzato et al., 2016b] and then summarized as *Loss-Evaluation Mismatch* by [Wiseman and Rush, 2016]. Because these metrics are often discrete, researchers generally seek help from reinforcement learning to add the evaluation metrics into the objective in the training phase. [Ranzato et al., 2016b] exploits the REINFORCE algorithm [Williams, 1992] and proposes several model variants. Similarly, [Liu et al., 2016] directly optimizes image caption metrics through policy gradient methods [Igel, 2005]. There exists a third issue, namely *Label Bias*, which makes it difficult for the MLE trained models to be optimized globally [Wiseman and Rush, 2016, Andor et al., 2016]

To addresses the abovementioned issues, we propose to formulate the problem under the setting of GANs. Initially proposed by [Goodfellow et al., 2014a], GANs have attracted exploding attention and have been successfully applied on image and video generation [Radford et al., 2015, Mirza and Osindero, 2014, Reed et al., 2016b, Zhang et al., 2016b, Nguyen et al.,

[3]On the model without latent variable, perplexity is available but previous work found it not a good metric for measuring the quality of responses [Tao et al., 2017, Lowe et al., 2017].

2016, Zhu et al., 2016, Sønderby et al., 2016, Ledig et al., 2016, Mathieu et al., 2015, Zhou and Berg, 2016, Saito and Matsumoto, 2016]. Despite these successes, work on applying GANs to text generation are not much explored yet noteworthy.

However, it is nontrivial to train GAN on discrete data due to its discontinuous nature. The instability inherent in GAN training makes things even worse [Salimans et al., 2016a, Che et al., 2016, Arjovsky and Bottou, 2017, Arjovsky et al., 2017b]. [Lamb et al., 2016] exploits adversarial domain adaption to regularize RNN's training. [Yu et al., 2017] applies GAN to discrete sequence generation by directly optimizing the discriminator's rewards. They adopt the Monte-Carlo tree search technique [Silver et al., 2016]. A similar technique is employed in [Li et al., 2017] which improves response generation by using adversarial learning.

In [Bornschein and Bengio, 2015], which inspired us, the authors propose a way of doing mini-batch reweighting when training latent variable models with discrete variables. However, they utilize an inference network which is infeasible in the GAN setting. Our work is also closely related to [Norouzi et al., 2016]. They propose to work with the objective $\mathrm{KL}(p_d||p_\theta)$ in a conditional generation setting. However, they directly sample from the augmented distribution because conditional generation metrics such as BLEU scores are decomposable into terms for each time step. This is not possible for sequence-level GANs, e.g., language modeling. Instead, we use importance sampling in our case.

2.6. Discussions and Future Work

When training discrete GANs, it is notoriously difficult to pass the discriminator information to train the generator. In this work, we solve this problem by first starting from the maximum likelihood training objective $\mathrm{KL}(p_d||p_\theta)$, and then using importance sampling combined with the discriminator output to derive a novel training objective. We prove that by estimating the partition function Z using samples, we are approximately doing normalized importance sampling with another distribution $p_{\setminus\Omega}$ which has much lower variance (Section 2.3.2). Practically, this single real sample normalization process combined with mixed training successfully avoided the missing mode problem by providing an equivalent training signal for each mode (Section 2.3.3.3).

In addition, our algorithm is surprisingly robust to overfitting. Teacher forcing is prone to overfit, because by maximizing the likelihood of the training data, the model can easily fit not only the regularities but also the noise in the data. However in our model, if the generator tries to fit too much noise in the data, the generated sample will become poorer

and the discriminator will very easily capture the differences between the generated and the real samples. As for future work, we plan to train the model on large datasets such as Google's one billion words [Chelba et al., 2014].

Chapter 3

Better Auto-Regressive Model Training with Energy-Based Models

3.1. Preface

3.1.1. My Role and Contributions In This Project

I proposed the idea for this project, derived the mathematical formulation, and supervised the experimental work, some debugging and paper writing. The first named author is responsible for most of the coding work. This is a paper in submission.

3.1.2. Motivation and Context

By factorizing the joint distribution into the product of a series of conditional distributions, auto-regressive generative models (abbr. ARGMs) [Vaswani et al., 2017, Dai et al., 2019b, van den Oord et al., 2016a,c, Salimans et al., 2017, Chen et al., 2018] simplify the difficult challenge of modeling high-dimensional joint distributions. They can be trained efficiently via maximum likelihood and generate samples of exceptional quality, making this technique popular for modeling distributions, especially for sequential data. Nonetheless, despite its potency and flexibility, ARGMs still have a slew of inherent flaws due to the intrinsic characteristics of chain-style conditional modeling. For example, ARGMs usually suffer from a discrepancy in the input context distributions between the training and inference stages, which causes consequent error propagation (i.e., Exposure Bias [Ranzato et al., 2016a, Bengio et al., 2015a]). Besides, due to the nature of greedy selection of beam search in the generation, the decoded results by ARGMs also lack long-range coherence. These remaining issues prevent ARGMs to produce good-quality results.

To address these defects, both heuristic and theoretical methods have been proposed. For instance, the exposure bias problem of ARGMs can be alleviated to some extent by mixing the input contexts with both real data and auto-regressively generated data during

the training stage, which is exactly the core idea of the scheduled sampling methods [Bengio et al., 2015a, Mihaylova and Martins, 2019]. However, one noteworthy trouble hiding inside this scheme is the over-correcting problem [Zhang et al., 2019]. In addition, at the inference stage, beam search enables model to choose more diverse candidates, improving the quality of generations. Nevertheless, its produced results only provide coherence marginally, since ARGM can only leverage previous decoded contexts without the consideration of the whole sequence information. Moreover, regardless its difficulty in optimization, energy-based models (EBMs) has demonstrated its effectiveness in modeling high-dimensional distributions in a variety of machine learning applications [Zhao et al., 2017, Arbel et al., 2021, Gao et al., 2021], which did not require a compromise of transforming the target distribution into a product of conditional distributions. As a result, some studies [Deng et al., 2020b, Bakhtin et al., 2021, Durkan and Nash, 2019] attempt to combine EBMs with ARGMs, expecting to take advantages of both models while eliminate their flaws. However, though obtained some positive results, the existing works preferred a two-stage optimization, which first obtain a well-trained ARGM and then train an additional EBM based on it. Such optimization strategy does not enable ARGM to benefit from the advances of EBM in modeling joint-distribution.

3.1.3. Main Contribution

In this chapter, we present a novel design for integrating the energy-based modeling approach into auto-regressive generative models seamlessly (E-ARM for short hereafter) and optimize it with an energy-based learning objective, which makes ARGMs to train auto-regressively with a constraint to align the joint sequence distribution at each time step. Thanks to our well-designed energy function, the two involved models can share a single base network without additional parameters, that is, the base network not only serves as a generator that provides fake data to facilitate the training of EBMs like previous works [Che et al., 2020a, Xiao et al., 2021, Durkan and Nash, 2019, Deng et al., 2020b], but also plays the role of modeling the energy surface.

Intuitively, the exposure bias in ARGMs is caused by the fact that the model is trained on real data rather than data generated by the model. On the other hand, in the EBM's optimization process for modeling joint densities, the negative phase of wake-sleep algorithms [Hinton, 2002b, Kim and Bengio, 2016a] requires sampling data from the EBM itself. Along with the fact that our method combines the EBM and the ARGM seamlessly as a whole, E-ARM can reduce the discrepancy between input data of the training and inference stage, which mitigates the exposure bias problem of the ARGM. On top of it, unlike

ARGMs, which factor the joint distribution into a product of conditional distributions, EBMs are able to model the joint distribution directly and score each input at the sequence level instead of at the token level, which makes them capable of modeling long-range coherence. Additionally, in order to optimize the proposed energy-based learning objective efficiently via gradient-based wake-sleep algorithms [Kim and Bengio, 2016a], we present a way to estimate the negative gradient (which is a necessary component in the gradient-based wake-sleep algorithms) through those samples generated by auto-regressive models instead of the EBM which requires expensive Markov chain monte carlo (MCMC) processes. This allows us to sidestep the extremely time-consuming MCMC procedures, accelerating the training processes.

In summary, the following contributions are described in this chapter: i) We introduce a novel scheme, E-ARM, to integrate an EBM into an ARGM seamlessly. ii) with the help of a proposed energy-based learning objective, we compel the ARGM to fit not only the conditional distribution but also the joint distribution at each time step, which can eliminate those inherent flaws such as exposure bias and enhance the global generational coherence. iii) We demonstrate how to efficiently optimize our model in a single network using wake-sleep algorithms without requiring an MCMC. iv) In a number of applications, such as language modeling, neural machine translation, and image generation, our model can achieve better results in comparison with baselines.

3.1.4. Follow-up Works and Impacts

This is a new work in submission and its impacts remains to be seen.

3.2. Exposure bias and incoherence problems in Auto-regressive models

In the discussion about the defects of sequential autoregressive generative models, the exposure bias problem [Bengio et al., 2015a, Ranzato et al., 2016a] is an important issue, which greatly affects the model's deployment performance. During the training stage, the autoregressive model is always conditioned on ground truth token sequences. In the generation stage, however, the model has to rely on its own previously generated tokens to predict the next token, when the model is deployed. If an incorrect token was generated by the model that conditions future generation, this error can be amplified in following steps because the next prediction will be made using an unusual input (one unlike those in the training set). Besides, out of the consideration of efficiency, greedy autoregressive decoding

usually selects the most probable token at each time step, given the ones previously selected. Such a scheme assumes the largest joint probability of the whole sequence can be achieved by separately choosing the most probable next token (given its previous context) over all time steps, which is generally suboptimal.

3.3. Integrate EBMs into auto-regressive models seamlessly

For a long time, as a result of compromises for improved training stability and efficiency (*e.g.*, modeling a joint distribution by decomposing it and using a teacher-forcing training strategy), conventional auto-regressive generative models have suffered from flaws such as the exposure bias and the lack of long-range coherence. To tackle these issues, we attempt to integrate energy-based models into auto-regressive models seamlessly (E-ARM), which can be regarded as a variant of ARGMs blending with an energy-based learning objective. Similar to conventional ARGMs, given a joint sequential distribution, E-ARM also addresses it auto-regressively, that is, tackling tokens step by step under a specific order. However, what differs from conventional approaches is that we attempt to model both the conditional and the joint distributions simultaneously at each time step. In this way, E-ARM can model distributions conveniently in an auto-regressive manner while avoiding those potential problems brought about by ARGMs.

Formally, given a sequence of random variables $(x_1, x_2, \ldots, x_K)$ with length K, we introduce a parametric ARGM $q_\theta(x_k|\mathbf{x}_{<k})$ (k denotes the time step) with parameters θ, and we define $p_\theta(x_k, \mathbf{x}_{<k})$ as a product of the base ARGM $q(\mathbf{x}_{<k}) = \prod_{l=1}^{k-1} q(x_l|\mathbf{x}_{<l})$ and an EBM as follows,

$$p_\theta(x_k, \mathbf{x}_{<k}) = q(\mathbf{x}_{<k}) \cdot \frac{e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}}{\mathbf{Z}_\theta}, \tag{3.3.1}$$

where the energy function $\phi_\theta(x_k, \mathbf{x}_{<k})$ is defined as the x_k's negative corresponding component of the base network's output logit with the input prefix context $\mathbf{x}_{<k} = (x_1, x_2, \ldots, x_{k-1})$ (*e.g.* given a sequence "This is Friday." and assuming the corresponding index of the token "Friday" in the vocabulary is i, then the value of $-\phi_\theta$("Friday", "This is") is the i-th component of the output logit, which is the straight input tensor of the final softmax layer), and the normalization term $\mathbf{Z}_\theta = \mathbb{E}_{\mathbf{x}'_{<k} \sim q_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}'_{<k})}]$.

Practically speaking, q and ϕ share the same parameters, however, in our model, during the optimization of the EBMs, we treat q as a fixed distribution and only optimize ϕ. This

is reasonable because the model is also trained with the auto-regressive loss and thus q is expected to approximate p_d as training progresses. In automatic gradient frameworks, this is equivalent to putting a stop_gradient operator over q.

Our primary goal is to make the distribution $q_\theta(x_k|\mathbf{x}_{<k})$ approach the real conditional $p_d(x_k|\mathbf{x}_{<k})$ whilst maintaining $p_\theta(x_k, \mathbf{x}_{<k})$ as close to the real joint $p_d(x_k, \mathbf{x}_{<k})$ as possible, which can be achieved by minimizing the appropriate Kullback-Leibler (KL) divergences,

$$\theta^* = \arg\min_\theta \left[\mathbf{D}_{KL}\Big(p_d(x_k|\mathbf{x}_{<k})||p_\theta(x_k|\mathbf{x}_{<k})\Big) + \lambda \mathbf{D}_{KL}\Big(p_d(x_k, \mathbf{x}_{<k})||p_\theta(x_k, \mathbf{x}_{<k})\Big)\right], \tag{3.3.2}$$

where λ adjusts the ratio between the two objectives. In Eq. 3.3.2, the first objective can be easily optimized with the usual cross entropy supervised learning objective while the second one is optimized in the sense of EBMs by wake-sleep algorithms [Hinton et al., 1995, Kim and Bengio, 2016a] which minimize the objective by descending the following gradient of θ according to Eq. 1.6.2[1]

$$\underbrace{\mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_d(x_k, \mathbf{x}_{<k})}\left[\frac{\partial}{\partial \theta}\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})\right]}_{\textbf{Positive Phase}} - \underbrace{\mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_\theta(x_k, \mathbf{x}_{<k})}\left[\frac{\partial}{\partial \theta}\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})\right]}_{\textbf{Negative Phase}}, \tag{3.3.3}$$

where we have $\mathbf{E}_\theta(x_k, \mathbf{x}_{<k}) = \phi_\theta(x_k, \mathbf{x}_{<k}) - \log q_\theta(\mathbf{x}_{<k})$. Since the optimization via Eq. 1.6.2 involves sampling data from the model and can lead to the discovery of non-data-like samples, whose likelihood is then explicitly reduced by the energy function, E-ARM will not be plagued by the exposure bias problem. Besides, because we model the joint distribution throughout the training process, our model can assess the entire sequence as a whole and generate more coherent data using energy sampling [Deng et al., 2020b]. Next, we show how to efficiently optimize this model.

3.4. Optimization

We optimize the first objective in Eq. 3.3.2 as in conventional auto-regressive models by reducing the cross-entropy loss. As for the second objective, we resort to descend the estimated gradient as shown in Eq. 3.3.3. Thanks to the importance sampling technique and our well-defined energy function, we show that the improved version of Eq. 3.3.3 has a simple

[1]here, we take a minimization version of the Eq. 1.6.2. As a result, the sign before each phase in two equations is converse.

and symmetric form that can be easily estimated whilst it does not require the expensive MCMC process.

Specifically, by replacing $\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})$ with the specific form $\phi_\theta(x_k, \mathbf{x}_{<k}) - \log q_\theta(\mathbf{x}_{<k})$, the gradient w.r.t. θ in the positive phase of Eq. 3.3.3 can be written as

$$-\mathbb{E}_{\mathbf{x}_{<k}\sim p_d}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_d}[\frac{\partial}{\partial\theta}\phi_\theta(x_k, \mathbf{x}_{<k})]. \tag{3.4.1}$$

Similarly, we can get the negative phase gradient as

$$-\mathbb{E}_{\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\phi_\theta(x_k, \mathbf{x}_{<k})]. \tag{3.4.2}$$

The first term $-\mathbb{E}_{\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})]$ in Eq. 3.4.1 is equivalent to the log-likelihood gradient for $q_\theta(\mathbf{x}_{<k})$, which means this gradient part will automatically descend as the optimization (the first KL-divergence in Eq. 3.3.2) of ARGM $q_\theta(\mathbf{x}_{<k}) = \prod_{l=1}^{k-1} q_\theta(x_l|\mathbf{x}_{<l})$ is carried on. Besides, because the estimation of the expectation operator over the data distribution p_d is easy, and the score $\phi_\theta(x_k, \mathbf{x}_{<k})$ can be acquired by simply accessing the ARGM's output logit (see the definition of ϕ_θ in Sec. 3.3), the second term can likewise be readily estimated and optimized. As a result, the positive phase optimization is both feasible and efficient.

The negative phase gradient estimation, on the other hand, is more involved. In Eq. 3.4.2, sampling data from p_θ is required for estimating the expectation $\mathbb{E}_{p_\theta}$, whereas p_θ is a parametric joint density involving an energy-based unnormalized density estimator that requires time-consuming MCMC methods to produce data. However, thanks to importance sampling, we can substitute the troublesome computation of the expectation over the distribution p_θ with the expectation over the distribution q_θ, which can generate samples auto-regressively without MCMC. Formally, we have the following equations:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] &= \sum_{\mathbf{x}_{<k}} p_\theta(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k}) \\
&= \sum_{\mathbf{x}_{<k}}\sum_{x_k} p_\theta(x_k, \mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k}) \\
&= \sum_{\mathbf{x}_{<k}} q_\theta(\mathbf{x}_{<k})\frac{\sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}}\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k}) \\
&= \mathbb{E}_{\mathbf{x}_{<k}\sim q_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})],
\end{aligned} \tag{3.4.3}$$

where we have

$$\begin{aligned}\mathbf{w}(\mathbf{x}_{<k}) &= \frac{\sum_{x_k} e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}} = \frac{\sum_{x_k} e^{-\phi(x_k,\mathbf{x}_{<k})}}{\sum_{\mathbf{x}_{<k}}\sum_{x_k} q_\theta(\mathbf{x}_{<k}) e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}} \\ &= \frac{\sum_{x_k} e^{-\phi(x_k,\mathbf{x}_{<k})}}{\sum_{\mathbf{x}_{<k}} q_\theta(\mathbf{x}_{<k}) \sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}} \\ &= \frac{\sum_{x_k} e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}_{<k}\sim q_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}]} \simeq \frac{\sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\frac{1}{N}\sum_{\mathbf{x}_{<k}}\sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}.\end{aligned} \tag{3.4.4}$$

From the above equations, we can derive the negative phase gradient $\mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(x_k,\mathbf{x}_{<k})]$ is equivalent to the following formulation,

$$-\mathbb{E}_{\mathbf{x}\sim p_\theta}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim q_\theta(x_k,\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})], \tag{3.4.5}$$

$$\textbf{where} \quad \mathbf{w}(\mathbf{x}_{<k}) = \frac{\sum_{x_k} e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k}\sim q_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k,\mathbf{x}'_{<k})}]}. \tag{3.4.6}$$

According to Eq. 3.4.5, all expectation estimations only need to sample data from the auto-regressive model q_θ, rather than the distribution p_θ, and the reweighing weight $\mathbf{w}$ in Eq. 3.4.6 also does not involve an expectation computation over the distribution p_θ. Generally, producing data from an auto-regressive model is a very simplified process compared to sampling straight from an EBM, which needs MCMC approaches [Durkan and Nash, 2019]. On account of that, the optimization process can be much more efficient.

Besides, the term $\mathbb{E}_{\mathbf{x}_{<k}\sim q_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})]$ in Eq.3.4.5 is equivalent to a re-weighted version of the gradient of q_θ's information entropy with respect to θ. This term can be optimized similarly to the teacher-forcing training of auto-regressive models with the "teacher" sequence generated auto-regressively by the model itself. Actually, scheduled sampling methods [Bengio et al., 2015a, Ranzato et al., 2016a, Mihaylova and Martins, 2019] shared a similar idea but without reweighting.

Ultimately, combining Eq. 3.4.1 and Eq. 3.4.5, we can optimize $p_\theta(x_k,\mathbf{x}_{<k})$ via descending the estimated gradient of θ as follows,

$$\underbrace{\begin{pmatrix} -\ \mathbb{E}_{\mathbf{x}_{<k}\sim p_d}[\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] \\ +\mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_d}[\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]\end{pmatrix}}_{\textbf{Positive Phase}} - \underbrace{\begin{pmatrix} -\ \mathbb{E}_{\mathbf{x}_{<k}\sim q_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log q_\theta(\mathbf{x}_{<k})] \\ +\mathbb{E}_{x_k,\mathbf{x}_{<k}\sim q_\theta(x_k,\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]\end{pmatrix}}_{\textbf{Negative Phase}}. \tag{3.4.7}$$

From Eq. 3.4.7, we can see that the only difference between the two phases is that in the negative phase, the expectation over q_θ has a reweighing weight $\mathbf{w}$ for each sample. The weight $\mathbf{w}$ in Eq. 3.4.6 and Eq. 3.4.7 can be deduced a little bit further and we can observe that

$$\mathbf{w}(\mathbf{x}_{<k}) = \frac{\mu(\mathbf{x}_{<k})}{\mathbb{E}_{\mathbf{x}'_{<k}}\mu(\mathbf{x}_{<k})}, \tag{3.4.8}$$

where $\mu(\mathbf{x}_{<k}) = \frac{p_\theta(\mathbf{x}_{<k})}{q_\theta(\mathbf{x}_{<k})}$ indicating the possibility of which distribution the prefix context $\mathbf{x}_{<k}$ is most likely to come from, the distribution p_θ or the distribution q_θ. Correspondingly, $\mathbf{w}(\mathbf{x}_{<k})$ reflects the context $\mathbf{x}_{<k}$'s relative magnitude of $\mu(\mathbf{x}_{<k})$ compared to the average among all potential contexts—the larger the value of $\mathbf{w}(\mathbf{x}_{<k})$, the more likely the context $\mathbf{x}_{<k}$ in the data space is close to the p_θ which is modeled by the product of auto-regressive models and EBMs. In the training stage, those input sequences with contexts which are closer to the modeled p_θ will be assigned larger weights $\mathbf{w}$ while those input sequences with contexts that are far from p_θ will be assigned with smaller weights $\mathbf{w}$.

3.5. Training and Inference

In practice, we approximate Eq. 3.4.7 with the auto-regressive model q_θ as a constant in each update so that we can ignore those gradient terms with respect to q_θ in Eq. 3.4.7. Thus we have

$$\mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_d}[\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})] - \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim q_\theta}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]. \tag{3.5.1}$$

Compared with the original version shown in Eq. 3.4.7, this approximation Eq. 3.5.1 achieves marginally better performance. We argue this is because of the imbalanced training signals among different timesteps. Concretely, at different timesteps during auto-regressive generation, the global likelihood can be naturally broken up into pieces: the $p_\theta(x_k, \mathbf{x}_{<k})$ for each k is an independent probabilistic density. The tricky part is that one cannot simply sum up the losses at all timesteps, because this means summing up $\log q_\theta(x_1) + \log q_\theta(x_1,x_2) + \cdots + \log q_\theta(x_1,\ldots,x_K)$ and will result in earlier timesteps to get stronger training signals.

We are unable to only train using the EBM training objective mentioned above. The reason for this is that when q_θ does not closely match p_θ, the importance weight will have high variance, and the expectation estimation will be dominated by the data with the most weight in the process of importance sampling. Moreover, importance sampling underestimates the gradient w.r.t. θ in the negative phase when q_θ does not fully cover regions of high density under p_θ [Salakhutdinov and Murray, 2008]. As a result, in order to make the optimization

more stable, we must maintain the cross-entropy loss throughout training and warm up the ARGM for a few epochs. The EBM training objective should be viewed as a regularizer that ensures our base model does not deviate from the real distribution p_d.

Following the excellent work [Deng et al., 2020b, Bakhtin et al., 2021], we also adopt Top-K energy re-sampling in the inference stage, which means that in the generative procedure, we first gather multiple candidates of produced sequences auto-regressively, and then re-sample from them based on their energy scores estimated by the network's logit at the last time step where the entire sequence has been processed. Since we employ the EBM to model the joint distribution at each time step, such a re-sampled strategy can mitigate the undesirable impact of the greedy selection of one token at a time, which increases the coherence of generated samples.

3.6. Experiments

To empirically corroborate the effectiveness of E-ARM and show its general applicability, we conduct extensive experiments covering 3 machine learning applications, which are neural machine translation (NMT), language modeling, and image generation. In this section, we will introduce the experimental setup for all applications one by one, followed by an analysis of the obtained results.

3.6.1. Application to Neural Machine Translation

Model	Label Smoothing	Scheduled Sampling	Beam Searching	BLEU4 ↑ DE→ EN	EN→ DE	EN→ IT	IT→ EN	ES→ EN	EN→ ES	Avg.
Base	-	-	-	32.44±0.06	26.64±0.10	27.92±0.03	30.48±0.08	38.61±0.11	35.42±0.09	31.92
			5 B	33.62±0.07	27.41±0.08	28.72±0.04	31.39±0.05	39.55±0.12	36.38±0.07	32.85
	✔	-	-	33.68±0.03	27.62±0.04	28.81±0.07	31.42±0.07	39.85±0.13	36.71±0.09	33.02
			5 B	34.61±0.08	28.46±0.06	29.72±0.10	32.29±0.03	40.64±0.07	37.48±0.05	33.87
	✔	✔	-	34.23±0.06	27.96±0.03	29.26±0.11	31.93±0.08	40.16±0.03	37.21±0.04	33.46
			5 B	35.10±0.04	28.73±0.04	29.97±0.07	32.64±0.12	40.91±0.06	37.93±0.10	34.21
E-ARM	-	-	-	32.99±0.10	27.15±0.03	28.33±0.12	31.13±0.04	39.56±0.01	36.07±0.02	32.54
			5 B	34.06±0.06	27.97±0.08	29.26±0.09	31.90 ±0.13	40.30 ±0.03	36.92 ±0.09	33.40
	✔	-	-	33.97 ±0.08	28.03 ±0.04	29.13 ±0.02	31.84 ±0.11	40.32 ±0.03	36.96 ±0.07	33.38
			5 B	34.93 ±0.05	28.91 ±0.12	30.04 ±0.11	32.56 ±0.04	41.01 ±0.06	37.73 ±0.12	34.20
	✔	✔	-	34.58 ±0.09	28.38 ±0.12	29.56 ±0.10	32.11 ±0.03	40.93 ±0.03	37.56 ±0.07	33.85
			5 B	**35.36** ±0.05	**29.11** ±0.04	**30.25** ±0.09	**32.82** ±0.11	**41.58** ±0.07	**38.19** ±0.03	**34.55**

Table 1. Comparison of BLEU4 scores between our approach E-ARM and the base ARGM trained just with cross-entropy loss on six translation pairs of IWSLT14 datasets. We use "-" to denote that the training trick (either label smoothing or scheduled sampling) is not used while "✔" indicates we use it. "**5 B**" means that we use beam searching with 5 beams.

E-ARM is initially evaluated under the neural machine translation (NMT) problem, which can be considered a conditional generation task and is particularly important in the natural language processing (NLP) field. We first analyze our E-ARM on IWSLT14 dataset, which includes six different language pairs ({German, Spanish, Italian} $\rightarrow$ English and English $\rightarrow$ {German, Spanish, Italian}). In addition, we test our model on the WMT16 (English $\rightarrow$ German) benchmark for evaluating E-ARM's performance on a larger scale. Hereafter we abbreviate English, German, Spanish, Italian as "En", "De", "Es", "It". The weight λ in Eq. 3.3.2 is set as 5e-2 for all translation tasks. We use one size type of transformer ("Base-IWSLT") for IWSLT14 benchmark and two size type of transformer ("Base-WMT", "Large-WMT") for WMT16 benchmark [2]. Scheduled Sampling is carried out following [Mihaylova and Martins, 2019].

[2]Implementation is developed on Fairseq [Ott et al., 2019].

Model	L.S.	S.S.	w/E-ARM	BLEU4 ↑
Base-WMT	-	-	-	27.56
	✔	-	-	28.04
	✔	✔	-	28.36
	✔	✔	✔	**28.62**
Large-WMT	-	-	-	28.70
	✔	-	-	29.05
	✔	✔	-	29.23
	✔	✔	✔	**29.44**

Table 2. Translation performance of our E-ARM on WMT16 English→German, evaluated by BLEU4. We uniformly use 5 beams to apply beams searching. "**L.S.**" denotes Label Smoothing and "**S.S.**" denotes Scheduled Sampling.

The results of IWSLT14 tasks are shown in Table 1. We test not only the pure performance of our E-ARM but also the compatibility with other training techniques. Specifically, we can observe that (1) without any particular training skills, our E-ARM outperforms the base auto-regressive translation model trained with cross-entropy singly by 0.62 (31.92 → 32.54) in average, especially on three translation pairs—38.61 → 39.56 on Spanish-to-English, 30.48 → 31.13 on Italian-to-English, 35.42 → 36.07 on English-to-Spanish. (2) Our E-ARM is compatible with other training technique like scheduled sampling or beam search which can reduce the exposure bias problem and the lack of long-range coherence to some extent. They are not mutually exclusive and can work together to further improve the performance of the base ARGM. (3) However, since scheduled sampling can reduce exposure bias and beam search can somewhat alleviate the flaws caused by greedy selection at each time step, the performance gain of E-ARM when all training tactics are used is only 0.34 (34.21 → 34.55), which is lower than 0.62 (31.92 → 32.54) obtained when pure training without other training techniques.

Additionally, we show the performance of our method on WMT16 English → German task in Table 2. When enabling label smoothing (L.S.), the model performance increases by 0.52 and 0.35 respectively in two different model scales. When further applying scheduled sampling (S.S.), the base transformer model performance improves to 28.36 while the larger one obtains 29.23. The best scores 28.62 and 29.44 are both obtained using E-ARM conbined with label smoothing and scheduled sampling. Overall, our training scheme performs favorably against the vanilla teacher-forcing training of ARGM and can universally take positive effects under different scales of models and datasets.

3.6.2. Application to Language Modeling

Model	#Params	PPL ↓
Tr-Base	149M	30.56
Tr-Base (w/E-ARM)	149M	**29.89**
Standard Tr-XL	151M	24.20
Standard Tr-XL (w/E-ARM)	151M	**23.81**

Table 3. Test set performance of different models on WikiText103. Evaluation is conducted using perplexity.

To further demonstrate its resilience to flaws of auto-regressive generating models, we also conduct experiments on the Language Modeling task. The WikiText-103 dataset [Merity et al., 2017], which is the largest word-level language modeling benchmark accessible with long-term reliance, was chosen as the testbed. It comprises 103 million training tokens from 28 thousand articles, with an average length of 3.6 thousand tokens per article, which allows evaluating the ability of long-term dependency modeling. Two network structures are mainly tested, which are TransformerBase [Vaswani et al., 2017] and Transformer-XL [Dai et al., 2019b](Tr-Base and Tr-XL for short respectively hereafter).

		Start Epoch		
		5	**15**	**25**
λ	**0.00**	30.56	30.56	30.56
	0.01	30.48	30.12	30.22
	0.05	30.43	**29.89**	30.16
	0.1	30.60	30.03	30.14
	0.5	30.71	30.36	30.47

Table 4. Ablation of different λ and the start epoch where we introduce the E-ARM into the training on WikiText103. Evaluation is conducted using perplexity.

The final results are reported in Table 3. From the results, we observe that E-ARM outperforms baselines with clear margins in different scale of models. Specifically, the Transformer-Base achieves 0.67 PPL points performance gain (from 30.56 to 29.89) and the Transformer-XL increases from 24.20 to 23.81. Our method neither modifies the structure of the base network nor do we introduce any extra module or learnable parameters, that is, the performance boost comes only from the introduced energy-based learning objective. Apart from this, we also conduct some ablation study of language modeling, which can be found in Table 4.

3.6.3. Application to Image Generation

In order to examine the effectiveness of our method on additional data modalities in addition to neural language processing (NLP) tasks, we show the results of applying E-ARM to the image generating task in this section. We test E-ARM above Pixel-CNN [Van Oord et al., 2016] and its variant Gated Pixel-CNN [Oord et al., 2016]. Experiments are carried out on the MNIST and CIFAR-10 datasets.

Table 5 summarizes the quantitative results measured by negative log-likelihood (NLL), while Figure 1 depicts some of the generated samples. From which we can see that with the

Model	Test (Train) NLL ↓	
	MNIST	CIFAR-10
Pixel-CNN	0.17 (0.13)	3.14 (3.08)
Pixel-CNN (w/E-ARM)	**0.14 (0.12)**	**3.08 (2.95)**
Gated Pixel-CNN	0.14 (0.11)	2.98 (2.90)
Gated Pixel-CNN (w/E-ARM)	0.12 (0.10)	**2.94 (2.88)**

Table 5. Test set performance of different models on MNIST and CIFAR-10 in bits/dim (lower is better), training performance in brackets.

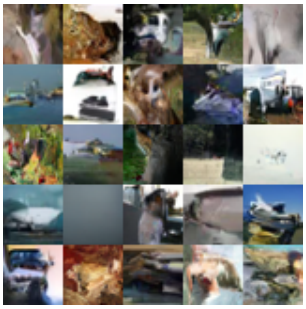


Fig. 1. Samples of CIFAR-10 from Gated Pixel-CNN (w/E-ARM).

help of our E-ARM, both the pixel-cnn and the gated pixel-cnn are improved in all datasets tested, which suggests a clear advantage of our design of the energy-based learning objective for improving auto-regressive models.

3.7. Related Works

3.7.1. auto-regressive generative models

Modeling high-dimensional data distribution directly is usually a rather challenging task due to "the curse of dimensionality" [Bellman, 1954]. One alternative method is to sequentialize the random variables and then factorize the joint probability distribution into the product of conditionals based on the appointed sequence, which is exactly the core idea of auto-regressive generative models. In general, this type of generative models has achieved prominent improvement in the world of generative modeling, particularly in tasks where sequential data is involved. For example, ARGMs have been widely used in language modeling [Vaswani et al., 2017, Dai et al., 2019b, Radford et al., 2019], audio synthesis [van den Oord et al., 2016a], and even image generation [van den Oord et al., 2016b,c, Salimans et al., 2017]. Nonetheless, what comes with the advantages, such as strong feasibility and high efficiency, of auto-regressive sequential models are several intrinsic defects: the exposure bias problem [Ranzato et al., 2016a, Bengio et al., 2015a, Song et al., 2020a], which emerges due to the discrepancy in input context distributions between the training and inference stages, as well as the lack of long-range coherence, which occurs because of the inherent greedy selection of one token at a time without look ahead.

3.7.2. Energy-based models

In the field of generative modeling, energy-based models have been widely used [Zhao et al., 2017, Arbel et al., 2021, Gao et al., 2021]. The primary idea behind EBMs is to understand the dependencies between variables (*e.g.* images and labels) represented by energy functions, and to assign low energies to proper configurations while assigning high energies to incorrect ones [LeCun et al., 2006].

Due to sampling difficulties [Kim and Bengio, 2016a, Grathwohl et al., 2021], training EBMs has been notoriously difficult, especially on high-dimensional data like images or texts. Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011a] is a frequently used gradient-based MCMC approach that injects noises into parameter updates and anneals the step size during the course of training, which has been adopted by numerous prior works [Nijkamp et al., 2019, Du and Mordatch, 2019, Grathwohl et al., 2020]. However, these MCMC methods always require enormous extra computing overheads and are not applicable when the input is discrete like in text applications [Deng et al., 2020b].

As a result, a variety of recent works attempt to explore the way of training an EBM without MCMC processes. In particular, [Bakhtin et al., 2021, Xu et al., 2021] optimize the EBMs using noise contrastive estimation (NCE) [Gutmann and Hyvärinen, 2010, Ma and Collins, 2018]. [Durkan and Nash, 2019] estimate the intractable normalization component by utilizing ARGMs and importance sampling. [Che et al., 2020a, Wang et al., 2021] skirt the challenge of collecting data in the high-dimensional data space by producing data in the lower-dimensional feature space, which improves the efficiency of sampling.

3.8. Conclusion

In this chapter, we proposed a novel method E-ARM to integrate energy-based models into the ARGM, as well as a well-designed training objective to train it efficiently. In particular, we introduce an energy function, defined using the base auto-regressive network's output logit, to model the unnormalized joint distribution of real data across the sequence's time steps, and importance sampling to avoid the requirement of MCMC processes during energy-based training, which makes the optimization much more tractable. Experimental results on two language tasks and one vision task demonstrate that our method has the ability to ameliorate the current ARGMs by reducing the exposure bias and increasing the long-range coherence of generated samples. In the future, we expect to extend our method on other sequential generation tasks, *e.g.* text summarization, audio generation, and incorporate our energy-based auto-regressive mechanism into other advanced structures.

Chapter 4

Improved GAN sampling with Energy-Based Models

4.1. Preface

4.1.1. My Role and Contributions In This Project

I proposed the core idea of for this project, worked out the mathematical formulation through discussions with my co-authors and carried out a small portion of experimental work. This project was published in [Che et al., 2020b].

4.1.2. Motivation and Context

Despite the ability of GANs to generate high-resolution, sharp samples, the samples of GAN models sometimes contain bad artifacts or are even not recognizable [Karras et al., 2019]. It is conjectured that this is due to the inherent difficulty of generating high dimensional complex data, such as natural images, and the optimization challenge of the adversarial formulation. In order to improve sample quality, conventional sampling techniques, such as increasing the temperature, are commonly adopted for GAN models [Brock et al., 2019]. Recently, new sampling methods such as Discriminator Rejection Sampling (DRS) [Azadi et al., 2018], Metropolis-Hastings Generative Adversarial Network (MH-GAN) [Turner et al., 2019], and Discriminator Optimal Transport (DOT) [Tanaka, 2019] have shown promising results by utilizing the information provided by both the generator and the discriminator. However, these sampling techniques are either inefficient or lack theoretical guarantees, possibly reducing the sample diversity and making the mode dropping problem more severe.

In this chapter, we show that GANs can be better understood through the lens of Energy-Based Models (EBM). In our formulation, GAN generators and discriminators collaboratively learn an "implicit" energy-based model. However, efficient sampling from this energy based

model directly in pixel space is *extremely* challenging for several reasons. One is that there is no tractable closed form for the implicit energy function in pixel space.

This motivates an intriguing possibility: that Markov Chain Monte Carlo (MCMC) sampling may prove more tractable in the GAN's latent space.

4.1.3. Main Contribution

Surprisingly, we find that the implicit energy based model defined jointly by a GAN generator and discriminator takes on a simpler, tractable form when it is written as an energy-based model over the generator's latent space. In this way, we propose a theoretically grounded way of generating high quality samples from GANs through what we call Discriminator Driven Latent Sampling (DDLS). DDLS leverages the information contained in the discriminator to re-weight and correct the biases and errors in the generator. Through experiments, we show that our proposed method is highly efficient in terms of mixing time, is generally applicable to a variety of GAN models (e.g. Minimax, Non-Saturating, and Wasserstein GANs), and is robust across a wide range of hyper-parameters. An energy-based model similar to our work is also obtained simultaneously in independent work [Arbel et al., 2020] in the form of an approximate MLE lower bound.

We highlight our main contributions as follows:

- We provide more evidence that it is beneficial to sample from the energy-based model defined both by the generator and the discriminator instead of from the generator only.
- We derive an equivalent formulation of the pixel-space energy-based model in the latent space, and show that sampling is much more efficient in the latent space.
- We show experimentally that samples from this energy-based model are of higher quality than samples from the generator alone.
- We show that our method can approximately extend to other GAN formulations, such as Wasserstein GANs.

4.1.4. Follow-up Works and Impacts

Although our paper was published only less than two years ago, it has attracted significant attention from the generative model community. Some follow-up works are directly generalizations, and some are improvement or applications of our algorithm. For instance, [Ansari et al., 2021] is a generalization of our algorithm to a larger class of gradient flows. In their view, our model is a special case of their gradient flows. Another line of work focusing

on applying our algorithm to compositional or controlled generalization [Nie et al., 2021]. The idea is simple — if we replace the discriminator with a K-way classifier, we can control the properties of the generated images.

Our idea of moving generative modeling to the latent space has inspired a large number of high quality research works. [DeVries et al., 2020] considers doing latent space reweighting during training not sampling, thus improving the training performance of GANs. [Zhou et al., 2021] considers learning neural Fokker Planck equations in the latent space to model high dimensional distributions.

Till now, our work has been cited more than 43 times.

4.2. Methodology

4.2.1. GANs as an Energy-Based Model

Suppose we have a GAN model trained on a data distribution p_d with a generator $G(z)$ with generator distribution p_g and a discriminator $D(x)$. We assume that p_g and p_d have the same support. This can be guaranteed by adding small Gaussian noise to these two distributions.

The training of GANs is an adversarial game which generally does not converge to the optimal generator, so usually p_d and p_g do not match perfectly at the end of training. However, the discriminator provides a quantitative estimate for how much these two distributions (mis)match. Let's assume the discriminator is near optimality, namely [Goodfellow et al., 2014b] $D(x) \approx \frac{p_d(x)}{p_d(x)+p_g(x)}$. From this equation, let $d(x)$ be the logit of $D(x)$, in which case $\frac{p_d(x)}{p_d(x)+p_g(x)} = \frac{1}{1+\frac{p_g(x)}{p_d(x)}} \approx \frac{1}{1+\exp(-d(x))}$, and we have $e^{d(x)} \approx p_d/p_g$, and $p_d(x) \approx p_g(x)e^{d(x)}$. Normalization of $p_g(x)e^{d(x)}$ is not guaranteed, and it will not typically be a valid probabilistic model. We therefore consider the energy-based model $p_d^* = p_g(x)e^{d(x)}/Z_0$, where Z_0 is a normalization constant. Intuitively, this formulation has two desirable properties. First, as we elaborate later, if $D = D^*$ where D^* is the optimal discriminator, then $p_d^* = p_d$. Secondly, it corrects the bias in the generator via weighting and normalization. If we can sample from this distribution, it should improve our samples.

There are two difficulties in sampling efficiently from p_d^*:

(1) Doing MCMC in pixel space to sample from the model is impractical due to the high dimensionality and long mixing time.

(2) $p_g(x)$ is implicitly defined and its density cannot be computed directly.

In the next section we show how to overcome these two difficulties.

4.2.2. Rejection Sampling and MCMC in Latent Space

Our approach to the above two problems is to formulate an equivalent energy-based model in the latent space. To derive this formulation, we first review rejection sampling [Casella et al., 2004]. With p_g as the proposal distribution, we have $e^{d(x)} / Z_0 = p_d^*(x) / p_g(x)$. Denote $M = \max_x p_d^*(x) / p_g(x)$ (this is well-defined if we add a Gaussian noise to the output of the generator and x is in a compact space). If we accept samples from proposal distribution p_g with probability $p_d^* / (Mp_g)$, then the samples we produce have the distribution p_d^*.

We can alternatively interpret the rejection sampling procedure above as occurring in the latent space z. In this interpretation, we first sample z from $p(z)$, and then perform rejection sampling on z with acceptance probability $e^{d(G(z))} / (MZ_0)$. Only once a latent sample z has been accepted do we generate the pixel level sample $x = G(z)$.

This rejection sampling procedure on z induces a new probability distribution $p_t(z)$. To explicitly compute this distribution we need to conceptually reverse the definition of rejection sampling. We formally write down the "reverse" lemma of rejection sampling as Lemma 1, to be used in our main theorem.

Lemma 4.2.1. *On space X there is a probability distribution $p(x)$. $r(x) : X \to [0,1]$ is a measurable function on X. We consider sampling from p, accepting with probability $r(x)$, and repeating this procedure until a sample is accepted. We denote the resulting probability measure of the accepted samples $q(x)$. Then we have $q(x) = p(x)r(x) / Z$,where $Z = \mathbb{E}_p[r(x)]$.*

Namely, we have the prior proposal distribution $p_0(z)$ and an acceptance probability $r(z) = e^{d(G(z))} / (MZ_0)$. We want to compute the distribution after the rejection sampling procedure with $r(z)$. With Lemma 1, we can see that $p_t(z) = p_0(z)r(z) / Z'$. We expand on the details in our main theorem.

4.2.3. Main Theorem

Lemma 4.2.2. *On space X there is a probability distribution $p(x)$. $r(x) : X \to [0,1]$ is a measurable function on X. We consider sampling from p, accepting with probability $r(x)$, and repeating this procedure until a sample is accepted. We denote the resulting probability measure of the accepted samples $q(x)$. Then we have:*

$$q(x) = p(x)r(x) / Z,\ Z = \mathbb{E}_p[r(x)]. \tag{4.2.1}$$

Proof. From the definition of rejection sampling, we can see that in order to get the distribution $q(x)$, we can sample x from $p(x)$ and do rejection sampling with probability

$r'(x) = q(x) \,/\, (Mp(x))$, where $M \geq q(x) / \, p(x)$ for all x. So we have $r'(x) = r(x) \,/\, (ZM)$. If we choose $M = 1 / \, Z$, then from $r(x) \leq 1$ for all x, we can see that M satisfies $M \geq q(x) / \, p(x) = r(x) \,/\, Z$, for all x. So we can choose $M = 1 \,/\, Z$, resulting in $r(x) = r'(x)$. □

Theorem 4.2.3. *Assume p_d is the data generating distribution, and p_g is the generator distribution induced by the generator $G : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{Z}$ is the latent space with prior distribution $p_0(z)$. Define $p_d^* = e^{\log p_g(x) + d(x)} / \, Z_0$, where Z_0 is the normalization constant.*

Assume p_g and p_d have the same support. This assumption is typically satisfied when $\dim(z) \geq \dim(x)$. We address the case that $\dim(z) < \dim(x)$ in Corollary 4.2.3.1. Further, let $D(x)$ be the discriminator, and $d(x)$ be the logit of D, namely $D(x) = \sigma\left(d(x)\right)$. We define the energy function $E(z) = -\log p_0(z) - d(G(z))$, and its Boltzmann distribution $p_t(z) = e^{-E(z)} / \, Z$. Then we have:

(1) $p_d^* = p_d$ when D is the optimal discriminator.

(2) If we sample $z \sim p_t$, and $x = G(z)$, then we have $x \sim p_d^*$. Namely, the induced probability measure $G \circ p_t = p_d^*$.

Proof. (1) follows from the fact that when D is optimal, $D(x) = \frac{p_g}{p_d + p_g}$, so $D(x) = \sigma(\log p_d - \log p_g)$, which implies that $d(x) = \log p_d - \log p_g$ (which is finite on the support of p_g due to the fact that they have the same support). Thus, $p_d^*(x) = p_d(x) / \, Z_0$, we must have $Z_0 = 1$ for normalization, so $p_d^* = p_d$.

For (2), for samples $x \sim p_g$, if we do rejection sampling with probability $p_d^*(x) / \, (Mp_g(x)) = e^{d(x)} / \, (MZ_0)$ (where M is a constant with $M \geq p_d^*(x) / \, p_g(x)$), we get samples from the distribution p_d^*. We can view this rejection sampling as a rejection sampling in the latent space $\mathcal{Z}$, where we perform rejection sampling on $p_0(z)$ with acceptance probability $r(z) = p_d^*(G(z)) / \, (Mp_g(G(z))) = e^{d(G(z))} / \, M$. Applying lemma 1, we see that this rejection sampling procedure induces a probability distribution $p_t(z) = p_0(z) r(z) / C$ on the latent space $\mathcal{Z}$. C is the normalization constant. Thus sampling from $p_d^*(x)$ is equivalent to sampling from $p_t(z)$ and generating with $G(z)$. □

Interestingly, $p_t(z)$ has the form of an energy-based model, $p_t(z) = e^{-E(z)} / \, Z'$, with *tractable* energy function $E(z) = -\log p_0(z) - d(G(z))$. In order to sample from this Boltzmann distribution, one can use an MCMC sampler, such as Langevin dynamics or Hamiltonian Monte Carlo.

4.2.4. Sampling Wasserstein GANs with Langevin Dynamics

Wasserstein GANs are different from original GANs in that they target the Wassertein loss. Although when the discriminator is trained to optimality, the discriminator can recover the Kantorovich dual [Arjovsky et al., 2017c] of the optimal transport between p_g and p_d, the target distribution p_d cannot be exactly recovered using the information in p_g and D. However, in the following we show that in practice, the optimization of WGAN can be viewed as an approximation of an energy-based model, which can also be sampled with our method.

The objectives of Wasserstein GANs can be summarized as:

$$L_D = \mathbb{E}_{p_g}[D(x)] - \mathbb{E}_{p_d}[D(x)] \quad , \quad L_G = -\mathbb{E}_{p_0}[D(G(z))] \tag{4.2.2}$$

where D is restricted to be a K-Lipschitz function.

On the other hand, consider a new energy-based generative model (which also has a generator and a discriminator) trained with the following objectives:

(1) Discriminator training phase (D-phase). Unlike GANs, our energy-based model tries to match the distribution $p_t(x) = p_g(x)e^{D_\phi(x)} / Z$ with the data distribution p_d, where $p_t(x)$ can be interpreted as an EBM with energy $D_\phi(x) - \log p_g(x)$. In this phase, the generator is kept fixed, and the discriminator is trained.

(2) Generator training phase (G-phase). The generator is trained such that $p_g(x)$ matches $p_t(x)$, in this phase we treat D as fixed and train G.

In the D-phase, we are training an EBM with data from p_d. The gradient of the KL-divergence (which is our loss function for D-phase) can be written as [MacKay, 2003]:

$$\nabla_\phi \mathrm{KL}(p_d||p_t) = \mathbb{E}_{p_t}[\nabla_\phi D(x)] - \mathbb{E}_{p_d}[\nabla_\phi D(x)] \tag{4.2.3}$$

Namely we are trying to maximize D on real data and trying to minimize it on fake data. Note that the fake data distribution p_t is a function of both the generator and discriminator, and cannot be sampled directly. As with other energy-based models, we can use an MCMC procedure such as Langevin dynamics to generate samples from p_t [Tieleman, 2008].

In the G-phase, we can train the model with the gradient of KL-divergence $\mathrm{KL}(p_g \,||\, p_t')$ as our loss. Let p_t' be a fixed copy of p_t, we can compute the gradient as:

$$\nabla_\theta \mathrm{KL}(p_g \,||\, p_t') = -\mathbb{E}[\nabla_\theta D(G(z))]. \tag{4.2.4}$$

Note that the losses above coincide with what we are optimizing in WGANs, with two differences:

(1) In WGAN, we optimize D on p_g instead of p_t. This may not be a big difference in practice, since as training progresses p_t is expected to approach p_g, as the optimizing

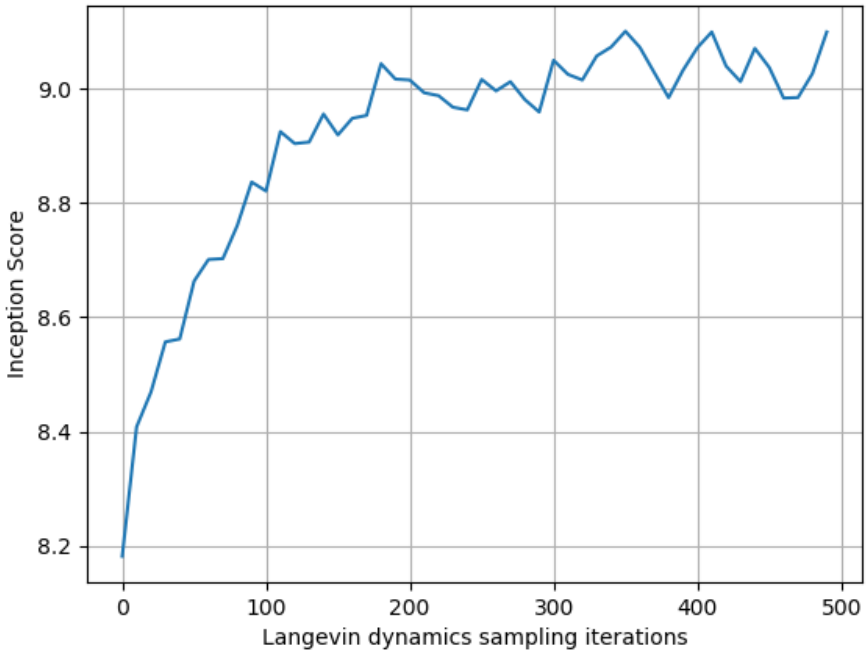


Fig. 1. Progression of Inception Score with more Langevin dynamics sampling steps.

loss for the generator explicitly acts to bring p_g closer to p_t (Equation 4.2.4). Moreover, it has recently been found in LOGAN [Wu et al., 2019] that optimizing D on p_t rather than p_g can lead to better performance.

(2) In WGAN, we impose a Lipschitz constraint on D. This constraint can be viewed as a smoothness regularizer. Intuitively it will make the distribution $p_t(x) = p_g(x)e^{-D_\phi(x)} / Z$ more "flat" than p_d, but $p_t(x)$ (which lies in a distribution family parameterized by D) remains an approximator to p_d subject to this constraint.

Thus, we can conclude that for a Wasserstein GAN with discriminator D, WGAN approximately optimizes the KL divergence of $p_t = p_g(x)e^{-D(x)} / Z$ with p_d, with the constraint that D is K-Lipschitz. This suggests that one can also perform DDLS on the WGAN latent space to generate improved samples, using an energy function $E(z) = -\log p_0(z) - D(G(z))$.

4.2.5. DDLS Algorithm

We show the detailed algorithm using Langevin dynamics in Alg. 4.

Algorithm 4 Discriminator Langevin Sampling

Input: $N \in \mathbb{N}_+, \epsilon > 0$
Output: Latent code $z_N \sim p_t(z)$
Sample $z_0 \sim p_0(z)$.
for $i < N$ **do**
  $n_i \sim N(0,1)$
  $z_{i+1} = z_i - \epsilon/2 \nabla_z E(z) + \sqrt{\epsilon} n_i$
  $i = i + 1$
end for

Algorithm 5 WGAN-EBM Hybrid Algorithm

Input: $N \in \mathbb{N}_+, \epsilon > 0, \delta > 0$, Initialized $D_\phi(x), G_\theta(z)$
Output: Trained $D_\phi(x), G_\theta(z)$
for Model Not Converged **do**
  Sample a batch $z_0^k \sim p_0(z), k = 1,2,\cdots M$.
  Sample a batch of real data $x^k, k = 1,2,\cdots M$.
  for $i < N$ **do**
    $n_i^k \sim N(0,1)$
    $z_{i+1}^k = z_i^k - \epsilon/2 \nabla_z E(z_i^k) + \sqrt{\epsilon} n_i^k,$ ▷ $E(z) = -\log p_0(x) - D(G(z))$
    $i = i + 1$
  end for
  $\phi = \phi - \delta(\sum_{k=1}^M \nabla_\phi D(G(z_N^k)) - \nabla_\phi D(x^k))$
  $\theta = \theta + \delta \sum_{k=1}^M \nabla_\theta D(G(z_0^k))$
end for

We described an EBM algorithm which WGAN is approximately optimizing. Here we detail this algorithm in Alg. 5.

4.2.6. Practical Issues and the Mode Dropping Problem

Mode dropping is a major problem in training GANs. In our main theorem it is assumed that p_g and p_d have the same support. We also assumed that $G : \mathcal{Z} \to \mathcal{X}$ is a deterministic function. Thus, if G cannot recover some of the modes in p_d, p_d^* also cannot recover these modes.

However, we can partially solve the mode dropping problem by introducing an additional Gaussian noise $z' \sim N(0,1; z') = p_1(z')$ to the output of the generator, namely we define the new deterministic generator $G^*(z,z') = G(z) + \epsilon z'$. We treat z' as a part of the generator, and do DDLS on joint latent variables (z,z'). The Langevin dynamics on this joint energy will help the model to move data points that are a little bit off-mode to the data manifold, and we have the follwing Corollary:

Corollary 4.2.3.1. *Assume p_d is the data generating distribution with small Gaussian noise added. The generator $G : \mathcal{Z} \to \mathcal{X}$ is a deterministic function, where $\mathcal{Z}$ is the latent space endowed with prior distribution $p_0(z)$. Assume $z' \sim p_1(z') = N(0,1;z)$ is an additional Gaussian noise variable with $\dim z' = \dim \mathcal{X}$. Let $\epsilon > 0$, denote the distribution of the extended generator $G^*(z,z') = G(z) + \epsilon z'$ as p_g. $D(x)$ is the discriminator trained between p_g and p_d. Let $d(x)$ be the logit of D, namely $D(x) = \sigma(d(x))$. Define $p_d^* = e^{\log p_g(x) + d(x)} / Z_0$, where Z_0 is the normalization constant. We define the energy function in the extended latent space $E(z,z') = -\log p_0(z) - \log p_1(z') - d(G^*(z,z'))$, and its Boltzmann distribution $p_t(z,z') = e^{-E(z,z')} / Z$. Then we have:*

(1) $p_d^* = p_d$ when D is the optimal discriminator.

(2) If we sample $(z,z') \sim p_t$, and $x = G^*(z,z')$, then we have $x \sim p_d^*$. Namely, the induced probability measure $G^* \circ p_t = p_d^*$.

Proof. Let $G^*(z,z')$ be the generator G defined in Theorem 1, we can see that p_d and p_g have the same support. Apply Theorem 1 and we deduce the corollary. □

4.3. Related Work

Previous work has considered utilizing the discriminator to achieve better sampling for GANs. Discriminator rejection sampling [Azadi et al., 2018] and Metropolis-Hastings GANs [Turner et al., 2019] use p_g as the proposal distribution and D as the criterion of acceptance or rejection. However, these methods are inefficient as they may need to reject a wechlot of samples. Intuitively, one major drawback of these methods is that since they operate in the pixel space, their algorithm can use discriminators to reject samples when they are bad, but cannot easily guide latent space updates using the discriminator which would improve these samples. The advantage of DDLS over DRS or MH-GAN is similar to the advantage of SGD over zero-th order optimization algorithms.

Trained classifiers have similarly been used to correct probabilistic models in other contexts [Cranmer et al., 2015]. Discriminator optimal transport (DOT) [Tanaka, 2019] is another way of sampling GANs. They use deterministic gradient descent in the latent space to get samples with higher D-values, However, since p_g and D cannot recover the data distribution exactly, DOT has to make the optimization local in a small neighborhood of generated samples, which hurts the sample performance. Also, DOT is not guaranteed to converge to the data distribution even under ideal assumptions (D is optimal).

Other previous work considered the usage of probabilistic models defined jointly by the generator and discriminator. In [Deng et al., 2020a], the authors use the idea of training an EBM defined jointly by a generator and an additional critic function in the text generation

setting. [Grover et al., 2019] uses an additional discriminator as a bias corrector for generative models via importance weighting. [Grover et al., 2018] considered rejection sampling in latent space in encoder-decoder models.

Energy-based models [Hinton and Salakhutdinov, 2006, Tao et al., 2019, Gao et al., 2020, 2018, Xie et al., 2018, Han et al., 2017, 2019, 2020, Pang et al., 2020] have gained significant attention in recent years. Most work focuses on the maximum likelihood learning of energy-based models [LeCun et al., 2006, Du and Mordatch, 2019, Salakhutdinov and Hinton, 2009b]. Other work has built new connections between energy based models and classifiers [Grathwohl et al., 2019]. The primary difficulty in training energy-based models comes from effectively estimating and sampling the partition function. The contribution to training from the partition function can be estimated via MCMC [Du and Mordatch, 2019, Hinton, 2002c, Nijkamp et al., 2019], via training another generator network [Kim and Bengio, 2016b, Kumar et al., 2019], or via surrogate objectives to maximum likelihood [Hyvärinen, 2005, Gutmann and Hyvärinen, 2010, Sohl-Dickstein et al., 2011]. The connection between GANs and EBMs has been studied by many authors [Zhao et al., 2016, Finn et al., 2016, Dai et al., 2019a, Zhai et al., 2019]. Our result can be viewed as establishing a new connection between GANs and EBMs which allows efficient latent MCMC sampling.

4.4. Experimental results

In this section we present a set of experiments demonstrating the effectiveness of our method on both synthetic and real-world datasets. In section 4.4.1 we illustrate how the proposed method, DDLS, can improve the distribution modeling of a trained GAN and compare with other baseline methods. In section 4.4.2 we show that DDLS can improve the sample quality on real world datasets, both qualitatively and quantitatively.

4.4.1. Synthetic dataset

Table 1. DDLS suffers less from mode dropping when modeling the 2d synthetic distribution in Figure 3. Table shows number of recovered modes, and fraction of "high quality" (less than four standard deviations from mode center) recovered modes.

	# recovered modes	% "high quality"	std "high quality"
Generator only	24.8 ± 0.2	70 ± 9	0.11 ± 0.01
DRS	24.8 ± 0.2	90 ± 2	$\mathbf{0.10 \pm 0.01}$
GAN w. DDLS	24.8 ± 0.2	$\mathbf{98 \pm 2}$	$\mathbf{0.10 \pm 0.01}$

Table 2. DDLS has lower Earth Mover's Distance (EMD) to the true distribution for the 2d synthetic distribution in Figure 3, and matches the performance of DOT on the Swiss Roll distribution.

	EMD 25-Gaussian	EMD Swiss Roll
Generator only([Tanaka, 2019])	0.052(08)	0.021(05)
DOT([Tanaka, 2019])	0.052(10)	**0.020**(06)
Generator only(Our imple.)	0.043(04)	0.026(03)
GAN as EBM with DDLS	**0.036**(04)	**0.020**(05)

Following the same setting used in [Azadi et al., 2018, Turner et al., 2019, Tanaka, 2019], we apply DDLS to a WGAN model trained on two synthetic datasets, 25-gaussians and Swiss Roll, and investigate the effect and performance of the proposed sampling method.

Implementation details We follow the same synthetic experiments design as in DOT [Tanaka, 2019], while parameterizing the prior with a standard normal distribution instead of a uniform distribution. The 25-Gaussians dataset is generated by a mixture of twenty-five two-dimensional isotropic Gaussian distributions with variance 0.01, and means separated by 1, arranged in a grid. The Swiss Roll dataset is a standard dataset for testing dimensionality reduction algorithms. We use the implementation from scikit-learn, and rescale the coordinates as suggested by [Tanaka, 2019]. We train a Wasserstein GAN model with the standard WGAN-GP objective. Both the generator and discriminator are fully connected neural networks with ReLU nonlinearities, and we follow the same architecture design as in DOT [Tanaka, 2019], while parameterizing the prior with a standard normal distribution instead of a uniform distribution. We optimize the model using the Adam optimizer, with $\alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9$.

Qualitative results With the trained generator and discriminator, we generate 5000 samples from the generator, then apply DDLS in latent space to obtain enhanced samples. We also apply the DOT method as a baseline. All results are depicted in Figure 3 and Figure 4 together with the target dataset samples. For the 25-Gaussian dataset we can see that DDLS recovered and preserved all modes while significantly eliminating spurious modes compared to a vanilla generator and DOT. For the Swiss Roll dataset we can also observe that DDLS successfully improved the distribution and recovered the underlying low-dimensional manifold of the data distribution. This qualitative evidence supports the hypothesis that our GANs as energy based model formulation outperforms the noisy implicit distribution induced by the generator alone.

Quantitative results We first examine the performance of DDLS quantitavely by using the metrics proposed by DRS [Azadi et al., 2018]. We generate 10,000 samples with the DDLS algorithm, and each sample is assigned to its closest mixture component. A sample is of "high quality" if it is within four standard deviations of its assigned mixture component, and a mode is successfully "recovered" if at least one high-quality sample is assigned to it.

As shown in Table 1, our proposed model achieves a higher "high-quality" ratio. We also investigate the distance between the distribution induced by our GAN as EBM formulation and the true data distribution. We use the Earth Mover's Distance (EMD) between the two corresponding empirical distributions as a surrogate, as proposed in DOT [Tanaka, 2019]. As shown in Table 2, the EMD between our sampling distribution and the ground-truth distribution is significantly below the baselines. Note that we use our own re-implementation, and numbers differ slightly from those previously published.

4.4.2. CIFAR-10 and CelebA

In this section we evaluate the performance of the proposed DDLS method on the CIFAR-10 dataset and CelebA dataset.

Implementation details We provide detailed description of baseline models, DDLS hyper-parameters and evaluation protocol as follows.

For CIFAR-10 dataset, we adopt the Spectral Normalization GAN (SN-GAN) [Miyato et al., 2018b] as our baseline GAN model. We take the publicly available pre-trained models of unconditional SN-GAN and apply DDLS. For CelebA dataset, we adopt DCGAN and WGAN as the baseline model following the same setting in [Radford et al., 2015]. We first sample latent codes from the prior distribution, then run the Langevin dynamics procedure with an initial step size 0.01 up to 1000 iterations to generate enhanced samples. Following the practice in [Welling and Teh, 2011b] we separately set the standard deviation of the Gaussian noise as 0.1. We optionally fine-tune the pre-trained discriminator with an additional fully-connected layer and a logistic output layer using the binary cross-entropy loss to calibrate the discriminator as suggested by [Azadi et al., 2018, Turner et al., 2019].

Quantitative results We evaluate the quality and diversity of generated samples via the Inception Score [Salimans et al., 2016b] and Fréchet Inception Distance (FID) [Heusel et al., 2017]. In Table 3, we show the Inception score improvements from DDLS on CIFAR-10 and CelebA, compared to MH-GAN [Turner et al., 2019] and DRS [Azadi et al., 2018], following the same evaluation protocol and using the same baseline models (DCGAN and WGAN) in [Turner et al., 2019]. On CIFAR-10, we applied DDLS to the unconditional generator of SN-GAN to generate 50000 samples and report all results in Table 3. We found that the proposed

Table 3. Inception and FID scores on CIFAR-10 and CelebA (grouped by corresponding baseline modles), showing a substantial quantitative advantage from DDLS, compared to MH-GAN [Turner et al., 2019], DRS [Azadi et al., 2018] and DOT [Tanaka, 2019] using the same architecture.

Model	CIFAR-10		CelebA
	Inception	FID	Inception
DCGAN w/o DRS or MH-GAN	2.8789	-	2.3317
DCGAN w/ DRS(cal) [Azadi et al., 2018]	3.073	-	2.869
DCGAN w/ MH-GAN(cal) [Turner et al., 2019]	3.379	-	3.106
WGAN w/o DRS or MH-GAN	3.0734	-	2.7876
WGAN w/ DRS(cal) [Azadi et al., 2018]	3.137	-	2.861
WGAN w/ MH-GAN(cal) [Turner et al., 2019]	3.305	-	2.889
Ours: DCGAN w/ DDLS	**3.681**	-	**3.372**
Ours: WGAN w/ DDLS	**3.614**	-	**3.093**
PixelCNN [van den Oord et al., 2016b]	4.60	65.93	-
EBM [Du and Mordatch, 2019]	6.02	40.58	-
WGAN-GP [Gulrajani et al., 2017]	$7.86 \pm .07$	36.4	-
ProgressiveGAN [Karras et al., 2018]	$8.80 \pm .05$	-	-
NCSN [Song and Ermon, 2019]	$8.87 \pm .12$	25.32	-
ResNet-SAGAN w/o DOT	$7.85 \pm .11$	21.53	-
ResNet-SAGAN w/ DOT	$8.50 \pm .12$	19.71	-
SNGAN w/o DDLS	$8.22 \pm .05$	21.7	-
Ours: SNGAN w/ DDLS	$9.05 \pm .11$	15.76	-
Ours: SNGAN w/ DDLS(cal)	$\mathbf{9.09 \pm 0.10}$	**15.42**	-

method significantly improves the Inception Score of the baseline SN-GAN model from 8.22 to 9.09 and reduces the FID from 21.7 to 15.42. Our unconditional model outperforms previous state-of-the-art GANs and other sampling-enhanced GANs [Azadi et al., 2018, Turner et al., 2019, Tanaka, 2019] and even approaches the performance of conditional BigGANs [Brock et al., 2019] which achieves an Inception Score 9.22 and an FID of 14.73, *without the need of additional class information, training and parameters.*

Qualitative results We illustrate the process of Langevin dynamics sampling in latent space in Figure 5 by generating samples for every 10 iterations. We find that our method helps correct the errors in the original generated image, and makes changes towards more semantically meaningful and sharp output by leveraging the pre-trained discriminator. To

demonstrate that our model is not simply memorizing the CIFAR-10 dataset, we find the nearest neighbors of generated samples in the training dataset and show the results in Figure 6.

Mixing time evaluation MCMC sampling methods often suffer from extremely long mixing times, especially for high-dimensional multi-modal data. For example, more than 600 MCMC iterations are need to obtain the most performance gain in MH-GAN [Turner et al., 2019] on real data. We demonstrate the sampling efficiency of our method by showing that we can expect a much shorter time to achieve competitive performance by migrating the Langevin sampling process to the latent space, compared to sampling in high-dimensional multi-modal pixel space.

4.4.3. ImageNet

Table 4. Inception score for ImageNet, showing the quantitative advantage of DDLS.

Model	Inception
SNGAN [Miyato et al., 2018b]	36.8
cGAN w/o DOT	36.23
cGAN w/ DOT	37.29
Ours: SNGAN w/ DDLS	40.2

In this section we evaluate the performance of the proposed DDLS method on the ImageNet dataset.

Implementation details As with CIFAR-10, we adopt the Spectral Normalization GAN (SN-GAN) [Miyato et al., 2018b] as our baseline GAN model. We take the publicly available pre-trained models of SN-GAN and apply DDLS. Fine tuning is performed on the discriminator, as described in Section 4.4.2. We introduce more details of the preliminary experimental results on Imagenet dataset here. We run the Langevin dynamics sampling algorithm with an initial step size 0.01 up to 1000 iterations. We decay the step size with a factor 0.1 for every 200 iterations. The standard deviation of Gaussian noise is annealed simultaneously with the step size. The discriminator is not yet calibrated in this preliminary experiment. We show the quantitative results in Table 4, where we substantially outperform the baseline.

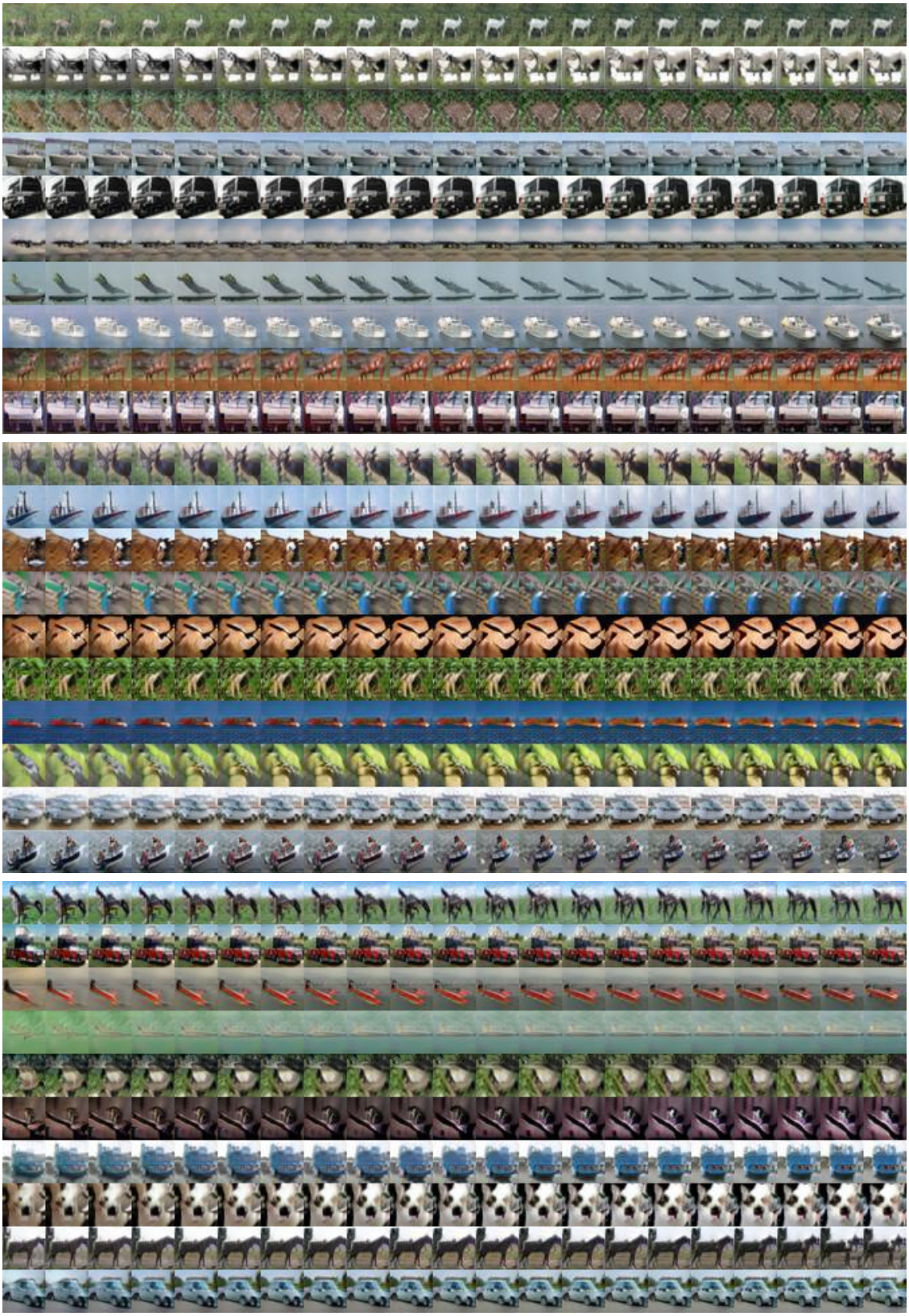


Fig. 2. CIFAR-10 Langevin dynamics visualization

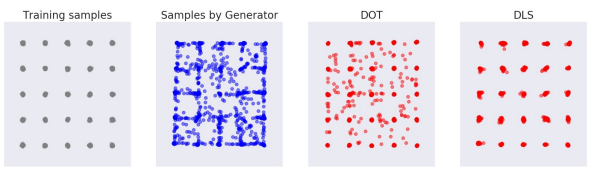


Fig. 3. DDLS, generator alone, and generator + DOT, on 2d mixture of Gaussians distribution

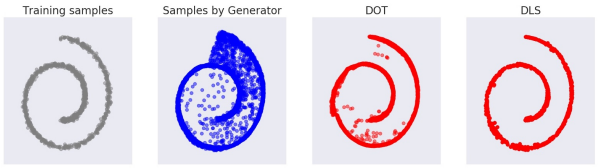


Fig. 4. DDLS, the generator alone, and generator + DOT, on the swiss roll dataset.

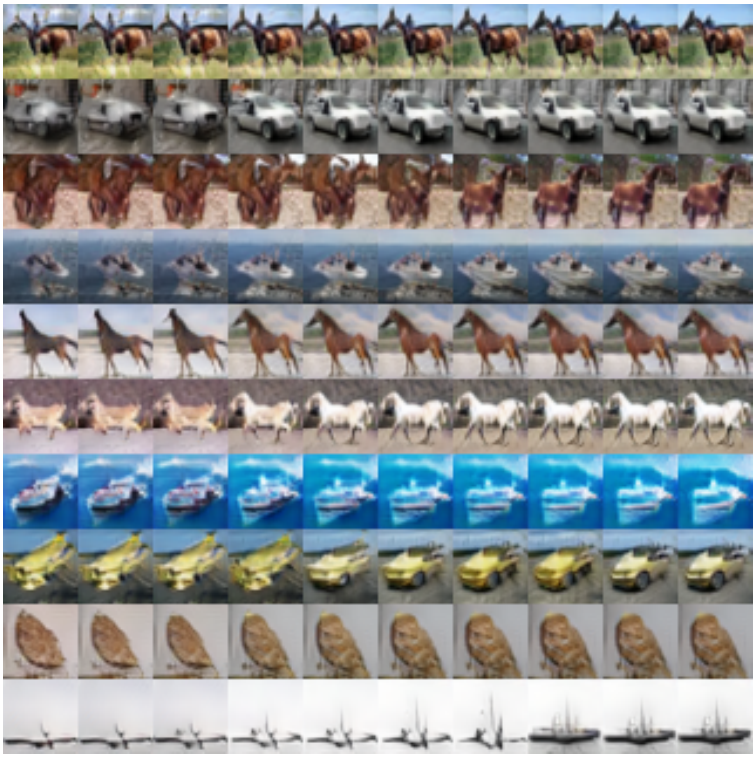


Fig. 5. CIFAR-10 Langevin dynamics visualization, initialized at a sample from the generator (left column). The latent space Markov chain appears to mix quickly, as evidenced by the diverse samples generated by a short chain. Additionally, the visual quality of the samples improves over the course of sampling, providing evidence that DDLS improves sample quality.

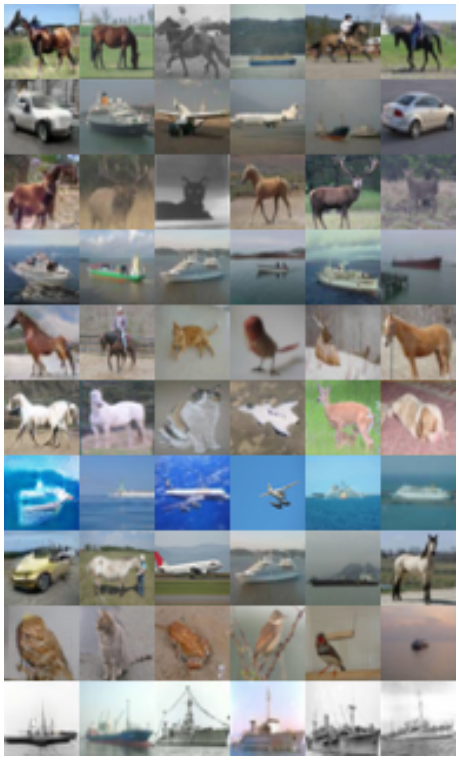


Fig. 6. Top-5 nearest neighbor images (right columns) of generated CIFAR-10 samples (left column).

Chapter 5

Application: Verifying Deep Discriminative Models with Deep Generative Models

5.1. Preface

5.1.1. My Role and Contributions In This Project

I proposed the idea for this project and worked out most of the details on the algorithms and experimental settings, and also worked on the paper writing. The experiments was carried out by the second, third and fourth authors. This project was published at AAAI 2021 [Che et al., 2021].

5.1.2. Motivation and Context

Deep learning models provide state-of-the-art performance in various applications such as image classification [Krizhevsky et al., 2012], caption generation [Xu et al., 2015], sequence modeling [Chung et al., 2014] and machine translation [Wu et al., 2016b]. However, such performance is based on the assumption that the training and test data are sampled from the same distribution [Goodfellow et al., 2016]. Without this assumption, deep learning models can fail silently by producing high confidence incorrect predictions even on completely unrecognizable or irrelevant inputs [Amodei et al., 2016]. For instance, the models trained on MNIST can produce 91% confidence on random noise images [Hendrycks and Gimpel, 2017]. Without additional assumptions, the behavior of a trained deep learning model on a slightly different test distribution is unpredictable. One such problematic case is also shown in Fig. 6.1. Unfortunately, there is very little control over the test distribution in real-world deployments due to dynamically changing environments or malicious attacks [Guo et al., 2017]. In fact, well calibrating the predictive uncertainty of DNNs is important for many production systems, *e.g.*, authentication devices, medical diagnosis and self-driving vehicles [Lee et al., 2018b].

Being overconfident on out-of-distribution (OOD) inputs has raised concerns about the safety of artificial intelligence (AI) systems. Recent research efforts try to address these concerns by developing models that can identify anomalous inputs, *i.e.*, OOD samples [Amodei et al., 2016]. Formally, the OOD detection problem can be formulated as a binary classification problem where the objective is to decide whether a test sample is from the training distribution (*i.e.*, in-distribution, ID) or from a different distribution (*i.e.*, OOD).

5.1.3. Main Contribution

In this chapter, we propose to verify the predictions of deep discriminative models by using deep generative models that try to generate the input conditioned on the label selected by the discriminative model. We call this concept "deep verifier". The high-level idea is simple: we train an inverse verification model $p(x|y)$ on the training data pairs (x,y). Intuitively speaking, for an input-output pair (x,y) with y picked by the predictive model, we verify whether the input x is consistent with y, by estimating if $p(x|y)$ is larger than a threshold. We design a density estimator of $p(x|y)$ using modified conditional VAEs. To ensure that the class code y is not ignored as a conditioning variable, we impose a disentanglement constraint based on minimizing mutual information between latent variable z and the label y. Although many different kinds of density estimators can be used in theory, we argue that the design of our model is robust to OOD samples and adversarial attacks, due to the use of latent variables with explicit and accurate density estimation.

Compared with previous approaches of OOD detection, our proposed method has four main advantages:

(1) The verifier is trained independently of OOD distributions. Users do not need to figure out OOD samples before deployment of the system.

(2) The verifier only needs to be trained once. No need to retrain the verifier for a different classifier trained on the same task.

(3) The verifier can detect ordinary OOD samples and malicious adversarial attacks in a unified manner.

(4) The framework is very general, so that it applies to structured prediction problems as well, such as image captioning.

We summarize the comparison of these four advantages with previous methods in Table 1.

The proposed solution achieves the state-of-the-art performance for detecting either OOD or adversarial samples in all tested classification scenarios, and can be generalized well for

	Hendrycks and Gimpel	Liang et al.	Devries and Taylor	Vyas et al.	Lee et al.	Choi et al.	Hendrycks et al.
1	√	×	×	×	×	×	×
2	√	×	×	×	√	×	×
3	×	×	×	×	√	×	×
4	×	×	×	×	×	×	×

Table 1. Summary comparison of the characteristics of the recent related methods.

structured prediction tasks (*e.g.*, image caption). In Sec 3.4, we analysed why DVN is useful for both OOD and Adversarial examples.

5.1.4. Follow-up Works and Impacts

As the first general framework to tackle both out-of-distribution and adversarial example detection, our DVN has a broad impact on these two challenging tasks [Ruff et al., 2021, Rahman et al., 2021]. The generative model for OOD detection has been further developed in [Chen et al., 2021]. In [Shao et al., 2020a], the authors investigate another form of confidence calibration and its corresponding neural network architecture, inspired by the framework proposed in this paper. The follow-up works [Yang and Ren, 2020, Shao et al., 2020b] propose to detect the adversarial attacks with an additional classifier, in addition to our DVN.

Our proposed method of measuring confidence has been applied in [Anil et al., 2021] to know when to trust the decisions made by machine learning systems. [Oneaţă et al., 2021] used it for word-level confidence estimation for end-to-end automatic speech recognition. [Schirrmeister et al., 2020] explained the performance of the inevitable network for OOD detection, which is investigated in our paper. In addition, our method also inspired the works on OOD generalization [Liang et al., 2021, Liu et al., 2021], which enable the deep learning networks to work well on the detected OOD data.

These show that our discovery has a real impact on the confidence calibration and its application on both out-of-distribution and adversarial example detection. Till today, the paper has been cited 36 times, this is an evidence of the real impacts brought by the paper.

5.2. Related Work

Detecting the OOD samples in a low-dimensional space using traditional non-parametric density estimation, nearest neighbor and clustering analysis have been well studied [Pimentel et al., 2014]. However, these methods are usually unreliable in high-dimensional complex spaces, *e.g.*, of images [Liang et al., 2018].

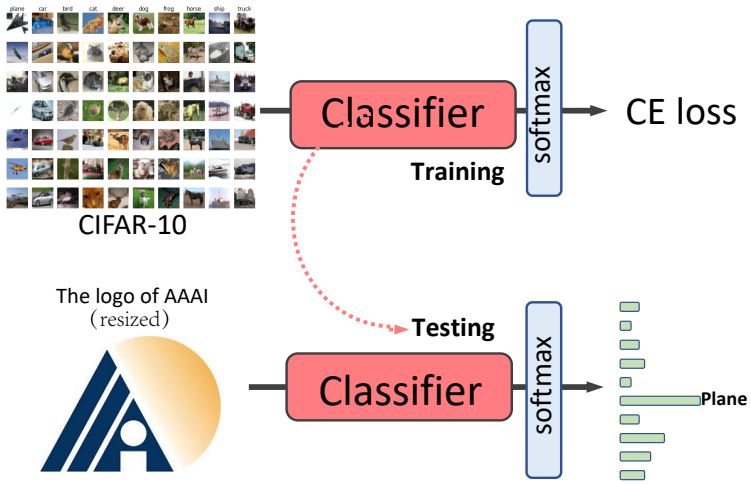


Fig. 1. A network trained on CIFAR-10 with ten-class softmax output will predict the resized 32x32x3 view of the AAAI logo (OOD sample w.r.t. CIFAR-10) as a plane with high confidence.

OOD detection with deep neural networks has recently been an active research topic. [Hendrycks and Gimpel, 2017] found that trained DNNs usually have higher maximum softmax output for in-distribution examples than anomalous one. A possible improvement of this baseline is to consider both the in-distribution and out-of-distribution training samples during training [Hendrycks et al., 2019]. However, enumerating all possible OOD distributions before deployment is usually not possible.

[Liang et al., 2018] proposed that the difference between maximum probabilities in softmax distributions on ID/OOD samples can be made more significant by using adversarial perturbation pre-processing during training. [Devries and Taylor, 2018] augmented the classifier with a confidence estimation branch, and adjusted the objective using the predicted confidence score for training. [Lee et al., 2018a] trained a classifier simultaneously with a GAN, with an additional objective to encourage low confidence on generated samples. [Hendrycks et al., 2019] proposed to use real OOD samples instead of generated ones to train the detector. [Vyas et al., 2018] labels a part of training data as OOD samples to train the classifier, and they dynamically change the partition of ID and OOD samples. These improvements based on [Hendrycks and Gimpel, 2017] typically need re-training a classifier with modified structures or optimization objectives. This can make it hard to maintain the original accuracy and is computationally expensive.

Recently, [Lee et al., 2018b] proposed a new framework for anomaly detection by first obtaining the class-conditional Gaussian distribution using Gaussian discriminative analysis,

and then define a confidence score using the Mahalanobis distance between the sample and the closest class-conditional Gaussian distribution. By modeling each class of in-distribution samples independently, it showed remarkable results for OOD and adversarial attack detection. Note however that their method also needs changing the input pre-processing and model. Besides, many previous methods [Liang et al., 2018, Vyas et al., 2018, Lee et al., 2018b] need OOD samples for hyper-parameter (*e.g.*, threshold for verification) selection, and these are usually not accessible in the real world.

Recently, [Choi et al., 2018] proposed an unsupervised OOD detector by estimating the Watanabe-Akaike Information Criterion, which is in turn estimated using an ensemble of generative models. The goal of our model is different from WAIC in that rather than just detecting OOD samples, DVNs aim to verify the predictions of a supervised predictive model, i.e., estimating $p(x|y)$ not just $p(x)$. We argue that modeling $p(x|y)$ is usually easier than directly modeling $p(x)$ as the former distribution contains less modes. Another motivation for modelling $p(x|y)$ instead of $p(x)$ is that for an adversarial attack and its classifier prediction (x',y'), $p(x')$ can be large because the adversarial input looks like a normal example by the eye, while $p(x'|y')$ is low (if the latter is compatible with the learned $p_\theta(y|x)$).

5.3. Methodology

This chapter targets the problem of verification of deep predictive models, as follows. Let $x \in \mathcal{X}$ be an input and $y \in \mathcal{Y}$ be the ground-truth value to be predicted. The in-distribution examples are sampled from the joint data-generating distribution $p_{\text{in}}(x,y) = p_{\text{in}}(y|x)p_{\text{in}}(x)$. We propose to reverse the order of the prediction process of $p(y|x)$ and try to compute the conditional probability $p(x|y)$, where y is the label value guessed by the classifier to be verified (e.g., the one with the highest probability according to the deep network). We evaluate whether the input x is consistent with that y.

The predictive model to be verified $p_\theta(y|x)$ is trained on a dataset set drawn from the $p_{\text{in}}(x,y)$, and may encounter samples from both $p_{\text{in}}(x,y)$ and $p_{\text{out}}(x,y)$ (*i.e.*, out-of-distribution or adversarial samples) at test time. Note there is some subtle difference between OOD (unlikely under $p_{\text{in}}(x)$) and adversarial examples (unlikely under the ground truth joint, but with high $p_{\text{in}}(x)$, especially if a small amount of noise is allowed).

Our goal is to verify if the pair (x,y) for y guessed by the predictive model given x is consistent with $p_{\text{in}}(x,y)$. We train a verifier network $q_\phi(x|y)$ as an approximation to the inverse posterior distribution $p(x|y)$. Modelling $p(x|y)$ instead of $p(x)$ as a verification has many advantages: (1) Usually $p(x)$ is much more diverse than the conditional distribution $p(x|y)$,

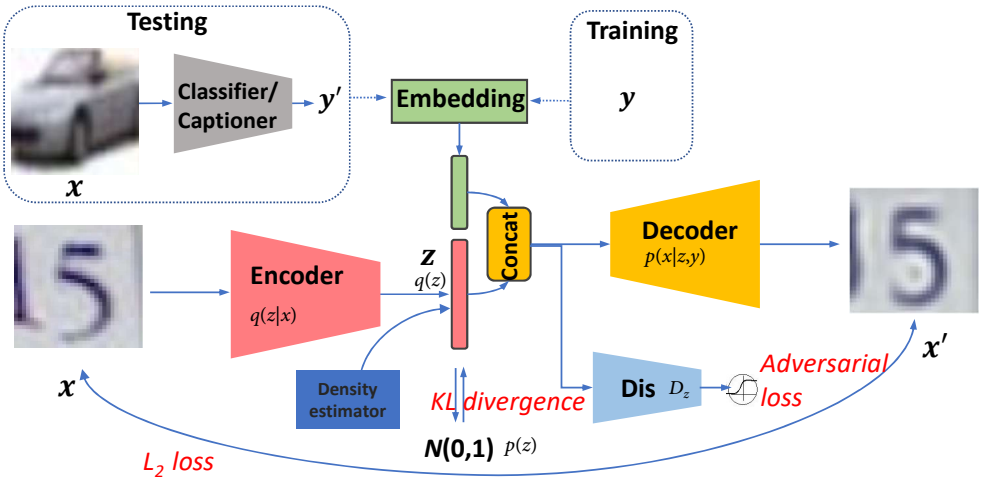


Fig. 2. The architecture of our proposed Deep Verifier Network (DVN). We use ground-truth label y of training example x during training while using the trained model prediction y' of testing image during testing.

so modelling $p(x|y)$ is much easier than modelling $p(x)$. (2) Modelling $p(x|y)$ allows us to provide a unified framework for verifying OODs, adversarial examples, and mis-classifications of the classifier.

5.3.1. Basic Model

Our basic model is a conditional variational auto-encoder shown in Fig. 2. The model is composed of two deep neural networks, a stochastic encoder $p_\phi(z|x)$ which takes input x to predict a latent variable z and a decoder $p(x|z,y)$ which takes both latent variable z and the label y to reconstruct x. One problem with training of conditional variational auto-encoders is that the decoder can ignore the effect of input label y, passing all information through the continuous latent variable z. This is not desirable as we want to use the decoder to model the conditional likelihood $p(x|y)$, not $p(x)$. Hence in this chapter, we train the encoder so that it outputs a z which is approximately independent of y. The encoder and decoder are thus jointly trained to maximize the evidence lower bound (ELBO):

$$\log p(x|y) \geq \mathbb{E}_{q(z|x)}[\log p(x|z,y)] - \mathrm{KL}(q(z|x)||p(z)) \tag{5.3.1}$$

The equality holds if and only if $q(z|x) = p(z|x,y)$, where $p(z|x,y)$ is the ground truth posterior. We note that the conditional GAN is not applicable here since its objective does not optimize the likelihood.

5.3.2. Disentanglement constraints for anomaly detection

To achieve this independence between z and y, we propose to add a disentanglement penalty to minimize the mutual information between z and y. Namely, besides the ELBO loss, we also minimize the mutual information estimator $\hat{I}(z,y)$ together with the loss, yielding:

$$L = -\mathbb{E}_{q(z|x)}[\log p(x|z,y) + \lambda \hat{I}(y,z)] + \mathrm{KL}(q(z|x)||p(z)) \tag{5.3.2}$$

In this chapter, we use deep Infomax [Hjelm et al., 2018] as the proxy for minimizing the mutual information (MI) between z and y. The mutual information estimator is defined as:

$$\hat{I}(z,y) = \mathbb{E}_{p(y,z)}[-s_+(-T(y,z))] - \mathbb{E}_{p(y)p(z)}[s_+(T(z,y))] \tag{5.3.3}$$

where s_+ is the softplus function and $T(y,z)$ is a discriminator network. Just like GAN discriminators, T is trained to maximize $\hat{I}(y,z)$, in order to get a better lower-bound estimation of the (JS-version) mutual information, while L (and in particular the encoder and decoder) is optimized (considering T fixed) to minimize $\hat{I}(y,z)$.

5.3.3. Measuring the likelihood as anomaly score

Our anomaly verification criterion is based on estimating the log-likelihood $\log p(x|y)$ for test samples. Importance sampling is a possible solution to provide an unbiased estimate of $p(x|y)$ when we have a VAE. Following IWAE [Burda et al., 2015], the k-sample importance weighting estimate of the log-likelihood is a lower bound of the ground truth likelihood $\mathcal{L}(x|y) = \mathbb{E}_{x\sim p(\cdot|y)}[\log p(x|y)]$:

$$\mathcal{L}_k(x|y) = \mathbb{E}_{z_1,\ldots,z_k\sim q(z|x)}[\log \frac{1}{k}\sum_{i=1}^{k} \frac{q(x,z_i|y)}{q(z_i|x)}]. \tag{5.3.4}$$

where $q(z)$ is a corrected density described below. We use the fact that $\mathcal{L}_k(x|y) \to \mathcal{L}(x|y)$ as $k \to \infty$ to estimate the likelihood. As will be discussed below, we want the decoder $q(x|z,y)$ be evaluated on the same input distribution for z as it is trained, which is not exactly the original Gaussian prior $p(z)$, so we will form a refined estimator of the prior, denoted $p*(z)$. The quantities $\mathcal{L}_k(x|y)$ form a monotonic series of lower bounds of the exact log-likelihood $\log p(x|y)$, with $\mathcal{L}_1 \leq \mathcal{L}_2 \leq \cdots \mathcal{L}_k \leq \log p(x|y))$. They have the property that when $k \to \infty$, $\mathcal{L}_k \to \log p(x|y)$. In our experiments we chose $k = 100$ for a good approximation of the exact likelihood,

In our algorithm, the distribution of z actually fed into decoder $p(x|z,y)$ during training is $q(z) = \int q(z|x)p_d(x)$. However, this distribution $q(z)$ can be drastically different from

the Gaussian prior $p(z)$. So instead of using the Gaussian $p(z)$ as a prior for the decoder network in Eq. 5.3.4, we use $q(z)$ and estimate the corrected likelihood of x under this directed generative model, as $p(x,z|y) = q(z)p(x|z,y)$. In order to estimate the density $q(z)$, we propose to train an additional discriminator D_z to distinguish $p(z)$ and $q(z)$. D_z is trained to discriminate the real distribution of latent variable $q(z) = \int p_d(x)e(z|x)dx$ ($p_d(x)$ is the data distribution of x, $e(z|x)$ is the encoder network) and Gaussian prior distribution $p(z)$, with ordinary GAN loss. Both $q(z)$ and $p(z)$ are easy to sample, so a discriminator is easy to train with the samples. In the GAN, the optimal discriminator D_z [Goodfellow, 2016] can be

$$D_z = \frac{p(z)}{p(z) + q(z)} \tag{5.3.5}$$

After D_z is trained (in theory optimally) and since $p(z)$ is known (*i.e.*, Gaussian), we can estimate $q(z) = \frac{1-D_z(z)}{D_z(z)}p(z)$.

We classify a sample x as an OOD sample if the log-likelihood is below the threshold δ and the x is an in-distribution sample, otherwise.

$$x \in \begin{cases} \text{in-distribution (ID), if } \ L_k \geq \delta \\ \text{out-of-distribution (OOD), otherwise} \end{cases} \tag{5.3.6}$$

We set δ to the threshold corresponding to 95% true positive rate (TPR), where the TPR refers to the probability of in-distribution validation samples being correctly verified as in-distribution. Therefore, the threshold selection in our model is only tuned on in-distribution validation datasets, while most previous methods also need OOD samples for hyper-parameter validation [Liang et al., 2018, Lee et al., 2018b]. We note that the distribution of OOD samples is usually not accessible before the system deployment.

5.3.4. Theoretical Justification

The loss function we optimize can be written as:

$$\begin{aligned} L =& L_1 + \lambda L_2 = \mathbb{E}_{x,y\sim p_d}[-\mathbb{E}_{q(z|x)}[\log p(x|z,y)] \\ &+ \mathrm{KL}(q(z|x)||p(z)) + \lambda \mathbb{E}_{q(z|x)}[\hat{I}(y,z)]] \end{aligned} \tag{5.3.7}$$

where $p(x|z,y)$ is the decoder we are training. In this section, we use the following convention. Symbol p means probability distributions induced by the decoder, and symbol q means

probability distributions induced by the encoder. Also denote p_d for real data distributions. Specifically, we define joint distribution $q(z,x,y) = q(z|x)p_d(x,y)$[1].

We have the following theorem that justifies the two parts of the above loss.

Theorem 5.3.1.

(i) $-L_1$ *is a variational lower bound of* $\mathbb{E}_{x,y\sim p_d}[\log p(x|y)]$. *The bound is tight when* q *is expressive enough and* z,y *are conditionally independent given* x.

(ii) If we have $I(y,z) = 0$, *where* $(y,z) \sim \mathbb{E}_{x\sim p_d}[p_d(y|x)q(z|x)]$ *(namely* $L_2 \approx 0$*), and assume that the decoder is perfect in the sense that* $p(x|y,z) = q(x|y,z)$, *then we have our evaluation metric* $\mathbb{E}_{z\sim q(z)}[p(x|y,z)] = p_d(x|y)$. *Namely, if* $I(y,z) = 0$, *and the decoder is trained optimally, , then no matter what the encoder looks like, the likelihood estimator we are using is* $\mathbb{E}_{z\sim q(z)}[p(x|y,z)]$, *which is equal to the ground truth likelihood. This justifies why we need loss* L_2.

Proof. For (i), we have:

$$\begin{aligned} -L_1 &= \mathbb{E}_{x,y\sim p_d}[\mathbb{E}_{q(z|x)}[\log p(x|z,y)] - \mathrm{KL}(q(z|x)||p(z))] \\ &= \mathbb{E}_{x,y\sim p_d}[\mathbb{E}_{q(z|x)}[\log p(x,z|y) - \log p(z|y)] \\ &\quad - \mathrm{KL}(q(z|x)||p(z))] \\ &= \mathbb{E}_{x,y\sim p_d}[\mathbb{E}_{q(z|x)}[\log p(x|y)] - \mathrm{KL}(q(z|x)||p(z|x,y))] \\ &\leq \mathbb{E}_{x,y\sim p_d}[\log p(x|y)] \end{aligned} \tag{5.3.8}$$

The bound is tight if $\mathbb{E}[\mathrm{KL}(q(z|x)||p(z|x,y))] = 0$, which is equivalent to $\mathbb{E}[\mathrm{KL}(q(z|x)||p(z|x))] = 0$ if z,y are conditionally independent give x.

For (ii), we have that if y,z are independent, namely $q(y,z) = p_d(y)q(z)$,we have: $q(x|y,z) = q(x,y,z)/q(y,z) = q(z|x)p_d(x,y)/p_d(y)q(z) = q(z|x)p_d(x|y)/q(z)$. So we have:

$$\begin{aligned} \mathbb{E}_{z\sim q(z)}[p(x|y,z)] &= \mathbb{E}_{z\sim q(z)}[q(x|y,z)] \\ &= \mathbb{E}_{z\sim q(z|x)}[p_d(x|y)] = p_d(x|y) \end{aligned}$$

□

5.3.5. Intuitive Justifications

We now present an intuitive justification for the above algorithm. First, consider the following part of our training loss:

$$L_1 = \mathbb{E}_{q(z|x)}[\log p(x|z,y)] - \mathrm{KL}(q(z|x)||p(z)) \tag{5.3.9}$$

[1]In this chapter we assume $q(z|x) = q(z|x,y)$, the motivation is during test time, y may be a wrong label, we don't want it to confuse the encoder.

It is well known that deep neural networks can generalize well for in-distribution samples, but their behavior out-of-distribution is less clear. Suppose x is an out-of-distribution sample, with y be the corresponding output of the classifier. Then the behavior of the stochastic encoder $q(z|x)$ is undefined. We denote $q(z) = \int q(z|x)p_d(x)$ the distribution to train $q(x|y,z)$. There are two cases: (1) $q(z|x)$ maps x to z with low density in $q(z)$. This case can be easily detected because $q(z)$ is easily computable. In this case the second term in Eq. 5.3.9 is a large negative number. (2) $q(z|x)$ maps x to z with high density in $q(z)$. Then since we train the decoder network with the input distribution $q(z)$ and because y and z are approximately independent, so (z,y) looks like an in-distribution input for decoder $p(x|z,y)$. Thus $p(x|y,z)$ should map to some in-distribution x' with class label y. Since input x is an OOD sample and reconstruction x' is an in-distribution sample, the reconstruction has to be bad. In this case, the first term in Eq. 5.3.9 is a large negative number. So in both cases, the log-likelihood score L_k derived from our model should be a large negative number. This is why our model is robust to both adversarial and OOD samples.

5.3.6. Can We Replace VAEs with Other Density Estimators?

In theory, we can use any other density estimator besides our modified conditional VAE (such as auto-regressive models and flow-based models) to estimate $p(x|y)$. However, our experiments and previous observations suggest that these other models may have drawbacks that would make them less suitable for this task. The comparisons with a DVN that is based on PixelCNN [Van den Oord et al., 2016] and Glow [Kingma and Dhariwal, 2018] are compared in Table 2, and they are consistently inferior to our VAE-based solution. Auto-regressive models are quite slow and may ignore the conditioning label y [Bowman et al., 2015]. Flow-based models were found to be less robust to adversarial examples, assigning higher likelihood on OOD samples than in-distribution samples [Nalisnick et al., 2018]. We have intuitively explained in Sec. 5.3.5 why our modified cVAE based model does not suffer from the same problem as flow-based models, thanks to our disentanglement regularizer, which relies on the existence of a latent space.

5.4. Experimental results

In this section, we demonstrate the effectiveness of the proposed DVN on several classification benchmarks, and show its potential for the image captioning task. We choose the DenseNet (Huang et al. 2017) and ResNet (He et al. 2016) architectures as the backbones

In-D	OOD	Validation on OOD samples			Validation on adversarial samples		
		TNR@TPR 95%	AUROC	Verification acc.	TNR@TPR 95%	AUROC	Verification acc.
		ODIN / SUF / **Our DVN** / Glow based DVN / Pixel CNN based DVN			ODIN / SUF /**Our DVN** / Glow based DVN / Pixel CNN based DVN		
C-10	SVHN	86.2/90.8/**92.4**/91.1/90.7	95.5/98.1/**99.0**/98.2/98.0	91.4/93.9/**95.1**/93.7/93.9	70.5/89.6/**91.2**/89.8/90.0	92.8/97.6/**98.1**/97.5/97.6	86.5/92.6/**94.2**/93.1/93.4
DenN	T-IN	92.4/95.0/**96.2**/95.1/94.8	98.5/98.8/**99.0**/98.4/98.2	93.9/95.0/**97.3**/96.4/96.6	87.1/94.9/**95.6**/94.7/94.3	97.2/98.8/**99.1**/98.8/98.6	92.1/95.0/**97.4**/96.5/96.2
	LSUN	96.2/97.2/**98.6**/97.5/97.3	99.2/**99.3**/**99.3**/98.9/98.9	95.7/96.3/**96.8**/96.2/96.0	92.9/97.2/**97.9**/97.2/97.3	98.5/99.2/**99.3**/98.7/98.8	94.3/96.2/**97.5**/96.6/96.3
C-100	SVHN	70.6/82.5/**85.2**/83.0/82.8	93.8/97.2/**97.3**/97.1/96.8	86.6/91.5/**93.4**/92.4/92.5	39.8/62.2/**70.5**/65.7/66.0	88.2/91.8/**92.2**/90.9/91.0	80.7/84.6/**86.3**/85.4/85.7
DenN	T-IN	42.6/86.6/**89.0**/86.4/86.5	85.2/**97.4**/**97.4**/96.8/95.6	77.0/92.2/**93.8**/91.8/92.0	43.2/87.2/**89.1**/88.5/88.5	85.3/97.0/**97.8**/96.9/96.4	77.2/91.8/**93.0**/92.3/92.0
	LSUN	41.2/91.4/**93.7**/92.5/93.1	85.5/98.0/**98.2**/97.6/97.5	77.1/93.9/**94.9**/93.0/93.2	42.1/91.4/**93.6**/91.8/92.0	85.7/97.9/**98.3**/97.9/97.8	77.3/93.8/**95.4**/94.2/94.6
SVHN	CFR-10	71.7/96.8/**97.4**/95.7/96.2	91.4/98.9/**99.2**/98.8/98.2	85.8/95.9/**96.5**/95.1/95.0	69.3/97.5/**97.8**/97.4/97.0	91.9/98.8/**99.1**/98.1/98.0	86.6/96.3/**97.4**/96.6/96.7
DenN	T-IN	84.1/99.9/**100**/98.3/98.0	95.1/**99.9**/**99.9**/98.5/98.4	90.4/98.9/**99.2**/98.0/97.7	79.8/**99.9**/**99.9**/96.4/98.3	94.8/99.8/**99.9**/96.7/97.1	90.2/98.9/**99.4**/97.6/97.7
	LSUN	81.1/**100**/**100**/98.7/98.5	94.5/**99.9**/**99.9**/97.9/98.2	89.2/99.3/**99.6**/98.8/98.4	77.1/**100**/**100**/98.2/98.5	94.1/99.9/**100**/96.8/96.5	89.1/99.2/**99.5**/97.2/98.1
C-10	SVHN	86.6/96.4/**98.4**/97.3/97.0	96.7/99.1/**99.2**/98.5/98.6	91.1/95.8/**97.3**/96.2/96.1	40.3/75.8/**78.5**/77.6/77.4	86.5/95.5/**96.1**/95.5/95.3	77.8/89.1/**92.2**/91.2/91.0
ResN	T-IN	72.5/97.1/**98.0**/97.0/96.9	94.0/99.5/**99.6**/98.5/98.5	86.5/96.3/**96.9**/94.7/94.9	96.6/95.5/**97.1**/96.2/95.9	93.9/99.0/**99.2**/98.3/98.1	86.0/95.4/**96.3**/95.5/95.1
	LSUN	73.8/98.9/**99.0**/97.6/97.7	94.1/**99.7**/**99.7**/97.8/97.5	86.7/97.7/**97.9**/96.3/96.0	70.0/98.1/**98.9**/96.8/96.5	93.7/**99.5**/**99.5**/97.6/97.7	85.8/97.2/**98.0**/96.8/96.5
C-100	SVHN	62.7/91.9/**93.5**/92.5/92.6	93.9/98.4/**98.8**/98.3/98.0	88.0/93.7/**94.8**/92.9/93.2	12.2/41.9/**46.2**/42.4/43.5	72.0/84.4/**86.3**/84.7/84.2	67.7/76.5/**79.4**/77.5/77.4
ResN	T-IN	49.2/90.9/**91.2**/90.6/90.3	87.6/98.2/**98.5**/98.0/97.7	80.1/93.3/**94.3**/93.0/93.1	33.5/70.3/**74.6**/72.2/71.8	83.6/87.9/**90.3**/86.6/86.5	75.9/84.6/**89.8**/85.2/85.8
	LSUN	45.6/90.9/**92.3**/89.5/88.8	85.6/98.2/**98.6**/96.7/97.0	78.3/93.5/**95.7**/93.9/93.7	31.6/56.6/**63.5**/60.2/60.1	81.9/82.3/**85.2**/82.9/82.8	74.6/79.7/**81.9**/80.0/78.9
SVHN	C-10	79.8/98.4/**99.4**/97.9/97.5	92.1/99.3/**99.9**/98.1/98.2	89.4/96.9/**97.5**/96.3/96.3	79.8/94.1/**94.5**/93.7/93.5	92.1/97.6/**98.7**/96.5/96.2	89.4/94.6/**94.8**/93.7/93.0
ResN	T-IN	82.1/99.9/**100**/98.5/98.4	92.0/**99.9**/**99.9**/96.3/96.5	89.4/99.1/**99.2**/95.8/96.7	80.5/99.2/**99.7**/98.5/98.3	92.9/99.3/**99.5**/97.2/97.0	90.1/98.8/**99.3**/98.0/98.2
	LSUN	77.3/**99.9**/**99.9**/96.4/96.4	89.4/**99.9**/**99.9**/97.6/97.4	87.2/99.5/**100**/99.0/98.9	76.3/**99.9**/**99.9**/96.5/97.4	90.7/**99.9**/99.8/96.8/96.7	88.2/99.5/**99.8**/98.3/98.1

Table 2. OOD verification results of image classification under different validation setups. All metrics are percentages and the best results are in bold. The backbone classifier in SUF [Lee et al., 2018b] and our DVN is ResNet34 [He et al., 2016], while ODIN [Liang et al., 2018] use more powerful wide ResNet 40 with width 4 [Zagoruyko and Komodakis, 2016].

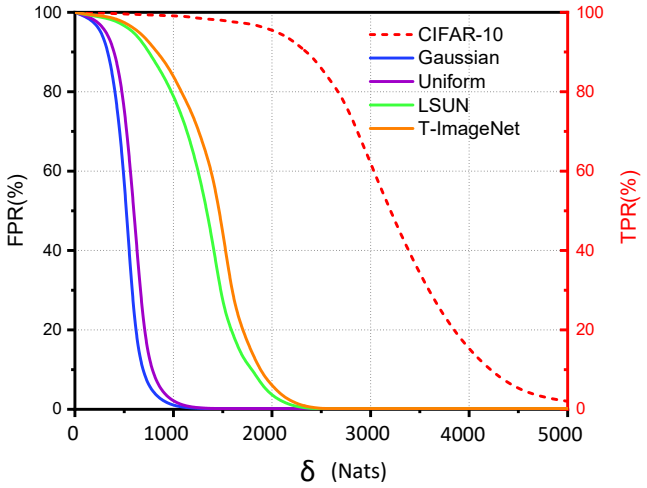


Fig. 3. FPR (for OOD) and TPR (for ID) under different δ when using CIFAR-10 as the in-distribution dataset, and use Tiny-ImageNet(resize), LSUN and Gaussian/Uniform noise as OOD. CIFAR-10 only applicable to the TPR which use the dashed red line and indicated by the right axis while the other OOD datasets use the left FPR axis.

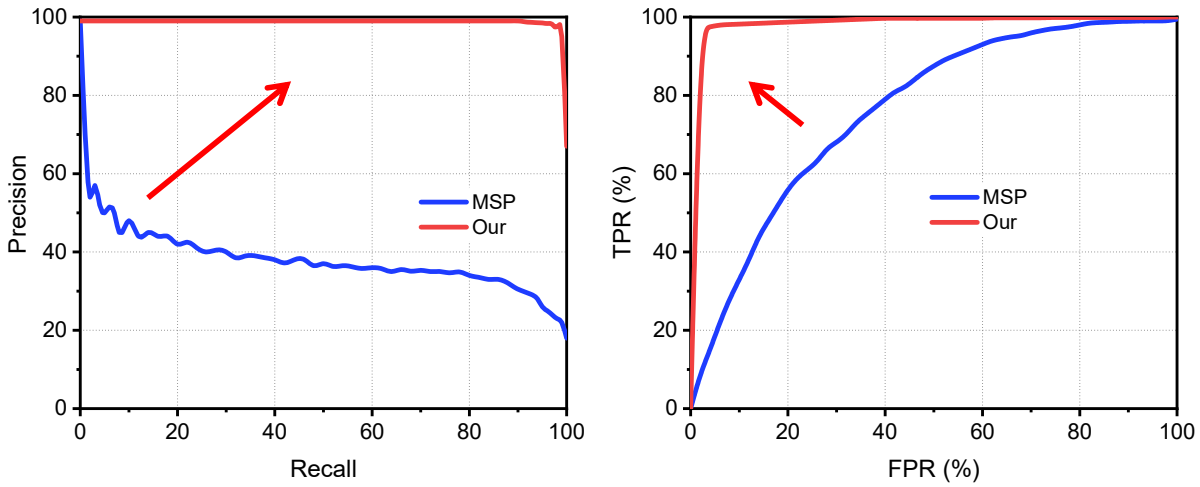


Fig. 4. Comparison with baseline MSP [Hendrycks and Gimpel, 2017] using DenseNet, with Tiny-ImageNet as in-distribution and LSUN as OOD.

of our experiments. Following the definitions in previous work, we denote the in-distribution images as positive samples, while the OOD ones as the negative samples.

For evaluation, we measure the True Negative Rate or False Positive Rate at 95% True Positive Rate (i.e., TNR@TPR95% or FPR@TPR95%), Area under the receiver operating

Table 3. Comparison of AUROC (%) under different validation setups. The best results are in bold. We also compared the use of negative samples for training and input image pre-processing.

	Dataset	Method	Negative Sample	Pre-proce	Deep Fool	CW	BIM		Dataset	Method	Negative Sample	Pre-proce	Deep Fool	CW	BIM
DenseNet	CIFAR -10	KD+PU	FGSM	-	68.34	53.21	3.10	ResNet	CIFAR -10	KD+PU	FGSM	-	76.80	56.30	16.16
		LID	FGSM	-	70.86	71.50	94.55			LID	FGSM	-	71.86	77.53	95.38
		SUF	FGSM	Yes	87.95	83.42	99.51			SUF	FGSM	Yes	78.06	93.90	98.91
		Our	-	-	**96.14**	**96.38**	**99.82**			Our	-	-	**95.45**	**99.51**	**99.57**
	CIFAR -100	KD+PU	FGSM	-	65.30	58.08	66.86		CIFAR -100	KD+PU	FGSM	-	57.78	73.72	68.85
		LID	FGSM	-	69.68	72.36	68.62			LID	FGSM	-	63.15	75.03	55.82
		SUF	FGSM	Yes	75.63	86.20	98.27			SUF	FGSM	Yes	81.95	90.96	96.38
		Our	-	-	**97.01**	**98.55**	**99.94**			Our	-	-	**97.22**	**99.38**	**99.72**
	SVHN	KD+PU	FGSM	-	84.38	82.94	83.28		SVHN	KD+PU	FGSM	-	84.30	67.85	43.21
		LID	FGSM	-	80.14	85.09	92.21			LID	FGSM	-	67.28	76.58	84.88
		SUF	FGSM	Yes	93.47	96.95	99.12			SUF	FGSM	Yes	72.20	86.73	95.39
		Our	-	-	**98.14**	**99.35**	**100.00**			Our	-	-	**97.13**	**99.76**	**100.00**

characteristic curve (AUROC), Area under the precision-recall curve (AUPR) and Verification accuracy. Noticing that AUROC, AUPR and verification accuracy are threshold (δ)-independent evaluation metrics.

5.4.1. Detecting OOD samples for classification

Datasets. The Street View Housing Numbers (**SVHN**) dataset (Netzer et al. 2011) consists of color images depicting house numbers, which range from 0 to 9. Images have a resolution of 32×32. For our tests, we use the official training set split which contains 73,257 images, and the test set split, which has 26,032 images. The **CIFAR-10/100** dataset (Krizhevsky and Hinton. 2009) consists of 10/100 classes colour images. The training set has 50,000 images, while the test set has 10,000 images. The dataset[2] is a subset of the ImageNet dataset (Deng et al. 2009). Its test set contains 10,000 images from 200 different classes. It contains the original images, downsampled to 32×32 pixels. The Large-scale Scene UNderstanding dataset (**LSUN**) (Yu et al. 2015) has a test set with 10,000 images from 10 different classes. The LSUN (crop) and LSUN (resize) are created in a similar downsampling manner to the TinyImageNet datasets. The **Uniform noise** and **Gaussian noise** dataset are with 10,000 samples respectively, which are generated by drawing each pixel in a 32×32 RGB image from an i.i.d uniform distribution of the range [0, 1] or an i.i.d Gaussian distribution with a mean of 0.5 and variance of 1 [Liang et al., 2018].

Setups. For fair comparisons, the backbones of the classifiers used here are the same 100-layer DenseNet with growth rate 12 [Liang et al., 2018, Lee et al., 2018b] and 34-layer

[2]https://tiny-imagenet.herokuapp.com/

Table 4. Test error rate of classification on CIFAR-10/100 using DenseNet as backbone. Our DVN does not re-train or modify the structure of the original trained classifier.

	CIFAR-10	CIFAR-100
ODIN/SUF	4.81	22.37
DenseNet/DVN	**4.51**	**22.27**

ResNet [Lee et al., 2018b]. They are trained to classify the SVHN, CIFAR-10, CIFAR-100 and Tiny-ImageNet datasets, whose test set is regarded as the in-distribution dataset in our testing stage. The dataset that is of a different type from the training dataset is considered as OOD. We use four convolution or deconvolution layers for the encoder and decoder structure, and z is a 128-dimension vector. The discriminator is a two-layer fully connected layer network with sigmoid output and binary cross-entropy loss. The hyper-parameters in previous methods [Liang et al., 2018, Lee et al., 2018b] need to be tuned on a validation set with 1,000 images from each in-distribution and OOD pair. We note that the threshold of the DVN is tuned on in-distribution only. This corresponds to a more realistic scenario, since the OOD nature of real-world applications is usually uncontrollable.

Effects of the threshold and performance across datasets. How the hyper-parameters (*e.g.*, δ) generalize across different OOD datasets is a challenging aspect of the system deployment. Most of the previous methods require a small set of OOD samples, with δ calibrated by evaluating the verification error at different values of δ. However, the more realistic scenario is that we do not have access to the OOD examples in the testing stage. A promising trend is improving the performance on an unknown OOD when using the model tuned on a similar OOD [Liang et al., 2018, Lee et al., 2018b]. We argue that our DVN is essentially free from such worries, since it does not need any OOD sample in the validation. To investigate how the threshold affects the FPR and TPR, Fig. 3 shows their relationship when training on CIFAR-10 and different OOD datasets are used in the test stage, with a DenseNet backbone. Note that the TPR (red axis) is used for in-distribution dataset CIFAR-10 (red dashed line), while FPR is used for OODs. We can observe that the threshold corresponding to 95% TPR can produce small FPRs on all OOD datasets. When the OOD images are sampled from some simple distributions (*e.g.*, Gaussian or Uniform), the available window of threshold δ can be larger.

Comparison with SOTA. The main results are summarised in Tab. 2. For each in- and out-of-distribution pair, we report the performance of ODIN [Liang et al., 2018], SUF [Lee et al., 2018b] and our DVN. Notably, DVN consistently outperforms the previous

Table 5. The performance of DVN w/o disentanglement of y from z with ResNet backbone, and using CIFAR-10/SVHN as in-distribution/OOD, respectively.

Disentangle	TNR@TPR95%	AUROC
$\surd$	**98.4**	**99.2**
-	62.6	84.7

methods and achieves a new state-of-the-art. As shown in Tab. 3, the pre-processing and model change in ODIN and SUF increase the error rate of the original classifier for the in-distribution test, while DVN does not affect the classification accuracy on the accepted in-distribution datasets (i.e., CIFAR-10 or 100), when the OOD examples are not presented.

Considering the technical route of DVN is essentially different from ODIN and SUF, we compare it with the baseline, maximum softmax probability (MSP) [Hendrycks and Gimpel, 2017], w.r.t. ROC and PR in Fig. 4. DVN shares some nice properties of MSP, *e.g.*, fixed classifier and single forward pass at the test stage. Moreover, DVN outperforms MSP by a large margin.

Ablation studies. To demonstrate the effectiveness of each module, we provide the detailed ablation study w.r.t. the choice of VAE/PixelCNN/Glow, disentanglement of y, modifying $p(z)$ to $q(z)$ with GAN and conditioned encoder.

• PixelCNN/Glow-based DVN. We also compared with the DVN that use pixel CNN or Glow rather than VAE as shown in Table 2. The pixelCNN/Glow-based DVN is consistently inferior than our solution. VAEs do have lower likelihood scores than Glow, but this gap is due to the different ways of computing likelihood of VAEs and flows. When computing the likelihood of a VAE, it is usually assumed that there is a unit Gaussian distribution at the output of the decoder. However the distribution of natural images is on a low dimensional manifold, so the likelihood number itself cannot be compared with Glow under this assumption. But VAEs are more robust than Glow due to the reason discussed in Sec 3.5, and in our experiments we found that Glows tend to put higher likelihood on OOD examples, which is bad for our usage.

• Disentangling y from z is critical to our model. Table 5 validates the contribution of this manipulation w.r.t. both threshold dependent and independent metrics. One can see that the DVN with disentanglement significantly outperforms its counterparts without disentanglement. This also implies the DVN has successfully learned to sufficiently minimize the mutual information between z and y to circumvent the challenge of conditioning x on y.

Table 6. The performance of DVN w/o replace $p(z)$ with $q(z)$. We use ResNet backbone, and choose CIFAR-10/SVHN as in-distribution/OOD.

$q(z)$	TNR@TPR95%	AUROC
$\surd$	**98.4**	**99.2**
-	95.3	96.7

Table 7. The performance of DVN use $q(z|x)$ and $q(z|x,y)$ encoder. We use ResNet backbone, and choose CIFAR-10/SVHN as in-distribution/OOD.

	TNR@TPR95%	AUROC
$q(z\|x)$	**98.4**	**99.2**
$q(z\|x,y)$	93.7	95.5

• Without modifying $p(z)$ with $q(z)$. Since modeling $p(x|y)$ is the core of DVN, we cannot remove y. Here, we give another ablation study that without modifying $p(z)$ with $q(z)$. As shown in Table 6, there is a large margin between the DVN with or without disentanglement w.r.t. TNR@TPR95 and AUROC. The results demonstrate that disentangle y from z is of essential important for our framework.

• Encoder condition on y. In this chapter we assume $q(z|x) = q(z|x,y)$, the motivation is during test time, y may be a wrong label, we don't want it to confuse the encoder. Table 7 gives a comparison of conditioning our encoder on x or (x,y).

5.4.2. Detecting adversarial examples

To detect adversarial examples, we train our DenseNet and ResNet-based classification network and DVN using the training set of CIFAR-10, CIFAR-100 or SVHN datasets, and their corresponding test sets are used as the positive samples for the test. Following the setting in [Lee et al., 2018b], we applied several attack methods to generate the negative samples, such as basic iterative method (BIM) [Kurakin et al., 2016], Deepfool [Moosavi-Dezfooli et al., 2016], and Carlini-Wagner (CW) [Carlini and Wagner, 2017]. The network structures are the same as for OOD verification.

We compare the DVN with the strategies in KD+PU [Feinman et al., 2017], LID [Ma et al., 2018], SUF [Lee et al., 2018b] in Tab. 4, and show that the DVN can achieve the state-of-the-art performance in most cases w.r.t. AUROC. In the "detection of unknown

Table 8. OOD verification results of image caption under different validation setups. We use CUB-200, LSUN and COCO as the OOD of Oxford-102, while using Oxford-102, LSUN and COCO as OOD of CUB-200.

In-Dist	OOD	Validation on OOD samples		
		TNR@TPR 95%	AUROC	Verif acc.
Oxford	CUB	55.6	72.3	79.5
	LSUN	50.5	71.8	76.2
	COCO	40.3	74.4	73.3
CUB	Oxford	39.8	68.4	72.5
	LSUN	36.3	65.4	69.5
	COCO	35.4	60.7	71.0

attack setting", we can not access the adversarial examples of the test stage in the training or validation. Therefore, the previous works choose to use another attack generation method, *i.e.*, fast gradient sign method (FGSM) [Goodfellow et al., 2014d], to construct a validation set of adversarial examples. In here, we do not need another attack method as a reference, since the threshold of the DVN is only related to the validation set of in-distribution samples. Moreover, the pre-processing and model change as in [Lee et al., 2018b] are not required in our proposed DVN.

5.4.3. OOD for image captioning

For detecting OOD samples in the image captioning task, we choose Oxford-102 and CUB-200 as the in-distribution datasets. Oxford-102 contains 8,189 images of 102 classes of flower. CUB-200 contains 200 bird species with 11,788 images. Each of them has 10 descriptions that are provided by [Reed et al., 2016a]. For these two datasets, we use 80% of the samples to train our caption generator, and the remaining 20% for testing in a cross-validation manner. The LSUN and Microsoft COCO datasets are used as our OOD dataset.

The captioner used in here is a classical image caption model [Xu et al., 2015]. We choose the generator of GAN-INT-CLS [Reed et al., 2016b] as our decoder's backbone, and replace its Normal distribution vector as the output of encoder z. A character-level CNN-RNN model [Reed et al., 2016a] is used for the text embedding which produces the 1,024-dimension vector given the description, and then projected to a 128-dimension code c. We configure the encoder and decoder with four convolutional layers and the latent vector

z is a 100-dimension vector. The input of the discriminator is the concatenation of z and c, which results in a 228-dimension vector. A two-layer fully connected network with sigmoid output unit is used as the discriminator. Tab. 8 summarizes the performance of DVN in image caption task and can be regarded as a powerful baseline.

5.5. Conclusions

In this chapter, we propose to enhance the performance of anomaly detection by verifying predictions of deep discriminative models using deep generative models. The idea is to train a conditional verifier network $q(x|y)$ as an approximation to the inverse posterior distribution. We propose Deep Verifier Networks (DVNs) which are based on a modified conditional variational auto-encoders with disentanglement constraints. We show our model is able to achieve state-of-the-art performance on benchmark OOD detection and adversarial example detection tasks.

Chapter 6

Future Works and Summary of Contributions

6.1. Overall Conclusions

In this thesis, we made several contributions surrounding the topic of generative models, including an algorithm to stabilize training of discrete generative models, better training of auto-regressive generative models by combining them with energy-based models, a better way to sample from generative adversarial networks, and two applications including a connection of energy-based models with residual networks, and a way to verify the predictions of discriminative models with deep generative models.

6.2. Future Developments of Generative Models

There are many important future directions in the field of generative models following up on the work in this thesis. I will list my favorite directions here and discuss future research possibilities.

6.2.1. Better Density Estimation

Currently, we have several methods for density estimation. For example, auto-regressive models [van den Oord et al., 2016a, Kolesnikov and Lampert, 2017, Germain et al., 2015] and flow-based models [Dinh et al., 2016, Danihelka et al., 2017]. However, both methods currently have major drawbacks. Auto-regressive density estimators can be extremely computational expensive. What's more, they are better at modeling local textures rather than global contents and consistency, because of the well-known difficulties to capture long-term dependencies. On the other hand, flow-based models have recently been proven not satisfying for capturing the likelihood of out-of-distribution samples [Nalisnick et al., 2019]. So there remains a very interesting open problem of how to train good density estimators on complex datasets. Fundamentally new ideas and methods are needed to address these problems.

6.2.2. Generating High Quality Samples and Modeling the Distribution Well

There seems to be a trade-off between generating high quality samples and modeling well the entire distribution with generative models. For instance, GAN-based models [Goodfellow et al., 2014c] are very good at generating high-quality samples but they are not able to model the entire distribution very well, probably leaving out some of its modes. Even the state-of-the-art GAN-based models are prone to drop modes, according to so-called collision analysis. On the other hand, autoencoder-based generative models [Kingma and Welling, 2013] and auto-regressive models [van den Oord et al., 2016a] are very good at modeling the entire distribution, however their sample quality is not as good. It is thus desirable to design the next generation of generative models which are able to achieve both.

Nowadays, diffusion-based generative models [Song et al., 2020b, Dhariwal and Nichol, 2021] seem to able to achieve both goals. But they are very expensive to train, as they try to train a diffusion process in the pixel space. It is an interesting research direction to see if we can combine benefits of different generative models, including diffusion generative models and avoid all their weaknesses.

6.2.3. Learning Disentangled/Causal Representations

Learning of disentangled representations [Bengio, 2013] aims to separate different factors of variations in the data. In many real-world datasets, such as images and natural language, it is assumed that each data point is generated by some latent causes. How to find the latent causes of data without supervision is a major challenge for unsupervised learning. Recently, the problem of learning disentangled representations has become a very popular topic. InfoGAN [Chen et al., 2016] and Beta-VAE are two typical models which try to produce good disentangled representations in a certain sense (of independent/causal latent factors). However, high-level variables manipulated by humans to understand their environment are not independent, and disentanglement is still an ambiguous concept leaving many questions open.

My proposed approach is to explore a variety of inductive biases which can encourage disentanglement. Because learning disentangled representations can only make sense on several special kinds of data, e.g. natural images, so incorporating good inductive biases is very important.

6.2.4. Learning Interpretable Hierarchical Models

It is widely believed that many complex datasets must be modelled with a hierarchy of latent variables. Each layer of latent variables corresponds to one layer of abstraction. However, currently the most successful generative models have only one layer of latent variables. It is thus very interesting to ask whether there are models which can naturally capture hierarchical structure in the datasets. An important class of temporal generative models is those with hierarchical temporal abstractions. These models are useful in modelling data with temporal structure, such as videos and time sequences. These models are essential for applications in agent learning [Wayne et al., 2018], for example model-based reinforcement learning. One difficulty for these models is that with we are not currently able to model the long-term future. By using temporal abstractions, we should be able to design new temporal generative models which are able to generate at different scales of time. Thus there is a very important open problem of how to design this kind of temporal generative model with temporal abstractions.

One direction that has already been explored somewhat is to try to learn hierarchical latent variable models that update their latent variables at different speeds at each layer. For example, one can think of imposing structures like Ordered LSTM in latent variable models.

6.2.5. Large-scale Generative Modelling for Foundations in AI

Recently, it is widely accepted that training large-scale generative models on Internet-scale datasets could be a key ingredient for next-generation AI systems[Bommasani et al., 2021]. For example, large-scale language models including BERT[Devlin et al., 2018], GPT-3[Brown et al., 2020], T5[Raffel et al., 2019] are widely used as prototypes for down-stream tasks. Multimodal large-scale generative models such as DALL-E-2 [Ramesh et al., 2022] can handle both natural languages and images. The emergence of these large-scale generative models are one of the most exciting developments of modern AI. These large-scale generative models have demonstrated ability to generalize to many down-stream tasks or sometimes, to out-of-distribution data.

A core future research direction is to see what kind of inductive biases is needed for these large-scale generative models, which can satisfy three goals: (1) Further boost generalization performance on out-of-distribution datasets. (2) Do not hurt model capacity and scalability to learn from large-scale datasets. (3) Can help reducing parameters and increase parameter efficiency of large-scale models. My current work in this direction including trying to boost

the generalization performance of large scale vision and NLP transformer based models via adding the inductive bias that the world is composed of different objects/concepts.

Besides the above research direction, another research topic which is of considerable importance is to find ways to better embed foundation models in down stream tasks and AI decision systems. Since foundation models are the most powerful deep learning models we have, it is natural to think about how to exploit their generalization power to boost the performance of many important down-stream tasks. As an instance, I am exploring how to use foundation models to boost exploration performance in reinforcement learning.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042, 2016.

Cem Anil, Guodong Zhang, Yuhuai Wu, and Roger Grosse. Learning to give checkable answers with prover-verifier games. *arXiv preprint arXiv:2108.12099*, 2021.

Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Zbc-ue9p_rE`.

Michael Arbel, Liang Zhou, and Arthur Gretton. Kale: When energy-based learning meets adversarial training, 2020.

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *CoRR*, abs/1701.04862, 2017.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017a.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *CoRR*, abs/1701.07875, 2017b.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017c.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, 2018.

Anton Bakhtin, Yuntian Deng, Sam Gross, Myle Ott, Marc'Aurelio Ranzato, and Arthur Szlam. Residual energy-based models for text. *J. Mach. Learn. Res.*, 22:40:1–40:41, 2021.

Richard Ernest Bellman. *The Theory of Dynamic Programming*. RAND Corporation, Santa Monica, CA, 1954.

Samy Bengio and Yoshua Bengio. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557, 2000.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179, 2015a.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015b.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.

Yoshua Bengio. Deep learning of representations: looking forward. In *Statistical Language and Speech Processing*, volume 7978 of *Lecture Notes in Computer Science*, pages 1–37. Springer, also in arXiv at http://arxiv.org/abs/1305.0445, 2013.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13, 2000.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan,

Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021. URL https://arxiv.org/abs/2108.07258.

Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgza6VtPB.

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

George Casella, Christian P. Robert, and Martin T. Wells. Generalized accept-reject sampling schemes. *Lecture Notes-Monograph Series*, 45:342–347, 2004. ISSN 07492170. URL http://www.jstor.org/stable/4356322.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv preprint arXiv:2003.06060*, 2020b.

Tong Che, Xiaofeng Liu, Site Li, Yubin Ge, Ruixiang Zhang, Caiming Xiong, and Yoshua Bengio. Deep verifier networks: Verification of deep discriminative models with deep generative models, 2021.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2014.

Wenxiao Chen, Xiaohui Nie, Mingliang Li, and Dan Pei. Doi: Divergence-based out-of-distribution indicators via deep generative models. *arXiv preprint arXiv:2108.05509*, 2021.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180, 2016.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 863–871, 2018.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

Hyunsun Choi, Eric Jang, and Alexander A Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*, 2015.

Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. In *Advances in Neural Information Processing Systems*, pages 10977–10988, 2019a.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988, 2019b.

Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, and Peter Dayan. Comparison of maximum likelihood and gan-based training of real nvps. *CoRR*, abs/1705.05263, 2017.

Cyprien de Masson d'Autume, Shakir Mohamed, Mihaela Rosca, and Jack Rae. Training language gans from scratch. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/a6ea8471c120fe8cc35a2954c9b9c595-Paper.pdf`.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *International Conference on Learning Representations*, 2020a. URL `https://openreview.net/forum?id=B1l4SgHKDH`.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Terrance Devries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks. 2018.

Terrance DeVries, Michal Drozdzal, and Graham W Taylor. Instance selection for gans. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13285–13296. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/99f6a934a7cf277f2eaece8e3ce619b2-Paper.pdf`.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL `https://arxiv.org/abs/2105.05233`.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, pages 3603–3613, 2019.

Conor Durkan and Charlie Nash. Autoregressive energy machines. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pages 1735–1744. PMLR, 2019.

S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models, 2016.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.

Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7518–7528, 2020.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. Learning energy-based models by diffusion recovery likelihood. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 881–889. JMLR.org, 2015.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014a.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014c.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014d.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Aditya Grover, Ramki Gummadi, Miguel Lazaro-Gredilla, Dale Schuurmans, and Stefano Ermon. Variational rejection sampling. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 823–832, Playa Blanca, Lanzarote, Canary Islands, 2018. PMLR. URL `http://proceedings.mlr.press/v84/grover18a.html`.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11058–11070. Curran Associates, Inc., 2019.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 297–304, 2010.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

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

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

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *ICLR*, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

G. Hinton, P. Dayan, B. Frey, and R. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 5214:1158–61, 1995.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, aug 2002a. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL https://doi.org/10.1162/089976602760128018.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, 2002b.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002c.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, jul 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL https://doi.org/10.1162/neco.2006.18.7.1527.

R Devon Hjelm, Athul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. 2017.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554. URL https://www.pnas.org/content/79/8/2554.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

C. Igel. Policy gradient methods for reinforcement learning with function approximation. 2005.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJa9iHgAZ.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=Hk99zCeAb`.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *CoRR*, abs/1606.03439, 2016a.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016b.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

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

Alexander Kolesnikov and Christoph H. Lampert. Pixelcnn models with auxiliary variables for natural image modeling. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1905–1914. PMLR, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Rithesh Kumar, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau, Ewa Kijak, Jacopo Staiano, and Benjamin Piwowarski. Generative cooperative networks for natural language generation, 2022.

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

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *ICLR*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *NIPS*, 2018b.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

Kaiqu Liang, Cem Anil, Yuhuai Wu, and Roger Grosse. Out-of-distribution generalization with deep equilibrium models. *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *ICLR*, 2018.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. Optimization of image description metrics using policy gradient methods. *CoRR*, abs/1612.00370, 2016.

Xiaofeng Liu, Bo Hu, Linghao Jin, Xu Han, Fangxu Xing, Jinsong Ouyang, Jun Lu, Georges EL Fakhri, and Jonghye Woo. Domain generalization under conditional and label shifts via variational bayesian inference. *arXiv preprint arXiv:2107.10931*, 2021.

Ryan Lowe, Michael Noseworthy, Iulian Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. 2017.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3698–3707, 2018.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. *CoRR*, abs/1511.06038, 2016.

Tsvetomila Mihaylova and André F. T. Martins. Scheduled sampling for transformers. In Fernando Emilio Alva-Manchego, Eunsol Choi, and Daniel Khashabi, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2*, pages 351–356, 2019.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018b. URL `https://openreview.net/forum?id=B1QRgziT-`.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

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

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1xwNhCcYm`.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34, 2021.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems*, pages 5233–5243, 2019.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.

Dan Oneaţă, Alexandru Caranica, Adriana Stan, and Horia Cucu. An evaluation of word-level confidence estimation for end-to-end automatic speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 258–265. IEEE, 2021.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. *Advances in Neural Information Processing Systems*, 33, 2020.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Marco A. F. Pimentel, David A. Clifton, Clifton Lei, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99(6):215–249, 2014.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional gans. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL `http://arxiv.org/abs/1910.10683`.

Quazi Marufur Rahman, Peter Corke, and Feras Dayoub. Run-time monitoring of machine learning for robotic perception: A survey of emerging trends. *IEEE Access*, 9:20067–20075, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL `https://arxiv.org/abs/2204.06125`.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016a.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016b.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016a.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016b.

Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

David E. Rumelhart and James L. McClelland. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pages 194–281. 1987.

Masaki Saito and Eiichi Matsumoto. Temporal generative adversarial nets. *arXiv preprint arXiv:1611.06624*, 2016.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009a. PMLR. URL `https://proceedings.mlr.press/v5/salakhutdinov09a.html`.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009b.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 872–879. ACM, 2008.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016a.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016b. URL `http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf`.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Robin Schirrmeister, Yuxuan Zhou, Tonio Ball, and Dan Zhang. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. *Advances in Neural Information Processing Systems*, 33:21038–21049, 2020.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Coldgans: Taming language gans with cautious sampling strategies. *Advances in Neural Information Processing Systems*, 2020a.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Discriminative adversarial search for abstractive summarization. *arXiv preprint arXiv:2002.10375*, 2020b.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. To beam or not to beam: That is a question of cooperation for language gans. *Advances in neural information processing systems*, 2021.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI-16*, 2016.

Iulian V Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

Zhihui Shao, Jianyi Yang, and Shaolei Ren. Calibrating deep neural network classifiers on out-of-distribution datasets. *arXiv preprint arXiv:2006.08914*, 2020a.

Zhihui Shao, Jianyi Yang, and Shaolei Ren. Increasing trustworthiness of deep neural networks via accuracy monitoring. *arXiv preprint arXiv:2007.01472*, 2020b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529 7587: 484–9, 2016.

Jascha Sohl-Dickstein, Peter B Battaglino, and Michael R DeWeese. New method for parameter estimation in probabilistic models: minimum probability flow. *Physical review letters*, 107(22):220601, 2011.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Kaitao Song, Xu Tan, and Jianfeng Lu. Neural machine translation with error correction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3891–3897, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11895–11907. Curran Associates, Inc., 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2020b. URL `https://arxiv.org/abs/2011.13456`.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*, 2015.

Akinori Tanaka. Discriminator optimal transport. *arXiv preprint arXiv:1910.06832*, 2019.

Chenyang Tao, Liqun Chen, Shuyang Dai, Junya Chen, Ke Bai, Dong Wang, Jianfeng Feng, Wenlian Lu, Georgiy V. Bobashev, and Lawrence Carin. On fenchel mini-max learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 10427–10439, 2019. URL `http://papers.nips.cc/paper/9230-on-fenchel-mini-max-learning`.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. *arXiv preprint arXiv:1701.03079*, 2017.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-Hastings generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6345–6353, Long Beach, California, USA, 2019. PMLR. URL `http://proceedings.mlr.press/v97/turner19a.html`.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125, 2016a.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016b. URL `http://papers.nips.cc/paper/6527-conditional-image-generation-with-pixelcnn-decoders`.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016c.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016a.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016b.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, and Theodore L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. *ECCV*, 2018.

Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. Energy-based open-world uncertainty modeling for confidence calibration. *CoRR*, abs/2107.12628, 2021.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack W. Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Jimenez Rezende, David Saxton, Adam Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matthew Botvinick, Demis Hassabis, and Timothy P. Lillicrap. Unsupervised predictive memory in a goal-directed agent. *CoRR*, abs/1803.10760, 2018.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688, 2011a.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011b. URL `https://icml.cc/2011/papers/398_icmlpaper.pdf`.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceeddings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Neural Information Processing Systems (NIPS)*, 2016a.

Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. Logan: Latent optimisation for generative adversarial networks. *arXiv preprint arXiv:1912.00953*,

2019.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016b.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A symbiosis between variational autoencoders and energy-based models. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *AAAI*, volume 1, page 7, 2018.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

Minkai Xu, Shitong Luo, Yoshua Bengio, Jian Peng, and Jian Tang. Learning neural generative dynamics for molecular conformation generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Fangfang Yang and Shaolei Ren. Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers. *arXiv preprint arXiv:2006.05594*, 2020.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Shuangfei Zhai, Walter Talbott, Carlos Guestrin, and Joshua Susskind. Adversarial fisher vectors for unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 11158–11168, 2019.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. Variational neural machine translation. In *EMNLP*, 2016a.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv:1612.03242*, 2016b.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume*

1, pages 4334–4343, 2019.

Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680, 2014.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Yipin Zhou and Tamara L Berg. Learning temporal transformations from time-lapse videos. In *European Conference on Computer Vision*, pages 262–277. Springer, 2016.

Yufan Zhou, Changyou Chen, and Jinhui Xu. Learning high-dimensional distributions with latent neural fokker-planck kernels, 2021.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.