Entropy-based aggregate posterior alignment techniques for deterministic autoencoders and implications for adversarial examples

by

Amur Ghose

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapters 1 and 3 consist of unpublished work solely written by myself, with proofreading and editing suggestions from my supervisor, Pascal Poupart. Chapter 2 is (with very minor changes) an UAI 2020 paper (paper link) on which I was the lead author, wrote the manuscript, formulated the core idea and ran the majority of the experiments. Some of the experiments were ran by Abdullah Rashwan, a co-author on the paper (credited in the acknowledgements of the thesis). My supervisor Pascal Poupart again proofread and edited the manuscript and made many valuable suggestions. Note that UAI 2020 proceedings are Open Access under Creative Commons, and as such, no copyright section is provided with the thesis, and as the official proceedings are as of yet unreleased a link has been provided in lieu of a citation.

Abstract

We present results obtained in the context of generative neural models — specifically autoencoders — utilizing standard results from coding theory. The methods are fairly elementary in principle, yet, combined with the ubiquitous practice of Batch Normalization in these models, yield excellent results when it comes to comparing with rival autoencoding architectures. In particular, we resolve a split that arises when comparing two different types of autoencoding models — VAEs versus regularized deterministic autoencoders — often simply called RAEs (Regularized Auto Encoder). The latter offer superior performance but lose guarantees on their latent space. Further, in the latter, a wide variety of regularizers are applied for excellent performance — ranging from L_2 regularization to spectral normalization. We, on the other hand, show that a simple entropy like term suffices to kill two birds with one stone — that of offering good performance while keeping a well behaved latent space.

The primary thrust of the thesis exactly consists of a paper presented at UAI 2020 on these matters, titled "Batch norm with entropic regularization turns deterministic autoencoders into generative models". This was a joint work with Abdullah Rashwan who was at the time with us at Waterloo as a postdoctoral associate, and is now at Google, and my supervisor, Pascal Poupart. This constitutes chapter 2. Extensions on this that relate to batch norm's interplay with adversarial examples are in chapter 3. An overall overview is presented in chapter 1, which serves jointly as an introduction.

Acknowledgements

There are innumerable possibilities here, but I will primarily thank my supervisor, professor Pascal Poupart, and my readers - professors Yu and Kamath - for agreeing to read the thesis.

A massive amount of appreciation goes out to the Vector Institute, without whose GPUs this work would have not been possible in its true empirical scale. My thanks go out to Abdullah Rashwan, who at the time was a postdoc with my supervisor, Pascal Poupart, and came in clutch when it came to running experiments for the paper which shares content and name with Chapter 2.

Resources used in preparing this research were provided by NSERC, the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. [1]

A similar credit in terms of discussions lies with friends at the Indian Institute of Technology Kanpur - namely Karttikeya Mangalam, Archit Sharma, Shubh Gupta, and Kanishk Gandhi - and at the University of Waterloo, namely Allen Wang and Pranav Subramani.

[1] www.vectorinstitute.ai/#partners

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Problem statements and overview

Deep generative models have, over the past decade, attained spectacular success in natural language, computer vision and all other manners of applied machine learning tasks. One of the premier families in this group is the Variational Auto-Encoder aka VAE [28], which has an encoder-decoder pair of networks creating latent codes for each data instance. These latent codes are usually coerced to follow a Gaussian distribution for the process of generation of new data instances. Within the broader VAE class, deterministic AEs have risen to the fore recently, with examples such as the Regularized Autoencoder [12] and others [52]. In deterministic AEs, one core problem that pops up is the loss of control on the shape of the latent space learnt this way, which often has to be re-estimated as it diverges from a Gaussian. This is usually circumvented by ad hoc ex-post density estimation steps.

Approach, motivation and contribution

Our approach to making a better deterministic autoencoder relies on using batch normalization [24, 23], a now-ubiquitous process in modern deep networks. The motivation is clear from the grounds of utility - despite having recently been overtaken by competitors such as GANs [13], the VAE remains a premier deep generative class of models, and as such improving it has direct benefits. Further, batch normalization is used in nearly every deep neural model, and as such requires no special cajoling to be put in. In the process, we will also gain insights into batch normalization's interplay with certain distributions,

that, orthogonally to VAEs, sheds light on batch normalization itself. Remarkably, the full extent of batch normalization's effects still remain unclear [46] and there is debate over whether it can help or harm [11], as such, investigating its properties forms a valid auxiliary motive in its own right.

We contribute by improving the subclass of deterministic AEs by creating our own variant - the Entropic Autoencoder, and we also contribute to elucidating the effects of batch normalization in general across deep neural nets. The improvements in the autoencoding models arise by FID scores [19], and in terms of elucidation we discuss cases of naturally Gaussian-like distributions in deep nets with batch norm, and explore interesting properties relating to the L_2 norm of adversarial examples.

An overview of batch normalization

Normalization - in itself - is understood simply as subtracting the mean and dividing by standard deviation in much of statistics, or simply projecting the data to some fixed range $[a, b]$. As elementary as this sounds, batch normalization [24] - where the mean and variance in question is computed over a minibatch of training examples and the data is normalized using the same - has become an inescapable part of training neural networks over the past decade, and when it is discarded, alternatives [56, 59, 5, 53, 2, 21] rush to take its place. We will call this operation simply **batchnorm** for the practice of subtracting the minibatch mean and dividing by the minibatch standard deviation.

And yet batch normalization is poorly understood. There is debate over whether to use it at all, whether it is beneficial, and whether the folklore explanation of 'internal covariate shift' [24, 23] holds any water. Nevertheless, this thesis does not mean to challenge the relevance of batch normalization to training neural networks, but merely notes that in nearly all standard architectures and methods across the empirical zoo of networks, there exists some form of normalization. We will take this as-is, and seek to leverage this fact.

A normalization associates a distribution

Our key observation is that normalizations are not just an arbitrary choice but that particular schemes link to particular distributions - specifically the Gibbs distribution related to its sufficient statistic.

For a rather contrived example, consider a model that seeks to capture $P(X)$, the probability of some data X. Let J denote a map from Z, of equal dimensionality as X, to X. Then,

$$\log P(X) = \log |\det J| + \log P(Z)$$

Suppose that $P(Z)$ is as per $\mathcal{N}(0, I)$. Suppose that Z is constrained to be zero mean and of unit variance in every direction. In this case, it is easy to see that the $\log P(Z)$ term is a constant in every minibatch, because a Gaussian's log likelihood function only involves the first and second moments of the random variable.

In other words, even if we take away the explicit likelihood on the prior space Z and simply put a constraint, along with maximizing $\log |\det J|$, the resultant optimization sees no change. This part comes from the $\log P(Z)$ term, being a constant, having no impact on the optimization process at all.

To see this clearly, consider removing the $\log P(Z)$ term from the optimization process and only instead requiring that $E(Z_i) = 0, E(Z_i^2) = 1$, while maximizing $\log |\det J|$. Suppose two models attain the same value of $\log P(X)$. Now, observe that with the LHS being a constant, the term $\log |\det J|$ is maximal when $\log P(Z)$ is as low as possible - in other words, $-\log P(Z)$ is as high as possible **in expectation** over the dataset, ergo, Z is of **maximum entropy** given its constraints.

Utilizing the link

The above observation can be put to use if we know for situations where entropy can be reliably controlled within a neural network - batchnorm layers, after all, are ubiquitous. Where may we find such high entropy conditions ?

- Using an explicit entropic regularizer
- Inducing situations where the network is incentivized to create maximally entropic Z

While the first approach sounds simple enough, the second deserves scrutiny. When is a representation of high entropy? Note that in general, differential entropy suffers from interpretability issues that its discrete counterpart does not, as differential entropy changes upon scaling the data or under bijections. Nevertheless, we heuristically claim that entropy

rises closer to the theoretical maximum when the representation both grows smaller in dimensionality and has complex functional maps both mapping it from data X and also mapping it to labels Y in the context of classifier networks. These form the bedrock of Chapter 2. Complexity here is a byword for other ways to measure the power of neural function approximators - depth, channels, and so on.

The implications of a high dimensional Gaussian

High dimensional probability in general [54] presents many counterintuitive geometric patterns, one of the most famous being that a Gaussian resembles a spherical distribution in high dimensions. If we have, from the above, a reliable way to find Gaussian-like distributions within the hidden layers of a neural network, can we efficiently exploit such properties to explain some other phenomena ? Chapter 3 takes a stab at this with respect to adversarial examples.

Future avenues

One of the most persistent and interesting findings encountered in the course of Chapter 3 is the low L_2 norm of adversarial examples. We have also found, in our investigations, cases of flow models [45, 42] failing to fit high dimensional spherical distributions and assigning mass to the interior of the high dimensional sphere. Given these two phenomena, it is natural to ask if, in an efficient manner, we can fool flow models with adversarial examples that exploit this geometric fact. I continue to experiment in this intersection.

Chapter 2

Batch Normalization and entropic regularization serves to turn deterministic autoencoders to generative models

2.1 Introduction

Modeling data with neural networks is often broken into the broad classes of discrimination and generation. We consider generation, which can be independent of related goals like density estimation, as the task of generating unseen samples from a data distribution, specifically by neural networks, or simply **deep generative models**.

The variational autoencoder [28] (VAE) is a well-known subclass of deep generative models, in which we have two distinct networks - a decoder and encoder. To generate data with the decoder, a Gaussian sampling step is introduced between the encoder and decoder, with the encoder supplying the parameters of the Gaussian. This sampling step complicates the optimization of autoencoders. Since it is not possible to differentiate through sampling, the process is slightly modified to allow backpropagation, and this is termed the **reparametrization trick**. The sampling distribution has to be optimized to approximate a canonical distribution such as a Gaussian. The log-likelihood objective is also approximated. Hence, it would be desirable to avoid the sampling step.

To that effect, [12] proposed regularized autoencoders (RAEs) where sampling is replaced by some regularization, since stochasticity introduced by sampling can be seen as a form of

regularization. By avoiding sampling, a deterministic autoencoder can be optimized more simply. However, they introduce multiple candidate regularizers, and picking the best one is not straightforward. Density estimation also becomes an additional task as [12] fit a density to the empirical latent codes after the autoencoder has been optimized.

In this work, we introduce a batch normalization step between the encoder and decoder and add a entropic regularizer on the batch norm layer. Batchnorm fixes some moments (mean and variance) of the empirical code distribution while the entropic regularizer maximizes the entropy of the empirical code distribution. Maximizing the entropy of a distribution with certain fixed moments induces Gibbs distributions of certain families (i.e., normal distribution for fixed mean and variance). Hence, we naturally obtain a distribution that we can sample from to obtain codes that can be decoded into realistic data. The introduction of a batchnorm step with entropic regularization does not complicate the optimization of the autoencoder which remains deterministic. Neither step is sufficient in isolation and requires the other, and we compare what happens when the entropic regularizer is absent. Our work parallels RAEs in determinism and regularization, though we differ in choice of regularizer and motivation, as well as ease of isotropic sampling.

The Chapter is organized as follows. In Section 2.2, we review background about variational autoencoders and batch normalization. In Section 2.3, we propose entropic autoencoders (EAEs) with batch normalization as a new deterministic generative model. Section 2.4 discusses the maximum entropy principle and how it promotes certain distributions over latent codes even without explicit entropic regularization. Section 2.5 demonstrates the generative performance of EAEs on three benchmark datasets (CELEBA, CIFAR-10 and MNIST). EAEs outperform previous deterministic and variational autoencoders in terms of FID scores. Section 2.6 concludes the Chapter with suggestions for future work.

2.2 Variational autoencoder

The variational autoencoder [28] (VAE) consists of a decoder followed by an encoder. The term autoencoder [41] is in general applied to any model that is trained to reconstruct its inputs. For a normal autoencoder, representing the decoder and encoder as $\mathcal{D}, \mathcal{E}$ respectively, for every input x_i we seek:

$$\mathcal{E}(x_i) = z_i, \mathcal{D}(z_i) = \hat{x_i} \approx x_i$$

Such a model is usually trained by minimizing $||\hat{x_i} - x_i||^2$ over all x_i in training set. In a variational autoencoder, there is no fixed codeword z_i for a x_i. Instead, we have

$$z_i = \mathcal{E}(x_i) \sim \mathcal{N}(\mathcal{E}_\mu(x_i), \mathcal{E}_{\sigma^2}(x_i))$$

The encoder network calculates means and variances via $\mathcal{E}_\mu, \mathcal{E}_{\sigma^2}$ layers for every data instance, from which a code is sampled. The loss function is of the form:

$$||\mathcal{D}(z_i) - x_i||^2 + \beta D_{KL}(\mathcal{N}(\mathcal{E}_\mu(x_i), \mathcal{E}_{\sigma^2}(x_i))||\mathcal{N}(0, I))$$

where D_{KL} denotes the Kullback-Leibler divergence and z_i denotes the sample from the distribution over codes. Upon minimizing the loss function over $x_i \in$ a training set, we can generate samples as: generate $z_i \sim \mathcal{N}(0, I)$, and output $\mathcal{D}(z_i)$. The KL term makes the implicitly learnt distribution of the encoder close to a spherical Gaussian. Usually, z_i is of a smaller dimensionality than x_i.

2.2.1 Variations on Variational autoencoders

In practice, the above objective is not easy to optimize. The original VAE formulation did not involve β, and simply set it to 1. Later, it was discovered that this parameter helps training the VAE correctly, giving rise to a class of architectures termed β-VAE. [20]

The primary problem with the VAE lies in the training objective. We seek to minimize KL divergence for every instance x_i, which is often too strong. The result is termed **posterior collapse** [18] where every x_i generates $\mathcal{E}_\mu(x_i) \approx 0, \mathcal{E}_{\sigma^2}(x_i) \approx 1$. Here, the latent variable z_i begins to relate less and less to x_i, because neither μ, σ^2 depend on it. Attempts to fix this [26] involve analyzing the mutual information between z_i, x_i pairs, resulting in architectures like InfoVAE [60], along with others such as δ-VAE [44]. Posterior collapse is notable when the decoder is especially 'powerful,' i.e. has great representational power. Practically, this manifests in the decoder's depth being increased, more deconvolutional channels, etc.

One VAE variation includes creating a deterministic architecture that minimizes an optimal transport based Wasserstein loss between the empirical data distribution and decoded images from aggregate posterior. Such models [52] work with the aggregate posterior instead of outputting a distribution per sample, by optimizing either the Maximum Mean Discrepancy (MMD) metric with a Gaussian kernel [16], or using a GAN to minimize this optimal transport loss via Kantorovich-Rubinstein duality. The GAN variant outperforms using MMD, and WAE techniques are usually considered as WAE-GAN for achieving state-of-the-art results.

2.2.2 Batch normalization

Normalization is often known in statistics as the procedure of subtracting the mean of a dataset and dividing by the standard deviation. This sets the sample mean to zero and variance to one. In neural networks, normalization for a minibatch [24] has become ubiquitous since its introduction and is now a key part of training all forms of deep generative models [23]. Given a minibatch of inputs x_i of dimensions n with μ_{ij}, σ_{ij} as its mean, standard deviation at index j respectively, we will call **BN** as the operation that satisfies:

$$[\text{BN}(x_i)]_j = \frac{x_{ij} - \mu_{ij}}{\sigma_{ij}} \tag{2.1}$$

Note that in practice, a batch normalization layer in a neural network computes a function of form $A \circ B$ with A as an affine function, and B as **BN**. This is done during training time using the empirical average of the minibatch, and at test time using the overall averages. Many variations on this technique such as L1 normalization, instance normalization, online adaptations, etc. exist [56, 59, 5, 53, 2, 21]. The mechanism by which this helps optimization was initially termed as "internal covariate shift," but later works challenge this perception [46, 58] and show it may have harmful effects [11].

2.3 The entropic autoencoder

Instead of outputting a distribution as VAEs do, we seek an approach that turns deterministic autoencoders into generative models on par with VAEs. Now, if we had a guarantee that, for a regular autoencoder that merely seeks to minimize reconstruction error, the distribution of all z_i's approached a spherical Gaussian, we could carry out generation just as in the VAE model. We do the following: we simply append a batch normalization step (BN as above, i.e. no affine shift) to the end of the encoder, and minimize the objective:

$$||\hat{x_i} - x_i||^2 - \beta H(z_i), \hat{x_i} = \mathcal{D}(z_i), z_i = \mathcal{E}(x_i) \tag{2.2}$$

where H represents the entropy function and is taken over a minibatch of the z_i. We recall and use the following property: let X be a random variable obeying $E[X] = 0, E[X^2] = 1$. Then, the maximum value of $H(X)$ is obtained iff $X \sim \mathcal{N}(0, 1)$. We later show that even when no entropic regularizer is applied, batch norm alone can yield passable samples when a Gaussian is used for generation purposes. However, for good performance, the entropic regularizer is necessary.

2.3.1 Equivalence to KL divergence minimization

Our method of maximizing entropy minimizes the KL divergence by a backdoor. Generally, minibatches are too small to construct a meaningful sample distribution that can be compared - in D_{KL} - to the sought spherical normal distribution without other constraints. However, suppose that we have the following problem with X being a random variable with some constraint functions C_k e.g. on its moments, with the expectations being over whatever density defines X:

$$\max H(X), E[C_k(X)] = c_k, k = 1, 2, \dots$$

In particular let the two constraints be $E[X] = 0, E[X^2] = 1$ in the above formulation i.e. $k = 1, 2$ with C_1 being the identity function and C_2 the squaring function. Consider a 'proposal' distribution Q that satisfies $E_Q[X] = 0, E_Q[X^2] = 1$ and also a maximum entropy distribution P that is the solution to the optimization problem above. In our setup, P can be thought of as the target distribution and Q the distribution the encoder is learning. The cross entropy of P with respect to Q is

$$E_Q[-\log P(X)]$$

In our case, P is a Gaussian and $-\log P(X)$ is a term of the form $aX^2 + bX + c$. In expectation of this w.r.t. Q, $E_Q[X], E_Q[X^2]$ are already fixed. Thus for all proposal distributions Q, cross entropy of P w.r.t. Q - written as $H(Q, P)$ obeys

$$H(Q, P) = H(Q) + D_{KL}(Q||P)$$

Pushing up $H(Q)$ thus directly reduces the KL divergence to P, as the left hand side is a constant. Over a minibatch, every proposal Q identically satisfies the two moment conditions due to normalization. Unlike KL divergence, involving estimating and integrating a conditional probability (both rapidly intractable in higher dimensions) entropy estimation is easier, involves no conditional probabilities, and forms the bedrock of estimating quantities derived from entropy such as MI. Due to interest in the Information Bottleneck method [51] which requires entropy estimation of hidden layers, we already have a nonparametric entropy estimator of choice - the Kozachenko Leonenko estimator [31], which also incurs low computational load and has already been used for neural networks. This principle of "cutting the middleman" builds on the fact that MI based methods for neural networks often use Kraskov-like estimators [32], a family of estimators that break the MI term into

H terms which are estimated by the Kozachenko-Leonenko estimator. Instead, we directly work with the entropy.

2.3.2 The Kozachenko-Leonenko estimator

The Kozachenko-Leonenko estimator [31] operates as follows. Let $N \geq 1$ and $X_1, \ldots, X_{N+1}$ be i.i.d. samples from an unknown distribution Q. Let each $X_i \in \mathbb{R}^d$.

For each X_i, define $R_i = \min ||X_i - X_j||_2, j \neq i$ and $Y_i = N(R_i)^d$. Let B_d be the volume of the unit ball in $\mathbb{R}^d$ and γ the Euler-mascheroni constant ≈ 0.577. The Kozachenko Leonenko estimator works as follows:

$$H(Q) \approx \frac{1}{N+1} \sum_{i=1}^{N+1} \log Y_i + \log B_d + \gamma$$

Intuitively, having a high distance to the nearest training example for each example pushes up the entropy via the Y_i term. Such "repulsion"-like nearest neighbour techniques have been employed elsewhere for likelihood-free techniques such as implicit maximum likelihood estimation [37, 36]. In general, the estimator is biased with known asymptotic orders [8] - however, when the bias stays relatively constant through training, optimization is unaffected. The complexity of the estimator when utilizing nearest neighbours per minibatch is quadratic in the size of the batch, which is reasonable for small batches.

2.3.3 Generalization to Gibbs distributions

A distribution that has the maximum entropy under constraints C_k as above is called the **Gibbs distribution** of the respective constraint set. When this distribution exists, we have the result that there exist Lagrange multipliers λ_k, such that if the maximum entropy distribution is P, $\log P(X)$ is of the form $\sum \lambda_k C_k$. For any candidate distribution Q, $E_Q[C_k(X)]$ is determined solely from the constraints, and thus the cross-entropy $E_Q[-\log P(X)]$ is also determined. Our technique of pushing up the entropy to reduce KL holds under this generalization. For instance, pushing up the entropy for L_1 normalization layers corresponds to inducing a Laplace distribution.

2.3.4 Parallels with constant variance VAEs

One variation on VAEs is the constant variance VAE [12], where the term $\mathcal{E}_{\sigma^2}$ is constant for every instance x_i. Writing Mutual Information as MI, consider transmitting a code via

the encoder that maximizes $\mathrm{MI}(X, Y)$ where X is the encoder's output and Y the input to the decoder. In the noiseless case, $Y = X$, and we work with $\mathrm{MI}(X, X)$.

For a discrete random variable X, $\mathrm{MI}(X, X) = H(X)$. If noiseless transmission was possible, the mutual information would depend solely on entropy. However, using continuous random variables, our analysis of the constant variance autoencoder would for $\sigma^2 = 0$ yield a MI of ∞, between the code emitted by the encoder and received by the decoder. This at first glance appears ill-defined.

However, suppose that we are in the test conditions i.e. the batch norm is using a fixed mean and variance and independent of minibatch. Now, if the decoder receives Y, $\mathrm{MI}(X, Y) = H(X) - H(X|Y)$. Since $(X|Y)$ is a Dirac distribution, it pushes the mutual information to ∞. If we ignore the infinite mutual information introduced by the deterministic mapping just as in the definition of differential entropy, the only term remaining is $H(X)$, maximizing which becomes equivalent to maximizing MI. We propose our model as the zero-variance limit of present constant variance VAE architectures, especially when batch size is large enough to allow accurate estimations of mean and variance.

2.3.5 Comparison to prior deterministic autoencoders

Our work is not the first to use a deterministic autoencoder as a generative one. Prior attempts in this regard such as regularized autoencoders (RAEs) [12] share the similarities of being deterministic and regularized autoencoders, but do not leverage batch normalization. Rather, these methods rely on taking the constant variance autoencoder, and imposing a regularization term on the architecture. This does not maintain the KL property that we show arises via entropy maximization, rather, it forms a latent space that has to be estimated such as via a Gaussian mixture model (GMM) on top of the regularization. The Gaussian latent space is thus lost, and has to be estimated post-training. In contrast to the varying regularization choices of RAEs, our method uses the specific Max Entropy regularizer forcing a particular latent structure. Compared to the prior Wasserstein autoencoder (WAE) [52], RAEs achieve better empirical results, however we further improve on these results while keeping the ability to sample from the prior i.e. isotropic Gaussians. As such, we combine the ability of WAE-like sampling with performance superior to RAEs, delivering the best of both worlds. This comparison excludes the much larger bigWAE models [52] which utilize ResNet encoder-decoder pairs.

In general, for all VAE and RAE-like models, the KL/Optimal Transport/Regularization terms compete against reconstruction loss and having perfect Gaussian latents is not always feasible, hence, EAEs, like RAEs, benefit from post-density estimation and GMM fitting.

The primary advantage they attain is not **requiring** such steps, and performing at a solid baseline without it.

2.4 The maximum entropy principle and regularizer-free latents

We now turn to a general framework that motivates our architecture and adds context. Given the possibility of choosing a distribution $Q \in \mathcal{D}$ that fits some given dataset $\mathcal{X}$ provided, what objective should we choose? One choice is to pick the maximum likelihood estimate:

$$Q_{MLE} = \arg\max_{Q \in \mathcal{D}} E_{\bar{\mathcal{X}}}[LL_Q(X)]$$

where $LL_Q(X)$ denotes the log likelihood of an instance X and $E_{\bar{\mathcal{X}}}$ indicates that the expectation is taken with the empirical distribution $\bar{\mathcal{X}}$ from $\mathcal{X}$, i.e. every point X is assigned a probability $\frac{1}{|\mathcal{X}|}$. An alternative is to pick the maximum entropy solution or maxent:

$$Q_{MAXENT} = \arg\max_{Q \in \mathcal{D}} H(Q) \tag{2.3}$$

$$\text{subject to } T_i(Q) = T_i(\mathcal{X}) \tag{2.4}$$

Where H is the entropy of Q, and $T_i(Q)$ are summary statistics of Q that match the summary statistics over the dataset. For instance, if all we know is the mean and variance of $\mathcal{X}$, the distribution Q with maximum entropy that has the same mean and variance is Gaussian. This so-called maximum entropy principle [3] has been used in reinforcement learning [61], natural language processing [4], normalizing flows [39], and computer vision [48] successfully. Maximum entropy is in terms of optimization the convex dual problem of maximum likelihood, and takes a different route of attacking the same objective. We will shorten maximum entropy to maxent for ease of reference.

2.4.1 The maxent principle and deterministic autoencoders

Now, consider the propagation of an input through an autoencoder. The autoencoder may be represented as:

$$X \approx \mathcal{D}(\mathcal{E}(X))$$

where $\mathcal{D}, \mathcal{E}$ respectively represent the decoder and encoder halves. Observe that if we add a BatchNorm of the form $A \circ B$ with A as an affine shift, B as **BN** (as defined in Equation 2.1) to $\mathcal{E}$ - the encoder - we try to find a distribution Z after B and before A, such that:

- $E[Z] = 0, E[Z^2] = 1$
- $B \circ \mathcal{E}(X) \sim Z$, $A \circ \mathcal{D}(Z) \sim X$

Observe that there are two conditions that do not depend on $\mathcal{E}, \mathcal{D}$: $E[Z] = 0, E[Z^2] = 1$. Consider two different optimization problems:

- O, which asks to find the max entropy distribution Q, i.e., with max $H(Q)$ over Z satisfying $E_Q[Z] = 0, E_Q[Z^2] = 1$.
- O', which asks to find $\mathcal{D}, \mathcal{E}, A$ and a distribution Q' over Z such that we maximize $H(Q')$, with $E_{Q'}[Z] = 0, E_{Q'}[Z^2] = 1, B \circ \mathcal{E}(X) \sim Z, A \circ \mathcal{D}(Z) \sim X, Z \sim Q'$.

Since O has fewer constraints, $H(Q) \geq H(Q')$. Furthermore, $H(Q)$ is known to be maximal iff Q is an isotropic Gaussian over Z. What happens as the capacity of $\mathcal{D}, \mathcal{E}$ rises to the point of possibly representing anything (e.g., by increasing depth)? The constraints $B \circ \mathcal{E}(X) \sim Z, A \circ \mathcal{D}(Z) \sim X, Z \sim Q'$ effectively vanish, since the functional ability to deform Z becomes arbitrarily high. We can take the solution of O, plug it into O', and find $\mathcal{E}, \mathcal{D}, A$ that (almost) meet the constraints of $B \circ \mathcal{E}(X) \sim Z, A \circ \mathcal{D}(Z) \sim X, Z \sim Q'$. If the algorithm chooses the max entropy solution, the solution of O' - the actual distribution after the BatchNorm layer - approaches the maxent distribution, an isotropic Gaussian, when the last three constraints in O affect the solution less.

2.4.2 Natural emergence of Gaussian latents in deep narrowly bottlenecked autoencoders

We make an interesting prediction: if we increase the depths of $\mathcal{E}, \mathcal{D}$ and constrain $\mathcal{E}$ to output a code Z obeying $E[Z] = 0, E[Z^2] = 1$, the distribution of Z should - even without an entropic regularizer - tend to go to a spherical Gaussian as depth increases relative to the bottleneck. In practical terms, this will manifest in less regularization being required at higher depths or narrower bottlenecks. This phenomenon also occurs in posterior collapse for VAEs and we should verify that our latent space stays meaningful under such conditions.

Under the information bottleneck principle, for a neural network with output Y from input X, we seek a hidden layer representation for Z that maximizes $\mathrm{MI}(Z, Y)$ while lowering $\mathrm{MI}(X, Z)$. For an autoencoder, $Y \approx X$. Since Z is fully determined from X in a deterministic autoencoder, increasing $H(Z)$ increases $\mathrm{MI}(Z, X)$ if we ignore the ∞ term that arises due to $H(Y|X)$ as Y approaches a deterministic function of X as before in our CV-VAE discussion. Increasing $H(Z)$ will be justified iff it gives rise to better reconstruction, i.e. making Z more entropic (informative) lowers the reconstruction loss.

Such increases are likelier when Z is of low dimensionality and struggles to summarize X. We predict the following: a deep, narrowly bottlenecked autoencoder with a batch normalized code, will, even without regularization, approach spherical Gaussian-like latent spaces. We show this in the datasets of interest, where narrow enough bottlenecks can yield samples even **without** regularization, a behaviour also anticipated in [12].

2.5 Empirical experiments

2.5.1 Baseline architectures with entropic regularization

We begin by generating images based on our architecture on 3 standard datasets, namely MNIST [35], CIFAR-10 [33] and CelebA [38]. We use convolutional channels of $[128, 256, 512, 1024]$ in the encoder half and deconvolutional channels of $[512, 256]$ for MNIST and CIFAR-10 and $[512, 256, 128]$ for CelebA, starting from a channel size of 1024 in the decoder half. For kernels we use 4×4 for CIFAR-10 and MNIST, and 5×5 for CelebA with strides of 2 for all layers except the terminal decoder layer. Each layer utilizes a subsequent batchnorm layer and ReLU activations, and the hidden bottleneck layer immediately after the encoder has a batch norm without affine shift. These architectures, preprocessing of datasets, etc. match exactly the previous architectures that we benchmark against [12, 52].

For optimization, we utilize the Adam optimizer. The minibatch size is set to 100, to match [12] with an entropic regularization based on the Kozachenko Leonenko estimator [31]. In general, larger batch sizes yielded better FID scores but harmed speed of optimization. In terms of latent dimensionality, we use 16 for MNIST, 128 for CIFAR-10 and 64 for CelebA. At most 100 epochs are used for MNIST and CIFAR-10 and at most 70 for CelebA.

In Figure 2.1, we present qualitative results on the MNIST dataset. We do not report the Frechet Inception Distance (FID) [19], a commonly used metric for gauging image quality, since it uses the Inception network, which is not calibrated on grayscale handwritten digits. In Figure 2.1, we show the quality of the generated images for two different regularization

weights β in Eq. 2.2 (0.05 and 1.0 respectively) and in the same Figure illustrate the quality of reconstructed digits.

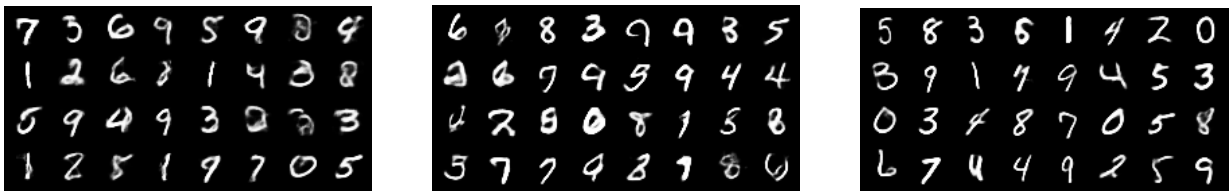

Figure 2.1: Left: Generated MNIST images with $\beta = 0.05$ in Eq. 2.2. Middle: Generated MNIST images with $\beta = 1.0$ in Eq. 2.2. Right: Reconstructed MNIST images with $\beta = 1.0$ in Eq. 2.2.

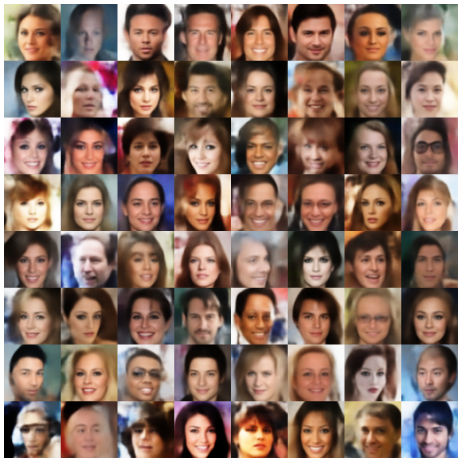

Figure 2.2: Generated images on CelebA

We move on to qualitative results for CelebA. We present a collage of generated samples in Figure 2.2. CIFAR-10 samples are presented in Figure 2.3. We also seek to compare, thoroughly, to the RAE architecture. For this, we present quantitative results in terms of FID scores in Table 2.1 (larger version in the additional results Section). We show results when sampling latent codes from an isotropic Gaussian as well as from densities fitted to the empirical distribution of latent codes after the AE has been optimized. We consider isotropic Gaussians and Gaussian mixture models (GMMs). In all cases, we improve on RAE-variant architectures proposed previously [12]. We refer to our architecture as the **entropic autoencoder (EAE)**. There is a tradeoff between Gaussian latent spaces and reconstruction loss, and results always improve with ex-post density estimation due to prior-posterior mismatch.

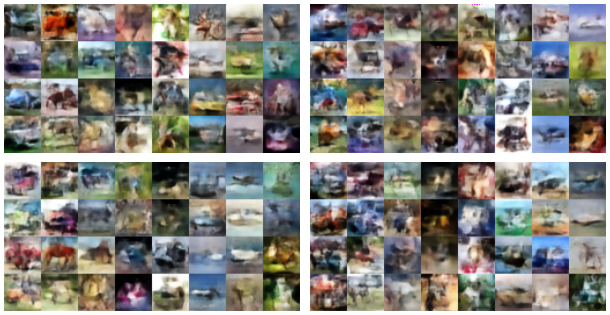

Figure 2.3: Generated images on CIFAR-10, under four different regularization weights (top left $\beta = 0.5$, top right $\beta = 0.7$, bottom left $\beta = 0.05$, bottom right $\beta = 0.07$).

Details on previous methods

In the consequent Tables and Figures, VAE/AE have their standard meanings. AE-L2 refers to an autoencoder with only reconstruction loss and L2 regularization, 2SVAE to the Two-Stage VAE as per [7], WAE to the Wasserstein Autoencoder as per [52], RAE to the Regularized Auto-encoder as per [12], with RAE-L2 referring to such with a L2 penalty, RAE-GP to such with a Gradient Penalty, RAE-SN to such with spectral normalization. We use spectral normalization in our EAE models for CelebA, and L2 regularization for CIFAR-10.

Qualitative comparison

While Table 2.1 captures the quantitative performance of our method, we seek to provide a qualitative comparison as well. This is done in Figure 2.4. We compare to all RAE variants, as well as 2SVAE, WAE, CV-VAE and the standard VAE and AE as in Table 2.1. Results for CIFAR-10 and MNIST appear later in the additional results Section.

2.5.2 Gaussian latents sans entropic regularization

A surprising result emerges as we make the latent space dimensionality lower while ensuring a complex enough decoder and encoder. Though we discussed this process earlier in the context of depth, our architectures are convolutional and a better heuristic proxy is the number of channels while keeping the depth constant. We note that all our encoders share a power of 2 framework, i.e. channels double every layer from 128. Keeping this doubling

	CIFAR-10		CelebA	
Architectures(Isotropic)	FID	Reconstruction	FID	Reconstruction
VAE	106.37	57.94	48.12	39.12
CV-VAE	94.75	37.74	48.87	40.41
WAE	117.44	35.97	53.67	**34.81**
2SVAE	109.77	62.54	49.70	42.04
EAE	**85.26(84.53)**	29.77	**44.63**	40.26
Architectures(GMM)	FID	Reconstruction	FID	Reconstruction
RAE	76.28	29.05	44.68	40.18
RAE-L2	74.16	32.24	47.97	43.52
RAE-GP	76.33	32.17	45.63	39.71
RAE-SN	75.30	**27.61**	40.95	36.01
AE	76.47	30.52	45.10	40.79
AE-L2	75.40	34.35	48.42	44.72
EAE	**73.12**	29.77	**39.76**	40.26

Table 2.1: FID scores for relevant VAEs & VAE-like architectures. Scores within parentheses for EAE denote regularization on a linear map. Isotropic denotes samples drawn from latent spaces of $\mathcal{N}(0, I)$. GMM denotes sampling from a mixture of 10 Gaussians of full covariance. These evaluations correspond to analogous benchmarking for RAEs [12]. Larger version in additional results Section.

Figure 2.4: Qualitative comparisons to RAE variants and other standard benchmarks on CelebA. On the left, we have reconstructions (top row being ground truth GT) , the middle has generated samples, the right has interpolations. From top to bottom ignoring GT: VAE, CV-VAE, WAE, 2SVAE, RAE-GP, RAE-L2, RAE-SN, RAE, AE, EAE. Non-EAE Figures reproduced from [12]

structure, we investigate the effect of width on the latent space with no entropic regularizer. We set the channels to double from 64, i.e. $64, 128, 256, 512$ and correspondingly in the decoder for MNIST. Figure 2.5 shows the samples with the latent dimension being set to 8, and the result when we take corresponding samples from an isotropic Gaussian when the number of latents is 32.

There is a large, visually evident drop in sample quality by going from a narrow autoencoder to a wide one for generation, when no constraints on the latent space are employed. To confirm the analysis, we provide the result for 16 dimensions in the Figure as well, which is intermediate in quality.

The aforesaid effect is not restricted to MNIST. We perform a similar study on CelebA

Figure 2.5: Variation in bottleneck width causes massive differences in generative quality without regularization. From left to right, we present samples (from $\mathcal{N}(0, I)$ for $8, 16, 32$ dimensional latent spaces)

taking the latent space from 48 to 128, and the results in Figures 2.6 and 2.7 show a corresponding change in sample quality. Of course, the results with 48 dimensional unregularized latents are worse than our regularized, 64 dimensional sample collage in Figure 2.2, but they retain facial quality without artifacts. FID scores (provided in caption) also follow this trend.

Figure 2.6: Generated images on CelebA with a narrow bottleneck of 48, unregularized. The associated FID score was 53.82.

In our formulation of the MaxEnt principle, we considered more complex maps (e.g., deeper or wider networks with possibly more channels) able to induce more arbitrary deformations between a latent space and the target space. A narrower bottleneck incentivizes Gaussianization - with a stronger bottleneck, each latent carries more information, with higher entropy in codes Z, as discussed in our parallels with Information Bottleneck-like methods.

We present a similar analysis between CIFAR-10 AEs without regularization. Unlike previous cases, CIFAR-10 samples suffer from the issue that visual quality is less evident to

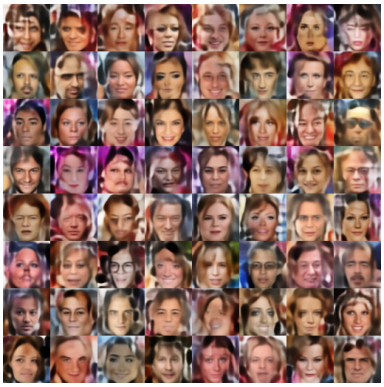

Figure 2.7: Generated images on CelebA with latent dimensions of 128, also unregularized. This associates a FID score of 64.72.

the human eye. These Figures are presented in Figures 2.8 and 2.9. The approximate FID difference between these two images is roughly 13 points (≈ 100 vs ≈ 87). While FID scores are not meaningful for MNIST, we can compare CelebA and CIFAR-10 in terms of FID scores (provided in Figure captions). These back up our assertions. For all comparisons, only the latent space is changed and the best checkpoint is taken for both models - we have a case of a less complex model outperforming another that can't be due to more channels allowing for better reconstruction, explainable in MaxEnt terms.

Figure 2.8: Generated images on CIFAR-10 with unregularized latent dimension of 128. The FID score is 100.62, with L2 regularization.

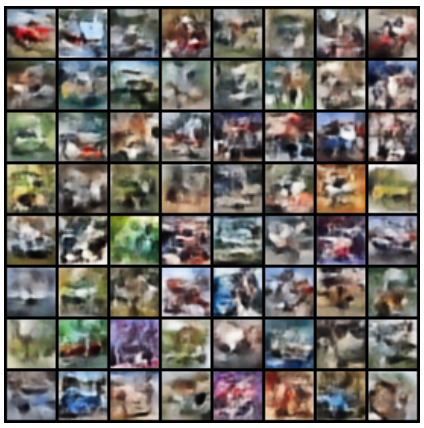

Figure 2.9: Generated images on CIFAR-10 with unregularized latent dimension equal to 64. The FID score associated with this checkpoint is 87.45, trained using L2 regularization.

Additional results

We present additional qualitative results that provide further insight into comparisons to RAE architectures, detailed on the next page.

We add in this page general notes on the training of EAEs, as requested by UAI 2020 reviewers and for general perusal of would-be EAE practitioners. In particular, across extensive experiments, we noted the following empirical trends and heuristics which we choose to pass on for the sake of ease of implementation.

- Results for all EAEs that use the Kozachenko-Leonenko or similar KNN based entropy estimators can be improved in general by using ≥ 2 neighbours per minibatch. However, we do not recommend this. Performance gains from this are slight, and for most applications, using 1 suffices.
- We recommend using at least 3 conv-deconv layers in the encoder and decoder for any autoencoding pair for all three datasets for best FID scores.
- For both CIFAR and CelebA, we recommend a minimum latent size of 32.
- For CIFAR-10 in particular, learning rate decay is critical when using the Adam optimizer. We use an exponential decay with a decay rate ≥ 0.98. It should be noted that [12] use a more complex schedule that involves looking at the validation loss. We did not require such.
- Good samples emerge early - samples for all three datasets generated by epoch 10 as evaluated by a human eye are highly predictive of eventual best performance in terms of FID. As such, it is recommended to periodically generate samples and visually inspect them.

Preprocessing datasets

Here, we detail the pre-processing of datasets common to our methods and the methods we benchmark against. We carry out no pre-processing for CIFAR-10. For MNIST, we pad with zeros to reach 32×32 as the shape. For CelebA, pre-processing is important and can vastly change FID scores. We perform a center-crop to 140×140 before resizing to 64×64.

Details of following material

In Table 2.2 below, we present a larger version of the FID results from Table 2.1 in the main Chapter. In Figures 2.10 and 2.11 below, we also present qualitative results including reconstruction and interpolations on the latent space that serve to show that the latent spaces obtained by EAEs are meaningful. These experiments on latent spaces mirror [12].

	CIFAR-10		CelebA	
Architectures(Isotropic)	FID	Reconstruction	FID	Reconstruction
VAE	106.37	57.94	48.12	39.12
CV-VAE	94.75	37.74	48.87	40.41
WAE	117.44	35.97	53.67	**34.81**
2SVAE	109.77	62.54	49.70	42.04
EAE	**85.26(84.53)**	29.77	**44.63**	40.26
Architectures(MVG)	FID	Reconstruction	FID	Reconstruction
RAE	83.87	29.05	48.20	40.18
RAE-L2	80.80	32.24	51.13	43.52
RAE-GP	83.05	32.17	116.30	39.71
RAE-SN	84.25	**27.61**	44.74	36.01
AE	84.74	30.52	127.85	40.79
AE-L2	247.48	34.35	346.29	44.72
EAE	**80.07**	29.77	**42.92**	40.26
Architectures(GMM)	FID	Reconstruction	FID	Reconstruction
VAE	103.78	57.94	45.52	39.12
CV-VAE	86.64	37.74	49.30	40.41
WAE	93.53	35.97	42.73	**34.81**
2SVAE	N/A	62.54	N/A	42.04
RAE	76.28	29.05	44.68	40.18
RAE-L2	74.16	32.24	47.97	43.52
RAE-GP	76.33	32.17	45.63	39.71
RAE-SN	75.30	**27.61**	40.95	36.01
AE	76.47	30.52	45.10	40.79
AE-L2	75.40	34.35	48.42	44.72
EAE	**73.12**	29.77	**39.76**	40.26

Table 2.2: FID scores for relevant VAEs and VAE-like architectures. Scores within parentheses for EAE denote a regularization on a linear map. Isotropic denotes samples drawn from a latent space of $\mathcal{N}(0, I)$. GMM denotes sampling from a mixture of 10 Gaussians of full covariance. These evaluations correspond to analogous benchmarking for RAEs [12]. Alongside FID values appearing in Table 2.1 of the main Chapter, we add results obtained when a Multivariate Gaussian (MVG) i.e. $\mathcal{N}(\mu, \Sigma)$ of full covariance is used for ex-post density estimation. Note that values for reconstruction are not changed by change of density estimators.

Figure 2.10: Qualitative comparisons to RAE variants and other standard benchmarks on CIFAR-10. On the left, we have reconstructions (top row being ground truth GT), the middle has generated samples, the right has interpolations. From top to bottom ignoring GT: VAE, CV-VAE, WAE, 2SVAE, RAE-GP, RAE-L2, RAE-SN, RAE, AE, EAE. Non-EAE Figures reproduced from [12]

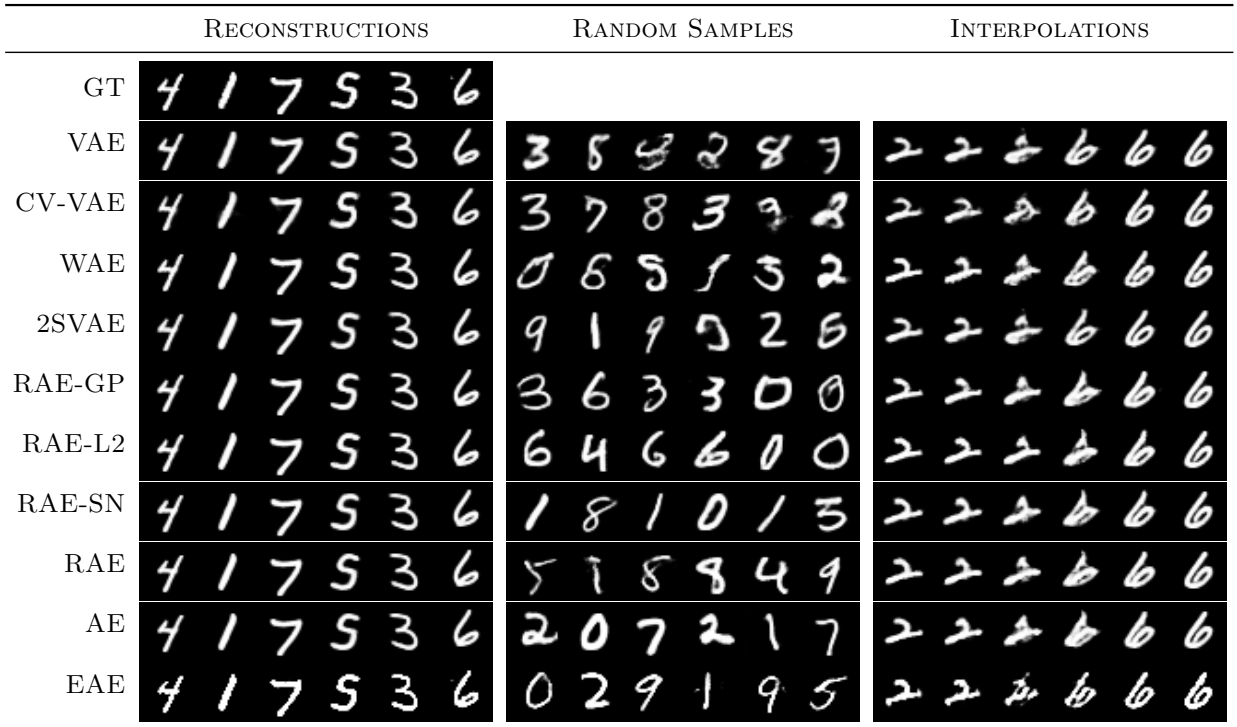

Figure 2.11: Qualitative comparisons to RAE variants and other standard benchmarks on MNIST. On the left, we have reconstructions (top row being ground truth GT) , the middle has generated samples, the right has interpolations. From top to bottom ignoring GT: VAE, CV-VAE, WAE, 2SVAE, RAE-GP, RAE-L2, RAE-SN, RAE, AE, EAE. Non-EAE Figures reproduced from [12]

2.6 Conclusions

The VAE has remained a popular deep generative model, while drawing criticism for its blurry images, posterior collapse and other issues. Deterministic encoders have been posited to escape blurriness, since they 'lock' codes into a single choice for each instance. We consider our work as reinforcing Wasserstein autoencoders and other recent work in deterministic autoencoders such as RAEs [12]. In particular, we consider our method of raising entropy to be generalizable whenever batch normalization exists, and note that it solves a more specific problem than reducing the KL between two arbitrary distributions P, Q, examining only the case where P, Q satisfy moment constraints. Such reductions can make difficult problems tractable via simple estimators.

We had initially hoped to obtain results via sampling from a prior distribution that were, without ex-post density estimation, already state of the art. In practice, we observed that using a GMM to fit the density improves results, regardless of architecture. These findings might be explained in light of the 2-stage VAE analysis [7], wherein it is postulated that single-stage VAEs inherently struggle to capture Gaussian latents, and a second stage is amenable. To this end, we might aim to design a 2-stage EAE. Numerically, we found such an architecture hard to tune, as opposed to a single stage EAE which was robust to the choice of hyperparameters. We believe this might be an interesting future direction.

We note that our results improve on the RAE, which in turn improved on the 2SVAE FID numbers. Though the latest GAN architectures remain out of reach in terms of FID scores for most VAE models, 2SVAE came within striking distance of older ones, such as the vanilla WGAN. Integrating state of the art techniques for VAEs as in, for instance, VQVAE2 [43] to challenge GAN-level benchmarks could form an interesting future direction. Quantized latent spaces also offer a more tractable framework for entropy based models and allow us to work with discrete entropy which is a more meaningful function. As noted earlier, we do not compare to the ResNet equipped bigWAE models [52], which are far larger but also deliver better results (up to 35 for CelebA).

The previous work on RAEs [12], which our method directly draws on deserves special addressal. The RAE method shows that deterministic autoencoders can succeed at generation, so long as regularizers are applied and post-density estimation is carried out. Yet, while regularization is certainly nothing out of the ordinary, the density estimation step robs RAEs of sampling from any isotropic prior. We improve on the RAE techniques when density estimation is in play, but more pertinently, we keep a method for isotropic sampling that is rigorously equivalent to cross entropy minimization. As such, we offer better performance while adding more features, and our isotropic results far outperform comparable isotropic benchmarks.

Chapter 3

Batch normalization, radial contraction and implications

3.1 Introduction

Let us begin by stating upfront that the results in this Chapter **merely, as of now,** reflect some interesting and consistent observations on the norms of adversarial examples. In the present state, this result is not sufficient to distinguish between adversarial and normal examples, due to significant overlap despite clear average differences. This caveat should be kept in mind when perusing the rest of the Chapter that these results are interesting but do not, as of this moment, have a **clear use case**. However, they are nevertheless deemed relevant enough to the core thrust of the thesis to be included.

Adversarial examples x_{adv} are commonly defined as data instances which lie only ϵ in some norm (usually L_∞) from an actual data instance x_{real}. To a human observer, ϵ is small and x_{real}, x_{adv} share the same label, yet to a classifier network, this is not the case. Since their initial discovery [50], they have spurred research in both attacking (generation of adversarial data instances) and defending neural networks against the same.

The common variants of generating adversarial images rely on gradient steps. One of the earliest and most effective attacks is the Fast Gradient Sign Method (FGSM) [14], and a far more effective variant can be found via Projected Gradient Descent (PGD) [40] . For defense, methods include adversarial training, in which adversarial images are fed to the network. Adversarial training, though expensive, is one of the best ways to defend. Many defenses have been shown to give a false sense of security by obfuscation of gradients [1].

Various models of defense rely on 'inspecting' the image to determine whether it is adversarial or not, and filtering correspondingly. A representative for this angle is PixelDefend [49]. In this method, we use a generative model which assigns likelihoods to data instances. The assumption is that adversarial examples possess lower likelihoods in comparison to real ones.

Can such models of inspection or filtering be used while working with the latent representations created by the data instances instead of in the raw image domain? This approach has the advantage of working with a space of much lower dimensionality compared to the input space, making building our generative model much easier. Therefore, it makes sense to look at the representations induced by adversarial examples, which we will call **adversarial representations** henceforth.

In this Chapter, we show that after batch normalization, adversarial representations tend to exhibit much lower L_2 norms than the representations of original instances. This is of potential interest if we view the instances as obeying a Gaussian likelihood, and as such, adversarial instances actually being **likelier**. We also put this result in the context of a possible explanation via the relation between Gaussians, batch normalization, and the geometry of Gaussians in high dimensions.

3.1.1 Gaussians, entropy and batchnorm

Suppose that in the process of training, the sample means and standard deviations in a pre-BN layer converge to actual sample statistics. After the batchnorm layer, the representation vector Z satisfies (a la Chapter 2):

$$E(Z_i) = 0, E(Z_i^2) = 1$$

Under the above constraints, the maximum entropy of Z is attained iff $Z \sim \mathcal{N}(0, I)$. The distribution over Z resembles a Gaussian (in KL divergence) as entropy rises - if $P = \mathcal{N}(0, I)$ and Z obeys above two constraints, with $Z \sim Q$, writing $H(Q)$ as the entropy of Z which is $\sim Q$, we have:

$$H(Q, P) = H(Q) + D_{KL}(Q||P)$$

where $H(Q, P)$ is $E_Q[-\log P(Z)]$ - a constant over any Q that generates random variables Z satisfying these moment constraints, since the expectation of $E_Q[-\log P(Z)]$ is wholly determined by $E_Q[Z], E_Q[Z^2]$ when P is the isotropic Gaussian.

3.2 Geometry of adversarial representations

Consider a real data instance x_{real}, and let us create x_{adv} from it via FGSM or PGD. Our experiments (discussed further below) demonstrate that x_{adv} produces a d-dimensional representation Z_{adv} (post batchnorm) which obeys $\|Z_{adv}\|_2 < \|Z_{real}\|_2$, often significantly.

3.2.1 Relation to gaussianization

We claim that the above is explainable if we assume that Z are distributed $\sim \mathcal{N}(0, I_d)$. For high d, the Gaussian distribution has almost all of its support over a thin shell [54] - the random variable $\|Z\|_2$ is tightly concentrated around $\sqrt{d}$. Given two representations Z_A, Z_B of different classes A, B on the surface of a hypersphere of radius $\sqrt{d}$, an adversarial example's representation Z_{adv} seeks to move from Z_A towards Z_B approximately along the chord joining them. Movement along a chord brings the representation 'inside' the sphere (Figure 3.1). Analogous cases arise for L_2 constrained representation learning [55].

We note that if the sphere is thought of as a manifold, previous works have discussed adversarial examples moving 'off' the manifold. Relatively less attention has been devoted to whether they move 'inside' the manifold, however.

3.2.2 Circumstances of Gaussianization

We assume above that $Z \sim \mathcal{N}(0, I_d)$, or, since Z follows a batch norm, equivalently that $H(Z)$ is high. The information bottleneck formulation of deep networks states deep nets seek to achieve intermediate representations Z lowering $\text{MI}(X, Z)$ where X is the input, and raise $\text{MI}(Y, Z)$ where Y is the output, where MI is the mutual information between two random variables. We recall that H, the entropy, is the expected value of $-\log P(X)$ over instances X generated from a data distribution X and the mutual information $\text{MI}(X, Y)$ is $H(Y) - H(Y|X)$.

Assuming that the deep network is deterministic - an assumption that can accommodate batch norm layers so long as they are set to use population statistics over minibatch ones at test time - we note that $\text{MI}(X, Z)$ is $H(Z) - H(Z|X)$, where the latter term is an undefined quantity as Z is a deterministic function of X. If we treat it as a constant and focus on $H(Z)$, raising $H(Z)$ 'raises' the MI. This is justified iff the corresponding rise in entropy improves classification accuracy wrt Y, as seen in the $\text{MI}(Y, Z)$ term which is $H(Y) - H(Y|Z)$. The first term, $H(Y)$ is a constant given the dataset, but the latter term $H(Y|Z)$ falls as classification accuracy rises.

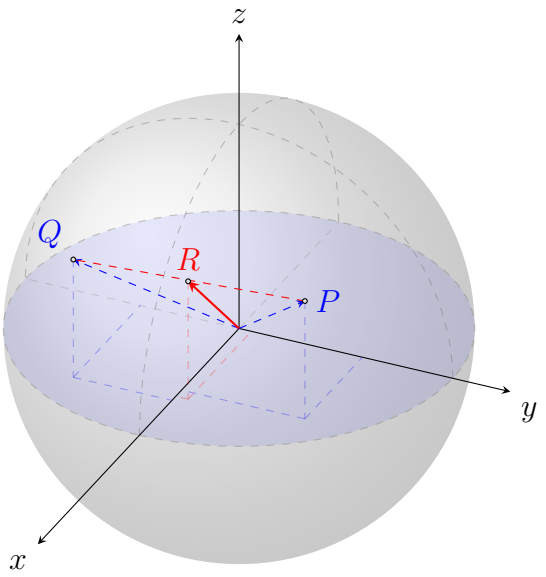

Figure 3.1: In the image above, P,Q are latent representations of actual images, while R lies on the chord joining them and has lower radial distance from the center

Such a tradeoff is thus likely when Z is low-dimensional and struggles to summarize X to improve Y. Narrow layers succeeding BatchNorm layers might be expected to have distributions close to Gaussians, especially as X grows more complex (such as switching to CIFAR-10 from MNIST).

Further, note that if we have $Z \sim$ as a Gaussian, X as the data, and Y is the label, the space of functions must be large enough to convert X to Z and then again to Y, wherein the distributions of X, Y are inherently unknown. In practical terms, increasing the space of functions largely boils down to adding more depth both before and after the batchnormed layer.

3.3 Empirical results

We create two separate classes of architectures for MNIST and CIFAR-10. The architectures for MNIST are similar to LeNet, while that of CIFAR-10 is similar to stacking VGG blocks. This is done in order to make sure our architectures are similar to popular models. For both CIFAR and MNIST, one model resembles a CNN, and the other is a CNN with a

	MNIST	CIFAR-10
CNN	$x \in \mathcal{R}^{28\times28\times1}$ $\to$ C_{10} $\to$ MP_2 $\to$ ReLU $\to$ C_{20} $\to$ MP_2 $\to$ ReLU $\to$ Flatten $\to$ L_{320} $\to$ BN $\to$ ReLU $\to$ L_W $\to$ BN $\to$ ReLU $\to$ L_{10} $\to$ Logsoftmax	$x \in \mathcal{R}^{32\times32\times3}$ $\to$ C_{32} $\to$ MP_2 $\to$ ReLU $\to$ C_{64} $\to$ MP_2 $\to$ ReLU $\to$ C_{128} $\to$ MP_2 $\to$ ReLU $\to$ Flatten $\to$ L_{2048} $\to$ BN $\to$ ReLU $\to$ L_W $\to$ BN $\to$ ReLU $\to$ L_{10} $\to$ Logsoftmax
AE-CNN	$x \in \mathcal{R}^{28\times28\times1}$ $\to$ C_{10} $\to$ MP_2 $\to$ ReLU $\to$ C_{20} $\to$ MP_2 $\to$ ReLU $\to$ Flatten $\to$ L_{320} $\to$ BN $\to$ ReLU $\to$ L_W $\to$ BN $\to$ ReLU $\to$ $L_{W'}$ $\to$ BN $\to$ ReLU $\to$ L_{10} $\to$ Logsoftmax	$x \in \mathcal{R}^{32\times32\times3}$ $\to$ C_{32} $\to$ MP_2 $\to$ ReLU $\to$ C_{64} $\to$ MP_2 $\to$ ReLU $\to$ C_{128} $\to$ MP_2 $\to$ ReLU $\to$ Flatten $\to$ L_{2048} $\to$ BN $\to$ ReLU $\to$ L_W $\to$ BN $\to$ ReLU $\to$ $L_{W'}$ $\to$ BN $\to$ ReLU $\to$ L_{10} $\to$ Logsoftmax

Table 3.1: Architectural details for the experiments. One extra layer is added for performance after the bottleneck in AE-CNN. L denotes a linear layer, MP a MaxPool layer. The BN layers contain affine shifts (attached separately as a linear layer, not shown). L_W denotes a linear layer of width W. We list the values used for W, W' : $W = 30, 40, 50$ is used for both MNIST and CIFAR-10 for CNN. For CIFAR-10, we additionally use $120, 160, 200$ and for MNIST $90, 120, 150$. For $W' = 100$, we use $10, 20, 30, 40$ for both MNIST and CIFAR-10. $W' = 320$ for MNIST has $W = 40, 80, 120, 160$ and for CIFAR-10 as $50, 100, 150, 200$.

narrow bottleneck and one extra layer, to resemble autoencoder-like compression and test our hypothesis. Architectural details appear in Table 3.1. All models are trained using the Adam optimizer with a learning rate of 0.001 and checkpointed every 5 epochs.

3.3.1 Radial contraction

Given the data instance X, we calculate Z which is the representation of X after passing through the layer L_W, normalized to zero mean and unit variance but not affine-shifted. Now, observe that the classification loss - in this case negative log likelihood on the log

MNIST	W	$\#\langle\nabla_Z L, Z\rangle < 0$	$\#\langle\nabla_Z L, Z\rangle \geq 0$
Normal CNN	30	8432	1568
	40	8378	1622
	50	7407	2593
	90	7504	2496
	120	7885	2115
	150	7286	2714
AE CNN	10	9073	927
	20	9826	174
	30	9838	162
	40	9861	139
	40	9826	174
	80	9809	191
	120	9680	320
	160	9801	199

Table 3.2: Gradient inner products for datasets for MNIST, gray for $W' = 320$

softmax outputs - is a function of Z (by feeding it forward), and if moving Z towards the origin increased this, then the gradient of the loss wrt Z i.e. $\nabla_Z L$ would obey $\langle\nabla_Z L, Z\rangle \leq 0$. This is empirically verified for both MNIST and CIFAR in Tables 3.2 and 3.3. The above property is especially obeyed for bottlenecked architectures.

The above result indicates - but does not necessarily show - that moving X towards the loss direction (via FGSM/PGD) creates Z of lower L_2 norms. We now move on to the actual statistics of adversarial representations.

3.3.2 L_2 norms of adversarial representations

From the above, we can see that AE-like bottlenecking strongly promotes representations Z which obey $\langle\nabla_Z L, Z\rangle \leq 0$. We now show the average squared L_2 norms of Z and adversarial representations of the same, obtained via FGSM and PGD for all architectures. In all these cases, $\epsilon = 0.01$ for CIFAR-10, and 0.03 for MNIST. The point is to choose a reasonably small ϵ, since our arguments about adversarial representations rely on them being close to real ones.

Overall, we see a consistent pattern of lower squared L_2 norm when adversarial instances are used to generate the representations, in Tables 3.5, 3.7, 3.4 and 3.6.

CIFAR-10	W	$\#\langle \nabla_Z L, Z \rangle < 0$	$\#\langle \nabla_Z L, Z \rangle \geq 0$
Normal CNN	30	5453	4547
	40	5288	4712
	50	5208	4792
	120	5034	4966
	160	4874	5126
	200	4716	5284
AE CNN	10	8157	1843
	20	8775	1225
	30	8840	1160
	40	8982	1018
	50	8593	1407
	100	8220	1780
	150	8304	1696
	200	7507	2493

Table 3.3: Gradient inner products for CIFAR-10, gray for $W' = 320$

3.3.3 Pairwise comparisons of L_2 norms

In this segment, we seek to find out if for corresponding pairs x_{real}, x_{adv}, most latent pairs Z_{real}, Z_{adv} reflect $\|Z_{real}\|_2 \geq \|Z_{adv}\|_2$. Note that this is not equivalent to stating anything about the average of $\|Z_{real}\|_2, \|Z_{adv}\|_2$, which can be skewed by massive outliers in either direction. We test this on the AE-CNN architecture for MNIST of $W = 40, W' = 100$ which exhibits a slanted ratio for $\langle \nabla_Z L, Z \rangle < 0$. We plot the L_2 norm of the adversarial representations vis-a-vis real ones in Figure 3.2. The resultant trend can be seen to be a consistent one.

Width	Normal	FGSM	PGD
10	5.26	3.91	3.31
20	19.05	12.46	11.22
30	40.24	21.67	22.49
40	47.06	23.26	26.86
50	66.48	34.73	35.27
100	105.09	61.91	71.83
150	162.62	103.89	111.54
200	215.80	176.28	211.01

Table 3.4: CIFAR-10 L_2 norms vs different widths W, gray for $W' = 320$

Width	Normal	FGSM	PGD
10	3.71	2.32	2.12
20	7.69	4.94	4.76
30	11.43	8.00	7.46
40	15.72	10.58	10.94
40	15.06	10.31	9.79
80	31.20	22.35	23.09
120	46.75	35.41	34.8
160	60.11	44.8	44.19

Table 3.5: MNIST L_2 norms vs different widths W, gray for $W' = 320$

Width	Normal	FGSM	PGD
30	40.77	16.97	26.71
40	45.05	18.58	25.59
50	59.61	28.96	38.81
120	135.81	91.18	116.90
160	166.47	112.29	142.46
200	208.27	186.99	280.37

Table 3.6: CIFAR CNN, L_2 norms on varying W

Width	Normal	FGSM	PGD
30	11.16	8.45	8.38
40	14.65	11.60	11.55
50	18.96	16.15	15.77
90	33.32	27.73	26.03
120	44.72	38.61	36.25
150	57.66	49.76	46.03

Table 3.7: MNIST CNN, L_2 norms on varying W

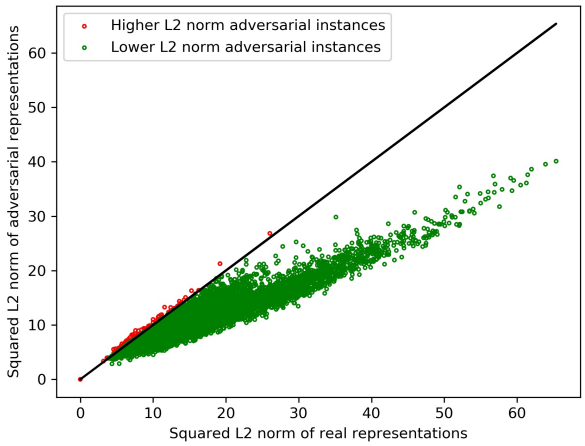

Figure 3.2: Pairwise comparisons of norms, MNIST

Chapter 4

Conclusion

The key theme of the works presented herein over the thesis is that batch normalization, in its choice of sufficient statistics, presents a hidden inductive bias towards Gaussian distributions. The idea that choosing some sufficient statistics biases us towards some distributions is not new, and such ideas are used to formulate, for instance, exponential families [17] and Gibbs distributions [10]. However, the usage of suitable entropic regularizers that exploit this relation is, to our knowledge, novel. This yields direct improvements for some models, as in the case of the entropic autoencoder presented.

More interestingly, examining situations where high entropy is incentivized, we gain insight into how this kind of entropic regularization might arise by itself without consciously put in as part of the optimization, thus, we can glean insight into the **inductive bias** of using batch normalization over any other choice. Such hidden biases may in general lead to further insights down the road.

We would like to conclude by noting that though these relations to distributions are most readily exploitable for generative models which use the distribution in question for a sampling step, as in the case of aggregate posterior methods, [52, 12], they can be used to work with classifier models as well. This is the direction Chapter 3 attempts to head in. However, as we mention straight away in the Chapter, this direction of the work remains in progress. It is hoped we can soon refine these findings into a solid use case, from merely interesting observations. This revolves around possibly using the L_2 norm as a diagnostic tool to detect adversarial examples solely from their latent representations, for instance, analogous to PixelDefend [49], or using the L_2 norm as a surrogate loss to generate adversarial examples for adversarial training, for the reverse role of being the attacker.

References

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

[2] J L Ba, J R Kiros, and G E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[3] AG Bashkirov. On maximum entropy principle, superstatistics, power-law distribution and renyi parameter. *Physica A: Statistical Mechanics and its Applications*, 340(1-3):153–162, 2004.

[4] A L Berger, V J D Pietra, and S A D Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[5] V Chiley, I Sharapov, A Kosson, U Koster, R Reece, S S de la Fuente, V Subbiah, and M James. Online normalization for training neural networks. *arXiv:1905.05894*, 2019.

[6] T M Cover and J A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[7] B Dai and D Wipf. Diagnosing and enhancing vae models. *arXiv:1903.05789*, 2019.

[8] Sylvain Delattre and Nicolas Fournier. On the kozachenko–leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.

[9] L Dinh, J Sohl-Dickstein, and S Bengio. Density estimation using real nvp. *arXiv:1605.08803*, 2016.

[10] Howard Elliott, Haluk Derin, Roberto Cristi, and Donald Geman. Application of the gibbs distribution to image segmentation. In *ICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 9, pages 678–681. IEEE, 1984.

[11] A Galloway, A Golubeva, T Tanay, M Moussa, and G W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv:1905.02161*, 2019.

[12] P Ghosh, M SM Sajjadi, A Vergari, M Black, and B Schölkopf. From variational to deterministic autoencoders. *arXiv:1903.12436*, 2019.

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

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

[15] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[16] A Gretton, K M Borgwardt, M J Rasch, B Schölkopf, and A Smola. A kernel two-sample test. *JMLR*, 13(Mar):723–773, 2012.

[17] Rameshwar D Gupta and Debasis Kundu. Exponentiated exponential family: an alternative to gamma and weibull distributions. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 43(1):117–130, 2001.

[18] J He, D Spokoyny, G Neubig, and T Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv:1901.05534*, 2019.

[19] M Heusel, H Ramsauer, T Unterthiner, B Nessler, and S Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6626–6637, 2017.

[20] I Higgins, L Matthey, A Pal, C Burgess, X Glorot, M Botvinick, S Mohamed, and A Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[21] E Hoffer, R Banner, I Golan, and D Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *NeurIPS*, pages 2160–2170, 2018.

[22] C-W Huang, D Krueger, A Lacoste, and A Courville. Neural autoregressive flows. *arXiv:1804.00779*, 2018.

[23] S Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NeurIPS*, pages 1945–1953, 2017.

[24] S Ioffe and C Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[25] E T Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, 1982.

[26] Y Kim, S Wiseman, A C Miller, D Sontag, and A M Rush. Semi-amortized variational autoencoders. *arXiv:1802.02550*, 2018.

[27] D P Kingma and P Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, pages 10215–10224, 2018.

[28] D P Kingma and M Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[29] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[30] S Kolouri, P E Pope, C E Martin, and G K Rohde. Sliced-wasserstein autoencoder: an embarrassingly simple generative model. *arXiv:1804.01947*, 2018.

[31] LF Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

[32] A Kraskov, H Stögbauer, and P Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[33] A Krizhevsky, V Nair, and G Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[34] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[35] Y LeCun, C Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

[36] K Li and J Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

[37] K Li, T Zhang, and J Malik. Diverse image synthesis from semantic layouts via conditional imle. In *ICCV*, pages 4220–4229, 2019.

[38] Z Liu, P Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

[39] Gabriel Loaiza-Ganem, Yuanjun Gao, and John P Cunningham. Maximum entropy flow networks. *arXiv preprint arXiv:1701.03504*, 2017.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[41] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[42] G Papamakarios, T Pavlakou, and I Murray. Masked autoregressive flow for density estimation. In *NeurIPS*, pages 2338–2347, 2017.

[43] A Razavi, A van den Oord, and O Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, pages 14837–14847, 2019.

[44] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

[45] D J Rezende and S Mohamed. Variational inference with normalizing flows. *arXiv:1505.05770*, 2015.

[46] S Santurkar, D Tsipras, A Ilyas, and A Madry. How does batch normalization help optimization? In *NeurIPS*, pages 2483–2493, 2018.

[47] J Shore and R Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[48] J Skilling and RK Bryan. Maximum entropy image reconstruction-general algorithm. *Monthly notices of the royal astronomical society*, 211:111, 1984.

[49] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

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

[51] N Tishby, F C Pereira, and W Bialek. The information bottleneck method. *arXiv*, 2000.

[52] I Tolstikhin, O Bousquet, S Gelly, and B Schoelkopf. Wasserstein auto-encoders. *arXiv:1711.01558*, 2017.

[53] D Ulyanov, A Vedaldi, and V Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[54] Roman Vershynin. High-dimensional probability, 2019.

[55] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020.

[56] S Wu, G Li, L Deng, L Liu, D Wu, Y Xie, and L Shi. L1-norm batch normalization for efficient training of deep neural networks. *IEEE transactions on neural networks and learning systems*, 2018.

[57] G Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv:1902.04760*, 2019.

[58] G Yang, J Pennington, V Rao, J Sohl-Dickstein, and S S Schoenholz. A mean field theory of batch normalization. *arXiv:1902.08129*, 2019.

[59] H Zhang, Y N Dauphin, and T Ma. Fixup initialization: Residual learning without normalization. *arXiv:1901.09321*, 2019.

[60] S Zhao, J Song, and S Ermon. Infovae: Information maximizing variational autoencoders. *arXiv:1706.02262*, 2017.

[61] B D Ziebart, A Maas, J A Bagnell, and A K Dey. Maximum entropy inverse reinforcement learning. 2008.