,,,

Université de Montréal

Generative Models: A Critical Review

par Alex Lamb

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Avril, 2018

Résumé

Dans cette thèse, nous introduisons et motivons la modélisation générative comme une tâche centrale pour l'apprentissage automatique et fournissons une vue critique des algorithmes qui ont été proposés pour résoudre cette tâche. Nous montrons comment la modélisation générative peut être définie mathématiquement en essayant de faire une distribution d'estimation identique à une distribution de vérité de terrain inconnue. Ceci peut ensuite être quantifié en termes de valeur d'une divergence statistique entre les deux distributions. Nous décrivons l'approche du maximum de vraisemblance et comment elle peut être interprétée comme minimisant la divergence KL. Nous explorons un certain nombre d'approches dans la famille du maximum de vraisemblance, tout en discutant de leurs limites. Enfin, nous explorons l'approche antagoniste alternative qui consiste à étudier les différences entre une distribution d'estimation et une distribution de données réelles. Nous discutons de la façon dont cette approche peut donner lieu à de nouvelles divergences et méthodes qui sont nécessaires pour réussir l'apprentissage par l'adversité. Nous discutons également des nouveaux paramètres d'évaluation requis par l'approche contradictoire.

Le chapitre ref chap: fortnet montre qu'en apprenant des modèles génératifs des couches cachées d'un réseau profond, on peut identifier quand le réseau fonctionne sur des données différentes des données observées pendant la formation. Cela nous permet d'étudier les différences entre les modes de fonctionnement libre et de forçage des enseignants dans les réseaux récurrents. Cela conduit également à une meilleure robustesse face aux attaques adverses.

Le chapitre ref chap: gibbsnet a exploré une procédure itérative pour la génération et l'inférence dans les réseaux profonds, qui est inspirée par la procédure MCMC de gibbs bloquées pour l'échantillonnage à partir de modèles basés sur l'énergie. Cela permet d'améliorer l'inpainting, la génération et l'inférence en supprimant l'exigence que les variables a priori sur les variables latentes aient une distribution connue.

Le chapitre ref chap: discreg a étudié si les modèles génératifs pouvaient être améliorés en exploitant les connaissances acquises par des modèles de classification discriminants. Nous avons étudié cela en augmentant les autoencoders avec des pertes supplémentaires définies dans les états cachés d'un classificateur fixe. Dans la pratique, nous avons montré que cela conduisait à des modèles générateurs mettant davantage l'accent sur les aspects saillants des données, et discutait également des limites de cette approche.

Mots clés: réseaux de neurones, apprentissage automatique, apprentissage de représentations profondes, apprentissage supervisé, modèles génératifs, prédiction structurée

Summary

In this thesis we introduce and motivate generative modeling as a central task for machine learning and provide a critical view of the algorithms which have been proposed for solving this task. We overview how generative modeling can be defined mathematically as trying to make an estimating distribution the same as an unknown ground truth distribution. This can then be quantified in terms of the value of a statistical divergence between the two distributions. We outline the maximum likelihood approach and how it can be interpreted as minimizing KL-divergence. We explore a number of approaches in the maximum likelihood family, while discussing their limitations. Finally, we explore the alternative adversarial approach which involves studying the differences between an estimating distribution and a real data distribution. We discuss how this approach can give rise to new divergences and methods that are necessary to make adversarial learning successful. We also discuss new evaluation metrics which are required by the adversarial approach.

Chapter 2 shows that by learning generative models of the hidden layers of a deep network can identify when the network is being run on data differing from the data seen during training. This allows us to study differences between free-running and teacher forcing modes in recurrent networks. It also leads to improved robustness to adversarial attacks.

Chapter 3 explored an iterative procedure for generation and inference in deep networks, which is inspired by the blocked gibbs MCMC procedure for sampling from energy-based models. This achieves improved inpainting, generation, and inference by removing the requirement that the prior over the latent variables have a known distribution.

Chapter 4 studied whether generative models could be improved by exploiting the knowledge learned by discriminative classification models. We studied this by augmenting autoencoders with additional losses defined in the hidden states of a fixed classifier. In practice we showed that this led to generative models with better focus on salient aspects of the data, and also discussed limitations in this approach.

Keywords: neural networks, machine learning, deep learning, supervised learning, generative modeling, structured prediction

Contents

Résumé . . . . . **ii**

Summary . . . . . **iv**

Contents . . . . . **v**

List of Figures . . . . . **viii**

List of Tables . . . . . **xi**

List of Abbreviations . . . . . **xiii**

Acknowledgments . . . . . **xiv**

1 Introduction **1**
1.1 What are Generative Models? 1
1.1.1 Formalizing the Generative Modeling Problem 2
1.2 Algorithms 4
1.2.1 The Likelihood Maximization Approach 4
1.2.2 Energy-Based Models 7
1.2.3 The Adversarial Approach 14

2 Fortified Networks **24**
2.1 Prologue to the Article 24
2.2 Abstract 25
2.3 Introduction 26
2.4 Background 28
2.4.1 The Empirical Risk Minimization Framework 28
2.4.2 Adversarial Attacks and Robustness 28
2.4.3 Denoising Autoencoders 28
2.5 Fortified Networks 29
2.6 Experiments 33
2.6.1 Attacks 33
2.6.2 The Gradient Masking and Gradient Obfuscation Problem . 34

2.6.3 Reconstruction Error as a Heuristic for Deviation from the Manifold . . . 34
2.7 Results . . . 35
2.7.1 Recurrent Networks . . . 36
2.8 Related Work . . . 37
2.8.1 Using Generative Models as a Defense . . . 37
2.8.2 Adversarial Hidden State Matching . . . 39
2.8.3 Denoising Feature Matching . . . 40
2.8.4 Adversarial Spheres . . . 41
2.9 Conclusion . . . 42

3 GibbsNet . . . 44
3.1 Prologue to the Article . . . 44
3.2 Abstract . . . 45
3.3 Introduction . . . 46
3.4 Proposed Approach: GibbsNet . . . 48
3.4.1 Theoretical Analysis . . . 50
3.4.2 Architecture . . . 51
3.5 Related Work . . . 52
3.6 Experiments and Results . . . 54
3.6.1 Expressiveness of GibbsNet's Learned Latent Variables . . . 55
3.6.2 Inception Scores . . . 56
3.6.3 Generation, Inpainting, and Learning the Image-Attribute Joint Distribution . . . 57
3.7 Conclusion . . . 60

4 Discriminative Regularization for Generative Models . . . 61
4.1 Prologue to the Article . . . 61
4.2 Abstract . . . 61
4.3 Introduction . . . 62
4.4 VAEs as Generative models of images . . . 63
4.4.1 The generative model . . . 64
4.4.2 The approximate inference model . . . 65
4.4.3 Reparametrization Trick . . . 66
4.4.4 The problem with the Independent Gaussian Assumption . . . 66
4.5 Discriminative Regularization . . . 67
4.6 Related Work . . . 69
4.7 Experiments . . . 70
4.7.1 Samples . . . 71
4.7.2 Reconstructions . . . 72
4.7.3 Interpolations in the Latent Space . . . 72
4.7.4 Explaining Visual Artifacts . . . 73

4.8 Conclusion 76

5 Conclusion 77

Bibliography 79

List of Figures

2.1 Illustration of the autoencoder dynamics in the input space (top) and in abstract hidden space (bottom). The leftmost panels show data points from three different classes, the middle panels show vector fields describing the autoencoder dynamics, and the rightmost panels show a number of resulting trajectories and basins of attraction. The key motivation behind Fortified Networks is that directions which point off the data manifold are easier to identify in an abstract space with simpler statistical structure, making it easier to map adversarial examples back to the projected data manifold. 27

2.2 An illustration of the process of mapping back to the manifold in the visible space (left) and the hidden space (right). The shaded regions represent the areas in the space which are occupied by data points from a given class (they do *not* represent decision boundaries). . . 30

2.3 Diagram illustrating a one-layer fortified network. A network is evaluated with a data sample x and its corresponding adversarial example $\widetilde{x}$. Hidden units h_k and $\widetilde{h_k}$ are corrupted with noise, encoded with the encoder *Enc*, and decoded with the decoder *Dec*. The autoencoder (denoted by the red color) is trained to reconstruct the hidden unit h_k that corresponds to the clean input. Dotted lines are two reconstruction costs: for a benign ($\mathcal{L}_{rec}$) and adversarial examples ($\mathcal{L}_{adv}$). Note that a layer can be fortified at any position in the network. 31

2.4 We added fortified layers with different capacities to MLPs trained on MNIST, and display the value of the total reconstruction errors for adversarial examples divided by the total reconstruction errors for clean examples. A high value indicates that adversarial examples have high reconstruction error. We considered fortified layers with autoencoders of different capacities. Our results support the central motivation for fortified networks: that off-manifold points can much more easily be detected in the hidden space (as seen by the relatively constant ratio for the autoencoder in h space) and are much harder to detect in the input space (as seen by this ratio rapidly falling to zero as the autoencoder's capacity is reduced). 40

2.5 We ran a fortified network on Fashion-MNIST using adversarial training with PGD for a variety of ε values, each for 5 epochs. The motivation behind this experiment, suggested by Athalye et al. (2018) is confirming if unbounded ($\varepsilon = 1$) adversarial attacks are able to succeed. A defense which succeeds primarily by masking or obfuscating the gradients would fail to bring the accuracy to zero even with an unbounded attack. As can be seen, unbounded attacks against Fortified Networks succeed when given a sufficiently large ε, which is evidence against gradient masking. 42

3.1 Diagram illustrating the training procedure for GibbsNet. The **unclamped chain** (dashed box) starts with a sample from an isotropic Gaussian distribution $\mathcal{N}(0, I)$ and runs for N steps. The last step (iteration N) shown as a solid pink box is then compared with a single step from the **clamped chain** (solid blue box) using joint discriminator D. 46

3.2 Evolution of samples for 20 iterations from the unclamped chain, trained on the SVHN dataset starting on the left and ending on the right. 50

3.3 Illustration of the distribution over inferred latent variables for real data points from the MNIST digits (0, 1, 9) learned with different models trained for roughly the same amount of time: GibbsNet with a determinstic decoder and the latent variables not given to the discriminator (a), GibbsNet with a stochastic decoder and the latent variables not given to the discriminator (b), ALI (c), GibbsNet with a deterministic decoder (f), GibbsNet with a stochastic decoder with two different runs (g and h), GibbsNet with a stochastic decoder's inferred latent states in an unclamped chain at 1, 2 , 3, and 15 steps (d, e, i, and j, respectively) into the P-chain (d, e, i, and j, respectively). Note that we continue to see refinement in the marginal distribution of z when running for far more steps (15 steps) than we used during training (3 steps). 56

3.4 CIFAR samples on methods which learn transition operators. Non-Equilibrium Thermodynamics (Sohl-Dickstein et al., 2015) after 1000 steps (left) and GibbsNet after 20 steps (right). 58

3.5 Inpainting results on SVHN, where the right side is given and the left side is inpainted. In both cases our model's trained procedure did not consider the inpainting or conditional generation task at all, and inpainting is done by repeatedly applying the transition operators and clamping the right side of the image to its observed value. GibbsNet's richer latent space allows the transition operator to keep more of the structure of the input image, allowing for tighter inpainting. 59
3.6 Demonstration of learning the joint distribution between images and a list of 40 binary attributes. Attributes (right) are generated from a multinomial distribution as part of the joint with the image (left). . 59

4.1 The discriminative regularization model. Layers f_1, f_2, f_3, d_1, d_2 and d_3 represent convolutional layers, whereas layers g_3, g_4 and μ_θ represent fractionally strided convolutional layers. 64
4.2 Face samples generated with and without discriminative regularization. On balance, details of the face are better captured and more varied in the samples generated with discriminative regularization. . 71
4.3 Face reconstructions with (top row) and without (bottom row) discriminative regularization. The face images used for the reconstructions (middle row) are from the held-out validation set and were not seen by the model during training. The architecture and the hyperparameters (except those directly related to discriminative regularization) are the same for both models. Discriminative regularization greatly enhances the model's ability to preserve identity, ethnicity, gender, and expressions. Note that the model does not improve the visual quality of the image background, which likely reflects the fact that the classifier's labels all describe facial attributes. Additional reconstructions can be seen in the appendix. 72
4.4 Latent space interpolations with discriminative regularization. On each row, the first and last image correspond to reconstructions of randomly selected examples. 74
4.5 From left to right: input examples, convolutional (non-variational) autoencoder reconstructions (no blurring applied to the classifier's hidden representations), model reconstructions (trained with discriminative regularization), convolutional autoencoder reconstructions (blurring applied to the classifier's hidden representations). 75

List of Tables

2.1 Accuracies against white-box MNIST attacks with FGSM, where the model is a convolutional net. We used the standard FGSM attack parameters with an ε of 0.3 and compare against published adversarial training defenses. We also performed ablations where we considered removing the reconstruction error on adversarial examples $\mathcal{L}_{adv}$ as well as switching the activation function in the fortified layers from leaky relu to tanh, which we found to slightly help in this case. While our baseline and pre-fortified networks used relu activations, we found that by using a leaky relu in all layers the accuracy on FGSM $\varepsilon = 0.3$ could be improved to 99.2% with standard adversarial training, suggesting that both our own baselines and those reported in the past have been too weak. 35

2.2 Accuracies against white-box MNIST attacks with PGD with an ε of 0.1, where our model is a convnet. 36

2.3 Accuracies against white-box CIFAR attacks with FGSM using ($\varepsilon = 0.3$), where each model is a convnet. Our baseline adversarial training is the resnet model provided in (Nicolas Papernot, 2017) 36

2.4 Accuracies against white-box CIFAR attacks with FGSM using the standard ($\varepsilon = 0.03$), where each model is a convnet. Our baseline adversarial training is the resnet model provided in (Nicolas Papernot, 2017) 37

2.5 Accuracies against white-box attacks on Fashion MNIST. For PGD we used $\varepsilon = 0.1$ and for FGSM we experimented with $\varepsilon = 0.1$ and $\varepsilon = 0.3$. Compared with DefenseGAN (Pouya Samangouei, 2018). . 38

2.6 Accuracies against blackbox MNIST attacks with adversarial training. Reporting 50/50 results compared to previous works (Jacob Buckman, 2018, JB) and (Pouya Samangouei, 2018, PS). The test error on clean examples is in parenthesis. 39

2.7 We trained Fortified Networks on a single-layer LSTM on the Text-8 dataset, with fortified layers added between each step. We recorded the ratio between reconstruction error on the testing set during both teacher forcing mode and sampling mode (where the model is supplied with its own outputs as inputs for the next step). The motivation is that the outputs should gradually move off of the manifold with more sampling steps, which is indicated by a high reconstruction error ratio, which makes it an interesting tool for monitoring or potentially fixing this problem. 41

3.1 Inception Scores from different models. Inpainting results were achieved by fixing the left half of the image while running the chain for four steps. Sampling refers to unconditional sampling. 57

4.1 A list of the binary targets that we predict with our celebA classifier. 73

List of Abbreviations

AE	Auto-Encoder
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
GD	Gradient Descent
GMM	Gaussian Mixture Model
KLD	Kullback-Liebler Divergence
LSTM	Long-Short Term Memory
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MSE	Mean Squired Error
NLL	Negative Log-Likelihood
PMF	Probability Mass Function
PDF	Probability Density Function
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
VAE	Variational Auto-Encoder
WGAN	Wasserstein GAN
WGAN-GP	Wasserstein GAN with Gradient Penalty
XE	Cross Entropy

Acknowledgments

I am extremely grateful for an amazing set of family, friends, colleagues, and advisers without whom my work would not be possible.

Thank you to Dr. Aaron Courville for serving as an outstanding adviser. Your direction, research mentorship, and guidance has allowed me to grow as a researcher and follow the path I am on today.

To my many friends, co-authors, co-workers, and colleagues (alphabetically):
Adam Roberts, Adriana Romero, Alexandre Abraham, Amjad Almahairi, Anirudh Goyal, Anna Huang, Arnaud Bergeron, Barron Bichon, Bart van Merriënboer, Ben Poole, Brandon Sammons, Çaglar Gülçehre, Casper Kaae Sønderby, César Laurent, Chris Smith, Christof Angermueller, Cinjon Resnick, Colin Raffel, Colin Vaz, Curtis "Fjord" Hawthorne, Daniel Trejo, Daniel Jiwoong Im, Danlan Chen, David Ha, David Krueger, David Warde-Farley, Dmitriy Serdyuk, Dzmitry Bahdanau, Elvis Dohmatob, Eduardo Gonzalez, Eugene Belilovsky, Fabian Pedregosa, Faruk Ahmed, Felix Hill, Francesco Visin, Fred Bertsch, Frédéric Bastien, Guillaume Alain, Harm de Vries, Ishaan Gulrajani, Ishmael Belghazi, Iulian Vlad Serban, Jacob Limmer, Jan Chorowski, Jesse Engel, João Felipe Santos, Jörg Bornschein, José Sotelo, Junyoung Chung, Kelvin Xu, Kratarth Goel, Kyunghyun Cho, Laurent Dinh, Li Yao, Luke Vilnis, Marcelo Grave, Mark Pillion, Mary Otto, Mathieu Germain, Matthew Coursen, Mehdi Mirza, Michael Eickenberg, Michael Talbert, Mohammad Pezeshski, Myriam Côté, Natasha Jaques, Nan Rosemary Ke, Nicolas Ballas, Negar Rostamzadeh, Orhan Firat, Pascal Lamblin, Patrick Sammons, Pierre-Luc Carrier, Philemon Brakel, Rick Parker, Sam Stavinoha, Sebastian Öberg, Shawn Tan, Simon Receveur, Sina Honari, Søren Sønderby, Soroush Mehri, Stanislas Lauly, Sungjin Ahn, Tim Cooijmans, Vincent Dumoulin, and Yann Dauphin; Thank you for everything you do.

In addition, thanks to the rest of my friends and coworkers in MILA and my former colleagues at Amazon. I am sure I missed some people I shouldn't have forgotten. Thanks to all of you.

To my friends, family, and mentors who are no longer with us, RIP and thank you for everything.

Finally, the work reported in this thesis would not have been possible without the financial support from: Ubisoft, Samsung, IBM, NSERC, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

1 Introduction

1.1 What are Generative Models?

One of the most distinctive and powerful aspects of human cognition is the ability to imagine: to synthesize mental objects which are not bound by what is immediately present in reality. There are many potential reasons why humans evolved this capability. One is that it allows humans to do planning by imagining how their actions could effect the future. Another is that by imagining how the future will unfold, humans can test hypotheses about the dynamics of the world, and learn about its properties without explicit supervision. The subfield of Machine Learning which aims to endow machines with this same essential capacity to imagine and synthesize new entities is referred to as generative modeling. Along with these lofty ambitions, there are many practical motivations for generative models. One common argument is that humans may use generative models as a way of doing supervised and reinforcement learning while using less labeled data. Consider the task of children learning a new language. While children do receive occasional explicit supervised feedback, i.e. from their parents telling them that they misspelled a word or that they've overgeneralized a word, this feedback is relatively rare. Reward signals are perhaps just as rare. Because of this, cognitive scientists are curious about how humans are able to learn with relatively little overfitting. They refer to this as the "poverty of the stimulus" problem. An appealing hypothesis is that humans use unsupervised generative models to build robust representations of the world and then use those same representations to do supervised and reinforcement learning from small amounts of explicitly labeled data. Since humans are constantly absorbing perceptual data (sound, vision, touch), humans should have enormous amounts of unlabeled data which can be used to train generative models without overfitting. We could also imagine that many of these generative models require learning features which are also useful for supervised learning. For example, one

potential generative model would learn to construct the sequence of future visual stimuli conditioned on a window of past visual stimuli $p(X_{t:T}|X_{1:t})$. A model capable of performing this task well would need to have a strong model for how the world works (i.e. what things form objects, what things are near and far away, what things are large and small, whether something is alive or not, etc.).

Another practical motivation for generative models is that they could provide better evaluations of the effectiveness of Machine Learning models. Because classifiers produce the same output for a wide class of inputs, it can be hard to evaluate what a classifier has really learned about a type of data. Suppose we have a dataset where a model generates text captions from an image. Consider an image of a giraffe, which the model describes as "A giraffe walking next to tall green grass". It's possible that the model has learned enough to be able to recognize that there is a giraffe, and that the giraffe is walking (and not running or sitting), and that there is tall green grass. However, another possibility is that the model recognizes the giraffe, but simply says that the giraffe is walking because giraffes are usually walking, and says that the giraffe is near tall grass because giraffes are usually near tall grass, and says that the grass is green because grass is usually green. Thus it's difficult to know if the model really understands the image, or if it's merely making reasonable guesses based on what types of images are common. However, consider what could be done if we had a generative model which produced sample images conditioned on the caption $P(X|C)$. Since humans can generate arbitrary text by hand, we could easily supply the model with counterfactuals like "A running giraffe next to short red grass" or "A giraffe lying down next to tall blue grass". Since humans easily generate detailed counterfactuals in text, it would be easy to verify how well the model understands the world.

1.1.1 Formalizing the Generative Modeling Problem

So far, we have discussed generative modeling in qualitative terms. We want models which can simulate from the dynamics of the world. We want models that can synthesize realistic looking data. However, before going further it is useful to understand the probabilistic interpretation of generative model, which gives a formal framework for studying generative models. The essential idea is that we treat observations from the world as samples from a distribution $x \sim p(x)$. For

example, we could consider the distribution over all human faces which can occur in reality to be $p(x)$ and consider each face as a sample. If we have access to a recorded dataset (for example a set of faces), we may also choose to treat these points as a finite collection of samples from this distribution.

At the same time, we can interpret our generative model as an estimating distribution $q_\theta(x)$, which is described by a set of parameters θ. Then we can frame generative modeling as trying to ensure that $p(x)$ and $q_\theta(x)$ become as similar as possible. Statistical divergences give a natural mathematical framework for this. A divergence is a function $D(p||q) : S \times S \rightarrow R$ taking two distributions p and q over a space of distributions S as inputs, with the properties:

$$D(p||q) \geq 0.0 \tag{1.1}$$

$$D(p||q) = 0.0 \iff p = q \tag{1.2}$$

Notably, there is no symmetry assumption, so in general $D(p||q) \neq D(q||p)$. The probabilistic approach to generative modeling frames learning as an optimization problem where the loss corresponds to a given divergence.

$$\mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}}\, D(p||q_\theta(x)) \tag{1.3}$$

Alternatives to the Probabilistic Generative Models Formalization

At this point, it is worth noting that not all generative models use the probabilistic generative model framework. Deep style transfer and texture synthesis (Gatys et al., 2015a,b) search for an image with "style features", defined using a fixed neural network, matching a real image. Deep Image Prior searches for network parameters which produce a single real image (or part of a real image, in the case of inpainting). Many of these approaches have the distinctive property that they define a rule for modifying a single real image (Gatys et al., 2015a; Ulyanov et al., 2017), which limits their ability to generalize. There is also a line of work which has competing objectives: one which encourages the generations to have novel traits and another which encourages the generations to look realistic. One such example is the Creative Adversarial Network (Elgammal et al., 2017) which has

one objective encouraging the generated images to differ from styles that occur in the data and another objective encouraging the generated images to follow the data distribution. These approaches radically differ from the probabilistic framework in that they encourage the production of data points which do not have density under any observed distribution $p(x)$.

1.2 Algorithms

We briefly overview the two major approaches which are used for probabilistic generative modeling with deep learning. The first, and considerably older approach, defines a density for generative model and directly maximizes the value of this density on observed data points. A newer and quite distinctive approach involves modeling the difference between a given generative model and the real data, and then encouraging the generative model to minimize that distance.

1.2.1 The Likelihood Maximization Approach

What is the right algorithm for finding a distribution $q_\theta(x)$ which minimizes a divergence between itself and $p(x)$. Before selecting the type of divergence to minimize, a natural question is to consider what types of expressions we are capable of optimizing, and work backwards to find a suitable divergence. In general, we only have access to samples from the distribution $p(x)$ and not any additional information about the distribution. At the same time, $q_\theta(x)$ is a model that we control, so it's reasonable to believe that we'll be able to design it so that it has a density that we can compute as well as the ability to draw samples.

The KL-divergence can be rewritten as an expression in which the only term that depends on the parameters is an expectation on $q_\theta(x)$ over samples from $p(x)$. Beginning with two distributions $p(x)$ (the empirical distribution) and $q(x)$ (the model distribution), we write the KL-divergence (Nowak, 2009).

$$D_{KL}(p(x)||q(x)) = \int p(x) \log p(x) dx - \int p(x) \log q(x) dx \tag{1.4}$$

$$= \int p(x) \log \frac{p(x)}{q(x)} dx \tag{1.5}$$

$$= \mathbb{E}_{x \sim p}[\log \frac{p(x)}{q(x)}] \tag{1.6}$$

$$= \mathbb{E}_{x \sim p}[\log p(x) - \log q(x)] \tag{1.7}$$

Then we can show the maximum likelihood estimation for a set of N data points.

$$\theta* = \arg\max_{\theta} \prod_{i=1}^{N} q(x_i) \tag{1.8}$$

$$= \arg\max_{\theta} \sum_{i=1}^{N} \log q(x_i) \tag{1.9}$$

$$= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} - \log q(x_i) \tag{1.10}$$

$$\sim= \arg\min_{\theta} \mathbb{E}_{l \to \infty, x \sim p}[- \log q(x)] \tag{1.11}$$

The objective for maximum likelihood is maximizing the log-density $log(q_\theta(x))$ over real data points sampled from the distribution $p(x)$.

$$D_{KL}(p(x)||q(x)) = \int p(x) \log p(x) dx - \int p(x) \log q(x) dx \tag{1.12}$$

$$= -H(p(x)) + CE(p(x), q(x)) \tag{1.13}$$

Thus we can see that the KL-divergence decomposes into two terms 1.12: a cross-entropy term (likelihood) and a term for the entropy of the true data distribution. Because the entropy of the true data distribution doesn't depend on the estimator, the KL-divergence can be minimized by maximizing likelihood. Another useful consequence of this is that the entropy of the true data distribution can be estimated by such a generative model if it maximizes likelihood.

Estimation with Tractable Densities

Now that we have established a statistical divergence that we can minimize by optimizing over the parameters of a distribution $\mathbb{Q}_\theta$, we turn to the question of deciding what to use as $\mathbb{Q}$, which will occupy our attention for the remainder of the section on the maximum likelihood approach.

The maximum likelihood approach only requires that we be able to sample uniformly from the real data and evaluate the log-density of our estimating distribution $q_\theta(x)$ at these points. What is the simplest choice for q, if we want to frame our problem in terms of optimizing over functions? Indeed, q cannot simply be an arbitrary function, because it could simply assign a high value to every point in the space. For q to correspond to the density of an actual probability distribution, it only needs to satisfy two simple properties 1.14: that it be non-negative everywhere and integrate to 1.0 over the region where its value is defined (called the support of the distribution). To simplify, we'll write the definition using the real numbers $\mathbb{R}$ as the support.

$$q(x) \geq 0 \tag{1.14}$$

$$\int_{x \in \mathbb{R}} q(x) = 1 \tag{1.15}$$

One of the most straightforward ways to satisfy 1.14 and 1.15 is to analytically prove that functions with specific parameters satisfy these properties. While this approach is very limited many such useful functions have been derived and are in widespread use. One of the most prominent is the normal distribution, which has a density parameterized by μ and σ.

$$q(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-(x-\mu)^2}{2\sigma^2} \tag{1.16}$$

The gaussian integral has a simple value, allowing for a straightforward proof for 1.15 for a variety of distributions involving an exponential over the variable x, which includes the exponential family.

Mixture Models

A major limitation of most closed-form densities, is that they are unimodal, which makes them ill-suited to problems where very distinct points can have high density with regions of low density separating them. For example, nearly all natural distributions such as image and audio datasets are multimodal. Moreover, most of these tractable closed-form densities either assume independence between dimensions or only pairwise dependence (for example, the multivariate gaussian). This further limits the applicability to real data, where density is generally concentrated along specific low-dimensional manifolds (Bengio et al., 2012a).

One straightforward way to get around these limitations is to replace any density with a mixture over densities with distinct parameters. This can greatly increase the capacity of the model and has been used in deep generative models (Graves, 2012; Salimans et al., 2017). It has a simple closed-form, where each component in the mixture has a weighting π_k and a distribution q_{θ_k}.

$$q_\theta(x) = \sum_{k=1}^{C} \pi_k q_{\theta_k}(x) \tag{1.17}$$

This form is guaranteed to be normalized with the only condition that the π_k sum to 1.0. This approach has the clear advantage that it allows for a much higher capacity model, yet it has the issue that the only way to achieve more modes is to add more mixture components. This turns out to be a significant limitation that restricts the utility of mixture models to relatively simple, low-dimensional distributions when the mixture components are unimodal distributions.

1.2.2 Energy-Based Models

Considering the properties of a distribution in 1.14 and 1.15, one potential path forward is to recognize that the non-negativity is quite easy to enforce by construction, while the constraint on the integral often requires a non-trivial proof. Based on this, we can define a function called an energy, which is non-negative by construction, but does not necessarily sum to 1.0 over its support. One way to define this energy is by using an exponential.

$$E_\theta(x) = e^{-f_\theta(x)} \tag{1.18}$$

We can then compute the sum over the space which x occupies, which we refer to as Z_θ.

$$Z_\theta = \int_{x \in R} e^{-f_\theta(x)} \tag{1.19}$$

$$q_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta} \tag{1.20}$$

When $Z_\theta = 1$, our energy function already defines a probability distribution. Otherwise, we can divide the energy by Z_θ to have a probability distribution which satisfies 1.15. For energy functions defined by neural networks, there are no general methods of determining this integral over the entire space. Typically neural networks are evaluated at specific points from some distribution (for example, a data distribution), which is insufficient for computing the integral over the entire space.

However, computing the gradient of the log-likelihood for energy-based models reveals an interesting form.

$$q_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta} \tag{1.21}$$

$$log(q_\theta(x)) = -f_\theta(x) - log(Z_\theta) \tag{1.22}$$

$$-\frac{dlog(q_\theta(x))}{d\theta} = \frac{df_\theta(x)}{d\theta} + \frac{dlog(Z_\theta)}{d\theta} \tag{1.23}$$

$$\frac{dlog(Z_\theta)}{d\theta} = \frac{d}{d\theta}\left[log \sum_{\tilde{x}} e^{-f_\theta(x)}\right] \tag{1.24}$$

$$\frac{dlog(Z_\theta)}{d\theta} = \frac{1}{Z_\theta} \sum_{\tilde{x}} \frac{d}{d\theta}\left[e^{-f_\theta(\tilde{x})}\right] \tag{1.25}$$

$$\frac{dlog(Z_\theta)}{d\theta} = \sum_{\tilde{x}} -q(\tilde{x}) \frac{df_\theta(\tilde{x})}{d\theta} \tag{1.26}$$

$$-\frac{dlog(Z_\theta)}{d\theta} = \mathop{\mathbb{E}}_{\tilde{x} \in q_\theta(x)}\left[\frac{df_\theta(\tilde{x})}{d\theta}\right] \tag{1.27}$$

When we consider the expected value of the gradient over multiple data samples, the resulting gradient then has a particularly elegant form consisting of a positive phase, in which the function is maximized over samples from the real data, and a negative phase in which the function's value is pushed down at samples from the model distribution.

$$\mathop{\mathbb{E}}_{x \in p(x)}\left[-\frac{dlog(q_\theta(x))}{d\theta}\right] = \mathop{\mathbb{E}}_{x \in p(x)}\left[\frac{df_\theta(x)}{d\theta}\right] - \mathop{\mathbb{E}}_{\tilde{x} \in q_\theta(x)}\left[\frac{df_\theta(\tilde{x})}{d\theta}\right] \tag{1.28}$$

Although this gives a simple form for the gradient, it assumes that we have the ability to sample from our model distribution as well as calculate its energy. In practice this has been a major obstacle to the use of energy-based probabilistic generative models. Some research has explored the use of boltzmann machine and an approximation called contrastive divergence which replaces the model distribution $q_\theta(x)$ in 1.28 with short gibbs sampling chains which start from the real data distribution. The discovery of a general way of defining energy functions with deep networks which also allow for fast and exact sampling would make energy-based models significantly more appealing.

Another challenge with energy-based models is that this straightforward form

only applies to the gradient of the likelihood with respect to the parameters. Computing the likelihood itself still turns out to be quite difficult, due to the partition function being an integral over the entire space, while neural networks are typically only evaluated at specific points. Two solutions have been proposed for estimating the partition function, Annealed Importance Sampling (Neal, 1998) and Reverse Annealed Importance Sampling (Burda et al., 2014), however both are only approximations and require an iterative procedure.

Autoregressive Models

The previous approaches that we've explored for likelihood maximization have tried to increase the expressiveness of $q_\theta(x)$, but this has proved to be difficult for complicated multivariate distributions. An alternative way to achieve the goal of increased expressiveness for multivariate distributions is to factorize the joint distribution into a chain of conditionals. Often this can be done with an RNN in the context of sequence modeling. The density is represented via a fully-observed directed graphical model: it decomposes the distribution over the discrete time sequence $x_1, x_2, \ldots x_T$ into an ordered product of conditional distributions over tokens

$$q_\theta(x_1, x_2, \ldots x_T) = q_\theta(x_1) \prod_{t=1}^{T} q_\theta(x_t \mid x_1, \ldots x_{t-1}).$$

When using this autoregressive approach to train RNNs in practice, it is known as *teacher forcing* (Williams and Zipser, 1989), due to the use of the ground-truth samples y_t being fed back into the model to be conditioned on for the prediction of later outputs (analogous to a teacher directly replacing a student's attempted steps in a solution with correctly completed steps so that they may continue learning). These fed back samples force the RNN to stay close to the ground-truth sequence during training.

When sampling from an autoregressive model, the ground-truth sequence is not available for conditioning and we sample from the joint distribution over the sequence by sampling each y_t from its conditional distribution given the previously generated samples. Unfortunately, this procedure can result in problems in generation as small prediction error compound in the conditioning context. This can lead to poor prediction performance as the conditioning context (the sequence of previously generated samples) diverges from the distribution seen during training.

(Bengio et al., 2015) proposed to address this exposure bias issue by sometimes feeding the model's predicted values back into the network as inputs during training (as is done during sampling from autoregressive models). However, when the model generates several consecutive y_t's, it is not clear anymore that the correct target (in terms of its distribution) remains the one in the ground truth sequence. In general these sampled values could be completely inconsistent with the ground truth sequence providing targets for the outputs. This is mitigated in various ways, by making the self-generated subsequences short and annealing the probability of using self-generated vs ground truth samples. However, as remarked by Huszár (2015), scheduled sampling yields a biased estimator, in that even as the number of examples and the capacity go to infinity, this procedure may not converge to the correct model. Nonetheless, in some experiments scheduled sampling still had value as a regularizer. A consistent way of improving autoregressive models by using adversarial training 1.2.3 was proposed by (Lamb et al., 2016).

In general, the strength of autoregressive models is that they have a straightforward and general statistical formulation in terms of defining a density and directly maximizing likelihood. Additionally, if each step in the sequence is a scalar, it only requires us to define univariate conditional distributions, and the set of univariate distributions with with closed-form densities is quite general. For example, a univariate multinomial distribution can be multimodal, can closely approximate a wide range of distributions, and is quite tractable.

The major weakness of autoregressive models are that the one-step-ahead loss is often not a good fit for long-term measures of error (due to the compounding error effects not observed during training) and that representing uncertainty directly in the space of single steps could be very unnatural. This may be related to the phenomenon in humans of "writer's block", where it's difficult to begin a writing task from scratch. In the context of autoregressive models, the first few steps often contain a great deal of entropy as they practically constrain the content of all of the text to follow, yet figuring out how the beginning of the text will need to lead to the desired topic or distribution of topics in a long-document can be a challenging task. Perhaps for this reason, writing often proceeds by an iterative or hierarchical process, instead of as a purely sequential process.

Variational Autoencoders

Another approach for increasing the expressive of learned density functions is to introduce probabilistic latent variables z which capture much of the learned uncertainty, and then represent the distribution $p(x, z) = p(x|z)p(z)$. Samples from $p(x)$ can then be achieved by marginalizing out over z.

The key appeal of such an approach is that the statistical structure of a learned latent space can often be much simpler than the statistical structure in the visible space. A density with latent variables has a straightforward form, and from a conceptual perspective, leads to straightforward maximum likelihood estimation.

$$p(x) = \prod_{x \in p_{data}(x)} \sum_{z} p(x, z) \tag{1.29}$$

$$log(p(x)) = \sum x \in p_{data}(x) \log(\sum_{z} p(x, z)) \tag{1.30}$$

If the z variable is discrete and has a relatively small number of values, then computing this density is quite straightforward and reasonable. However, if z is continuous or has many potential values, then computing this sum/integral on each update is either slow or impossible. It might be tempting to sample a few values of z for each update, and treat the expression as an expectation over both x and z. However this is both biased and quite misguided, as it ignores the interaction between the log and the sum in the expression. If only a few values of z give rise to a large $p(x, z)$, it's sufficient to give $\log(\sum_z p(x, z))$ a large value. However, if we simply sampled a single z or small number of z on each update, then we would essentially require each z to lead to a $p(x, z)$ with a large value. Intuitively, it is fine if a few or even a single value of the latent variable explains our observed data, and it is not necessary for all of the z values to explain all of the data points.

The variational bound provides a mathematical tool for decomposing this likelihood term involving a log-sum structure into a tractable expectation over both x and z.

Introducing the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ allows us to decompose the marginal log-likelihood of the data under the generative model in terms of the variational free energy and the Kullback-Leibler divergence between the approximate

and true posteriors:

$$\log p_\theta(x) = \mathcal{L}(\theta, \phi; x) + D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right) \tag{1.31}$$

where the Kullback-Leibler divergence is given by

$$D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right]$$

and the variational free energy is given by

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right].$$

Since $D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right)$ measures the divergence between $q_\phi(z \mid x)$ and $p_\theta(z \mid x)$, it is guaranteed to be non-negative. As a consequence, the variational free energy $\mathcal{L}(\theta, \phi; x)$ is always a lower bound on the likelihood, which is sometimes referred to as the variational lower bound.

In the VAE framework, $\mathcal{L}(\theta, \phi; x)$ is often rearranged into two terms:

$$\mathcal{L}(\theta, \phi; x) = \mathcal{L}_z(\theta, \phi; x) + \mathcal{L}_x(\theta, \phi; x) \tag{1.32}$$

where

$$\begin{aligned}
\mathcal{L}_z(\theta, \phi; x) &= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) \\
\mathcal{L}_x(\theta, \phi; x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]
\end{aligned}$$

$\mathcal{L}_x$ can be interpreted as the (negative) expected reconstruction error of x under the conditional likelihood with respect to $q_\phi(z \mid x)$. Maximizing this lower bound pushes the model towards minimizing reconstruction error and minimizing the KL divergence between the approximate posterior $q_\phi(z \mid x)$ and the prior $p_\theta(z)$.

With real-valued z, the *reparametrization trick* Kingma and Welling (2013); Bengio et al. (2013) can be used to propagate the gradient from the decoder network to the encoder network. The mean of z is computed as a deterministic function of x along with the noise term $\varepsilon \sim \mathcal{N}(0, I)$ such that $\mathbf{z}$ has the desired distribution.

Typically teh gaussian distribution is used for the posterior.

$$q_\phi(z \mid x) = \mathcal{N}(z \mid \mu_\phi(x), \text{diag}(\sigma^2_\phi(x))), \tag{1.33}$$

and the *reparametrization trick* allows the value of a sample to be written in terms of the parameters of the distributed estimated by the network and a noise variable ε.

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

which produces values with the desired distribution while permitting gradients to propagate into the encoder network through both $\mu_\phi(x)$ and $\sigma^2_\phi(x)$.

Evaluation Criteria

Under the maximum likelihood approach, a straightforward way of quantifying the success of the model is to compute the model's average likelihood $q_\theta(x)$ on datapoints from a held-out distribution (often referred to as a test dataset). This criteria has some desirable qualities: it is able to detect overfitting and it has a consistency guarantee. On the other hand, it has significant limitations. One discussed by (Arjovsky et al., 2017) is that $q_\theta(x)$ has a value approaching 0 at any points where $p(x)$ has a value greater than zero, the log-likelihood approaches negative infinity (intuitively this can be seen by observing that $log(x)$ approaches negative infinity as x approaches 0). This property is unlikely to be suitable for most applications.

1.2.3 The Adversarial Approach

An alternative to the likelihood maximization approach involves studying a candidate generative model and the differences between the samples from this model and the original data. In practice this usually takes the form of estimating the density ratio between the generating distribution $q_\theta(x)$ and the real data distribution $p(x)$.

$$D_\theta(x) = \frac{p(x)}{q_\theta(x) + p(x)} \tag{1.34}$$

A key motivation behind this approach is that the learning procedure for

$D_\theta(x)$ is equivalent to learning a classifier between the real data and the model's distribution. Classifiers have been extremely successful in deep learning and methods for successfully training classifiers have been widely studied, and inductive biases that are known to be good for classification could also be good for determining the quality of generations. Another motivation for modeling the difference between a model and the data is that it allows the model to become sensitive to any clear difference between real samples and generated samples, which may be a much easier task than simply determining the density of a distribution at a given point.

Noise Contrastive Estimation

(Gutmann and Hyvärinen, 2010) proposed to use a fixed generative model $q_\theta(x)$ and learn a classifier to distinguish between these samples and the real data distribution. Once this density ratio $D_\theta(x)$ is learned, the estimator can be sampled from by using importance sampling. A markov chain monte carlo method could also be used for sampling.

$$D_\theta(x) = \frac{p(x)}{q_\theta(x) + p(x)} \tag{1.35}$$

$$\hat{p}(x) = \frac{D(x)}{1 - D(x)} q_\theta(x) \tag{1.36}$$

A significant limitation in this approach is that a $q_\theta(x)$ must be selected which is very similar to $p(x)$. For example, the expression isn't even well defined if $q_\theta(x)$ doesn't have support everywhere that $p(x)$ has support. And if $q_\theta(x)$ has very small values where $p(x)$ has large values, this pushes $D_\theta(x)$ to 1.0, which leads to very large importance weights and high variance sampling. Intuitively, in a high-dimensional space like an image, a random prior such as a gaussian distribution for $q_\theta(x)$ has no realistic chance of ever producing a realistic image, even though it can happen in theory.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), aimed to leverage the strengths of using a classifier for generation, while avoiding the major weaknesses of noise contrastive estimation. The GAN framework approachs the

generative modeling problem from a perspective inspired by game theory. The model involves training two networks in an adversarial fashion. Rather than using a fixed $q_\theta(x)$, a generator network is trained to produce samples which are similar to the training examples and a discriminator network $D_\theta(x)$ is trained to classify between examples from the training set and examples produced by the generator. The generator is optimized to maximize the probability that the discriminator will classify the generated example as "real". This setup is described as adversarial because the loss for the generator's loss is the opposite of a term in the discriminator's loss.

$$\min_\theta \left[\ell(\mathbb{Q}_\theta; \mathcal{F}) := \sup_{F \in \mathcal{F}} F\left(\mathbb{P}, \mathbb{Q}_\theta\right) \right] . \tag{1.37}$$

For the usual cross-entropy classification objective, this can be rewritten more directly.

$$V(G, D) := \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim q(z)}[\log(1 - D(G(z)))], \tag{1.38}$$

A practical observation from (Goodfellow et al., 2014) is that optimizing the generator network to maximize the value of $V(G, D)$ performs poorly in practice, especially when the support of p and q don't overlap. Instead a non-saturing objective is often used.

$$\max_\theta \mathbb{E}_{z \sim q(z)}[\log(D(G_\theta(z)))] \tag{1.39}$$

Principled Methods for Training GANs

(Arjovsky and Bottou, 2017) showed that the gradient for a GAN generator is not well behaved when the support of $q_\theta(x)$ and $p(x)$ don't overlap. More concretely, they showed that **??** leads to saturation and vanishing gradients for the generator and **??** leads to instability and a lack of useful gradient signal in this situation.

They proposed injecting noise into both $p(x)$ and $q_\theta(x)$ as a way of overcoming this issue in theory, while still learning an estimator of the data distribution.

$$F_\gamma(\mathbb{P}, \mathbb{Q}; \phi) := F(\mathbb{P} * \Lambda, \mathbb{Q} * \Lambda; \phi), \quad \Lambda = \mathcal{N}(\mathbf{0}, \gamma\mathbf{I})\,. \tag{1.40}$$

In this model, the generator is also trained on gradient from samples with the noise injected. So long as sufficient noise is injected, this provides the generator with a density ratio which is well-defined even when the support of the real data and generator distribution don't overlap. It was also shown that the divergence between the distributions with noise injected gives an upper bound on the wasserstein distance between the real distribution and estimating distribution where the tightness of the bound depends on the variance of the noise, which could be controlled by annealing the noise over the course of training.

Wasserstein GAN

A serious problem with GAN training, noted even in its original formulation (Goodfellow et al., 2014) is that on difficult problems, especially early in training, it is difficult to select a generator which has overlapping support with the real data distribution without adding noise (which tends to degrade sample quality). When the generator and the real data distribution do not have overlapping support, the KL-divergence is undefined and the Jensen-Shannon divergence can be discontinuous around these points. While (Roth et al., 2017) proposed an analytical approximation to noise injection, the Wasserstein GAN proposes an alternative approach in which a statistical divergence is used which is continuous and differentiable even when the generating distribution and the real data distribution do not overlap.

A major contribution of (Arjovsky et al., 2017) is a formulation of a GAN objective which corresponds to optimizing the Earth Mover's distance, or wasserstein metric. This is based on the Kantorovich-Rubenstein duality which gives a definition of the wasserstein metric in terms of a supremum over all 1-Lipschitz continuous functions f.

$$W(p, q_\theta) = \sup_{\|f\|_L \leq 1} {}_{x\sim p}[f(x)] - {}_{x\sim q_\theta}\left[f(x)\right] \tag{1.41}$$

In practice, this is achieved by using a neural network discriminator as the function f. The lipschitz constraint on f was enforced in (Arjovsky et al., 2017)

by clipping all of the weights to be within a specified range after each update. (Gulrajani et al., 2017) proposed to use a penalty on the norm of the gradient of the discriminator's output with respect to its inputs. This achieved significant improvements over the clipping approach used in the original WGAN. (Roth et al., 2017) showed that applying the gradient penalty on the original GAN formulation can also achieve good results in practice and can be justified as an analytical approximation to injecting noise into the samples as was theoretically discussed in (Arjovsky and Bottou, 2017).

Spectral Normalization

(Miyato et al., 2018) provided a further refinement over gradient penalty based on two primary critiques: (1) that the gradient penalty only guarantees lipschitz continuity at the data points or wherever it is applied, and not everywhere in the input space and (2) that the gradient penalty has the effect of pushing down the rank of the weight matrices and lowering the expressiveness of the discriminator. (Miyato et al., 2018) showed that the lipschitz constant of the discriminator function is an upper-bound on the lipschitz constant of the function.

$$\begin{aligned}\|f\|_{\text{Lip}} \leq & \|(h_L \mapsto W^{L+1}h_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(h_{L-1} \mapsto W^L h_{L-1})\|_{\text{Lip}} \\ & \cdots \|a_1\|_{\text{Lip}} \cdot \|(h_0 \mapsto W^1 h_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(h_{l-1} \mapsto W^l h_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l).\end{aligned} \tag{1.42}$$

The lipschitz constant $\|g\|_{\text{Lip}}$ for a single linear layer $g : h_{in} \mapsto h_{out}$ is equal to the spectral norm of the weight matrix A, which is equivalent to its largest singular value (the largest eigenvalue of A^*A).

$$\sigma(A) := \max_{h:h\neq 0} \frac{\|Ah\|_2}{\|h\|_2} = \max_{\|h\|_2 \leq 1} \|Ah\|_2, \tag{1.43}$$

Therefore, for a linear layer $g(h) = Wh$, the norm is given by $\|g\|_{\text{Lip}} = \sup_h \sigma(\nabla g(h)) = \sup_h \sigma(W) = \sigma(W)$.

Spectral normalization directly normalizes the spectral norm of the weight matrix

W so that it satisfies the Lipschitz constraint $\sigma(W) = 1$:

$$\overline{W}_{\text{SN}}(W) := W/\sigma(W). \tag{1.44}$$

By normalizing all linear layers in this way, the inequality (1.42) and the fact that $\sigma\left(\overline{W}_{\text{SN}}(W)\right) = 1$ to see that $\|f\|_{\text{Lip}}$ is bounded from above by 1.

In practice, the eigenvalue of the largest singular value for each weight matrix is maintained by using the power method with a persistent estimate of the eigenvector corresponding to the largest eigenvalue. The power method consists of an iterative process of multiplying by matrix and re-normalizing. That the power method results in the eigenvector with the largest eigenvalue may be seen by considering its application on the Jordan-Normal form of the matrix, where the diagonal matrix containing the eigenvalues has the relative value of all but the largest eigenvalue decay with successive iterations.

As the eigenvector is only a single value and the weights change relatively slowly, the spectral normalization method has almost no computational cost, unlike the gradient penalty.

Aside from its computational advantages, there are two major advantages to spectral normalization over the gradient penalty. The first is that spectral normalization only penalizes based on the size of the largest eigenvalue, so there is no pressure to reduce the rank of the weight matrices. Moreover, dividing a matrix by a non-zero constant does not change its rank. On the other hand, (Miyato et al., 2018) showed that weight normalization and gradient penalty both have the effect of pushing down the rank of the weight matrices. This could make it more difficult for the network to learn expressive functions. A second advantage of spectral normalization is that it enforces the lipschitz constraint at all points in the space, whereas gradient penalty only enforces the lipschitz constraint at points where it is applied, usually around data points or on linear interpolations between data points.

Jacobian Clamping

So far we have looked at approaches for improving GAN training which consider modifications to the discriminator and its training objective. An alternative and potentially complementary approach was explored in (Odena et al., 2018) which

consists of an additional objective which encourages the generator's output to not change too much or too little as the latent value z is changed by a small amount. The eigenvalues $\lambda_1, \lambda_2, ...$ and corresponding eigenvectors $v_1, v_2, ...$ of the metric tensor associated with $G(z)$ and its jacobian can be written as follows (Odena et al., 2018).

$$\lim_{||\varepsilon|| \to 0} \frac{||G(z) - G(z + \varepsilon v_k)||}{||\varepsilon v_k||} = \sqrt{\lambda_k} \tag{1.45}$$

The condition number is given by the ratio $\frac{\lambda_k}{\lambda_1}$. The metric tensor is considered to be poorly conditioned if the condition number has a high value. (Odena et al., 2018) proposed to eschew the issue of computing the complete spectrum, which could be quite challenging, in favor of sampling random directions (essentially sampling small random values for εv_k and empirically computing 1.45, and then adding a penalty to encourage these values to fall within a specific range. In practice they achieved good results by setting λ_{min} to 1.0 and λ_{max} to 20.0. Making λ_{max} too small could have the effect of making it too hard for the model to be responsive to the latent variables and setting λ_{min} to be too large could have the effect of making it impossible for the model to learn to give large regions in the space relatively constant density. In practice these hyperparameters would likely need to be tuned depending on the dataset in accordance with these concerns.

Evaluation Criteria

The maximum likelihood approach provided a straightforward, if not completely well motivated, way to quantitatively evaluate generative models within its family. For adversarial approaches, no such criteria is readily apparent. Why is this? The discriminator's score provides an estimate of how much the model's density differs from the true data density at a given point. If the discriminator is able to correctly classify between real data points and generated data points reliably and in a way that generalizes, then it is a clear indicator that the generator is of poor quality. However, if the opposite is true, that the discriminator cannot classify between real and fake, then it could either be because the generator is of high quality, or it could be because the discriminator is somehow limited (in architecture, training procedure, or another characteristic). This means that the discriminator's score cannot reliably used as a way of discerning the quality of a generative model (although in the case

of the Wasserstein GAN, it is at least informative enough to gauge the progress of training) (Arjovsky et al., 2017; Gulrajani et al., 2017).

This basic limitation has motivated the exploration of novel quantitative evaluation criteria for generative models in the adversarial family. Despite having this motivation for their development, both of the criteria that we will discuss are agnostic to the actual form of the generative model, and could equally be applied to models trained using maximum likelihood.

Two different but very closely related methods have seen widespread adoption as methods for quantitatively evaluating adversarially trained generative models. In both cases a fixed pre-trained classifier is used as the basis for the scoring metric. The first is the Inception Score (Salimans et al., 2016), which is defined by:

$$\exp\left(\mathbb{E}_{\mathbf{x}\in q_\theta}[KL(p(y|\mathbf{x})\|p(y)]\right) \tag{1.46}$$

where $\mathbf{x}$ is a GAN sample, $p(y|\mathbf{x})$ is the probability for labels y given by a pre-trained classifier on $\mathbf{x}$, and $p(y)$ is the marginal distribution of the labels in the generated samples according to the classifier. Higher scores are considered better. The intuition behind this metric is that a generator should produce samples from many different classes while at the same time ensuring that each sample is clearly identifiable as belonging from a single class. For example, a generator which only produces samples of a single class will have a poor inception score because $p(y)$ and $p(y|x)$ will be very similar, as it will only reflect that single class. Likewise producing samples which do not give the classifier clear information about the class will tend to make $p(y|x)$ uncertain and more similar to $p(y)$, leading to a poor inception score.

While inception score has been shown to be highly correlated to visual sample quality (real data samples achieve better inception scores than any current models) and tends to give bad scores to clearly deficient GAN models (as well as models early in training), three limitations in Inception Score are readily apparent. One is that the inception score could be pushed beyond the values achievable with real data by a model which produces only a single and clearly identifiable example of each class that the classifier is aware of. This would make $p(y|x)$ very different from $p(y)$ while having high entropy in $p(y)$, and yet the model would clearly lack the diversity of real data, and would be a poor generative model from the statistical divergence perspective. Another limitation is that if the classifier is vulnerable to

adversarial examples, this could hypothetically be exploited to achieve unnaturally high inception scores. This was demonstrated directly in experiments by (Barratt and Sharma, 2018). While this is potentially an issue if researchers are unscrupulous and in a competitive setting, it is unclear if this will occur if a researcher does not intentionally set out to produce adversarial examples for the inception score classifier. Finally, a straightforward problem with inception score is that a very high score can be achieved just by returning the samples from the training set. Thus a generative model which merely memorizes the training data would achieve a high inception score, without doing any learning. The inception score will give low scores to model which underfits and fails to achieve clear samples, but it does not penalize a model at all for memorizing the training data or failing to generalize.

The Frechet Inception Distance (Heusel et al., 2017) was proposed to address some of the limitations inherent in the inception score. It shares the idea of using a fixed pre-trained classifier as its foundation, but instead of assessing the quality of $p(y|x)$ for samples, it instead takes hidden activations from the end of the classifier. The key idea is that the score is high when the distribution of these activations for generated samples is close to the distribution of these activations for real data points. How can we determine if these distributions are indeed close to each other, without simply reproducing the problem of having to train a generative model? While this remains an open question, the proposal in FID is to assume that these hidden states follow a multivariate gaussian distribution (but not necessarily with an isotropic variance). Because these hidden states are from the end of a deep classifier, this multivariate gaussian assumption is much more justified than it would be in the visible space.

To compute the FID score, one fits a multivariate gaussian $N(m, C)$ to the activations for the real test samples and fits a separate multivariate gaussian $N(m_w, C_w)$ to the activations for the generated samples. From this, the Frechet Distance between the distributions has a surprisingly tractable form which does not require inverting the covariance matrices C or C_w.

$$\|m - m_w\|_2^2 + \mathrm{Tr}\big(C + C_w - 2\big(CC_w\big)^{1/2}\big) \tag{1.47}$$

(Heusel et al., 2017) studied several artificial deformations of the data and showed that FID scores gradually became worse with increasing corruption. More specifically: they studied artifically injecting independent noise into the images,

removing random regions of the images, swirling the images, salt and pepper noise, and injecting examples from another dataset. On all of these increasing corruption led to worse FID scores, whereas only injecting unrelated samples led to worse inception scores.

Perhaps most important, FID can be evaluated on the test data, so it can directly test against overfitting, unlike inception scores. Moreover, generating a single high quality example for each class (at the expense of overall diversity) could still hurt FID by giving the hidden states of the generated samples an unnatural distribution. While these metrics are now widely used in measuring the quality of generative models (Odena et al., 2018; Miyato et al., 2018; Karras et al., 2017), they are still highly dependent on the choice of the classifier for evaluation and lack statistical consistency guarantees.

2 Fortified Networks

2.1 Prologue to the Article

Fortified Networks: Improving the Robustness of Deep Networks by Modeling the Manifold of Hidden Representations. Alex Lamb, Jonathan Binas, Anirudh Goyal, Dmitriy Serdyuk, Sandeep Subramanian, Ioannis Mitliagkas, Yoshua Bengio.

Under Review, International Conference on Machine Learning (ICML) 2018

Personal Contribution. The core idea emerged after Anirudh Goyal and Alex Lamb were working through the idea of whether the distribution of hidden states differed between adversarial examples and non-adversarial examples. We later refined this prototype to work on the cleverhands code base, with Jonathan Binas figuring out how to get strong results as well as blackbox attacks. Dima Serdyuk helped to get resnets working on the cifar dataset. Sandeep Subramanian was able to get fortified networks working with autoencoders on multiple hidden layers, significantly improving our results. Ioannis Mitliagkas and Yoshua Bengio contributed in discussions and helped to write much of the paper.

My contribution to this effort involved exploration of the initial concept, experimentation including data pipeline development, and content generation and review in the paper.

Affiliations

Alex Lamb: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Jonathan Binas: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Anirudh Goyal: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Dmitriy Serdyuk: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Sandeep Subramanian: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Ioannis Mitliagkas: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Yoshua Bengio: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CIFAR Senior Fellow

Funding We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

2.2 Abstract

Deep networks have achieved impressive results across a variety of important tasks. However a known weakness is a failure to perform well when evaluated on data which differ from the training distribution, even if these differences are very small, as is the case with adversarial examples. We propose Fortified Networks, a simple transformation of existing networks, which "fortifies" the hidden layers in a deep network by identifying when the hidden states are off of the data manifold, and maps these hidden states back to parts of the data manifold where the network performs well. Our principal contribution is to show that fortifying these hidden states improves the robustness of deep networks and our experiments (i) demonstrate improved robustness to standard adversarial attacks in both black-box and white-box threat models; (ii) suggest that our improvements are not primarily due to the gradient masking problem and (iii) show the advantage of doing this fortification in the hidden layers instead of the input space.

2.3 Introduction

Deep neural networks have been very successful across a variety of tasks. This success has also driven applications in domains where reliability and security are critical, including self-driving cars (Bojarski et al., 2016), health care, face recognition (Sharif et al., 2017), and the detection of malware (LeCun et al., 2015). Security concerns emerge when an agent using the system could benefit from the system performing poorly. Reliability concerns come about when the distribution of input data seen during training can differ from the distribution on which the model is evaluated.

Adversarial examples (Goodfellow et al., 2014) is a method to attack neural network models. This attack applies a small perturbation to the input that changes the predicted class. It is notable that it is possible to produce a perturbation small enough that it is not noticeable with a naked eye. It has been shown that simple gradient methods allow one to find a modification of the input that often changes the output class (Szegedy et al., 2013; Goodfellow et al., 2014). More recent work demonstrated that it is possible to create a patch such that even when presented on the camera, it changes the output class with high confidence (Brown et al., 2017).

Defences against adversarial examples have been developed as a response. Some of the most prominent classes of defences include feature squeezing (Xu et al., 2017), adapted encoding of the input (Jacob Buckman, 2018), and distillation-related approaches (Papernot et al., 2015). Existing defenses provide some robustness but most are not easy to deploy. In addition, many have been shown to be vulnerable to gradient masking. Still others require training a generative model directly in the visible space, which is still difficult today even on relatively simple datasets.

Our goal is to provide a method which (i) can be generically added into an existing network; (ii) robustifies the network against adversarial attacks and (iii) provides a reliable signal of the existence of input data that do not lie on the manifold on which it the network trained. The ability of generative models, used directly on the input data, to improve robustness is not new. Our main contribution is that we employ this robustification on the distribution on the learned hidden representations instead making the identification of off-manifold examples easier Figure 3.1.

We propose *Fortified Networks* ♖. Fortification consists of using denoising

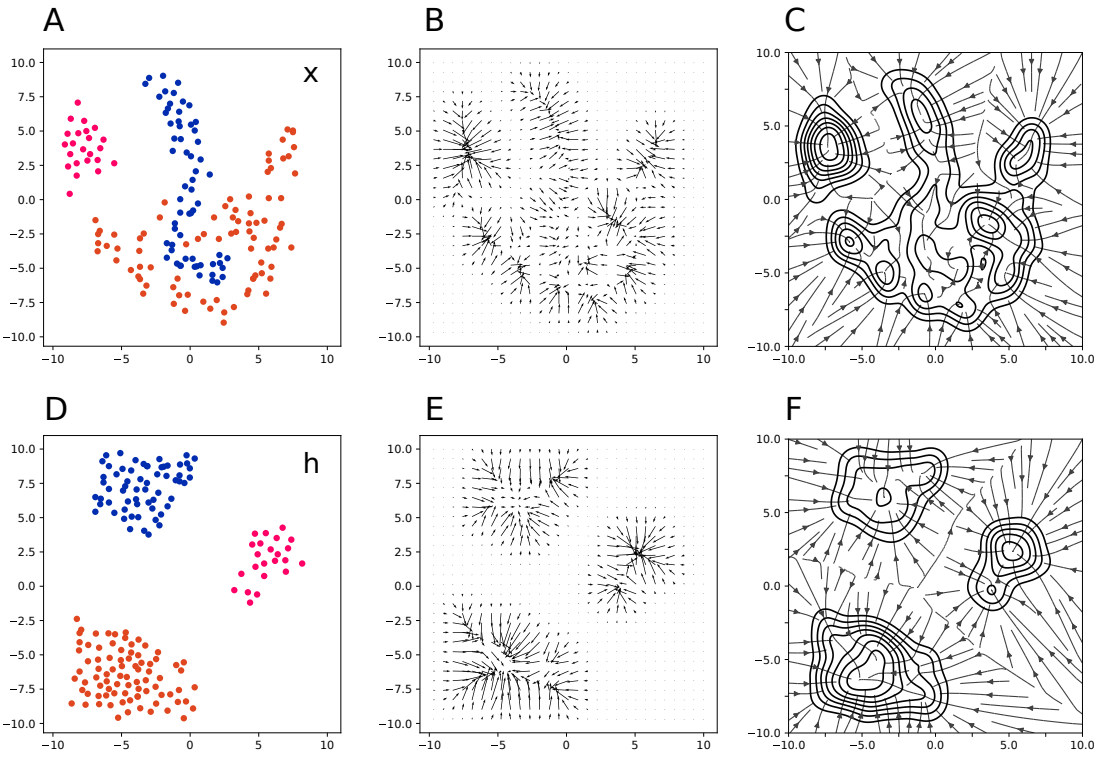

Figure 2.1 – Illustration of the autoencoder dynamics in the input space (top) and in abstract hidden space (bottom). The leftmost panels show data points from three different classes, the middle panels show vector fields describing the autoencoder dynamics, and the rightmost panels show a number of resulting trajectories and basins of attraction. The key motivation behind Fortified Networks is that directions which point off the data manifold are easier to identify in an abstract space with simpler statistical structure, making it easier to map adversarial examples back to the projected data manifold.

autoencoders to "decorate" the hidden layers of the original network. Fortification meets all three goals stated above. We discuss the intuition behind the fortification of hidden layers and lay out some of the method's salient properties. We evaluate our proposed approach on MNIST, Fashion-MNIST, CIFAR10 datasets against whitebox and blackbox attacks.

The rest of the paper is structured in the following way. Section 2.4 gives a detailed overview of the background on the adversarial attacks and denoising autoencoders used in this work. Section 2.5 presents our proposed methods for the defence against adversarial examples, Section 2.6 describes the experimental procedure and Section 2.7 provides the experimental results and a comparison to previous approaches. Finally, Section 2.8 puts this work into the context of previous

publications and Section 2.9 concludes.

2.4 Background

2.4.1 The Empirical Risk Minimization Framework

Let us consider a standard classification task with an underlying data distribution $\mathcal{D}$ over pairs of examples $x \in \mathbb{R}^d$ and corresponding labels $y \in [k]$. We also assume that we are given a suitable loss function $L(\theta, x, y)$, for instance the cross-entropy loss for a neural network. As usual, $\theta \in \mathbb{R}^p$ is the set of model parameters. Our goal then is to find model parameters θ that minimize the risk $\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x, y, \theta)]$. This expectation cannot be computed, therefore a common approach is to to minimize the empirical risk $1/N \sum_D L(x, y, \theta)$ taking into account only the examples in a given dataset D.

2.4.2 Adversarial Attacks and Robustness

While the empirical risk minimization framework has been very successful and often leads to excellent generalization, it has the significant limitation that it doesn't guarantee robustness, and more specifically performance on examples off the data manifold. Madry et al. (2017) proposed an optimization view of adversarial robustness, in which the adversarial robustness of a model is defined as a min-max problem

$$\min_{\theta} \rho(\theta), \qquad \text{where} \tag{2.1}$$

$$\rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta \in \mathcal{S}} L(\theta, x + \delta, y) \right] . \tag{2.2}$$

2.4.3 Denoising Autoencoders

Denoising autoencoders (DAEs) are neural networks which take a noisy version of an input (for example, an image) and are trained to predict the noiseless version of that input. This approach has been widely used for feature learning and generative

modeling in deep learning (Bengio et al., 2013). More formally, denoising autoencoders are trained to minimize a reconstruction error or negative log-likelihood of generating the clean input. For example, with Gaussian log-likelihood of the clean input given the corrected input, r the learned denoising function, C a corruption function with Gaussian noise of variance σ^2, the reconstruction loss is

$$\widehat{\mathcal{L}} = \frac{1}{N} \sum_{n=1}^{N} \left(\left\| r(C_\sigma(x^{(n)})) - x^{(n)} \right\|_2^2 \right). \tag{2.3}$$

Alain et al. (2012) demonstrated that with this loss function, an optimally trained denoising autoencoder's reconstruction vector is proportional to the gradient of the log-density:

$$\frac{r_\sigma(x) - x}{\sigma^2} \to \frac{\partial \log p(x)}{\partial x} \quad \text{as} \quad \sigma \to 0. \tag{2.4}$$

This theory establishes that the reconstruction vectors from a well-trained denoising autoencoder form a vector field which points in the direction of the data manifold. However, Alain et al. (2012) showed that this may not hold for points which are distant from the manifold, as these points are rarely sampled during training. In practice, denoising autoencoders are not just trained with tiny noise but also with large noises, which blurs the data distribution as seen by the learner but makes the network learn a useful vector field even far from the data.

2.5 Fortified Networks

We propose the use of DAEs inserted at crucial points between layers of the original neural network in order to clean up the transformed data points which may lie away from the original data manifold. Intuitively, the method aims to regularize the hidden representations by keeping the activations on the surface of the corresponding projected data manifold through the application of a DAE trained on the hidden representations (on the original clean data). We argue that applying the DAEs on the hidden layers—as opposed to the raw input signal—facilitates learning, while providing a stronger protection from adversarial attacks. As illustrated in Figure 3.1, we hypothesize that more abstract representations associated with deeper networks are easier to clean up because the transformed data manifolds are flatter.

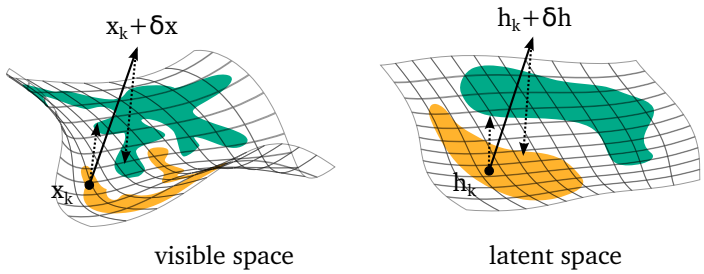

Figure 2.2 – An illustration of the process of mapping back to the manifold in the visible space (left) and the hidden space (right). The shaded regions represent the areas in the space which are occupied by data points from a given class (they do *not* represent decision boundaries).

The flattening of data manifolds in the deeper layers of a neural network was first noted experimentally by Bengio et al. (2013). We provide experimental support for these claims in Section 2.6.

♖ **Layer fortification** ♖ Our method works by substituting a hidden layer h_k with a denoised version. We feed the signal h_k through the encoder network, E_k, and decoder network, D_k, of a DAE for layer k, which yields the denoised version, $h_k^{decoded}$:

$$h_k^{decoded} = D_k(E_k(h_k + n_k)), \tag{2.5}$$

where n_k is white Gaussian noise of variance σ^2 and appropriate shape. We call the resulting layer, a *fortified layer* and the resulting network the *fortified network* corresponding to the original network.

For training purposes, we treat the DAEs as part of the fortified network, backpropagate through and train all weights jointly. Aside from the original classification loss, $\mathcal{L}_c$, we also include the classification loss from the adversarial objective, $\mathcal{L}_c(\widetilde{y})$ and we introduce a dual objective for the DAEs.

— **Reconstruction loss.** For a mini-batch of N clean examples, $x^{(1)}, \ldots, x^{(N)}$, each hidden layer $h_k^{(1)}, \ldots, h_k^{(N)}$ is fed into a DAE loss, similar to (2.3):

$$\mathcal{L}_{rec,k} = \frac{1}{N} \sum_{n=1}^{N} \left\| D_k \left(E_k \left(h_k^{(n)} + n_k \right) \right) - h_k^{(n)} \right\|_2^2.$$

— **Adversarial loss.** We use some adversarial training method to produce the perturbed version of the mini-batch, $\widetilde{x}^{(1)}, \ldots, \widetilde{x}^{(N)}$, where $\widetilde{x}^{(i)}$ is a small

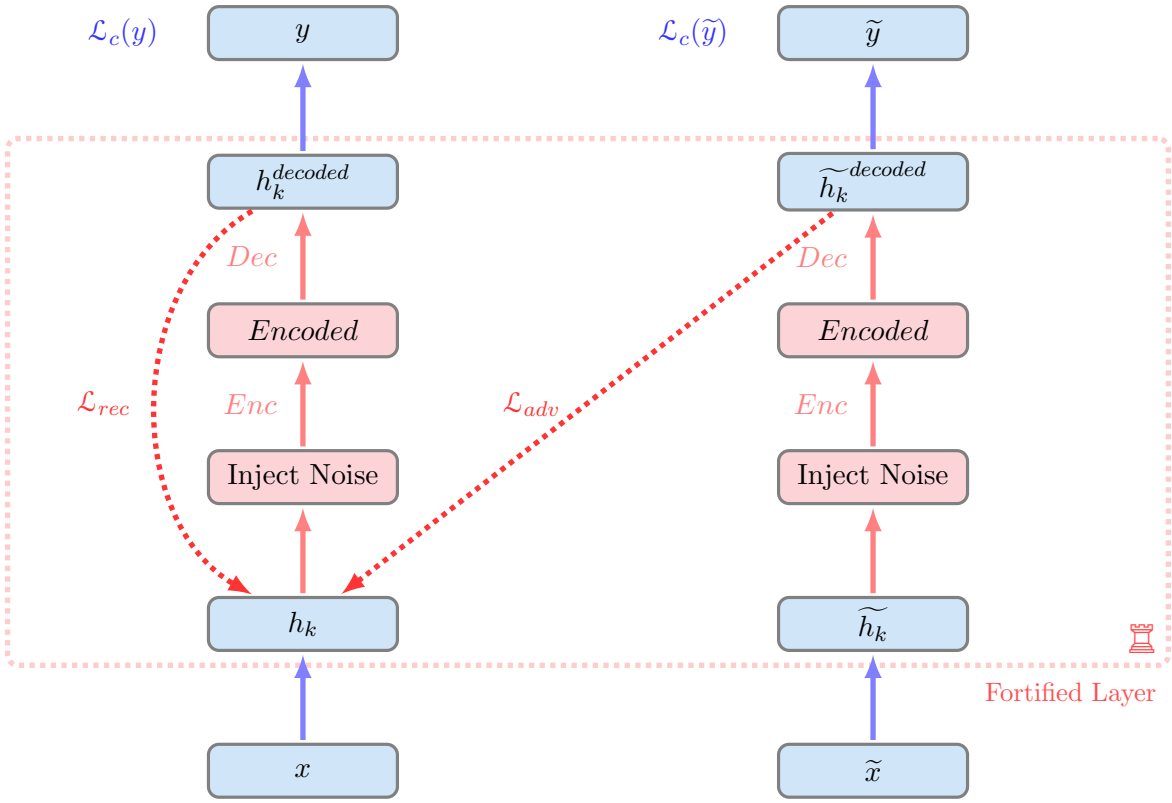

Figure 2.3 – Diagram illustrating a one-layer fortified network. A network is evaluated with a data sample x and its corresponding adversarial example $\widetilde{x}$. Hidden units h_k and $\widetilde{h_k}$ are corrupted with noise, encoded with the encoder *Enc*, and decoded with the decoder *Dec*. The autoencoder (denoted by the red color) is trained to reconstruct the hidden unit h_k that corresponds to the clean input. Dotted lines are two reconstruction costs: for a benign ($\mathcal{L}_{rec}$) and adversarial examples ($\mathcal{L}_{adv}$). Note that a layer can be fortified at any position in the network.

perturbation of $x^{(i)}$ which is designed to make the network produce the wrong answer. The corresponding hidden layer $\widetilde{h}_k^{(1)}, \ldots, \widetilde{h}_k^{(N)}$ (using the perturbed rather than original input) is fed into a similar DAE loss:

$$\mathcal{L}_{adv,k} = \frac{1}{N} \sum_{n=1}^{N} \left\| D_k \left(E_k \left(\widetilde{h}_k^{(n)} + n_k \right) \right) - h_k^{(n)} \right\|_2^2$$

where we note that the target reconstruction for denoising is the clean version of the hidden layer, without noise and without adversarial perturbation.

To build a fortified network, we can apply this fortification process to some or all the layers. The final objective used for training the fortified network includes

the classification loss and all reconstruction and adversarial losses:

$$\mathcal{L} = \mathcal{L}_c(y) + \mathcal{L}_c(\widetilde{y}) + \lambda_{rec} \sum_k \mathcal{L}_{rec,k} + \lambda_{adv} \sum_k \mathcal{L}_{adv,k},$$

where $\lambda_{rec} > 0$ and $\lambda_{adv} > 0$ tune the strength of the DAE terms. This kind of training process allows for the production of hidden representations robust to small perturbations, and in particular, to adversarial attacks.

Off-manifold signaling The reconstruction losses act as a reliable signal for detecting off-manifold examples (cf. Section 2.6). This is a particularly useful property in practice: not only can we provide more robust classification results, we can also sense and suggest to the analyst or system when the original example is either adversarial or from a significantly different distribution.

Motivation for when and where to use fortified layers We have discussed advantages to placing fortified layers in the hidden states instead of the input space (with further discussion in section 2.8.1), but the question of where exactly fortified layers need to be placed remains unanswered. Is it just the final hidden layer? Is it every hidden layer? We outline two important considerations regarding this issue:

1. In the higher-level hidden layers, it is much easier for the network to identify points which are off of the manifold or close to the margin. The former is directly experimentally demonstrated in 2.4.
2. At the same time, the higher level hidden layers may already look like points that are not adversarial due to the effect of the adversarial perturbations in the earlier layers. While we are not aware of any formal study of this phenomenon, it is clearly possible (imagine for example a fortified layer on the output from the softmax, which could only identify unnatural combinations of class probabilities).
3. Given these opposing objectives, we argue for the inclusion of multiple fortified layers across the network.

In the next section we describe a number of experiments to evaluate the practical merit of Fortified Networks.

2.6 Experiments

2.6.1 Attacks

We evaluated the performance of our model as a defense against adversarial attacks. We focused on two of the most popular and well-studied attacks, the Fast Gradient Sign Method (FGSM, Goodfellow et al., 2014) which is popular as it only requires a single step and can still be effective against many networks. Secondly, we consider the projected gradient descent attack (Kurakin et al., 2016) which is slower than FGSM as it requires many iterations, but has been shown to be a much stronger attack (Madry et al., 2017).

Additionally, we consider both white-box attacks (where the attackers knows the model) and black-box attacks (where they don't, but they have access to the training set).

Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) Goodfellow et al. (2014) is a simple one-step attack that produces ℓ_∞-bounded adversaries via the following gradient based perturbation.

$$\widetilde{x} = x + \varepsilon \operatorname{sgn}(\nabla_x L(\theta, x, y)). \tag{2.6}$$

Projected Gradient Descent

The projected gradient descent attack (Madry et al., 2017), sometimes referred to as FGSM^k, is a multi-step extension of the FGSM attack characterized as follows:

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x, y)) \right) \tag{2.7}$$

initialized with x^0 as the clean input x and with the corrupted input $\widetilde{x}$ as the last step in the sequence.

2.6.2 The Gradient Masking and Gradient Obfuscation Problem

A significant challenge with evaluating defenses against adversarial attacks is that many attacks rely upon a network's gradient. Methods which reduce the quality of this gradient, either by making it flatter or noisier can lead to methods which lower the effectiveness of gradient-based attacks, but which are not actually robust to adversarial examples (Athalye et al., 2017; Papernot et al., 2016). This process, which has been referred to as gradient masking or gradient obfuscation, must be analyzed when studying the strength of an adversarial defense.

One method for studying the extent to which an adversarial defense gives deceptively good results as a result of gradient masking relies on the observation that black-box attacks are a strict subset of white-box attacks, so white-box attacks should always be at least as strong as black-box attacks. If a method reports much better defense against white-box attacks, it suggests that the selected white-box attack is underpowered as a result of gradient masking. Another test for gradient masking is to run an iterative search, such as projected gradient descent (PGD) with an unlimited range for a large number of iterations. If such an attack is not completely successful, it indicates that the model's gradients are not an effective method for searching for adversarial images, and that gradient masking is occurring. Still another test is to confirm that iterative attacks with small step sizes always outperform single-step attacks with larger step sizes (such as FGSM). If this is not the case, it may suggest that the iterative attack becomes stuck in regions where optimization using gradients is poor due to gradient masking.

2.6.3 Reconstruction Error as a Heuristic for Deviation from the Manifold

The theory in Alain et al. (2012) established that a well-trained denoising autoencoder's reconstruction vector $r(x) - x$ points in the direction of the gradient of the log-density. Thus, critical points in the model's log-density will have a reconstruction error of zero. While it is not guaranteed to hold for arbitrary densities, we investigated whether reconstruction error can serve as a practical heuristic for how much a point deviates from the data manifold. If each data point were a local maximum in the log-density and the log-density has no other critical

points, then points on the data manifold are guaranteed to have lower reconstruction error.

2.7 Results

For details about the specifics of our model architectures and hyperparameters we refer readers to sections 1.1 and 1.2 of our supplementary material. With all experiments, we use the same attacks (with identical parameters) at training and test time to generate adversarial examples. An important point to note here is that all of the autoencoders in our fortified layers used a single hidden layer with tied weights. In the case of convolutional autoencoders we always used a stride of 1 and (5,5) kernels.

Table 2.1 – Accuracies against white-box MNIST attacks with FGSM, where the model is a convolutional net. We used the standard FGSM attack parameters with an ε of 0.3 and compare against published adversarial training defenses. We also performed ablations where we considered removing the reconstruction error on adversarial examples $\mathcal{L}_{adv}$ as well as switching the activation function in the fortified layers from leaky relu to tanh, which we found to slightly help in this case. While our baseline and pre-fortified networks used relu activations, we found that by using a leaky relu in all layers the accuracy on FGSM $\varepsilon = 0.3$ could be improved to 99.2% with standard adversarial training, suggesting that both our own baselines and those reported in the past have been too weak.

Model	FGSM
Adv. Train (Madry et al., 2017)	95.60
Adv. Train Jacob Buckman (2018)	96.17
Adv. Train (ours)	96.36
Adv. Train No-Rec (ours)	96.47
Quantized (Jacob Buckman, 2018)	96.29
One-Hot (Jacob Buckman, 2018)	96.22
Thermometer (Jacob Buckman, 2018)	95.84
Our Approaches	
Fortified Network - Conv, w/o $\mathcal{L}_{adv}$	96.46
Fortified Network - Conv	**97.97**

We also ran the above experiment with FGSM and an ε of 0.1 to compare directly with (Erraqabi et al., 2018) and obtain 98.34% accuracy on adversarial examples compared to their 96.10%.

Table 2.2 – Accuracies against white-box MNIST attacks with PGD with an ε of 0.1, where our model is a convnet.

Model	PGD
Baseline Adv. Train	96.98
Fortified Network - Conv (ours)	**98.09**

Table 2.3 – Accuracies against white-box CIFAR attacks with FGSM using ($\varepsilon = 0.3$), where each model is a convnet. Our baseline adversarial training is the resnet model provided in (Nicolas Papernot, 2017)

Model	FGSM
Baseline Adv. Train	79.57
Fortified Networks - Conv (ours)	**80.47**

2.7.1 Recurrent Networks

RNNs are often trained using teacher forcing, which refers to the use of the ground-truth samples y_t being fed back into the model and conditioning the prediction of later outputs. These fed back samples force the RNN to stay close to the ground-truth sequence. However, when generating at test time, during the ground truth sequence is not available. We investigated if Fortified Networks could be used to detect when sampling from a teacher-forcing model moves off the manifold. To this end we train a language model on the standard Text8 dataset, which is derived from Wikipedia articles. We trained a single-layer LSTM with 1000 units at the character-level, and included fortified layers between the hidden states and the output on each time step. As seen in table 2.7, the ratios of these reconstruction errors increases steadily as we increase the number of sampling steps and diverge away from the distribution of training sequences, providing empirical support for the notion that fortified layers effectively measure when the data moves off of the training manifold.

Table 2.4 – Accuracies against white-box CIFAR attacks with FGSM using the standard ($\varepsilon = 0.03$), where each model is a convnet. Our baseline adversarial training is the resnet model provided in (Nicolas Papernot, 2017)

Model	FGSM
Baseline Adv. Train (ours)	79.34
Quantized (Buckman)	53.53
One-Hot (Buckman)	68.76
Thermometer (Buckman)	80.97
Fortified Networks (autoencoder on input space with loss in hidden states)	79.77
Fortified Networks - Conv (ours)	**81.80**

2.8 Related Work

2.8.1 Using Generative Models as a Defense

The observation that adversarial examples often consist of points off of the data manifold and that deep networks may not generalize well to these points motivated (Gu and Rigazio, 2014; Ilyas et al., 2017; Pouya Samangouei, 2018; Liao et al., 2017) to consider the use of the generative models as a defense against adversarial attacks. Ilyas et al. (2017); Gilmer et al. (2018) also showed the existence of adversarial examples which lie on the data manifold, and (Ilyas et al., 2017) showed that training against adversarial examples forced to lie on the manifold is an effective defense. Our method shares a closely related motivation to these prior works, with a key difference being that we propose to consider the manifold in the space of learned representations, instead of considering the manifold directly in the visible space. One motivation for this is that the learned representations have a simpler statistical structure (Bengio et al., 2012a), which makes the task of modeling this manifold and detecting unnatural points much simpler. Learning the distribution directly in the visible space is still very difficult (even state of the art models fall short of real data on metrics like Inception Score) and requires a high capacity model. Additionally working in the space of learned representations allows for the use of a relatively simple generative model, in our case a small denoising autoencoder.

Table 2.5 – Accuracies against white-box attacks on Fashion MNIST. For PGD we used $\varepsilon = 0.1$ and for FGSM we experimented with $\varepsilon = 0.1$ and $\varepsilon = 0.3$. Compared with DefenseGAN (Pouya Samangouei, 2018).

Model	FGSM ($\varepsilon = 0.1$)	FGSM ($\varepsilon = 0.3$)	PGD ($\varepsilon = 0.1$)
DefenseGAN	n/a	89.60	n/a
n/a			
Our Approaches			
Baseline Adv. Train - Conv,ReLU	86.14	90.66	77.49
Baseline Adv. Train - Conv,LReLU	89.10	88.8	77.90
Fortified Nets - Conv (ours)	**89.86**	**91.31**	**79.54**

Ilyas et al. (2017) proposed to work around these challenges from working in the visible space by using the Deep Image Prior instead of an actual generative model. While this has the advantage of being a model that doesn't require a special training procedure (as deep image prior is a separate optimization process for each example) it may be limited in the types of adversarial attacks that it's resistant to, and it would provide no defense against adversarial attacks which are in the range of a convolutional network, which have been shown to exist (Chaowei Xiao, 2018).

Another key difference between our work and (Ilyas et al., 2017; Pouya Samangouei, 2018) is that both DefenseGAN and the Invert-and-Classify approach use an iterative search procedure at inference time to map observed data points onto nearby points on the range of the generator. On the other hand, our approach uses small denoising autoencoders that are used in the same way (i.e. a simple forward application) during both training and testing. The use of such an iterative procedure presents challenges for evaluation, as it is possible for gradients to vanish while doing backpropagation through such a procedure, which may lead to an overestimate in the strength of the defense due to the gradient masking problem (Papernot et al., 2016; Athalye et al., 2018). One indicator of the gradient masking problem is black-box attacks outperforming white-box attacks, which is an indicator of under-powered

Table 2.6 – Accuracies against blackbox MNIST attacks with adversarial training. Reporting 50/50 results compared to previous works (Jacob Buckman, 2018, JB) and (Pouya Samangouei, 2018, PS). The test error on clean examples is in parenthesis.

Model	FGSM
OneHot (JB)	95.96 (98.83)
ThermoEnc (JB)	96.97 (98.08)
DefenseGAN fc→conv (PS)	92.21 (n/a)
DefenseGAN conv→conv (PS)	93.12 (n/a)
Adv. Train fc→conv (PS)	96.68 (n/a)
Adv. Train conv→conv (PS)	96.54 (n/a)
Our Approaches	
Baseline Adv. Train	93.83 (98.95)
Fortified Network w/o $\mathcal{L}_{adv}$, $\mathcal{L}_{rec}$	96.98 (99.17)
Fortified Network	**97.82** (98.93)

attacks as black-box attacks are a strict subset of white-box attacks. This indicator of gradient obfuscation was present in the work of Pouya Samangouei (2018) where black-box attacks were generally stronger against their defense, but with our method we observe very similar defense quality against black-box and white-box attacks.

(Gu and Rigazio, 2014; Liao et al., 2017) both considered using an autoencoder as a pre-processing step in the input space. Interestingly (Liao et al., 2017) used a loss function defined in the space of the hidden states, but still used autoencoders directly in the input space.

2.8.2 Adversarial Hidden State Matching

Erraqabi et al. (2018) demonstrate that adversarially matching the hidden layer activations of regular and adversarial examples improves robustness. This work shared the same motivation of using the hidden states to improve robustness, but differed in that they used an adversarial objective and worked in the original hidden states instead of using a generative model (in our case, the DAE in the fortified layers). We present direct experimental comparisons with their work in section 2.7.

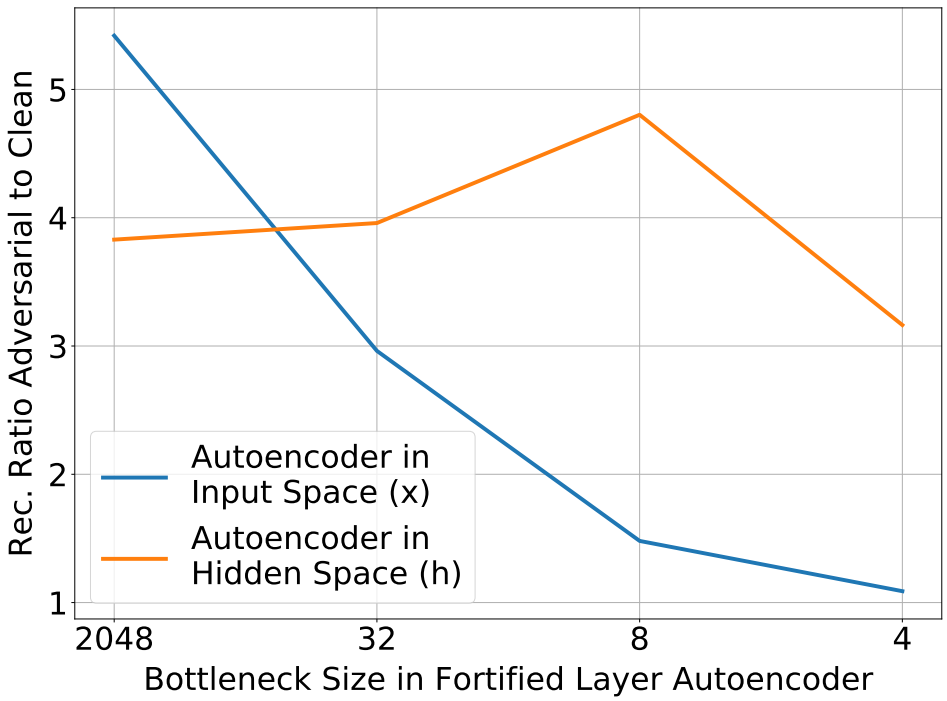

Figure 2.4 – We added fortified layers with different capacities to MLPs trained on MNIST, and display the value of the total reconstruction errors for adversarial examples divided by the total reconstruction errors for clean examples. A high value indicates that adversarial examples have high reconstruction error. We considered fortified layers with autoencoders of different capacities. Our results support the central motivation for fortified networks: that off-manifold points can much more easily be detected in the hidden space (as seen by the relatively constant ratio for the autoencoder in h space) and are much harder to detect in the input space (as seen by this ratio rapidly falling to zero as the autoencoder's capacity is reduced).

2.8.3 Denoising Feature Matching

Warde-Farley and Bengio (2016) proposed to train a denoising autoencoder in the hidden states of the discriminator in a generative adversarial network. The generator's parameters are then trained to make the reconstruction error of this autoencoder small. This has the effect of encouraging the generator to produce points which are easy for the model to reconstruct, which will include true data points. Both this and Fortified Networks use a learned denoising autoencoder in the hidden states of a network. A major difference is that the denoising feature matching work focused on generative adversarial networks and tried to minimize

Table 2.7 – We trained Fortified Networks on a single-layer LSTM on the Text-8 dataset, with fortified layers added between each step. We recorded the ratio between reconstruction error on the testing set during both teacher forcing mode and sampling mode (where the model is supplied with its own outputs as inputs for the next step). The motivation is that the outputs should gradually move off of the manifold with more sampling steps, which is indicated by a high reconstruction error ratio, which makes it an interesting tool for monitoring or potentially fixing this problem.

Sampling Steps	Error Ratio
50	1.03
180	1.12
300	1.34

reconstruction error through a learned generator network, whereas our approach targets the adversarial examples problem. Additionally, our objective encourages the output of the DAE to denoise adversarial examples so as to point back to the hidden state of the original example, which is different from the objective in the denoising feature matching work, which encouraged reconstruction error to be low on states from samples from the generator network.

2.8.4 Adversarial Spheres

Gilmer et al. (2018) studied the existence of adversarial examples in the task of classifying between two hollow concentric shells. Intriguingly, they prove and construct adversarial examples which lie on the data manifold (although Ilyas et al. (2017) also looked for such examples experimentally using GANs). The existence of such on-manifold adversarial examples demonstrates that a simplified version of our model trained with only $\mathcal{L}_{rec}$ could not protect against all adversarial examples. However, training with $\mathcal{L}_{adv}$ encourages the fortified layers to map back from points which are not only off of the manifold, but also to map back from points which are hard to classify, allowing Fortified Networks to also potentially help with on-manifold adversarial examples as well.

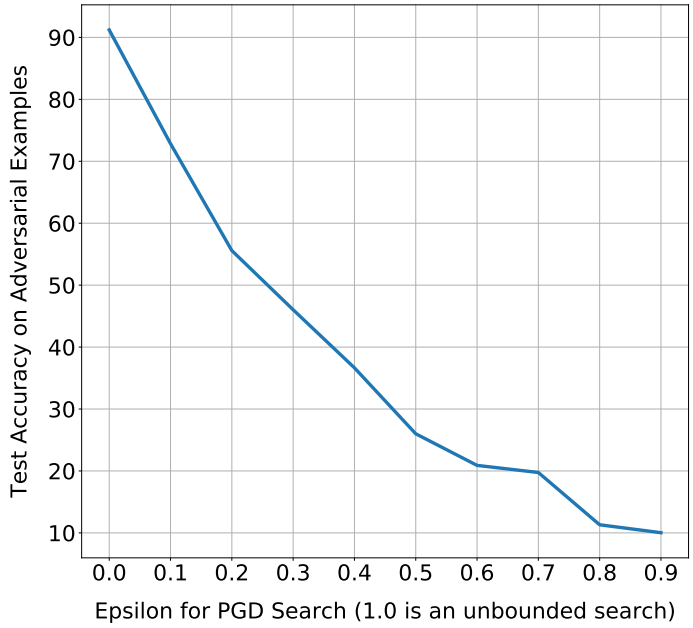

Figure 2.5 – We ran a fortified network on Fashion-MNIST using adversarial training with PGD for a variety of ε values, each for 5 epochs. The motivation behind this experiment, suggested by Athalye et al. (2018) is confirming if unbounded ($\varepsilon = 1$) adversarial attacks are able to succeed. A defense which succeeds primarily by masking or obfuscating the gradients would fail to bring the accuracy to zero even with an unbounded attack. As can be seen, unbounded attacks against Fortified Networks succeed when given a sufficiently large ε, which is evidence against gradient masking.

2.9 Conclusion

Protecting against adversarial examples could be of paramount importance in mission-critical applications. We have presented Fortified Networks, a simple method for the robustification of existing deep neural networks. Our method is

— Practical: fortifying an existing network entails introducing DAEs between the hidden layers of the network and can be automated. We are preparing a PyTorch module that does exactly that and will release it for the deep learning community to use shortly. Furthermore, the DAE reconstruction error at test time is a reliable signal of distribution shift, that is examples unlike those encountered during training. High error can signify either adversarial attacks or significant domain shift; both are important cases for

the analyst or system to be aware of.

— Effective: We showed results that improve upon the state of the art on defenses for adversarial attacks on MNIST and provides improvement on CIFAR and Fashion-MNIST.

Limitations The cost of the proposed method is the extended training time due to the search for an adversarial example and training the autoencoder. The added cost of the fortified layers over adversarial training by itself is relatively small, and is also much easier and simpler than training a full generative model (such as a GAN) in the input space. Layer fortification typically involves smaller DAEs that require less computation. Additionally, we have shown improvements on ResNets where only two fortified layers are added, and thus the change to the computational cost is very slightly. At the same time, fortified networks have only been shown to improve robustness when used alongside adversarial training, which is expensive for iterative attacks.

3 GibbsNet

3.1 Prologue to the Article

GibbsNet: Iterative Adversarial Inference for Deep Graphical Models. Alex Lamb, R Devon Hjelm, Yaroslav Ganin, Joseph Paul Cohen, Aaron Courville, Yoshua Bengio.

Neural Information Processing Systems (NIPS) 2017

Personal Contribution. The main idea came from Yoshua Bengio suggesting that hierarchical GANs with multiple discriminators and local generators could benefit from an undirected sampling procedure.

Alex Lamb did the initial prototyping and found that this model had interesting properties, even in the absence of a hierarchy. Yaroslav Ganin and Devon Hjelm worked to produce the main experiments and wrote much of the paper. Joseph Paul Cohen did the T-SNE experiments and helped with semi-supervised learning. Aaron Courville and Yoshua Bengio contributed in discussions and helped to write the paper.

My contribution to this effort involved exploration of the initial concept, experimentation including data pipeline development, and content generation and review in the paper.

Affiliations

Alex Lamb: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Devon Hjelm: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Yaroslav Ganin: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Joseph Paul Cohen: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Aaron Courville: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CIFAR Fellow

Yoshua Bengio: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CIFAR Senior Fellow

Funding We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

3.2 Abstract

Directed latent variable models that formulate the joint distribution as $p(x, z) = p(z)p(x \mid z)$ have the advantage of fast and exact sampling. However, these models have the weakness of needing to specify $p(z)$, often with a simple fixed prior that limits the expressiveness of the model. Undirected latent variable models discard the requirement that $p(z)$ be specified with a prior, yet sampling from them generally requires an iterative procedure such as blocked Gibbs-sampling that may require many steps to draw samples from the joint distribution $p(x, z)$. We propose a novel approach to learning the joint distribution between the data and a latent code which uses an adversarially learned iterative procedure to gradually refine the joint distribution, $p(x, z)$, to better match with the data distribution on each step. GibbsNet is the best of both worlds both in theory and in practice. Achieving the speed and simplicity of a directed latent variable model, it is guaranteed (assuming the adversarial game reaches the virtual training criteria global minimum) to produce samples from $p(x, z)$ with only a few sampling iterations. Achieving the expressiveness and flexibility of an undirected latent variable model, GibbsNet does away with the need for an explicit $p(z)$ and has the ability to do attribute prediction, class-conditional generation, and joint image-attribute modeling in a single model which is not trained for any of these specific tasks. We show empirically that GibbsNet is able to learn a more complex $p(z)$ and show that this leads to improved inpainting and iterative refinement of $p(x, z)$ for dozens of steps and stable generation without collapse for thousands of steps, despite being trained on only a few steps.

3.3 Introduction

Generative models are powerful tools for learning an underlying representation of complex data. While early undirected models, such as Deep Boltzmann Machines or DBMs (Salakhutdinov and Hinton, 2009), showed great promise, practically they did not scale well to complicated high-dimensional settings (beyond MNIST), possibly because of optimization and mixing difficulties (Bengio et al., 2012b). More recent work on Helmholtz machines (Bornschein et al., 2015) and on variational autoencoders (Kingma and Welling, 2013) borrow from deep learning tools and can achieve impressive results, having now been adopted in a large array of domains (Larsen et al., 2015a).

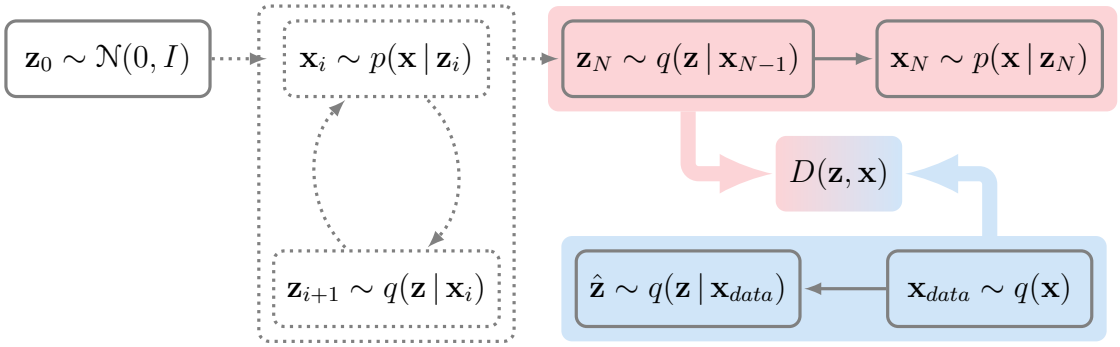

Figure 3.1 – Diagram illustrating the training procedure for GibbsNet. The **unclamped chain** (dashed box) starts with a sample from an isotropic Gaussian distribution $\mathcal{N}(0, I)$ and runs for N steps. The last step (iteration N) shown as a solid pink box is then compared with a single step from the **clamped chain** (solid blue box) using joint discriminator D.

Many of the important generative models available to us rely on a formulation of some sort of stochastic latent or hidden variables along with a generative relationship to the observed data. Arguably the simplest is the *directed graphical models* (such as the VAE) with a factorized decomposition $p(z, x) = p(z)p(x \mid z)$. In this, it is typical to assume that $p(z)$ follows some factorized prior with simple statistics (such as Gaussian). While sampling with directed models is simple, inference and learning tends to be difficult and often requires advanced techniques such as approximate inference using a proposal distribution for the true posterior.

The other dominant family of graphical models are *undirected graphical models*, such that the joint is represented by a product of clique potentials and a normalizing factor. It is common to assume that the clique potentials are positive, so that the un-normalized density can be represented by an energy function, E and the

joint is represented by $p(x, z) = e^{-E(z,x)}/Z$, where Z is the normalizing constant or partition function. These so-called energy-based models (of which the Boltzmann Machine is an example) are potentially very flexible and powerful, but are difficult to train in practice and do not seem to scale well. Note also how in such models, the marginal $p(z)$ can have a very rich form (as rich as that of $p(x)$).

The methods above rely on a fully parameterized joint distribution (and approximate posterior in the case of directed models), to train with approximate maximum likelihood estimation (MLE, Dempster et al., 1977). Recently, generative adversarial networks (GANs, Goodfellow et al., 2014) have provided a likelihood-free solution to generative modeling that provides an implicit distribution unconstrained by density assumptions on the data. In comparison to MLE-based latent variable methods, generated samples can be of very high quality (Radford et al., 2015), and do not suffer from well-known problems associated with parameterizing noise in the observation space (Goodfellow, 2016). Recently, there have been advances in incorporating latent variables in generative adversarial networks in a way reminiscent of Helmholtz machines (Dayan et al., 1995), such as adversarially learned inference (Dumoulin et al., 2017; Donahue et al., 2017) and implicit variational inference (Huszár, 2017).

These models, as being essentially complex directed graphical models, rely on approximate inference to train. While potentially powerful, there is good evidence that using an approximate posterior necessarily limits the generator in practice (Hjelm et al., 2016; Rezende and Mohamed, 2015). In contrast, it would perhaps be more appropriate to start with inference (encoder) and generative (decoder) processes and derive the prior directly from these processes. This approach, which we call GibbsNet, uses these two processes to define a transition operator of a Markov chain similar to Gibbs sampling, alternating between sampling observations and sampling latent variables. This is similar to the previously proposed generative stochastic networks (GSNs, Bengio et al., 2013) but with a GAN training framework rather than minimizing reconstruction error. By training a discriminator to place a decision boundary between the data-driven distribution (with x clamped) and the free-running model (which alternates between sampling x and z), we are able to train the model so that the two joint distributions (x, z) match. This approach is similar to Gibbs sampling in undirected models, yet, like traditional GANs, it lacks the strong parametric constraints, i.e., there is no explicit energy function. While

losing some the theoretical simplicity of undirected models, we gain great flexibility and ease of training. In summary, our method offers the following contributions:

— We introduce the theoretical foundation for a novel approach to learning and performing inference in deep graphical models. The resulting model of our algorithm is similar to undirected graphical models, but avoids the need for MLE-based training and also lacks an explicitly defined energy, instead being trained with a GAN-like discriminator.

— We present a stable way of performing inference in the adversarial framework, meaning that useful inference is performed under a wide range of architectures for the encoder and decoder networks. This stability comes from the fact that the encoder $q(z \mid x)$ appears in both the clamped and the unclamped chain, so gets its training signal from both the discriminator in the clamped chain and from the gradient in the unclamped chain.

— We show improvements in the quality of the latent space over models which use a simple prior for $p(z)$. This manifests itself in improved conditional generation. The expressiveness of the latent space is also demonstrated in cleaner inpainting, smoother mixing when running blocked Gibbs sampling, and better separation between classes in the inferred latent space.

— Our model has the flexibility of undirected graphical models, including the ability to do label prediction, class-conditional generation, and joint image-label generation in a single model which is not explicitly trained for any of these specific tasks. To our knowledge our model is the first model which combines this flexibility with the ability to produce high quality samples on natural images.

3.4 Proposed Approach: GibbsNet

The goal of GibbsNet is to train a graphical model with transition operators that are defined and learned directly by matching the joint distributions of the model expectation with that with the observations clamped to data. This is analogous to and inspired by undirected graphical models, except that the transition operators, which correspond to blocked Gibbs sampling, are defined to move along a defined energy manifold, so we will make this connection throughout our formulation.

We first explain GibbsNet in the simplest case where the graphical model consists of a single layer of observed units and a single layer of latent variable with stochastic mappings from one to the other as parameterized by arbitrary neural network. Like Professor Forcing (Lamb et al., 2016), GibbsNet uses a GAN-like discriminator to make two distributions match, one corresponding to the model iteratively sampling both observation, x, and latent variables, z (free-running), and one corresponding to the same generative model but with the observations, x, clamped. The free-running generator is analogous to Gibbs sampling in Restricted Boltzmann Machines (RBM, Hinton et al., 2006) or Deep Boltzmann Machines (DBM, Salakhutdinov and Hinton, 2009). In the simplest case, the free-running generator is defined by conditional distributions $q(z|x)$ and $p(x|z)$ which stochastically map back and forth between data space x and latent space z.

To begin our free-running process, we start the chain with a latent variable sampled from a normal distribution: $z \sim \mathcal{N}(0, I)$, and follow this by N steps of alternating between sampling from $p(x|z)$ and $q(z|x)$. For the clamped version, we do simple ancestral sampling from $q(z|x)$, given x_{data} is drawn from the data distribution $q(x)$ (a training example). When the model has more layers (e.g., a hierarchy of layers with stochastic latent variables, à la DBM), the data-driven model also needs to iterate to correctly sample from the joint. While this situation highly resembles that of undirected graphical models, GibbsNet is trained adversarially so that its free-running generative states become indistinguishable from its data-driven states. In addition, while in principle undirected graphical models need to either start their chains from data or sample a very large number of steps, we find in practice GibbsNet only requires a very small number of steps (on the order of 3 to 5 with very complex datasets) from noise.

An example of the free-running (unclamped) chain can be seen in Figure 3.2. An interesting aspect of GibbsNet is that we found that it was enough and in fact best experimentally to back-propagate discriminator gradients *through a single step of the iterative procedure*, yielding more stable training. An intuition for why this helps is that each step of the procedure is supposed to generate increasingly realistic samples. However, if we passed gradients through the iterative procedure, then this gradient could encourage the earlier steps to store features which have downstream value instead of immediate realistic x-values.

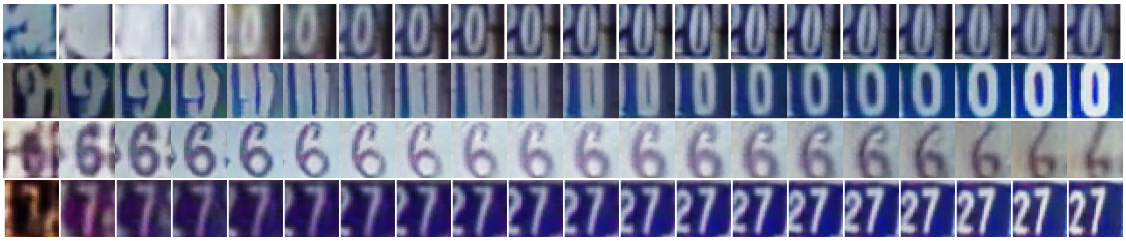

Figure 3.2 – Evolution of samples for 20 iterations from the unclamped chain, trained on the SVHN dataset starting on the left and ending on the right.

3.4.1 Theoretical Analysis

We consider a simple case of an undirected graph with single layers of visible and latent units trained with alternating 2-step (p then q) unclamped chains and the asymptotic scenario where the GAN objective is properly optimized. We then ask the following questions: in spite of training for a bounded number of Markov chain steps, are we learning a transition operator? Are the encoder and decoder estimating compatible conditionals associated with the stationary distribution of that transition operator? We find positive answers to both questions.

A high level explanation of our argument is that if the discriminator is fooled, then the consecutive (z, x) pairs from the chain match the data-driven (z, x) pair. Because the two marginals on x from these two distributions match, we can show that the next z in the chain will form again the same joint distribution. Similarly, we can show that the next x in the chain also forms the same joint with the previous z. Because the state only depends on the previous value of the chain (as it's Markov), then all following steps of the chain will also match the clamped distribution. This explains the result, validated experimentally, that even though we train for just a few steps, we can generate high quality samples for thousands or more steps.

Proposition 1. *If **(a)** the stochastic encoder $q(z|x)$ and stochastic decoder $p(x|z)$ inject noise such that the transition operator defined by their composition (p followed by q or vice-versa) allows for all possible x-to-x or z-to-z transitions ($x \to z \to x$ or $z \to x \to z$), and if **(b)** those GAN objectives are properly trained in the sense that the discriminator is fooled in spite of having sufficient capacity and training time, then **(1)** the Markov chain which alternates the stochastic encoder followed by the stochastic decoder as its transition operator T (or vice-versa) has the data-driven*

distribution π_D as its stationary distribution π_T, ***(2)*** *the two conditionals $q(z|x)$ and $p(x|z)$ converge to compatible conditionals associated with the joint $\pi_D = \pi_T$.*

Proof. When the stochastic decoder and encoder inject noise so that their composition forms a transition operator T with paths with non-zero probability from any state to any other state, then T is ergodic. So condition (a) implies that T has a stationary distribution π_T. The properly trained GAN discriminators for each of these two steps (condition (b)) forces the matching of the distributions of the pairs (z_t, x_t) (from the generative trajectory) and (x, z) with $x \sim q(x)$, the data distribution and $z \sim q(z \mid x)$, both pairs converging to the same data-driven distribution π_D. Because (z_t, x_t) has the same joint distribution as (z, x), it means that x_t has the same distribution as x. Since $z \sim q(z \mid x)$, when we apply q to x_t, we get z_{t+1} which must form a joint (z_{t+1}, x_t) which has the same distribution as (z, x). Similarly, since we just showed that z_{t+1} has the same distribution as z and thus the same as z_t, if we apply p to z_{t+1}, we get x_{t+1} and the joint (z_{t+1}, x_{t+1}) must have the same distribution as (z, x). Because the two pairs (z_t, x_t) and (z_{t+1}, x_{t+1}) have the same joint distribution π_D, it means that the transition operator T, that maps samples (z_t, x_t) to samples (z_{t+1}, x_{t+1}), maps π_D to itself, i.e., $\pi_D = \pi_T$ is both the data distribution and the stationary distribution of T and result (1) is obtained. Now consider the "odd" pairs (z_{t+1}, x_t) and (z_{t+2}, x_{t+1}) in the generated sequences. Because of (1), x_t and x_{t+1} have the same marginal distribution $\pi_D(x)$. Thus when we apply the same $q(z|x)$ to these x's we obtain that (z_{t+1}, x_t) and (z_{t+2}, x_{t+1}) also have the same distribution. Following the same reasoning as for proving (1), we conclude that the associated transition operator T_{odd} has also π_D as stationary distribution. So starting from $z \sim \pi_D(z)$ and applying $p(x \mid z)$ gives an x so that the pair (z, x) has π_D as joint distribution, i.e., $\pi_D(z, x) = \pi_D(z)p(x \mid z)$. This means that $p(x \mid z) = \frac{\pi_D(x,z)}{\pi_D(z)}$ is the $x \mid z$ conditional of π_D. Since (z_t, x_t) also converges to joint distribution π_D, we can apply the same argument when starting from an $x \sim \pi_D(x)$ followed by q and obtain that $\pi_D(z, x) = \pi_D(x)q(z \mid x)$ and so $q(z|x) = \frac{\pi_D(z,x)}{\pi_D(x)}$ is the $z \mid x$ conditional of π_D. This proves result (2). □

3.4.2 Architecture

GibbsNet always involves three networks: the inference network $q(z|x)$, the generation network $p(x|z)$, and the joint discriminator. In general, our architecture

for these networks closely follow Dumoulin et al. (2017), except that we use the boundary-seeking GAN (BGAN, Hjelm et al., 2017) as it explicitly optimizes on matching the opposing distributions (in this case, the model expectation and the data-driven joint distributions), allows us to use discrete variables where we consider learning graphs with labels or discrete attributes, and worked well across our experiments.

3.5 Related Work

Energy Models and Deep Boltzmann Machines The training and sampling procedure for generating from GibbsNet is very similar to that of a deep Boltzmann machine (DBM, Salakhutdinov and Hinton, 2009): both involve blocked Gibbs sampling between observation- and latent-variable layers. A major difference is that in a deep Boltzmann machine, the "decoder" $p(x|z)$ and "encoder" $p(z|x)$ exactly correspond to conditionals of a joint distribution $p(x, z)$, which is parameterized by an energy function. This, in turn, puts strong constraints on the forms of the encoder and decoder.

In a restricted Boltzmann machine (RBM, Hinton, 2010), the visible units are conditionally independent given the hidden units on the adjacent layer, and likewise the hidden units are conditionally independent given the visible units. This may force the layers close to the data to need to be nearly deterministic, which could cause poor mixing and thus make learning difficult. These conditional independence assumptions in RBMs and DBMs have been discussed before in the literature as a potential weakness in these models (Bengio et al., 2012b).

In our model, $p(x|z)$ and $q(z|x)$ are modeled by separate deep neural networks with no shared parameters. The disadvantage is that the networks are over-parameterized, but this has the added flexibility that these conditionals can be much deeper, can take advantage of all the recent advances in deep architectures, and have fewer conditional independence assumptions than DBMs and RBMs.

Generative Stochastic Networks Like GibbsNet, generative stochastic networks (GSNs, Bengio et al., 2013) also directly parameterizes a transition operator of a Markov chain using deep neural networks. However, GSNs and GibbsNet have

completely different training procedures. In GSNs, the training procedure is based on an objective that is similar to de-noising autoencoders (Vincent et al., 2008).

GSNs begin by drawing a sampling from the data, iteratively corrupting it, then learning a transition operator which de-noises it (i.e., reverses that corruption), so that the reconstruction after k steps is brought closer to the original un-corrupted input.

In GibbsNet, there is no corruption in the visible space, and the learning procedure never involves "walk-back" (de-noising) towards a real data-point. Instead, the processes from and to data are modeled by different networks, with the constraint of the marginal, $p(x)$, matches the real distribution imposed through the GAN loss on the joint distributions from the clamped and unclamped phases.

Non-Equilibrium Thermodynamics The Non-Equilibrium Thermodynamics method (Sohl-Dickstein et al., 2015) learns a reverse diffusion process against a forward diffusion process which starts from real data points and gradually injects noise until the data distribution matches a analytically tractible / simple distribution. This is similar to GibbsNet in that generation involves a stochastic process which is initialized from noise, but differs in that Non-Equilibrium Thermodynamics is trained using MLE and relies on noising + reversal for training, similar to GSNs above.

Generative Adversarial Learning of Markov Chains The Adversarial Markov Chain algorithm (AMC, Song et al., 2017) learns a markov chain over the data distribution in the visible space. GibbsNet and AMC are related in that they both involve adversarial training and an iterative procedure for generation. However there are major differences. GibbsNet learns deep graphical models with latent variables, whereas the AMC method learns a transition operator directly in the visible space. The AMC approach involves running chains which start from real data points and repeatedly apply the transition operator, which is different from the clamped chain used in GibbsNet. The experiments shown in Figure 3.3 demonstrate that giving the latent variables to the discriminator in our method has a significant impact on inference.

Adversarially Learned Inference (ALI) Adversarially learned inference (ALI, Dumoulin et al., 2017) learns to match distributions generative and inference distributions, $p(x, z)$ and $q(x, z)$ (can be thought of forward and backward models) with a discriminator, so that $p(z)p(x \mid z) = q(x)q(z \mid x)$. In the single latent layer case, GibbsNet also has forward and reverse models, $p(x \mid z)$ and $q(z \mid x)$. The un-clamped chain is sampled as $p(z), p(x \mid z), q(z \mid x), p(x \mid z), \ldots$ and the clamped chain is sampled as $q(x), q(z \mid x)$. We then adversarially encourage the clamped chain to match the equilibrium distribution of the unclamped chain. When the number of iterations is set to $N = 1$, GibbsNet reduces to ALI. However, in the general setting of $N > 1$, Gibbsnet should learn a richer representation than ALI, as the prior, $p(z)$, is no longer forced to be the simple one at the beginning of the unclamped phase.

3.6 Experiments and Results

The goal of our experiments is to explore and give insight into the joint distribution $p(x, z)$ learned by GibbsNet and to understand how this joint distribution evolves over the course of the iterative inference procedure. Since ALI is identical to GibbsNet when the number of iterative inference steps is $N = 1$, results obtained with ALI serve as an informative baseline.

From our experiments, the clearest result (covered in detail below) is that the $p(z)$ obtained with GibbsNet can be more complex than in ALI (or other directed graphical models). This is demonstrated directly in experiments with 2-D latent spaces and indirectly by improvements in classification when directly using the variables $q(z \mid x)$. We achieve strong improvements over ALI using GibbsNet even when $q(z \mid x)$ has exactly the same architecture in both models.

We also show that GibbsNet allows for gradual refinement of the joint, (x, z), in the sampling chain $q(z \mid x), p(x \mid z)$. This is a result of the sampling chain making small steps towards the equilibrium distribution. This allows GibbsNet to gradually improve sampling quality when running for many iterations. Additionally it allows for inpainting and conditional generation where the conditioning information is not fixed during training, and indeed where the model is not trained specifically for these tasks.

3.6.1 Expressiveness of GibbsNet's Learned Latent Variables

Latent structure of GibbsNet The latent variables from $q(z \mid x)$ learned from GibbsNet are more expressive than those learned with ALI. We show this in two ways. First, we train a model on the MNIST digits 0, 1, and 9 with a 2-D latent space which allows us to easily visualize inference. As seen in Figure 3.3, we show that GibbsNet is able to learn a latent space which is not Gaussian and has a structure that makes the different classes well separated.

Semi-supervised learning Following from this, we show that the latent variables learned by GibbsNet are better for classification. The goal here is not to show state of the art results on classification, but instead to show that the requirement that $p(z)$ be something simple (like a Gaussian, as in ALI) is undesirable as it forces the latent space to be filled. This means that different classes need to be packed closely together in that latent space, which makes it hard for such a latent space to maintain the class during inference and reconstruction.

We evaluate this property on two datasets: Street View House Number (SVHN, Netzer et al., 2011) and permutation invariant MNIST. In both cases we use the latent features $q(z \mid x)$ directly from a trained model, and train a 2-layer MLP on top of the latent variables, without passing gradient from the classifier through to $q(z \mid x)$. ALI and GibbsNet were trained for the same amount of time and with exactly the same architecture for the discriminator, the generative network, $p(x \mid z)$, and the inference network, $q(z \mid x)$.

On permutation invariant MNIST, ALI achieves 91% test accuracy and GibbsNet achieves 97.7% test accuracy. On SVHN, ALI achieves 66.7% test accuracy and GibbsNet achieves 79.6% test accuracy. This does not demonstrate a competitive classifier in either case, but rather demonstrates that the latent space inferred by GibbsNet keeps more information about its input image than the encoder learned by ALI. This is consistent with the reported ALI reconstructions (Dumoulin et al., 2017) on SVHN where the reconstructed image and the input image show the same digit roughly half of the time.

We found that ALI's inferred latent variables not being effective for classification is a fairly robust result that holds across a variety of architectures for the inference

network. For example, with 1024 units, we varied the number of fully-connected layers in ALI's inference network between 2 and 8 and found that the classification accuracies on the MNIST validation set ranged from 89.4% to 91.0%. Using 6 layers with 2048 units on each layer and a 256 dimensional latent prior achieved 91.2% accuracy. This suggests that the weak performance of the latent variables for classification is due to ALI's prior, and is probably not due to a lack of capacity in the inference network.

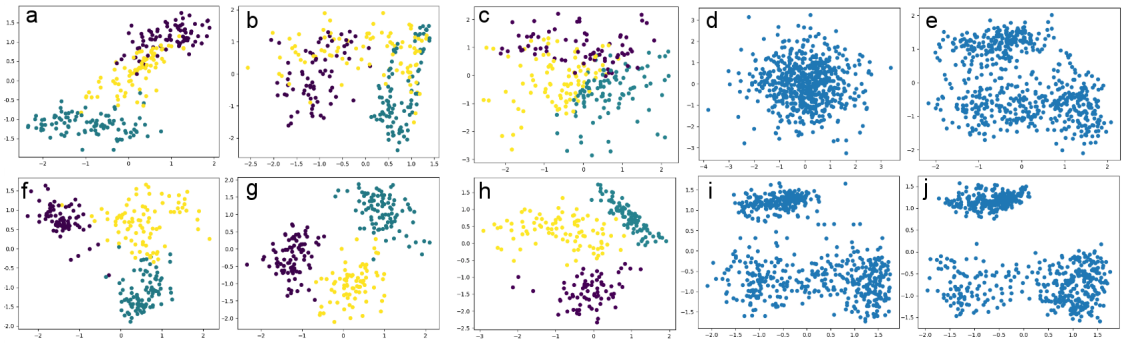

Figure 3.3 – Illustration of the distribution over inferred latent variables for real data points from the MNIST digits (0, 1, 9) learned with different models trained for roughly the same amount of time: GibbsNet with a determinstic decoder and the latent variables not given to the discriminator (a), GibbsNet with a stochastic decoder and the latent variables not given to the discriminator (b), ALI (c), GibbsNet with a deterministic decoder (f), GibbsNet with a stochastic decoder with two different runs (g and h), GibbsNet with a stochastic decoder's inferred latent states in an unclamped chain at 1, 2 , 3, and 15 steps (d, e, i, and j, respectively) into the P-chain (d, e, i, and j, respectively). Note that we continue to see refinement in the marginal distribution of z when running for far more steps (15 steps) than we used during training (3 steps).

3.6.2 Inception Scores

The GAN literature is limited in terms of quantitative evaluation, with none of the existing techniques being satisfactory Theis et al. (2015a). Inception scores Salimans et al. (2016) have become widely used and are correlated with human assessments of quality, but lack a statistical consistency guarantee. Nonetheless, we computed inception scores on CIFAR-10 using the standard method and code released from Salimans et al. (2016). In our experiments, we compared the inception scores from samples from GibbsNet and ALI on two tasks, generation and inpainting.

Our conclusion from the inception scores (Table 3.1) is that GibbsNet slightly improves sample quality but greatly improves the expressiveness of the latent space z, which leads to more detail being preserved in the inpainting chain and a much

larger improvement in inception scores in this setting. The supplementary materials includes examples of sampling and inpainting chains for both ALI and GibbsNet which shows differences between sampling and inpainting quality that are consistent with the inception scores.

Table 3.1 – Inception Scores from different models. Inpainting results were achieved by fixing the left half of the image while running the chain for four steps. Sampling refers to unconditional sampling.

Source	Samples	Inpainting
Real Images	11.24	11.24
ALI (ours)	5.41	5.59
ALI (Dumoulin)	5.34	N/A
GibbsNet	5.69	6.15

3.6.3 Generation, Inpainting, and Learning the Image-Attribute Joint Distribution

Generation Here, we compare generation on the CIFAR dataset against Non-Equilibrium Thermodynamics method (Sohl-Dickstein et al., 2015), which also begins its sampling procedure from noise. We show in Figure 3.4 that, even with a relatively small number of steps (20) in its sampling procedure, GibbsNet outperforms the Non-Equilibrium Thermodynamics approach in sample quality, even after many more steps (1000).

Inpainting The inpainting that can be done with the transition operator in GibbsNet is stronger than what can be done with an explicit conditional generative model, such as Conditional GANs, which are only suited to inpainting when the conditioning information is known about during training or there is a strong prior over what types of conditioning will be performed at test time. We show here that GibbsNet performs more consistent and higher quality inpainting than ALI, even when the two networks share exactly the same architecture for $p(x \mid z)$ and $q(z \mid x)$ (Figure 3.5), which is consistent with our results on latent structure above.

Joint generation Finally, we show that GibbsNet is able to learn the joint distribution between face images and their attributes (CelebA, Liu et al., 2015a)

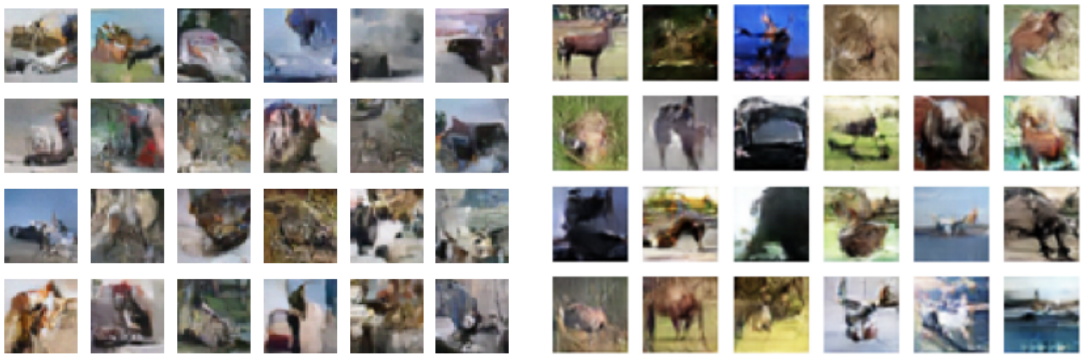

Figure 3.4 – CIFAR samples on methods which learn transition operators. Non-Equilibrium Thermodynamics (Sohl-Dickstein et al., 2015) after 1000 steps (left) and GibbsNet after 20 steps (right).

(Figure 3.6). In this case, $q(z \mid x, y)$ (y is the attribute) is a network that takes both the image and attribute, separately processing the two modalities before joining them into one network. $p(x, y \mid z)$ is one network that splits into two networks to predict the modalities separately. Training was done with continuous boundary-seeking GAN (BGAN, Hjelm et al., 2017) on the image side (same as our other experiments) and discrete BGAN on the attribute side, which is an importance-sampling-based technique for training GANs with discrete data.

(a) SVHN inpainting after 20 steps (ALI).

(b) SVHN inpainting after 20 steps (GibbsNet).

Figure 3.5 – Inpainting results on SVHN, where the right side is given and the left side is inpainted. In both cases our model's trained procedure did not consider the inpainting or conditional generation task at all, and inpainting is done by repeatedly applying the transition operators and clamping the right side of the image to its observed value. GibbsNet's richer latent space allows the transition operator to keep more of the structure of the input image, allowing for tighter inpainting.

Figure 3.6 – Demonstration of learning the joint distribution between images and a list of 40 binary attributes. Attributes (right) are generated from a multinomial distribution as part of the joint with the image (left).

3.7 Conclusion

We have introduced GibbsNet, a powerful new model for performing iterative inference and generation in deep graphical models. Although models like the RBM and the GSN have become less investigated in recent years, their theoretical properties worth pursuing, and we follow the theoretical motivations here using a GAN-like objective. With a training and sampling procedure that is closely related to undirected graphical models, GibbsNet is able to learn a joint distribution which converges in a very small number of steps of its Markov chain, and with no requirement that the marginal $p(z)$ match a simple prior. We prove that at convergence of training, in spite of unrolling only a few steps of the chain during training, we obtain a transition operator whose stationary distribution also matches the data and makes the conditionals $p(x \mid z)$ and $q(z \mid x)$ consistent with that unique joint stationary distribution. We show that this allows the prior, $p(z)$, to be shaped into a complicated distribution (not a simple one, e.g., a spherical Gaussian) where different classes have representations that are easily separable in the latent space. This leads to improved classification when the inferred latent variables $q(z|x)$ are used directly. Finally, we show that GibbsNet's flexible prior produces a flexible model which can simultaneously perform inpainting, conditional image generation, and prediction with a single model not explicitly trained for any of these specific tasks, outperforming a competitive ALI baseline with the same setup.

4 Discriminative Regularization for Generative Models

4.1 Prologue to the Article

Discriminative Regularization for Generative Models. Alex Lamb, Vincent Dumoulin, Aaron Courville.

DeepVision Workshop 2016

Personal Contribution.

Affiliations

Alex Lamb: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Vincent Dumoulin: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Aaron Courville: MILA, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CIFAR Fellow

Funding We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

4.2 Abstract

We explore the question of whether the representations learned by classifiers can be used to enhance the quality of generative models. Our conjecture is that labels correspond to characteristics of natural data which are most salient to humans: identity in faces, objects in images, and utterances in speech. We propose to take advantage of this by using the representations from discriminative classifiers to augment the objective function corresponding to a generative model. In particular we enhance the objective function of the variational autoencoder, a popular generative

model, with a discriminative regularization term. We show that enhancing the objective function in this way leads to samples that are clearer and have higher visual quality than the samples from the standard variational autoencoders.

4.3 Introduction

Discriminative neural network models have had a tremendous impact in many traditional application areas of machine learning such as object recognition and detection in images Krizhevsky et al. (2012); Simonyan and Zisserman (2014), speech recognition Hinton et al. (2012) and a host of other application domains Schmidhuber (2014). While progress in the longstanding problem of learning generative models capable of producing novel and compelling examples of natural data has not quite kept pace with the advances in discriminative modeling, there have been a number of important developments.

Within the context of generative models that support tractable approximate inference, the variational autoencoder (VAE) Kingma and Welling (2013) has emerged as a popular framework. The VAE leverages deep neural networks both for the generative model (mapping from a set of latent random variables to a conditional distribution over the observed data) and for an approximate inference model (mapping from the observed data to a conditional distribution over the latent random variables).

Images generated from the VAE (and most other generative frameworks) diverge from natural images in two distinct ways:

1. *Missing high frequency information.* Compared to natural data, generated samples often lack detail and appear blurry. Generative models of natural data such as images are largely limited to the maximum likelihood setting where the data was modeled as Gaussian distributed (with diagonal covariance), given some setting of the latent variables. Under a Gaussian, the quality of reconstruction is essentially evaluated on the basis of a generalized L^2 distance. As a measure of similarity between images, L^2 distance does not closely match human perception. For instance, the same image translated

by a few pixels could have relatively high L^2 distance, yet humans may not even perceive the difference.

2. *Missing semantic information.* Human perception is goal driven: we perceive our environment so that we can interact with it in meaningful ways. This implies that semantic information is going to be particularly salient to the human perceptual system. The current state-of-the-art in generative models, even when they capture high frequency information, produce samples which often lack semantically-relevant details. Generative models of natural images often lack a clear sense of "objectness". It is not enough to capture the correct local statistics over the data. For example, generative models trained on faces often produce inconsistencies in gender and identity, which may be subtle in pixel space but immediately apparent to humans viewing the samples.

In this work we explore an alternative VAE training objective by augmenting the standard VAE lower bound on the likelihood objective with additional discriminative terms that encourage the model's reconstructions to be close to the data example in a representation space defined by the hidden layers of highly-discriminative, neural network-based classifiers. We refer to this strategy as discriminative regularization of generative models.

In this effort we are heavily inspired by recently introduced texture synthesis method of Gatys et al. (2015b) as well as the DeepStyle model of Gatys et al. (2015a). These works showed that surprisingly detailed and semantically-rich information regarding natural images is preserved in the hidden-layer representations of ImageNet-trained object recognition networks such as VGG Simonyan and Zisserman (2014). Our goal is to incorporate this insight into the VAE framework and to render the synthetic data perceptually closer to the real data.

4.4 VAEs as Generative models of images

In this section we lay out the variational autoencoder (VAE) framework Kingma and Welling (2013); Rezende et al. (2014) on which we build. The VAE is a neural

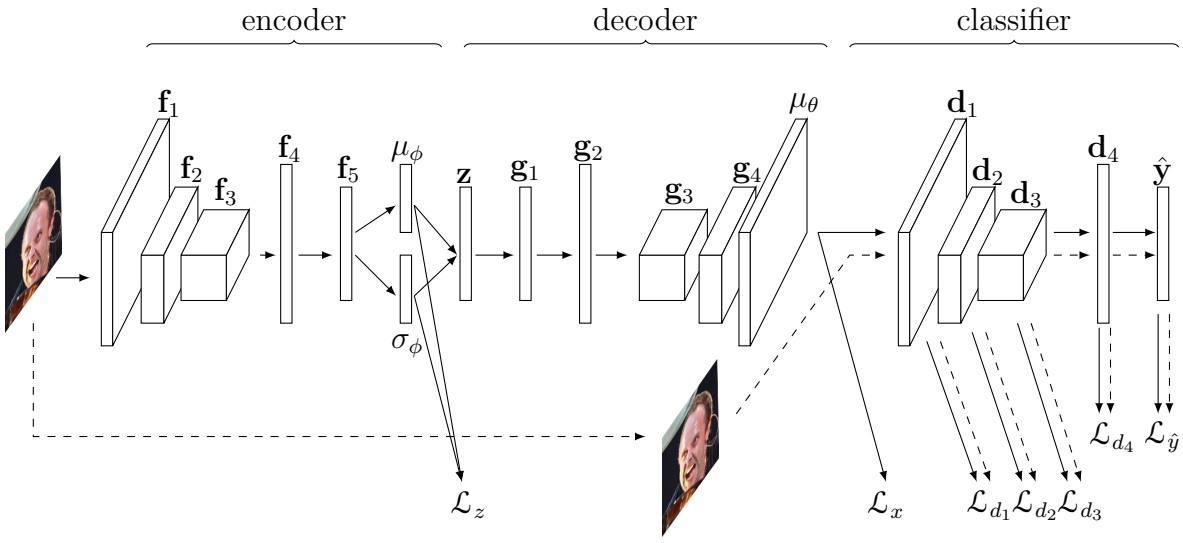

Figure 4.1 – The discriminative regularization model. Layers f_1, f_2, f_3, d_1, d_2 and d_3 represent convolutional layers, whereas layers g_3, g_4 and μ_θ represent fractionally strided convolutional layers.

network-based approach to latent variable modeling where the natural, richly-structured dependencies found in the data are disentangled into the relatively simple dependencies between a set of latent variables. Formally, let $\mathbf{x}$ be a random real-valued vector representing the observed data and let $\mathbf{z}$ be a random real-valued vector representing the latent variables that reflect the principle directions of variation in the input data.

4.4.1 The generative model

We specify the generative model over the pair (x, z) as $p_\theta(x, z) = p_\theta(x \mid z)p_\theta(z)$, where $p_\theta(z)$ is the prior distribution over the latent variables and $p_\theta(x \mid z)$ is the conditional likelihood of the data given the latents. θ represents the generative model parameters. As is typical in the VAE framework, we assume a standard Normal (Gaussian) prior distribution over z: $p_\theta(z) = \mathcal{N}(z \mid 0, I)$.

For real-valued data such as natural images, by far the most common conditional likelihood is the Gaussian distribution: $p(x \mid z) = \mathcal{N}(x \mid \mu_\theta(z), \text{diag}(\sigma_\theta^2))$, where the mean $\mu_x(z)$ is a nonlinear function of the latent variables specified by a neural network, which following autoencoder terminology, we refer to as the *decoder network*,

$f(x)$. In the natural image setting, $\mu_\theta(z)$ is parameterized by a CNN (see Figure 4.1) and σ_θ^2 is a vector of independent variance parameters over the pixels.

4.4.2 The approximate inference model

Given the generative model described above, inference is intractable, as is standard parameter learning paradigms such as maximizing the likelihood of the data. The VAE resolves these issues by introducing a learned approximate posterior distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$, specified by another neural network known as the *encoder network*, $g(z)$ and parametrized by ϕ.

Introducing the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ allows us to decompose the marginal log-likelihood of the data under the generative model in terms of the variational free energy and the Kullback-Leibler divergence between the approximate and true posteriors:

$$\log p_\theta(x) = \mathcal{L}(\theta, \phi; x) + D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right) \tag{4.1}$$

where the Kullback-Leibler divergence is given by

$$D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right]$$

and the variational free energy is given by

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right].$$

Since $D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right)$ measures the divergence between $q_\phi(z \mid x)$ and $p_\theta(z \mid x)$, it is guaranteed to be non-negative. As a consequence, the variational free energy $\mathcal{L}(\theta, \phi; x)$ is always a lower bound on the likelihood. As such it is sometimes called the variational lower bound or the evidence lower bound (ELBO).

In the VAE framework, $\mathcal{L}(\theta, \phi; x)$ is often rearranged into two terms:

$$\mathcal{L}(\theta, \phi; x) = \mathcal{L}_z(\theta, \phi; x) + \mathcal{L}_x(\theta, \phi; x) \tag{4.2}$$

where

$$\mathcal{L}_z(\theta, \phi; x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right)$$
$$\mathcal{L}_x(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

$\mathcal{L}_x$ can be interpreted as the (negative) expected reconstruction error of x under the conditional likelihood with respect to $q_\phi(z \mid x)$. Maximizing this lower bound strikes a balance between minimizing reconstruction error and minimizing the KL divergence between the approximate posterior $q_\phi(z \mid x)$ and the prior $p_\theta(z)$.

4.4.3 Reparametrization Trick

The power of the VAE approach can be credited to how the model is trained. With real-valued z, we can exploit a *reparametrization trick* Kingma and Welling (2013); Bengio et al. (2013) to propagate the gradient from the decoder network to the encoder network. Instead of sampling directly from $q_\phi(z \mid x)$, z is computed as a deterministic function of x and some noise term $\varepsilon \sim \mathcal{N}(0, I)$ such that $\mathbf{z}$ has the desired distribution. For instance, if

$$q_\phi(z \mid x) = \mathcal{N}(z \mid \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))), \tag{4.3}$$

then we would express z as

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

to produce values with the desired distribution while permitting gradients to propagate through both $\mu_\phi(x)$ and $\sigma^2_\phi(x)$.

4.4.4 The problem with the Independent Gaussian Assumption

The derivation of the variational autoencoder allows for different choices for the reconstruction model $p_\theta(x \mid z)$. However, as previously mentioned the majority of applications on real-valued data use a multivariate Gaussian with diagonal covariance matrix as the conditional likelihood of the data given the latent variables Gregor et al. (2015); Mansimov et al. (2015). Maximizing the conditional likelihood of this

distribution corresponds to minimizing an elementwise L^2 reconstruction penalty. One major weakness with this approach is that elementwise distance metrics are a poor fit for human notions of similarity. For example, shifting an image by only a few pixels will cause it to look very different under elementwise distance metrics but will not change its semantic properties or how it is perceived by humans Theis et al. (2015b).

In addition to the issues surrounding elementwise independence, there is nothing in a Gaussian conditional likelihood that will cause the model to render semantically-salient perceptual features of the data to be captured by the model.

4.5 Discriminative Regularization

In this section we describe our modification to the VAE lower bound training objective. Our goal is to modify the VAE training objective to render generated images perceptually closer to natural images. As previously discussed, generated images from the VAE (or other generative frameworks) often diverge from natural images in two distinct directions: (1) by being excessively blurry and (2) by lacking semantically meaningful cues such as depictions of well-defined objects. We conjecture that both of these issues can be ameliorated by encouraging the generator to render reconstructions that match the original data example in a representation space defined by the hidden layers of a classifier trained on a discrimination task relevant to the input data.

Let $d_1(x), d_2(x), \ldots, d_L(x)$ represent the L hidden layer representations of a *pre-trained* classifier. The classifier could be trained on a task specifically relevant to the data we wish to model. For example, in learning to generate images of faces we may wish to leverage a classifier trained to either identify individuals Huang et al. (2007) or trained to recognize certain facial characteristics Liu et al. (2015a). On the other hand, we could also follow the example of Gatys et al. (2015b) and use one of the high performing ImageNet trained models such as VGG Simonyan and Zisserman (2014) as a general purpose classifier for natural images.

In the standard VAE variational lower bound objective, we include a term that aims to minimize the reconstruction error in the space of the observed data. To

this we add additional terms aimed at minimizing the reconstruction error in the space defined by the hidden layer representations, $d_1, \ldots, d_L$, of the classifier.

$$\begin{aligned}\mathcal{L}_{disc}(\theta, \phi; x) = &- \mathcal{L}_z(\theta, \phi; x) + \mathcal{L}_x(\theta, \phi; x) \\ &+ \sum_{l=1}^{L} \mathcal{L}_{d_l}(\theta, \phi; x),\end{aligned} \tag{4.4}$$

where

$$\mathcal{L}_{d_l}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(d_l(x) \mid z). \tag{4.5}$$

We take the conditional likelihood of each $d_l(x) \mid z$ to be Gaussian with its mean $\mu_{d_l}(z)$ defined by forward propagating the conditional mean $\mu_x(z)$ through the layers of the classifier from d_1 to d_l:

$$\begin{aligned}
d_1(x) \mid z &\sim \mathcal{N}((d_1 \circ \mu_\theta)(z), \mathrm{diag}(\sigma^2_{d,1})), \\
d_2(x) \mid z &\sim \mathcal{N}((d_2 \circ d_1 \circ \mu_\theta)(z), \mathrm{diag}(\sigma^2_{d,2})), \\
&\ldots \\
d_L(x) \mid z &\sim \mathcal{N}((d_L \circ \cdots \circ d_2 \circ d_1 \circ \mu_\theta)(z), \mathrm{diag}(\sigma^2_{d,L})).
\end{aligned}$$

The discriminative regularization approach can be considered a kind of multitask regularization of the standard VAE, where in addition to the standard VAE objective, we include the additional tasks of predicting each of the hidden layer representations of a classifier.

We can understand the impact that these additional terms would have on the VAE parameters by considering matching in the different layers of the classifier. Since the classifiers we will consider will all be convolutional neural networks, the different layers will tend to have different characteristics, especially with respect to spatial translations. Matching the lower layer representations is going to encourage visual features such as edges to be well-defined and in the right location. The upper layers of a convolutional neural network classifier have been shown to be both highly invariant to spatial transformations (particularly translation), while simultaneously showing high specificity to semantically-relevant stimuli. Matching in the upper layers will likely de-emphasize exact spatial alignment, but will pressure semantic elements apparent in the example, such as the identity of objects, to be well matched between the data example x and the mean of the conditional likelihood μ_x.

It is important to assess the impact that the addition of our discriminative regularization terms have on the VAE. By adding the discriminative regularization terms we are no longer directly optimizing the variational lower bound.

Furthermore, since we are backpropagating the gradient of the combined objective $\mathcal{L}_{disc}$ through the decoder network and into the encoder network (the network responsible for approximating the posterior distribution), we are no longer directly optimizing the encoder network to minimize $\mathrm{KL}\left(q(z \mid x), p(z \mid x)\right)$. Doing so implies that we risk deteriorating our approximate posterior in favor of improving the example reconstructions (w.r.t the combined objective). One consequence could be an overall deterioration of the generated sample quality as the marginal $q(z) = \int q(z \mid x)q(x)\ dx$ diverges from the prior $p(z)$.

In our experiments, we did not observe any negative impact in sample quality, however if such an issue did arise, we could simply have elected not to propagate the the gradient contribution due to our discriminative regularization through the encoder network and thus preserve direct minimization of $\mathrm{KL}\left(q(z \mid x), p(z \mid x)\right)$ w.r.t. the parameters of the encoder network.

4.6 Related Work

Recent work has used the structural similarity metric Wang et al. (2004) as an auxiliary loss function for training variational autoencoders Ridgeway et al. (2015). They showed that using this metric instead of pixel-wise square loss dramatically improved human ratings of the generated images. Our approach differs from theirs in a few ways. First, we use the representations from a discriminatively trained classifier to augment our objective function, whereas they use a hand-crafted measure for image similarity. Second, discriminative regularization describes both local and global properties of the image (the local properties coming from lower layers and the global properties coming from higher layers), whereas their method only compares the true image and the reconstructed image around local 11x11 patches centered at each pixel. An interesting area for future work would be to study which method does a better job at improving the generation of local data, or if results can be improved by using both methods simultaneously.

Recently there has been a focus on alternative measures to be used during the training of generative models. Probably the most established of these is the generative adversarial networks (GANs) that leverage discriminative machinery and apply it to a two player game scenario between a generator and a discriminator Goodfellow et al. (2014). While the discriminator is trained to distinguish between true training samples and those generated from the generator, the generator is trained to try to fool the discriminator. While this joint optimization of the generator and discriminator is prone to instabilities, the end result are often generated images that capture realistic local texture. Recent applications of the GAN formalism have show very impressive results Denton et al. (2015); Radford et al. (2015).

Of all the proposed GAN-based methods, the one that most closely resembles the approach we propose here is the discriminative VAE Larsen et al. (2015b). In this work, the authors integrate the VAE within a GAN framework, in part, by maximizing a lower bound on a representation of the image defined by a given hidden layer of the GAN discriminator network. The authors show that their integration of the GAN and the VAE leads to impressive samples.

While generative adversarial networks have been a driving force in the relatively rapid improvement in the quality of image generation models, there are ways in which VAEs are preferable. GAN models do not optimize likelihood and are not trained directly for coverage of the training set, i.e. they use their capacity to convincingly mimic natural images. On the other hand the VAE more explicitly encourages coverage by maximizing a lower bound on the log likelihood. Another disadvantage of GANs is that in their original formulation there is no clear way to perform inference in the model, i.e. to recover the posterior distribution $p(z \mid x)$. However, there has been a few very recent efforts that are working to address this shortcoming of the GAN framework Makhzani et al. (2015); Larsen et al. (2015b).

4.7 Experiments

We evaluated the impact of the discriminative regularization on CelebA Liu et al. (2015b). The aligned and cropped version of the CelebA dataset was scaled from 218×178 pixels to 78×64 pixels and center cropped at 64×64 pixels. We trained our own classifier to predict all of the labels.

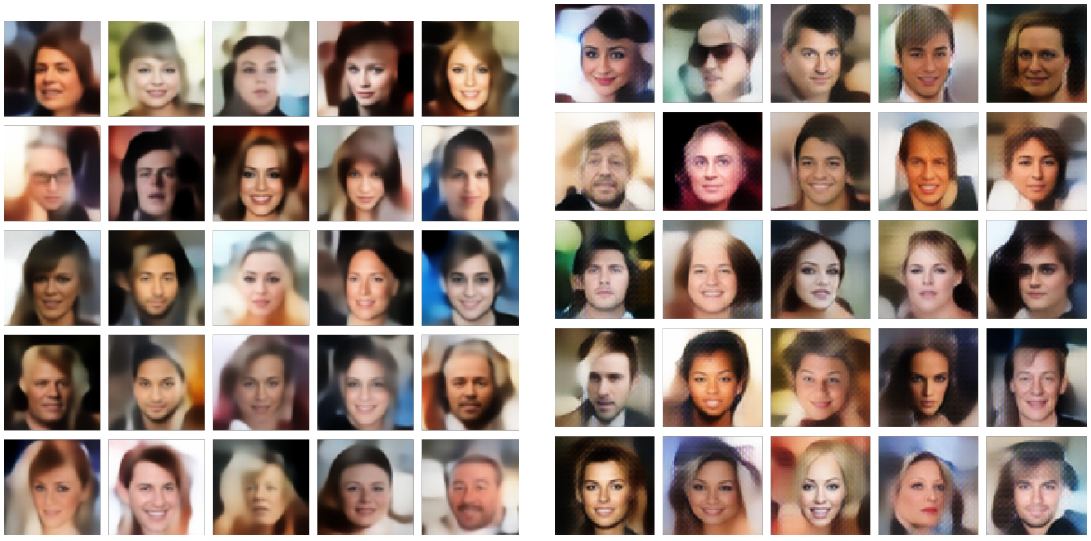

Figure 4.2 – Face samples generated with and without discriminative regularization. On balance, details of the face are better captured and more varied in the samples generated with discriminative regularization.

All VAE models, regularized or not, as well as the CelebA classifier were trained using Adam and batch normalization. Our architecture closely follows Radford et al. (2015), with convolutional layers in the encoder and fractionally-strided convolutions in the decoder. In each convolutional layer in the encoder we double the number of filters present in the previous layer and use a convolutional stride of 2. In each convolutional layer in the decoder we use a fractional stride of 2 and halve the number of filters on each layer.

Evaluating generative models quantitatively is a challenging task Theis et al. (2015c). One common evaluation metric is the likelihood of held-out samples. However, the usefulness of this metric is limited. If we compare the log-likelihood using the independent Gaussian in the pixel space, then we suffer from the limitations of pixel-wise distance metrics for comparing images. On the other hand, if we compare using the log-likelihood over the hidden states of the discriminative classifier, then we bias our evaluation criteria towards the criteria that we trained on.

4.7.1 Samples

Trained models were sampled from by sampling $z \sim p_\theta(z)$ and computing $\mathbb{E}_{q_\phi(z|x)}[p(x \mid z)]$ (in our case $\mu_\theta(z)$), which is standard practice in generative

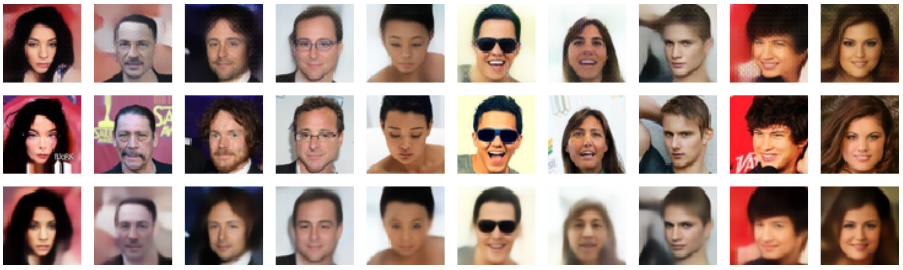

Figure 4.3 – Face reconstructions with (top row) and without (bottom row) discriminative regularization. The face images used for the reconstructions (middle row) are from the held-out validation set and were not seen by the model during training. The architecture and the hyperparameters (except those directly related to discriminative regularization) are the same for both models. Discriminative regularization greatly enhances the model's ability to preserve identity, ethnicity, gender, and expressions. Note that the model does not improve the visual quality of the image background, which likely reflects the fact that the classifier's labels all describe facial attributes. Additional reconstructions can be seen in the appendix.

modeling work.

Faces in CelebA samples (Figure 4.2) look more "in focus" when discriminative regularization is used during training.

4.7.2 Reconstructions

Reconstructions were obtained by sampling $z \sim q_\phi(z \mid x)$ and computing $\mathbb{E}_{q_\phi(z|x)}\left[p(x \mid z)\right]$ (in our case $\mu_\theta(z)$), which is also standard practice in generative modeling work.

Using discriminative regularization during training leads to improved reconstructions (Figure 4.3). In addition to producing sharper reconstructions, this approach helps maintaining the identity better. This is especially noticeable in the eyes region: VAE reconstructions tend to produce stereotypical eyes, whereas our approach better captures the overall eye shape.

4.7.3 Interpolations in the Latent Space

To evaluate the quality of the learned latent representation, we visualize the result of linearly interpolating between latent configurations. We choose pairs of images whose latent representation we obtain by computing $\mu_\phi(x)$. We then compute intermediary latent representations z by linearly interpolating between the latent representation pairs, and we display the corresponding $\mu_\theta(z)$.

5 Shadow	Arch. Eyebrows	Attractive
Bags under eyes	Bald	Blurry
Bangs	Big Lips	Brown Hair
Big Nose	Black Hair	Bushy Eyebrows
Blond Hair	Goatee	Gray Hair
Eyeglasses	Double Chin	Heavy Makeup
Heavy Cheekbones	Gender	Mouth Open
Mustache	Narrow Eyes	No Beard
Oval Face	Pale Skin	Pointy Nose
Recced. Hairline	Rosey Cheeks	Sideburns
Smiling	Straight Hair	Wavy Hair
Earrings	Wearing Hat	Lipstick
Necklace	Necktie	Young

Table 4.1 – A list of the binary targets that we predict with our celebA classifier.

The resulting trajectory in pixel space (Figure 4.4) exhibits smooth and realistic transitions between face pose and orientation, hair color and gender.

4.7.4 Explaining Visual Artifacts

In the samples generated from a model trained with discriminative regularization, we sometimes see unnatural patterns or texturing. In the faces samples, we mostly observe these patterns in the background. They occur to some extent in nearly all samples. These patterns are not seen in samples from the standard variational autoencoders.

One explanation for the visual artifacts is that the variational autoencoder with discriminative regularization produces unnaturally blurred activations in the classifier's convolutional layers in the same way that the standard variational autoencoder outputs unnaturally blurred images.

To support this hypothesis, we visualize what happens when a convolutional autoencoder explicitly tries to generate a reconstruction which produces a blurred representation in the classifier. To do so, we train a convolutional autoencoder on a batch of 100 examples. The examples are reconstructed as usual, but we propagate both the input and the reconstruction through the first two layers of the classifier. The propagated input is then blurred by adding gaussian blur (applied separately to each filter), and the cost is computed as the squared error between the propagated reconstruction and the blurred propagated input.

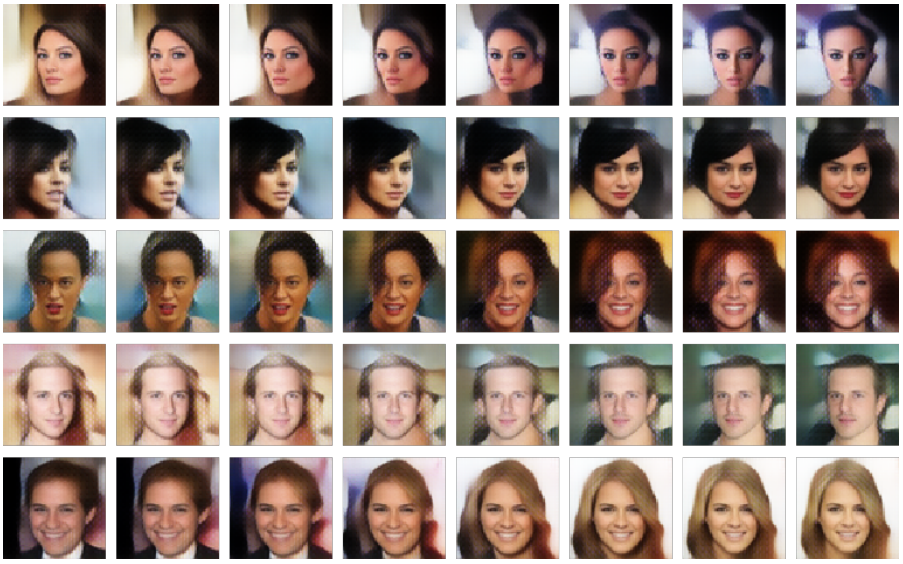

Figure 4.4 – Latent space interpolations with discriminative regularization. On each row, the first and last image correspond to reconstructions of randomly selected examples.

Figure 4.5 provides a visual summary of the experiment. We see that when no blurring is applied to the hidden representation, the autoencoder does a perfect job of matching the hidden representations (middle left column), which is indicated by an excellent reconstruction at the input level. When blurring is applied, we see that the resulting reconstructions (right column) exhibit visual patterns resembling those of our model's reconstructions (middle right column).

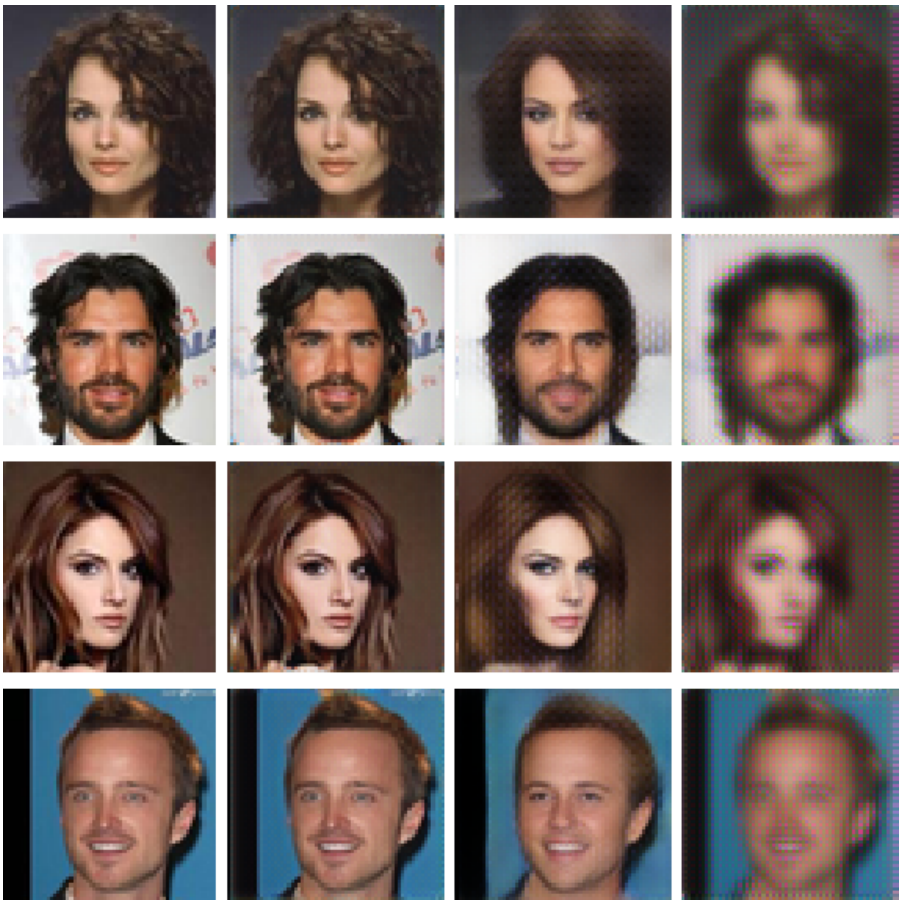

Figure 4.5 – From left to right: input examples, convolutional (non-variational) autoencoder reconstructions (no blurring applied to the classifier's hidden representations), model reconstructions (trained with discriminative regularization), convolutional autoencoder reconstructions (blurring applied to the classifier's hidden representations).

4.8 Conclusion

A common view in cognitive science is that generative modeling will play a central role in the development of artificial intelligence by enabling feature learning where labeled data and reward signals are sparse. In this view generative models serve to assist other models by learning representations and discovering causal factors from the nearly unlimited supply of unlabeled data. Our paper shows that this interaction ought to be a two-way street, in which supervised learning contributes to generative modeling by determining which attributes of the data are worth learning to represent. We have demonstrated that discriminative information can be used to regularize generative models to improve the perceptual quality of their samples.

5 Conclusion

This thesis opens with a high level overview of what generative models are and how the task can be formalized. We then overview the two most prominent approaches in generative modeling: likelihood maximization approaches and adversarial approaches. We then motivate the problem by discussing applications of generative models. After introducing some core concepts necessary for the later chapters, we give an overview of three recently published papers which focus on new algorithms for generative modeling and a new application for generative models. Summarizing a few of the key points from this thesis:

— Generative models have both a qualitative definition and a formal definition based on minimizing statistical divergence between a real distribution and an estimating distribution.

— Maximizing likelihood is perhaps the most widely studied method in generative modeling, yet its limitations and attempts to get around those limitations are discussed. Autoregressive models and variational autoencoders are discussed in detail.

— The adversarial approach to generative modeling is discussed, as well as the progression of methods leading to successful training of GANs in practice. In particular, the method of injecting noise provides a better theoretical grounding for training GANs, and the gradient penalty and spectral normalization techniques are better ways of achieving the same effect as noise injection.

— Fortified Networks are introduced as a way of adding simple generative models (denoising autoencoders) to the hidden layers of a deep neural network, which makes it possible for the network to map off-manifold points back onto the manifold. This improves robustness to adversarial examples and allows for measurement of differences between teacher forcing and sampling modes when training RNNs.

— GibbsNet is introduced as a new framework for doing inference iterative generation in an adversarial framework which is inspired by the blocked-gibbs

sampling procedure for sampling from energy-based models. This framework allows for iterative improvement in the quality of samples as well as improved quality inference as the gaussian prior on the latent variables is removed and replaced with a constraint on the dimensionality of the latent space.

— Discriminative Regularization is proposed as a way of making generative models correspond to divergences which correspond to categories salient to human perception, as opposed to solely being divergences which are easy to define and quantify.

The ability to imagine and conceptualize in the mind things which are not immediately present in reality has long been seen as a defining feature of human existence. The field of generative models has made great progress in creating machines which are also endowed with the ability to imagine. At the same time, a considerable gap remains between human abilities and the abilities of our best generative models.

Bibliography

Alain, G., Y. Bengio, and S. Rifai (2012). Regularized auto-encoders estimate local statistics. *CoRR abs/1211.4246*.

Arjovsky, M. and L. Bottou (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

Arjovsky, M., S. Chintala, and L. Bottou (2017, January). Wasserstein GAN. *ArXiv e-prints*.

Athalye, A., N. Carlini, and D. Wagner (2018, February). Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ArXiv e-prints*.

Athalye, A., L. Engstrom, A. Ilyas, and K. Kwok (2017). Synthesizing robust adversarial examples. *CoRR abs/1707.07397*.

Barratt, S. and R. Sharma (2018, January). A Note on the Inception Score. *ArXiv e-prints*.

Bengio, S., O. Vinyals, N. Jaitly, and N. Shazeer (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179.

Bengio, Y., A. Courville, and P. Vincent (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 35*(8), 1798–1828.

Bengio, Y., N. Léonard, and A. Courville (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432.

Bengio, Y., G. Mesnil, Y. Dauphin, and S. Rifai (2012a). Better mixing via deep representations. *CoRR abs/1207.4404*.

Bengio, Y., G. Mesnil, Y. Dauphin, and S. Rifai (2012b). Better mixing via deep representations. *CoRR abs/1207.4404*.

Bengio, Y., G. Mesnil, Y. Dauphin, and S. Rifai (2013). Better mixing via deep representations. In *ICML'2013*.

Bengio, Y., E. Thibodeau-Laufer, and J. Yosinski (2013). Deep generative stochastic networks trainable by backprop. *CoRR abs/1306.1091*.

Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bornschein, J., S. Shabanian, A. Fischer, and Y. Bengio (2015). Training opposing directed models using geometric mean matching. *CoRR abs/1506.03877*.

Brown, T. B., D. Mané, A. Roy, M. Abadi, and J. Gilmer (2017, December). Adversarial Patch. *ArXiv e-prints*.

Burda, Y., R. B. Grosse, and R. Salakhutdinov (2014). Accurate and conservative estimates of MRF log-likelihood using reverse annealing. *CoRR abs/1412.8566*.

Chaowei Xiao, Bo Li, J.-Y. Z. W. H. M. L. D. S. (2018). Generating adversarial examples with adversarial networks.

Dayan, P., G. E. Hinton, R. M. Neal, and R. S. Zemel (1995). The helmholtz machine. *Neural computation* 7(5), 889–904.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1–38.

Denton, E. L., S. Chintala, A. Szlam, and R. Fergus (2015). Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR abs/1506.05751*.

Donahue, J., P. Krähenbühl, and T. Darrell (2017). Adversarial feature learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*. abs/1605.09782.

Dumoulin, V., I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville (2017). Adversarially learned inference. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv:1606.00704.

Elgammal, A. M., B. Liu, M. Elhoseiny, and M. Mazzone (2017). CAN: creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *CoRR abs/1706.07068*.

Erraqabi, A., A. Baratin, Y. Bengio, and S. Lacoste-Julien (2018). A3t: Adversarially augmented adversarial training. *arXiv preprint arXiv:1801.04055*.

Gatys, L. A., A. S. Ecker, and M. Bethge (2015a). A neural algorithm of artistic style. *CoRR abs/1508.06576*.

Gatys, L. A., A. S. Ecker, and M. Bethge (2015b). Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *CoRR abs/1505.07376*.

Gilmer, J., L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow (2018, January). Adversarial Spheres. *ArXiv e-prints*.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014, June). Generative Adversarial Networks. *ArXiv e-prints*.

Goodfellow, I. J., J. Shlens, and C. Szegedy (2014, December). Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*.

Graves, A. (2012). Sequence transduction with recurrent neural networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*.

Gregor, K., I. Danihelka, A. Graves, and D. Wierstra (2015). Draw: A recurrent neural network for image generation. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*.

Gu, S. and L. Rigazio (2014). Towards deep neural network architectures robust to adversarial examples. *CoRR abs/1412.5068*.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville (2017). Improved training of wasserstein gans. *CoRR abs/1704.00028*.

Gutmann, M. and A. Hyvärinen (2010, 13–15 May). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y. W. Teh and M. Titterington (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Volume 9 of *Proceedings of Machine Learning Research*, Chia Laguna Resort, Sardinia, Italy, pp. 297–304. PMLR.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR abs/1706.08500*.

Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum 9*(1), 926.

Hinton, G., L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury (2012, Nov). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE 29*(6), 82–97.

Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

Hjelm, D., R. R. Salakhutdinov, K. Cho, N. Jojic, V. Calhoun, and J. Chung (2016). Iterative refinement of the approximate posterior for directed belief networks. In *Advances in Neural Information Processing Systems*, pp. 4691–4699.

Hjelm, R. D., A. P. Jacob, T. Che, K. Cho, and Y. Bengio (2017). Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*.

Huang, G. B., M. Ramesh, T. Berg, and E. Learned-Miller (2007, October). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst.

Huszár, F. (2015, November). How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *ArXiv e-prints*.

Huszár, F. (2017, February). Variational Inference using Implicit Distributions. *ArXiv e-prints*.

Ilyas, A., A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis (2017, December). The Robust Manifold Defense: Adversarial Training using Generative Models. *ArXiv e-prints*.

Jacob Buckman, Aurko Roy, C. R. I. G. (2018). Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*.

Karras, T., T. Aila, S. Laine, and J. Lehtinen (2017). Progressive growing of gans for improved quality, stability, and variation. *CoRR abs/1710.10196*.

Kingma, D. P. and M. Welling (2013, December). Auto-Encoding Variational Bayes. *ArXiv e-prints*.

Kingma, D. P. and M. Welling (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Kurakin, A., I. J. Goodfellow, and S. Bengio (2016). Adversarial machine learning at scale. *CoRR abs/1611.01236*.

Lamb, A., A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio (2016, October). Professor Forcing: A New Algorithm for Training Recurrent Networks. *ArXiv e-prints*.

Larsen, A. B. L., S. K. Sønderby, and O. Winther (2015a). Autoencoding beyond pixels using a learned similarity metric. *CoRR abs/1512.09300*.

Larsen, A. B. L., S. K. Sønderby, and O. Winther (2015b). Autoencoding beyond pixels using a learned similarity metric. *CoRR abs/1512.09300*.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature 521*(7553), 436.

Liao, F., M. Liang, Y. Dong, T. Pang, J. Zhu, and X. Hu (2017, December). Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser. *ArXiv e-prints*.

Liu, Z., P. Luo, X. Wang, and X. Tang (2015a). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Liu, Z., P. Luo, X. Wang, and X. Tang (2015b). Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738.

Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2017, June). Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*.

Makhzani, A., J. Shlens, N. Jaitly, and I. J. Goodfellow (2015). Adversarial autoencoders. *CoRR abs/1511.05644*.

Mansimov, E., E. Parisotto, L. J. Ba, and R. Salakhutdinov (2015). Generating images from captions with attention. *CoRR abs/1511.02793*.

Miyato, T., T. Kataoka, M. Koyama, and Y. Yoshida (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Neal, R. M. (1998, March). Annealed Importance Sampling. *ArXiv Physics e-prints*.

Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, Volume 2011, pp. 5.

Nicolas Papernot, Nicholas Carlini, I. G. R. F. F. F. A. M. K. H. Y.-L. J. A. K. R. S. A. G. Y.-C. L. (2017). cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.

Nowak, R. (2009). University Lecture http://nowak.ece.wisc.edu/SLT09/lecture13.pdf.

Odena, A., J. Buckman, C. Olsson, T. B. Brown, C. Olah, C. Raffel, and I. Goodfellow (2018, February). Is Generator Conditioning Causally Related to GAN Performance? *ArXiv e-prints*.

Papernot, N., P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami (2016). Practical black-box attacks against deep learning systems using adversarial examples. *CoRR abs/1602.02697*.

Papernot, N., P. D. McDaniel, A. Sinha, and M. P. Wellman (2016). Towards the science of security and privacy in machine learning. *CoRR abs/1611.03814*.

Papernot, N., P. D. McDaniel, X. Wu, S. Jha, and A. Swami (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR abs/1511.04508*.

Pouya Samangouei, Maya Kabkab, R. C. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*.

Radford, A., L. Metz, and S. Chintala (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR abs/1511.06434*.

Rezende, D. J. and S. Mohamed (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning (ICML)*, pp. 1278–1286.

Ridgeway, K., J. Snell, B. Roads, R. S. Zemel, and M. C. Mozer (2015). Learning to generate images with perceptual similarity metrics. *CoRR abs/1511.06409*.

Roth, K., A. Lucchi, S. Nowozin, and T. Hofmann (2017). Stabilizing training of generative adversarial networks through regularization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, pp. 2018–2028. Curran Associates, Inc.

Salakhutdinov, R. and G. Hinton (2009). Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pp. 448–455.

Salimans, T., I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved techniques for training gans. *CoRR abs/1606.03498*.

Salimans, T., A. Karpathy, X. Chen, and D. P. Kingma (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR abs/1701.05517*.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *CoRR abs/1404.7828*.

Sharif, M., S. Bhagavatula, L. Bauer, and M. K. Reiter (2017). Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*.

Simonyan, K. and A. Zisserman (2014). Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556*.

Sohl-Dickstein, J., E. A. Weiss, N. Maheswaranathan, and S. Ganguli (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR abs/1503.03585*.

Song, J., S. Zhao, and S. Ermon (2017). Generative adversarial learning of markov chains. *ICLR Workshop Track*.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2013, December). Intriguing properties of neural networks. *ArXiv e-prints*.

Theis, L., A. van den Oord, and M. Bethge (2015a, November). A note on the evaluation of generative models. *ArXiv e-prints*.

Theis, L., A. van den Oord, and M. Bethge (2015b, November). A note on the evaluation of generative models. *ArXiv e-prints*.

Theis, L., A. van den Oord, and M. Bethge (2015c, November). A note on the evaluation of generative models. *ArXiv e-prints*.

Ulyanov, D., A. Vedaldi, and V. S. Lempitsky (2017). Deep image prior. *CoRR abs/1711.10925*.

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM.

Wang, Z., A. Bovik, H. Sheikh, and E. Simoncelli (2004, April). Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on 13*(4), 600–612.

Warde-Farley, D. and Y. Bengio (2016). Improving generative adversarial networks with denoising feature matching.

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

Xu, W., D. Evans, and Y. Qi (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR abs/1704.01155*.