Université de Montréal

Improved Training of Energy-Based Models

par Rithesh Kumar

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

June, 2019

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Improved Training of Energy-Based Models

présenté par:

Rithesh Kumar

a été évalué par un jury composé des personnes suivantes:

Stefan Monnier,	président-rapporteur
Yoshua Bengio,	directeur de recherche
Neil Stewart,	membre du jury

Mémoire accepté le: .....

Résumé

L'estimation du maximum de vraisemblance des modèles basés sur l'énergie est un problème difficile à résoudre en raison de l'insolubilité du gradient du logarithmique de la vraisemblance. Dans ce travail, nous proposons d'apprendre à la fois la fonction d'énergie et un mécanisme d'échantillonnage approximatif amorti à l'aide d'un réseau de générateurs neuronaux, qui fournit une approximation efficace du gradient de la log-vraisemblance. L'objectif qui en résulte exige la maximisation de l'entropie des échantillons générés, que nous réalisons en utilisant des estimateurs d'information mutuelle non paramétriques récemment proposés. Enfin, pour stabiliser le jeu antagoniste qui en résulte, nous utilisons une pénalité du gradient, centrée en zéro, dérivée comme condition nécessaire issue de la littérature sur l'alignement des scores. La technique proposée peut générer des images nettes avec des scores d'Inception et de FID compétitifs par rapport aux techniques récentes de GAN, ne souffrant pas d'effondrement de mode, et compétitive par rapport aux techniques de détection d'anomalies les plus récentes.

Le chapitre 1 introduit les concepts essentiels à la compréhension des travaux présentés dans cette thèse, tels que les modèles graphiques fondés sur l'énergie, les méthodes de Monte-Carlo par chaînes de Markov, les réseaux antagonistes génératifs et l'estimation de l'information mutuelle. Le chapitre 2 contient un article détaillant notre travail sur l'amélioration de l'entraînement des fonctions d'énergie. Enfin, le chapitre 3 présente quelques conclusions tirées de ce travail de thèse, la portée des travaux futurs, ainsi que des questions ouvertes qui restent sans réponse.

môts-cles: apprentissage profond, apprentissage non supervisé, modèles génératifs, modèles basés sur l'énergie

Summary

Maximum likelihood estimation of energy-based models is a challenging problem due to the intractability of the log-likelihood gradient. In this work, we propose learning both the energy function and an amortized approximate sampling mechanism using a neural generator network, which provides an efficient approximation of the log-likelihood gradient. The resulting objective requires maximizing entropy of the generated samples, which we perform using recently proposed nonparametric mutual information estimators. Finally, to stabilize the resulting adversarial game, we use a zero-centered gradient penalty derived as a necessary condition from the score matching literature. The proposed technique can generate sharp images with Inception and FID scores competitive with recent GAN techniques, does not suffer from mode collapse, and is competitive with state-of-the-art anomaly detection techniques.

Chapter 1 introduces concepts that are crucial to understanding the work presented in the thesis, such as Energy-based graphical models, Markov Chain Monte Carlo, Generative Adversarial Networks and Mutual Information Estimation. Chapter 2 contains a detailed article about our work on improved training of energy functions. Chapter 3 provides some conclusions drawn from this thesis work and scope for future work and open questions that have been left unanswered.

Keywords: deep learning, unsupervised learning, generative models, energy-based models

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

MEG	Maximum Entropy Generators
MCMC	Markov Chain Monte Carlo
GAN	Generative Adversarial Networks
PCA	Principal Components Analyis
t-SNE	t-Distributed Stochastic Neighbour Embedding
CI	Conditional Independence
GM	Graphical Model
DGM	Directed Graphical Model
DAG	Directed Acyclic Graph
HMM	Hidden Markov Model
VAE	Variational Auto-Encoder
UGM	Undirected Graphical Model
CPD	Conditional Probability Distribution
MALA	Metropolis Adjusted Langevin Algorithm
MH	Metropolis Hastings
RBM	Restricted Boltzmann Machine
WGAN	Wasserstein GAN
RNN	Recurrent Neural Networks
EBM	Energy Based Model
CD	Contrastive Divergence
MI	Mutual Information
PDF	Probability Density Function

Acknowledgments

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

I'm grateful to Prof. Yoshua Bengio for accepting me as a graduate student and for encouraging me to explore deep learning research in a variety of disciplines such as speech, vision and natural language processing. His guidance has been instrumental in helping me become the researcher that I am today. I would also like to thank Prof. Aaron Courville for all the helpful research guidance he has provided throughout my study.

I would like to the thank the following co-authors, co-workers, and colleagues (random order): Kundan Kumar, Kyle Kastner, Soroush Mehri, Jose Sotelo, Alexandre de Brébisson, Sandeep Subramanian, Sherjil Ozair, Anirudh Goyal, Tristan Deleu, Shagun Sodhani, Evan Racah, Sai Krishna, Ankesh Anand, Pablo Piantanida, Alessandro Sordoni, Philip Bachman, Sarath Chandar, Dmitriy Serdyuk, Alex Lamb, Olexa Bilanuik, Sai Rajeshwar, Philemon Brakel, Shawn Tan, Chinwei Huang, Sina Honari, Anh Nguyen, Kris Sankaran, Florian Golemo, Devansh Arpit, Iulian Serban, Jae Hyun Lim, Faruk Ahmed ; Thank you for all the though-provoking research conversations and brainstorming sessions.

I owe a huge debt of gratitude to Frédéric Bastien, Simon Lefrançois, Ahmed Mamlouk and Quentin Lux for their support and managing the computing infrastructure at the lab.

My graduate life wouldn't be as easy without the help of Céline Begin and Linda Peinthiére for all their administrative help during my studies.

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

1 Introduction

1.1 Overview

Machine learning is an important part of modern computer science with important applications across many industries including commerce, finance, logistics, agriculture, and education. Many detailed references exist for deeply understanding machine learning, such as [Bishop, 2006, Hastie et al., 2005, Murphy, 2012, Goodfellow et al., 2016].

In this thesis, we present our work - **Maximum Entropy Generators for Energy-based Models** (MEG), which focuses on advancing the state of art in *unsupervised learning* and *generative modeling*. Unsupervised learning is regarded as crucial for artificial intelligence because it promises to take advantage from unlabelled data [Lake et al., 2017]. This work primarily focuses on improving a particular class of algorithms to solve unsupervised learning called *energy-based modeling* derived from the *probabilistic graphical modeling* literature. Our work uses *deep learning* techniques with *neural networks* to perform function approximation.

This chapter strives to provide an overview of the pre-requisite concepts and terminology required in order to understand the research work presented in this thesis and its important contributions. We begin by first motivating the importance of advancing research in the topics of *unsupervised learning* and *generative modeling*. Second, we provide a short introduction to the field of *probabilistic graphical modeling* - which uses graphs to express conditional dependence structure between random variables in a probabilistic model. Third, we discuss *Markov Chain Monte Carlo* methods, which are a popular class of algorithms for sampling from high-dimensional probability distributions. Next, we discuss a class of methods for performing unsupervised learning called *energy-based modeling*, which is the

core focus of this thesis. This section also provides a historical perspective on the previous seminal works that serve as an inspiration for the research presented in this thesis. Finally, we explain terminologies and short concepts from recent deep learning literature, such as neural estimators of mutual information, generative adversarial networks and evaluation metrics used for measuring quality of images.

1.2 Contributions

We propose a novel framework for training energy-based models called Maximum Entropy Generators (MEG) to perform unsupervised learning. A key impediment to train energy-based models has been the requirement to sample from the energy-function during maximum likelihood estimation which requires running an expensive Markov Chain Monte Carlo (MCMC) process in each step. In this work, we provide an alternate, fast and efficient method for maximum likelihood training of energy-based models using amortized neural generators and entropy maximization techniques.

We show that the resulting energy function can be successfully used for *anomaly detection* and strongly outperforms recently published results with energy-based models. We show that MEG generates *sharp images* (with competitive scores in quantitative evaluation metrics such as Inception and Fréchet Inception Distance) and does not suffer from the common mode-mixing issue of many maximum likelihood generative models which results in blurry samples.

We also show that our model accurately captures more modes in the data distribution than standard generative adversarial networks (GANs), thereby solving the common *mode collapse* issue of state of the art GAN-based generative models.

We note that many terms in the above contributions may seem enigmatic to an average reader. We hope that the following sections in the introduction help in resolving the gap in knowledge that is required to understand the remainder of this thesis.

1.3 Unsupervised Learning

The focus of this research work broadly falls under the category of unsupervised learning and generative modeling. This section provides a short introduction to unsupervised learning, which is a sub-field of machine learning that is concerned with learning without labeled data. Since our proposed model - MEG is also a type of *generative model*, this section also explains the topic of generative modeling and practical use-cases of generative models such as MEG.

Machine learning is typically divided into supervised and unsupervised learning. In predictive or **supervised** learning, the objective is typically to learn a mapping from inputs x to labels y, given a labeled **training** dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. The input variables can be any complex structured object such as images, sentences, audio, etc. The corresponding labels can be image categories, positive or negative sentiment of text and speaker identity. If the output labels are categorical, the task is known as **classification**. If the output labels are continuous, the task is known as **regression**.

Unsupervised learning is interesting since it is closer to human and animal learning. We are expected to deduce patterns from the sensory input we receive from the physical world. It is also advantageous to not require human experts to label the data. Unsupervised learning also has the potential learn more complex models because there is more information in the input data than just a simple mapping from the input to a single label.

In **unsupervised** learning, the objective is to discover interesting patterns in the data using only the input dataset. A few common types of unsupervised learning are:

1. *Density estimation*, in which the task is to recover the the data distribution p_{data}. Having access to p_{data} is useful for a variety of purposes, such as making predictions. Another popular application is *anomaly detection*. For ex: a credit card company might suspect fraud if a purchase is very unlikely given a model of a customer's spending habits. Mixture of gaussians is a popular example for density estimation model.

2. *Manifold learning*, in which the learning algorithm tries to explain the data as lying on a low-dimensional manifold embedded in the original space. A few examples of such models are nonlinear principal components analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE) [Maaten and Hinton, 2008].
3. *Clustering*, in which the task is to discover a set of categories that the data can be divided into neatly. For example: clustering speech data into groups based on the number of speakers. Example of clustering algorithms include k-means clustering and mean-shift clustering.

1.3.1 Generative Modeling

As defined in [Lake et al., 2017], generative modeling is concerned with learning a model that specifies a probability distribution over the data. For instance, in a classification task with examples X and class labels y, a generative model specifies the distribution of data given labels $P(X|y)$, as well as a prior on labels $P(y)$, which can be used for sampling new examples or for classification by using Bayes' rule to compute $P(y|X)$. A discriminative model in contrast specifies $P(y|X)$ directly, possibly by using a neural network to predict the label for a given data point, and cannot directly be used to sample new examples or to compute other queries regarding the data.

The intuition is that, generative models try to capture how the data was generated in order to perform other downstream tasks such as classification, semi-supervised learning, denoising, matrix completion, structured prediction etc. One important advantage is that generative models do not *require* human annotated data and labels. A good generative model captures the salient features and underlying factors of variability from a large amount of unsupervised data. Additionally, a generative model provides a mechanism for producing samples from the distribution learned by the model. In contrast, discriminative models do not care about how the data was generated and instead directly categorize the signal.

Some popular examples of generative models are gaussian mixtures models [Titterington et al., 1985], hidden Markov models [Rabiner, 1989], variational auto-

encoders [Kingma and Welling, 2013], generative adversarial networks [Goodfellow et al., 2014a], etc. Popular examples of discriminative models are support vector machines [Cortes and Vapnik, 1995], k-nearest neighbours [Altman, 1992], conditional random fields [Lafferty et al., 2001], etc.

Recently, generative models have been utilized for purposes such as representation learning and semi-supervised learning [Radford et al., 2015, Odena et al., 2017, Salimans et al., 2016], domain adaptation [Ganin et al., 2016, Tzeng et al., 2017], text to image synthesis [Reed et al., 2016], speech recognition [Graves et al., 2013], speech synthesis [Oord et al., 2016], image compression [Theis et al., 2017], super resolution [Ledig et al., 2017], inpainting [Pathak et al., 2016, Yeh et al., 2017], image enhancement [Zhang et al., 2019] , style transfer and texture synthesis [Gatys et al., 2016, Johnson et al., 2016], image-to-image translation [Isola et al., 2017, Zhu et al., 2017], and video generation and prediction [Vondrick et al., 2016].

1.4 Graphical Model

MEG is an energy-based model, which is a type of undirected graphical model, derived from the probabilistic graphical modeling literature. In this section we provide a background on probabilistic modeling using graphs. Specifically, we motivate the use of graphs to express conditional independence structure between random variables, explain graph terminology and discuss two major types of graphical models - directed and undirected. We also show a particular form of undirected graphical models which serves as the foundation for **energy-based models**.

Probabilistic modeling attempts to answer the core questions of how to compactly represent the joint distributions of multiple correlated random variables such as words in a document, pixels in an image, genes in a micro-array, etc [Murphy, 2012]. Related set of questions that are relevant to probabilistic modeling are inferring a set of variables given another and inferring the parameters of a distribution given a reasonable amount of data.

By the **chain rule** of probability, we can always represent a joint distribution as follows, using any ordering of the variables:

$$p(\mathbf{x}_{1:V}) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_2,\mathbf{x}_1)...p(\mathbf{x}_V|\mathbf{x}_{1:V-1}) \tag{1.1}$$

They key to efficiently represent large joint distributions is to make assumptions about their conditional independences (CI), where two random variables X and Y are conditionally independent given Z (denoted $X \perp Y|Z$) if and only if (iff) $p(X,Y|Z) = p(X|Z)p(Y|Z)$. A **graphical model** (GM) is a way to represent a joint distribution by making CI assumptions. In particular, the nodes in the graph represent random variables, and the (lack of) edges represent CI assumptions.

Terminology

A graph $G = (\mathcal{V}, \mathcal{E})$ consists of a set of **nodes** or **vertices**, $\mathcal{V} = 1, ..., V$ and a set of **edges**, $\mathcal{E} = \{(s,t) : s,t \in \mathcal{V}\}$. A graph can be represented by an adjacency matrix where $G(s,t) = 1$ is used to denote that $s \to t$ is an edge in the graph. If $G(s,t) = 1$ iff $G(t,s) = 1$, we say that the graph is **undirected**, otherwise it is **directed**. It is also assumed that the graph has no self loops, i.e. $G(s,s) = 0$.

For a directed graph, the **parents** of a node is the set of all nodes that feed into it: $\text{pa}(s) \triangleq \{t : G(t,s) = 1\}$. Correspondingly, the **children** of a node is the set of all nodes that feed out of it: $\text{ch}(s) \triangleq \{s : G(t,s) = 1\}$.

1.4.1 Directed Graphical Models

In directed graphical models (DGMs), probability distributions over the random variables are represented using a *directed acyclic graph* (DAG). The directed edges are used to represent the conditional independences exhibited by the probability distribution. The key property of DAGs is that the nodes can be ordered such that parents come before children (topological ordering). Given such an order, the **ordered Markov property** is defined to be the assumption that a node only depends on its immediate parents, not on all predecessors in the ordering, i.e.,

$$\mathbf{x}_s \perp \mathbf{x}_{\text{pred}(s)\setminus\text{pa}(s)}|\mathbf{x}_{\text{pa}(s)} \tag{1.2}$$

where pa(s) are the parents of the node s and pred(s) are the predecessors of the node s in the ordering.

In general, the joint probability distribution represented by the graph can be written as:

$$p(\mathbf{x}_{1:V}) = \prod_{t=1}^{V} p(\mathbf{x}_t|\mathbf{x}_{\text{pa}(t)}) \tag{1.3}$$

where $p(\mathbf{x}_t|\mathbf{x}_{\text{pa}(t)})$ denotes a non-negative function of the variables normalized such that $\int p(\mathbf{x}_t|\mathbf{x}_{\text{pa}(t)})d\mathbf{x}_t = 1$.

In recent works, directed versions of graphical models have been used to *synthesize* sensory data through a sampling process which often converts a simple distribution over latent (or hidden) variables[1] that models *causes* in the sensory data, into complex distributions over the data distribution. The hidden variables often represent quantities of interest, such as the identity of the word that someone is currently speaking. The observed variables are what we measure, such as an acoustic waveform. These models can also be used to *analyze* sensory data by computing the posterior distribution over latent variables given data. An early instantiation of this idea was the Helmholtz machine [Dayan et al., 1995], in which the analysis was performed by a *recognition* model and the synthesis was performed by a separate generative model, and the two were trained together to maximize the marginal probability of the data. Popular example of directed graphical modeling include the hidden Markov models (HMMs) [Rabiner, 1989] and the more recent work on variational auto-encoders (VAE) [Kingma and Welling, 2013, Rezende et al., 2014].

1.4.2 Undirected Graphical Models

In undirected graphical models (UGMs), also called **Markov random fields**, the probability distribution over the random variables is represented using an undirected graph, which is more natural for certain problems such as image analysis and spatial statistics. From Murphy [2012], UGMs define CI relationships via simple graph separation as follows: for sets of nodes A, B, and C, we say $\mathbf{x}_A \perp_G \mathbf{x}_B \mid \mathbf{x}_C$ iff C separates A from B in the graph G. This means that, when we remove all

1. In statistics, latent variables (as opposed to observable variables), are variables that are not directly observed but are rather inferred (through a mathematical model) from other variables that are observed (directly measured)[Wikipedia, 2019a].

the nodes in C, if there are no paths connecting any node in A to any node in B, then the CI property holds. This is called the **global Markov property** for UGMs.

The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket** ; denote by $\text{mb}(t)$. Formally, the Markov blanket satisfies the following property:

$$t \perp \mathcal{V} \setminus \text{cl}(t) | \text{mb}(t) \tag{1.4}$$

where $\text{cl}(t) \triangleq \text{mb}(t) \cup \{t\}$ is the **closure** of node t. In a UGM, a node's Markov blanket is its set of immediate neighbours. This is called the **undirected local Markov property**. From the local Markov property, we can also easily see that two nodes are conditionally independent given the rest if there is no direct edge between them. This is called the **pairwise Markov property**.

Unlike DGMs which associate a conditional probability distribution (CPD) with each node in the graph (of the form $p(\mathbf{x}_s|\mathbf{x}_{\text{pa}(s)})$, UGMs associate **potential functions** or **factors** with each maximal clique in the graph. The potential function for clique c is given by $\psi_c(\mathbf{x}_c|\boldsymbol{\theta_c})$ where $\boldsymbol{\theta_c}$ denotes the parameters of the potential function for clique c. The potential function can be any non-negative function of its arguments. The joint distribution is then defined to be proportional to the product of clique potentials:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c|\boldsymbol{\theta_c}) \tag{1.5}$$

where $\mathcal{C}$ is the set of all maximal cliques in the graph G and $Z(\boldsymbol{\theta})$ is the **partition function** given by:

$$Z(\boldsymbol{\theta}) \triangleq \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c|\boldsymbol{\theta_c}) \tag{1.6}$$

Note that the partition function ensures that the overall distributions sums to 1. Hence it is also called the **normalization constant**.

Drawing inspiration from the **Gibbs distribution** in statistical physics, the

potential function or clique potential can also be represented as an energy function $E(\mathbf{x}_c) > 0$ which denotes the energy associated with the variables in clique c:

$$\psi_c(\mathbf{x}_c|\boldsymbol{\theta}_c) = \exp(-E(\mathbf{x}_c|\boldsymbol{\theta}_c)) \tag{1.7}$$

It can be seen that high probability states correspond to low energy configurations and low probability states correspond to high energy configurations. Models of this form are known as **energy based models**.

In recent work, these methods model the data as the *stationary* distribution of a stochastic process (e.g. various Boltzmann machines ; Salakhutdinov and Hinton [2009]). Sampling under this method corresponds to a potentially powerful iterative process of repeatedly applying a fixed stochastic operator that can gradually turn simple initial distributions over data into complex stationary distributions over data. However a key impediment to this approach is the *mixing time problem*: if the stationary distribution has multiple modes, the sampling process can take a long time to mix, or reach the stationary distribution, due to the excessive time sampling methods can take to jump between modes.

1.5 Markov Chain Monte Carlo Inference

In this section, we explain the topic of Monte Carlo approximations and also inference using Markov Chain Monte Carlo (MCMC). Monte Carlo approximations are omnipresent in deep learning literature since the optimization of neural networks is typically performed using mini-batch stochastic gradient descent (which uses a stochastic (Monte Carlo) estimate of the true batch gradient across the entire dataset). Additionally, energy-based models such as MEG also use Monte Carlo approximations of the log-likelihood gradient (explained in detail in the energy-based models section).

MCMC is the most popular method for sampling from high-dimensional distributions and was placed in the top 10 most important algorithms of the 20th century.

MCMC methods are relevant in the context of energy-based modeling since it is required to sample from the energy-function during the maximum likelihood training of EBMs (explained in detail in the following section). Specifically in our work on MEG, we use a popular MCMC method called Metropolis-adjusted Langevin algorithm (MALA) [Wikipedia, 2019b] to generate high quality samples from our energy-model.

In general, **Monte Carlo approximations** use the principle that computing the distribution of a function f of a random variable X can be expensive to compute using the change of variables formula. Instead, we can approximate the distribution of $f(X)$ using the empirical distribution of the samples $\{f(\mathbf{x}^s)\}_{s=1}^S$, where $\mathbf{x}^1, ..., \mathbf{x}^S \sim p(X)$. Thus, we can use Monte Carlo to approximate the expected value of any function of a random variable as follows:

$$\mathbb{E}[f(X)] \approx \frac{1}{S} \sum_{s=1}^{S} f(\mathbf{x}^s) \tag{1.8}$$

However, drawing samples $\mathbf{x}^1, ..., \mathbf{x}^S \sim p(X)$ might be non-trivial in practical use-cases when $p(X)$ is a very high-dimensional probability distribution. This motivates the necessity for algorithms that can draw samples from high-dimensional probability distributions. Markov Chain Monte Carlo is a popular class of algorithms that attempt to solve this problem.

From Murphy [2012], the basic idea behind **Markov Chain Monte Carlo** is to construct a Markov chain on the state space $\mathcal{X}$ whose stationary distribution is the target density $p^*(\mathbf{x})$ of interest (this may be a prior or a posterior). That is, we perform a random walk on the state space, in such a way that the fraction of time we spend in each state $\mathbf{x}$ is proportional to $p^*(\mathbf{x})$. By drawing (correlated!) samples $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, ...$, from the chain, we can perform Monte Carlo integration wrt p^*.

1.5.1 Gibbs Sampling

Gibbs sampling is one of the most popular MCMC algorithms and is also the most widely used algorithm for sampling from energy-based models.

The basic idea of Gibbs sampling is that each variable is sampled in turn,

conditioned on the values of all the other variables in the distribution. That is, given a joint sample $\mathbf{x}^s$ of all the variables, we generate a new sample $\mathbf{x}^{s+1}$ by sampling each component in turn, based on the most recent values of the other variables. An example of a Gibbs sampling step with 3 variables:

$$\begin{aligned}
\mathbf{x}_1^{s+1} &\sim p(\mathbf{x}_1|\mathbf{x}_2^s, \mathbf{x}_3^s) \\
\mathbf{x}_2^{s+1} &\sim p(\mathbf{x}_2|\mathbf{x}_1^{s+1}, \mathbf{x}_3^s) \\
\mathbf{x}_3^{s+1} &\sim p(\mathbf{x}_3|\mathbf{x}_1^{s+1}, \mathbf{x}_2^{s+1})
\end{aligned}$$

The expression $p(\mathbf{x}_i|\mathbf{x}_{-i})$ is called the full conditional of the variable i. If $p(\mathbf{x})$ is represented as a graphical model, the full conditional for variable i will reduce to the Markov blanket of i, which are its neighbours in the graph.

The shortcoming of Gibbs sampling is that it is typically slow and sequential since each Gibbs step requires D steps where D is the number of variables in the graph.

1.5.2 Metropolis Hastings algorithm

Although Gibbs sampling is simple, it is restrictive in terms of the class of models to which it can be applied, such as when the corresponding graphical model has no useful Markov structure. In addition, Gibbs sampling can be slow as mentioned above. **Metropolis Hastings** (MH) algorithm is a more general algorithm that can alternatively be used to sample from high-dimensional probability distributions. This topic is specifically relevant in context to our work, since we use a variant of the Metropolis Hastings algorithm to draw samples from our energy model.

The basic idea of MH algorithm as defined in Murphy [2012] is, in each step, first a proposal is made to move to a new state $\mathbf{x}'$ from state $\mathbf{x}$ with probability $q(\mathbf{x}'|\mathbf{x})$, where q is known as the **proposal** distribution. Next, the proposal to move to state $\mathbf{x}'$ is **accepted** or rejected depending on a formula that ensures that the fraction of time spent on each state $\mathbf{x}$ is proportional to $p^*(\mathbf{x})$ (necessary since we want the stationary distribution of the Markov chain to be $p^*(\mathbf{x})$). If the proposal is accepted, the new state is $\mathbf{x}'$, else the new state is the same as the current state

$\mathbf{x}$. If the proposal distribution is *symmetric*, so $q(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}|\mathbf{x}')$, the acceptance probability of MH is given by:

$$r = \min\left(1, \frac{p^*(\mathbf{x}')}{p^*(\mathbf{x})}\right) \tag{1.9}$$

It can be seen that if $\mathbf{x}'$ is more probable than $\mathbf{x}$, we definitely move there (since $\frac{p^*(\mathbf{x}')}{p^*(\mathbf{x})} > 1$), but if $\mathbf{x}'$ is less probable, we may still move there anyway, depending on the relative probabilities. So instead of greedily moving to only more probable states, we occasionally allow "downhill" moves to less probable states. We direct the reader to [Murphy, 2012] for proof that this procedure ensures that the fraction of time we spend in each state x is proportional to $p^*(\mathbf{x})$.

If the proposal distribution is *asymmetric*, i.e $q(\mathbf{x}'|\mathbf{x}) \neq q(\mathbf{x}|\mathbf{x}')$, the **Hastings correction** is used to compute the acceptance probability:

$$r = \min(1, \alpha) \tag{1.10}$$

$$\alpha = \frac{p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} \tag{1.11}$$

Intuitively, it can be seen that this correction is required to fix the bias introduced by the proposal distribution that might itself favor certain states.

The most *important* reason why MH is a useful algorithm is that, the calculation of the acceptance probability α only requires to know the target density $p^*(\mathbf{x})$ up to a normalization constant. For example, supposed $p^*(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x})$ where Z is the normalization constant, then:

$$\alpha = \frac{(\tilde{p}(\mathbf{x}')/Z)q(\mathbf{x}|\mathbf{x}')}{(\tilde{p}(\mathbf{x})/Z)q(\mathbf{x}'|\mathbf{x})} \tag{1.12}$$

It can be seen that the Z's cancel. Therefore we can sample from the target distribution p^* even if Z is unknown. This will be especially important to sample from unnormalized graphical models such as energy-based models.

1.6 Energy Based Models

Our work on MEG is primarily an **energy-based model**. This section provides a background into energy-based modeling. Having provided a short introduction to energy-based models in the previous section on undirected graphical models, we further elucidate the topic in this section followed by a short description of seminal works such as Boltzmann machines and restricted Boltzmann machines (RBMs). We also discuss important impediments in this class of methods to motivate research in this direction and also explain prior attempts at alleviating these shortcomings such as contrastive divergence. We also shortly describe promising alternatives to contrastive divergence such as persistent MCMC and score matching. MEG uses a variant of score matching as one of the objectives to train the energy-function.

Additionally, we revisit the topic of Markov Chain Monte Carlo (MCMC) methods for sampling from EBMs. MCMC sampling is crucial for energy-based modeling since it is required in the training process and also useful for visualizing what the model has learned. Obtaining good samples can be a task of its own as well, for example - the task of unconditional generative modeling of music, speech or images. In this task, the objective is to synthesize new images after learning a model on a dataset of images. Much like standard EBMs, MEG uses MCMC algorithms (specifically, the MALA algorithm) to visualize samples from the energy-function

Energy-based models (EBMs) capture dependencies by associating a scalar value (called **energy**) to each configuration of the variables of interest [LeCun et al., 2006, LeCun and Huang, 2005, Boureau et al., 2007]. *Learning* corresponds to carving the energy function so that its shape has desirable properties. For example: we would like plausible (observed) configurations to have low energy and unobserved configurations to have high energy. *Inference* corresponds to clamping the value of the observed variable and finding configurations of the remaining variables that minimize the energy. *Loss functional* - minimized during learning, is used to measure the quality of the available energy functions.

Probabilistic models must be properly normalized, which may require evaluating intractable integrals over the space of all possible variable configurations. Since

EBMs have no requirement for proper normalization, this problem is naturally circumvented. EBMs therefore provide considerably more flexibility in the design of architectures and training criteria than approaches requiring explicit probability computations.

Energy-based probabilistic models define a probability distribution through an energy function $E(\mathbf{x})$, as follows:

$$P(\mathbf{x}) = \frac{e^{-\mathrm{E}(\mathbf{x})}}{Z}, \tag{1.13}$$

ie., energies operate in the log-probability domain.

The normalization factor Z is called the *partition function* by analogy with physical systems,

$$Z = \sum_{\mathbf{x}} e^{-\mathrm{E}(\mathbf{x})} \tag{1.14}$$

An energy-based model can be learnt by performing (stochastic) gradient descent on the empirical negative log-likelihood of the training data

$$\mathcal{L}(\theta, \mathcal{D}) = -\frac{1}{N} \sum_{\mathbf{x}^{(i)} \in \mathcal{D}} \log p(\mathbf{x}^{(i)}) \tag{1.15}$$

where $-\frac{\partial \log p(\mathbf{x}^{(i)})}{\partial \theta}$ is the stochastic gradient and θ represents the parameters of the energy function.

1.6.1 EBMs with Hidden Units

Usually, we want to introduce some non-observed (latent) variables to increase the expressive power of the model. So we consider an observed part $\mathbf{x}$ and a hidden part $\mathbf{h}$:

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z} \text{ and } P(\mathbf{x}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z} \tag{1.16}$$

To map to a formulation similar to (1.13), the notation of *free energy* $\mathcal{F}(\mathbf{x})$ is introduced and defined as follows:

$$P(\mathbf{x}) = \frac{e^{-F(\mathbf{x})}}{Z} \text{ where } Z = \sum_{\mathbf{x}} e^{-\mathcal{F}(\mathbf{x})} \tag{1.17}$$

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{h}} e^{-E(\mathbf{x},\mathbf{h})} \tag{1.18}$$

The free energy is just a marginalization of energies in the log-domain. The data log-likelihood gradient then has a particularly interesting form. Starting from (1.17), we obtain:

$$-\frac{\partial \log P(\mathbf{x})}{\partial \theta} = \frac{\partial F(\mathbf{x})}{\partial \theta} + \frac{1}{Z}\frac{\partial Z}{\partial \theta} \tag{1.19}$$

$$= \frac{\partial F(\mathbf{x})}{\partial \theta} + \frac{1}{\sum_{\tilde{\mathbf{x}}} e^{-\mathcal{F}(\tilde{\mathbf{x}})}} \sum_{\tilde{\mathbf{x}}} \frac{\partial e^{-\mathcal{F}(\tilde{\mathbf{x}})}}{\partial \theta} \tag{1.20}$$

$$= \frac{\partial F(\mathbf{x})}{\partial \theta} - \frac{1}{Z} \sum_{\tilde{\mathbf{x}}} e^{-\mathcal{F}(\tilde{\mathbf{x}})} \frac{\partial \mathcal{F}(\tilde{\mathbf{x}})}{\partial \theta} \tag{1.21}$$

$$= \frac{\partial F(\mathbf{x})}{\partial \theta} - \sum_{\tilde{\mathbf{x}}} P(\tilde{\mathbf{x}}) \frac{\partial \mathcal{F}(\tilde{\mathbf{x}})}{\partial \theta} \tag{1.22}$$

The average log-likelihood gradient over the training set $\mathcal{D}$ is:

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[-\frac{\partial \log P(\mathbf{x})}{\partial \theta} \right] = \underbrace{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta} \right]}_{\text{positive phase}} - \underbrace{\mathbb{E}_{\mathbf{x}\sim P}\left[\frac{\partial \mathcal{F}(\mathbf{x})}{\partial \theta} \right]}_{\text{negative phase}} \tag{1.23}$$

The terms **positive** and **negative** do not refer to the sign of each term in the equation, but rather reflect their effect on the probability density defined by the model. The first term increases the probability of training data (by reducing the corresponding free energy), while the second term decreases the probability of samples generated by the model.

Therefore, if we could sample from P and compute the free energy tractably, we would have a **Monte Carlo** method to obtain a *stochastic estimator* of the log-likelihood gradient. Thus, **Markov Chain Monte Carlo (MCMC)** methods are very important for energy-based models since the log-likelihood gradient requires

sampling from P.

1.6.2 Boltzmann Machines

The Boltzmann machine is a particular type of energy-based model with hidden variables. The energy function is a general second-order polynomial:

$$\text{Energy}(\mathbf{x}, \mathbf{h}) = -\mathbf{b}'\mathbf{x} - \mathbf{c}'\mathbf{h} - \mathbf{h}'W\mathbf{x} - \mathbf{x}'U\mathbf{x} - \mathbf{h}'V\mathbf{h}. \tag{1.24}$$

The parameters b_i and c_i are offsets, and W_{ij}, U_{ij} and V_{ij} are weight matrices. The parameters are collectively denoted θ.

The gradient of the log-likelihood can be written as:

$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}) \frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta} + \sum_{\tilde{\mathbf{x}}, \mathbf{h}} P(\tilde{\mathbf{x}}, \mathbf{h}) \frac{\partial E(\tilde{\mathbf{x}}, \mathbf{h})}{\partial \theta} \tag{1.25}$$

Similar to (1.23), in the *positive phase* $\mathbf{x}$ is clamped to the observed input vector and we sample $\mathbf{h}$ given $\mathbf{x}$; and in the *negative phase* both $\mathbf{x}$ and $\mathbf{h}$ are sampled from the model itself. In general, only approximate sampling can be achieved tractably, by using an iterative procedure that constructs an MCMC. *Gibbs sampling*, as explained in the previous sections, is a popular MCMC procedure used with RBMs [Hinton et al., 1986, Ackley et al., 1985].

Drawback of general Boltzmann Machines: Since an MCMC chain is required both for the positive phase and the negative phase for each example x, the computation of the gradient can be very expensive, and training time very long.

1.6.3 Restricted Boltzmann Machines

RBMs are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables. RBMs may be stacked (one on top of the other) to form deeper models.

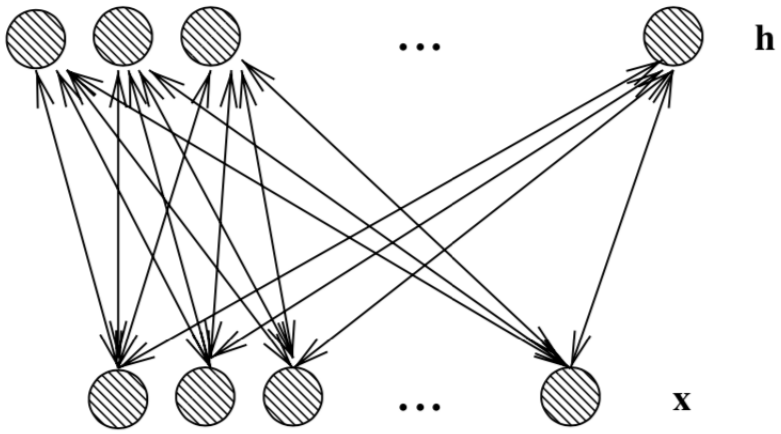


Figure 1.1 – Undirected graphical model of a Restricted Boltzmann Machine (RBM). There are no links between units of the same layer, only between input (or visible) units $\mathbf{x}_j$ and hidden units $\mathbf{h}_i$, making the conditionals $P(\mathbf{h}|\mathbf{x})$ and $P(\mathbf{x}|\mathbf{h})$ factorize conveniently.

From Figure 1.1 it can be seen that $\mathbf{h}_i$ are independent of each other when conditioning on $\mathbf{x}$ and the $\mathbf{x}_j$ are independent of each other when conditioning on $\mathbf{h}$ (It is a bipartite graph, with no connections permitted between any variables in the observed layer or between any units in the latent layer). Since the graph is bipartite in an RBM, $U = 0$ and $V = 0$ from (1.24). i.e., the only interaction terms are between a hidden unit and a visible unit, but not between units of the same layer. As a consequence, the energy function is bilinear:

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{b}'\mathbf{x} - \mathbf{c}'\mathbf{h} - \mathbf{h}'W\mathbf{x}. \tag{1.26}$$

In RBMs, factorization can be utilized to tractably compute the Free Energy and the conditional probabilities $P(\mathbf{h}|\mathbf{x})$ and $P(\mathbf{x}|\mathbf{h})$ required in the log-likelihood gradient (1.23). Thus:

$$F(\mathbf{x}) = -\mathbf{b}'\mathbf{x} - \sum_i \log \sum_{\mathbf{h}_i} e^{\mathbf{h}_i(\mathbf{c}_i + W_i\mathbf{x})} \tag{1.27}$$

$$P(\mathbf{h}|\mathbf{x}) = \prod_i P(\mathbf{h}_i|\mathbf{x}) \tag{1.28}$$

$$P(\mathbf{x}|\mathbf{h}) = \prod_i P(\mathbf{x}_i|\mathbf{h}) \tag{1.29}$$

The visible units $\mathbf{x}$ and hidden units $\mathbf{h}$ are typically modeled as bernoulli or gaussian units. For detailed derivation of the above equations, refer [Bengio, 2009].

1.6.4 Sampling in RBMs

Sampling from RBMs is useful for several reasons. First, it is useful in learning algorithms to get a stochastic estimator of the log-likelihood gradient. Second, it is also useful in sampling from the RBMs used as a generative model, or for visual inspection and to get an idea of what the model has captured about the data distribution.

Since RBMs enjoy the factorization introduced by the conditional independence structure, it brings two major benefits: First, we do not have to sample in the positive phase since free energy can be computed in closed form. Second, the set of variables in $(\mathbf{x}, \mathbf{h})$ can be sampled in only two sub-steps in each step of the Gibbs chain (as opposed to N sub-steps in Boltzmann machines). First we sample all the $\mathbf{h}_i$ given $\mathbf{x}$ in parallel, and then all the new $\mathbf{x}_j$ in parallel given $\mathbf{h}$. This type of Gibbs sampling in general is called *Blocked Gibbs Sampling*.

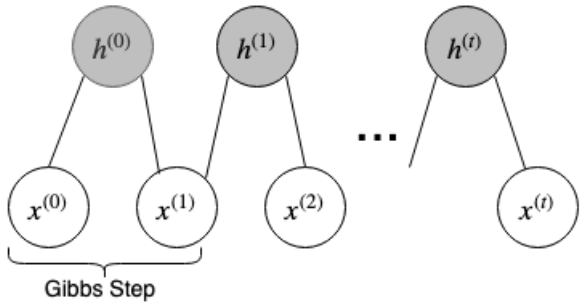


Figure 1.2 – Illustration of Blocked Gibbs Sampling in RBMs. As $t \to \infty$, sample $(x^{(t)}, h^{(t)})$ are guaranteed to be samples of $P(\mathbf{x}, \mathbf{h})$

Figure 1.2 shows an illustration of t steps of the blocked Gibbs chain for sampling from RBMs. Typically, the chain is seeded using an example from the training set. This makes sense because, as the model captures the training data better, the model distribution and training distribution become more similar. In theory, each parameter update in the learning process would require running one such chain to convergence. This would be computationally expensive. Several algorithms have been devised for RBMs in order to efficiently sample from $P(\mathbf{x}, \mathbf{h})$ during the training process.

1.6.5 Contrastive Divergence

Contrastive Divergence is an approximation of the log-likelihood gradient that has been found to be a successful update rule for training RBMs.

The **first approximation** replaces the average over all possible inputs (second term in (1.23)) by a single example. This is justified since we typically update parameters using stochastic or mini-batch gradient updates. The extra variance introduced from one or few MCMC samples instead of the complete summation might be partially cancelled during the online gradient updates, over consecutive parameter updates. The additional variance introduced by this approximation might not hurt much if it is comparable or smaller than the variance due to online gradient descent.

The **second approximation** combats the issue of running a long MCMC chain which is expensive. The idea of k-step Contrastive Divergence (CD-k) [Hinton, 1999, 2002] is to run the MCMC chain for only k steps $(\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{k+1})$ starting from the training example $\mathbf{x}^1 = \mathbf{x}$. The bias introduced by this approximation vanishes when $k \to \infty$. However a surprising empirical result was that $k = 1$ (CD-1) works well [Carreira-Perpinan and Hinton, 2005].

An intuitive interpretation of the Contrastive Divergence algorithm is that it approximates the log-likelihood gradient *locally* around the training example $\mathbf{x}^1$. The stochastic reconstruction $\tilde{\mathbf{x}} = \mathbf{x}^{k+1}$ (for CD-k) has a distribution centred around the training point $\mathbf{x}^1$ and spreads around as k increases. The CD-k update decreases the free energy of the training point $\mathbf{x}^1$ and increases the free energy of $\tilde{\mathbf{x}}$ in the neighbourhood of $\mathbf{x}^1$, thus "shoveling" energy elsewhere. Thus, the Contrastive Divergence algorithm is fueled by the *contrast* between the statistics collected when the input is a real training example and when the input is a chain sample since what is required by a training algorithm for an energy-based model is that it makes the energy of observed inputs smaller, shoveling energy elsewhere, and most importantly in areas of low energy (locally around the training example, here).

1.6.6 Alternative to Contrastive Divergence

Persistent MCMC [Salakhutdinov and Hinton, 2009, Tieleman, 2008b] This idea is to use a background (persistent) MCMC chain to obtain the negative phase samples, instead of running a new short chain as in CD-k. The approximation made is that we ignore the fact that parameters are changing as we move along the chain. However this approximation works very well in practice usually giving rise to better log-likelihood than CD-k probably because the parameters vary slowly during training.

Score Matching [Hyvärinen, 2005, Vincent, 2011] This is a general approach to energy-based model training in which energy can be computed tractably but not the normalization constant Z. The score function of a density $P(\mathbf{x})$ is $\psi = \frac{\partial \log P(\mathbf{x})}{\partial \mathbf{x}}$. The basic idea is to match the score function of the model with the score function of the empirical density. This idea exploits the fact that the score function does not depend on the normalization constant.

1.7 Recent Deep Learning Methods

The research work presented in this thesis lies at the intersection of deep learning and energy-based graphical modeling. MEG is an energy-based model that uses deep neural network based function approximators to model the energy-function. Having given a background on graphical models in the previous sections, this section strives to inform the reader about the various concepts popular in the recent deep learning literature. Specifically, we explain the recent methods to estimate mutual information using neural networks. This concept was instrumental in performing entropy maximization of the energy-function, that arises from the theoretical framework provided by MEG. We also provide a concise description of generative adversarial networks (GANs) [Goodfellow et al., 2014a] since the training of the energy-function in our method draws parallels with the adversarial training of GANs. Further, we also explain some of the evaluation metrics used in our paper for measuring the quality of generated image samples, such as Inception Score (IS) and Fréchet

Inception Distance (FID).

1.7.1 Neural Estimators of Mutual Information

Definitions

Entropy is a quantity that measures the unpredictability of a random variable. Entropy of a discrete random variable X with probability mass function (PMF) $p(x)$ is:

$$H(X) = -\sum_x p(x) \log p(x) = -\mathbb{E}[\log p(x)] \tag{1.30}$$

The entropy measures the expected uncertainty in X.

The **differential entropy** of a continuous random variable X with support $\mathcal{X}$ and probability density function (PDF) $f(x)$ is:

$$h(X) = -\int f(x) \log f(x) dx = -\mathbb{E}[\log(f(x))] \tag{1.31}$$

Mutual information (MI) is a quantity that measures a relationship between two random variables. In particular, it quantifies the "amount of information" (in units such as Shannons, commonly called bits) obtained about one random variable through observing the other random variable. Mutual information captures non-linear statistical dependencies between variables, and thus can act as a measure of true dependence [Kinney and Atwal, 2014]. Mutual Information quantifies the dependence between random variables X and Z as :

$$I(X;Z) = \int_{X\times Z} \log \frac{d\mathbb{P}_{XZ}}{d\mathbb{P}_X \otimes d\mathbb{P}_Z} d\mathbb{P}_{XZ}. \tag{1.32}$$

where $\mathbb{P}_{XZ}$ is the joint probability distribution, and $\mathbb{P}_X = \int_Z d\mathbb{P}_{XZ}$ and $\mathbb{P}_Z = \int_X d\mathbb{P}_{XZ}$ are the marginal distributions and $\otimes$ denotes the Cartesian product.

Mutual information can be *equivalently* expressed as the Kullback-Leibler divergence (KL divergence) between the joint and the product of the marginal probability

distributions:

$$I(X;Z) = D_{KL}(\mathbb{P}_{XZ} \mid\mid \mathbb{P}_X \otimes \mathbb{P}_Z) \tag{1.33}$$

KL Divergence (KLD) between two discrete probability distributions P and Q defined on the same probability space can be defined as:

$$D_{\mathrm{KL}}(P||Q) = -\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x}) \log\left(\frac{Q(\mathbf{x})}{P(\mathbf{x})}\right) \tag{1.34}$$

Jensen-Shannon Divergence (JSD) is a smoothed and symmetrized version of the KL divergence $D_{\mathrm{KL}}(P||Q)$ defined as:

$$\mathrm{JSD}(P||Q) = \frac{1}{2}D_{\mathrm{KL}}(P||M) + \frac{1}{2}D_{\mathrm{KL}}(Q||M), \text{where } M = \frac{1}{2}(P+Q) \tag{1.35}$$

Methods

Mutual Information Neural Estimator (MINE) [Belghazi et al., 2018] uses the the Donsker-Varadhan dual representation of the KL-divergence [Donsker and Varadhan, 1975] to exploit the bound:

$$I(X;Z) \geq \sup_{\theta\in\Theta} \mathbb{E}_{\mathbb{P}_{XZ}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}_X\otimes\mathbb{P}_Z}[e^{T_\theta}]). \tag{1.36}$$

where $T_\theta : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ is the family of functions parametrized by a neural network with parameters $\theta \in \Theta$.

DeepInfoMax (DIM) [Hjelm et al., 2018] uses the Jensen-Shannon MI estimator following the recent formulation of f-divergences by [Nowozin et al., 2016]. DIM showed more stable results for MI maximization using the Jensen-Shannon MI estimator, due to its bounded nature. MINE on the other hand leads to an unbounded estimate, rendering it unsuitable for MI maximization without tricks to adaptively clip the gradients during training.

1.7.2 Generative Adversarial Network

Generative Adversarial Network (GAN) [Goodfellow et al., 2014a] is a framework in which two networks - Discriminator (D) and Generator (G) are pitted against each other. The Discriminator attempts to determine whether a sample is from the model distribution or the data distribution. The Generator attempts to generate samples that are indistinguishable from the original data by the Discriminator.

Formally, let $p_{\text{data}}(x)$ denote the data distribution, $p_z(z)$ denote the prior distribution on the noise variables z and p_g denote the Generator's distribution. The Discriminator and Generator networks in GANs are trained to optimize the following objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \quad (1.37)$$

It has been shown in Goodfellow et al. [2014a] that GANs optimize the Jensen-Shannon divergence (JSD) between the distributions $p_{\text{data}}(x)$ and p_g.

Wasserstein GAN

A serious problem with GAN training as noted by [Arjovsky et al., 2017a] and in the original formula [Goodfellow et al., 2014a] is that on complex problems, it is difficult to select a generator that has overlapping support with the data distribution without adding noise. When the generator and the data distribution do not have overlapping support, KL divergence is undefined and the Jensen-Shannon divergence is discontinuous at these points. **Wasserstein GAN** (WGAN) solves this problem by providing a statistical divergence that is continuous and differentiable even when the supports do not overlap.

[Arjovsky et al., 2017a] thus provides a formulation of the GAN objective which corresponds to optimizing the Earth Mover's distance or the Wasserstein metric. The Wasserstein distance (or EM distance) can be intuitively thought of as the minimum amount of effort required to move mass distributed according to one distribution to match another distribution. WGAN uses the Kantorovich-Rubinstein

dual formulation for the EM distance:

$$W(\mathcal{P}_r, \mathcal{P}_g) = \sup_{||f||_L \leq 1} \Big(\mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)] \Big), \tag{1.38}$$

where the condition under the supremum indicates that it is over all 1-Lipschitz functions $D : \mathcal{X} \to \mathcal{R}$. In practice, the Lipschitz constraint was maintained by clipping the weights within a specific range after each update. [Gulrajani et al., 2017] instead proposed to use a penalty on the norm of the gradient of the discriminator's output with respect to its inputs. This achieved significantly better results in terms of the quality of samples and training stability over the weight clipping approach.

1.7.3 Evaluation Metrics

In our work, sample quality of generated images is a useful metric to evaluate the generative model. If the model has successfully modeled the data distribution (of images) really well, it should be possible to sample new images that are perceptually consistent and of high quality. Although we can qualitatively evaluate it by visual examination, quantitative metrics are useful to objectively compare competing models. We use two popular methods for this purpose, Inception Score [Salimans et al., 2016] and Fréchet Inception Distance [Heusel et al., 2017].

Inception Score

From Xu et al. [2018], Inception Score proposed by [Salimans et al., 2016] uses an image classification model $\mathcal{M}$, the Google Inception network [Szegedy et al., 2016], pre-trained on the ImageNet [Deng et al., 2009] dataset, to compute:

$$\text{IS}(P_g) = \exp(\mathbb{E}_{\mathbf{x} \sim P_g}[\text{KL}(p_{\mathcal{M}}(y|\mathbf{x}) \;||\; p_{\mathcal{M}}(y)]) \tag{1.39}$$

where P_g denotes the generator network's distribution, $p_{\mathcal{M}}(y|\mathbf{x})$ denotes the label distribution of $\mathbf{x}$ as prediction by $\mathcal{M}$ and $p_{\mathcal{M}}(y) = \int_{\mathbf{x}} p_{\mathcal{M}}(y|\mathbf{x}) dP_g$, i.e. the marginal of $p_{\mathcal{M}}(y|\mathbf{x})$ under the probability measure P_g. The expectation and the integral in $p_{\mathcal{M}}(y|\mathbf{x})$ can be approximated with i.i.d samples from P_g.

It can be seen that IS is high when $p_{\mathcal{M}}(y|\mathbf{x})$ is close to a point mass, which happens when the Inception network is very confident that the image belongs to a

particular ImageNet category, and $p_{\mathcal{M}}(y)$ is close to uniform, i.e. all categories are equally represented. This suggests that the generative model has both high quality and diversity. [Salimans et al., 2016] show that the Inception Score has a reasonable correlation with human judgment of image quality.

Fréchet Inception Distance

From Borji [2019], FID embeds a set of generated samples into a feature space given by a specific layer of Inception Net (or any CNN). Viewing the embedding layer as a continuous multivariate Gaussian, the mean and covariance are estimated for both the generated data and the real data. The Fréchet distance between these two Gaussians (a.k.a Wasserstein-2 distance) is then used to quantify the quality of generated samples, i.e

$$\text{FID}(P_r, P_g) = ||\mu_r - \mu_g||_2^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{1.40}$$

where μ_r, μ_g and Σ_r, Σ_g represent the mean and covariances of the real and generated distributions respectively.

2 Maximum Entropy Generators for Energy-based Models

Authors: Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville and Yoshua Bengio.

This chapter presents a joint work with Sherjil Ozair, Anirudh Goyal, Aaron Courville and Yoshua Bengio. It has been submitted to Neural Information Processing Systems (NeurIPS 2019) - Conference Track.

Contribution: The idea was initially conceptualized by my advisor - Prof. Yoshua Bengio. The idea was further developed and improved by myself through helpful discussions with Anirudh Goyal and Prof. Aaron Courville. I wrote all the code and performed the experiments listed in the paper. Prof. Yoshua Bengio helped write the initial draft of the introduction and background section and Sherjil Ozair helped refine the drafts. I wrote the sections on Maximum Entropy Generators for Energy-based Models (2.3), related work (2.4) and experiments (2.5). I am the first author of the paper.

Affiliation

— Rithesh Kumar, Mila, University of Montreal
— Sherjil Ozair, Mila, University of Montreal
— Anirudh Goyal, Mila, University of Montreal
— Aaron Courville, Mila, University of Montreal
— Yoshua Bengio, Mila, University of Montreal

2.1 Introduction

Unsupervised learning promises to take advantage of unlabelled data, and is regarded as crucial for artificial intelligence [Lake et al., 2017]. Energy-based modeling (EBMs, LeCun et al. [2006]) is a family of unsupervised learning methods focused on learning an energy function, i.e., an unnormalized log density of the data. This removes the need to make parametric assumptions about the data distribution to make the normalizing constant (Z) tractable. However, in practice, due to the very same lack of restrictions, learning high-quality energy-based models is fraught with challenges. To avoid explicitly computing Z or its gradient, Contrastive Divergence [Hinton, 2000] and Stochastic Maximum Likelihood [Younes, 1998, Tieleman, 2008a] rely on Markov Chain Monte Carlo (MCMC) to approximately sample from the energy-based model. However, MCMC-based sampling approaches frequently suffer from long mixing times for high-dimensional data. Thus, training of energy-based models has not remained competitive with other unsupervised learning techniques such as variational auto-encoders [Kingma and Welling, 2014] and generative adversarial networks or GANs [Goodfellow et al., 2014b].

In this work, we propose Maximum Entropy Generators (MEG), a framework in which we train both an energy function and an approximate sampler, which can either be fast (using a generator network G) or uses G to initialize a Markov chain in the latent space of the generator. Training such a generator properly requires entropy maximization of the generator's output distribution, for which we take advantage of recent advances in nonparametric mutual information maximization [Belghazi et al., 2018, Hjelm et al., 2018, Oord et al., 2018, Poole et al., 2018].

To evaluate the efficacy of the proposed technique, we compare against other state-of-the-art techniques on image generation, accurate mode representation, and anomaly detection. We demonstrate that the proposed technique is able to generate CIFAR-10 samples which are competitive with WGAN-GP [Gulrajani et al., 2017] according to the Fréchet Inception Distance [Heusel et al., 2017] and Inception Score [Salimans et al., 2016], and is able to generate samples of all the 10^4 modes of 4-StackedMNIST at the correct data frequencies.

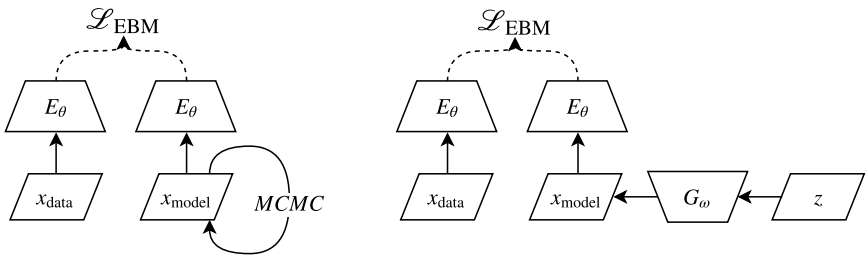


Figure 2.1 – **Left**: Traditional maximum likelihood training of energy-based models. **Right:** Training of maximum entropy generators for energy-based models

We demonstrate that our technique trains energy functions useful for anomaly detection on the KDD99 dataset [Lichman et al., 2013], and that it performs as well as state-of-the-art anomaly detection techniques which were specially designed for the task. Further it vastly outperforms other energy-based and generative models for anomaly detection.

To summarize our contributions, we propose maximum entropy generators (MEG), a novel framework for training energy-based models using amortized neural generators and mutual information maximization. We show that the resulting energy function can be successfully used for anomaly detection, and outperforms recently published results with energy-based models. We show that MEG generates sharp images – with competitive Inception and FID scores – and accurately captures more modes than standard GANs, while not suffering from the common mode-mixing issue of many maximum likelihood generative models which results in blurry samples.

2.2 Background

Let $\boldsymbol{x}$ denote a sample in the data space $\mathcal{X}$ and $E_\theta : \mathcal{X} \to \mathbb{R}$ an energy function corresponding to the negative logarithm of an unnormalized estimated density density function

$$p_\theta(\boldsymbol{x}) = \frac{e^{-E_\theta(\boldsymbol{x})}}{Z_\theta} \propto e^{-E_\theta(\boldsymbol{x})} \tag{2.1}$$

where $Z_\theta := \int e^{-E_{\boldsymbol{\theta}}(\boldsymbol{x})} d\boldsymbol{x}$ is the normalizing constant or partition function. Let p_D be the training distribution, from which the training set is drawn. Towards optimizing the parameters θ of the energy function, the maximum likelihood parameter gradient is

$$\frac{\partial \mathbb{E}_{\boldsymbol{x}\sim p_D}[-\log p_\theta(\boldsymbol{x})]}{\partial \theta} = \mathbb{E}_{\boldsymbol{x}\sim p_D}\left[\frac{\partial E_\theta(\boldsymbol{x})}{\partial \theta}\right] - \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x})}\left[\frac{\partial E_\theta(\boldsymbol{x})}{\partial \theta}\right] \tag{2.2}$$

where the second term is the gradient of $\log Z_\theta$, and the sum of the two expectations is zero when training has converged, with expected energy gradients in the positive phase (under the data p_D) matching those under the negative phase (under $p_\theta(\boldsymbol{x})$). Training thus consists in trying to separate two distributions: the positive phase distribution (associated with the data) and the negative phase distribution (where the model is free-running and generating configurations by itself). This observation has motivated the pre-GAN idea presented by Bengio [2009] that "model samples are negative examples" and a classifier could be used to learn an energy function if it separated the data distribution from the model's own samples. Shortly after introducing GANs, Goodfellow [2014] also made a similar connection, related to noise-contrastive estimation [Gutmann and Hyvarinen, 2010]. One should also recognize the similarity between Eq. 2.2 and the objective function for Wasserstein GANs or WGAN [Arjovsky et al., 2017b].

The main challenge in Eq. 2.2 is to obtain samples from the distribution p_θ associated with the energy function E_θ. Although having an energy function is convenient to obtain a score allowing comparison of the relative probability for different $\boldsymbol{x}$'s, it is difficult to convert an energy function into a generative process. The commonly studied approaches for this are based on Markov Chain Monte Carlo, in which one iteratively updates a candidate configuration, until these configurations converge in distribution to the desired distribution p_θ. For the RBM, the most commonly used algorithms have been Contrastive Divergence [Hinton, 2000] and Stochastic Maximum Likelihood [Younes, 1998, Tieleman, 2008a], relying on the particular structure of the RBM to perform Gibbs sampling. Although these MCMC-based methods are appealing, RBMs (and their deeper form, the deep Boltzmann machine) have not been competitive in recent years compared to autoregressive models [van den Oord et al., 2016], variational auto-encoders [Kingma and Welling,

2014] and generative adversarial networks or GANs [Goodfellow et al., 2014b].

What has been hypothesized as a reason for poorer results obtained with energy-based models trained with an MCMC estimator for the negative phase gradient is that running a Markov chain in data space is fundamentally difficult when the distribution is concentrated (e.g, near manifolds) and has many modes separated by vast areas of low probability. This mixing challenge is discussed by Bengio et al. [2013] who argue that a Markov chain is very likely to produce only sequences of highly probable configurations: if two modes are far from each other and only local moves are possible (which is typically the case when performing MCMC), it becomes exponentially unlikely to traverse the "desert" of low probability that can separate two modes. This makes mixing between modes difficult in high-dimensional spaces with strong concentration of probability mass in some regions (e.g. corresponding to different categories) and very low probability elsewhere.

2.3 Maximum Entropy Generators for Energy-Based Models

We thus propose using an amortized neural sampler to perform fast approximate sampling to train the energy model. We begin by replacing the model distribution p_θ in in Eq. 2.2 by a neural generator G parametrized by $\boldsymbol{w}$. We define P_G as the distribution of the outputs $G(z)$ for $\boldsymbol{z} \sim p_z$ where p_z is a simple prior distribution such as a standard Normal distribution.

$$\frac{\partial \mathcal{L}_E}{\partial \theta} = \mathbb{E}_{\boldsymbol{x} \sim p_D}\left[\frac{\partial E_\theta(\boldsymbol{x})}{\partial \theta}\right] - \mathbb{E}_{\boldsymbol{x} \sim p_G(\boldsymbol{x})}\left[\frac{\partial E_\theta(\boldsymbol{x})}{\partial \theta}\right] \tag{2.3}$$

To minimize the approximation error, p_G must be close to p_θ. To do so, we tune G to minimize the KL divergence $KL(p_G||p_\theta)$, which can be rewritten in terms of minimizing the energy of the samples from the generator while maximizing the

entropy at the output of the generator:

$$KL(p_G||p_\theta) = -H[p_G] - E_{p_G}[\log p_\theta(\boldsymbol{x})] \tag{2.4}$$
$$= -H[p_G] + E_{p_G}[E_\theta(\boldsymbol{x})] + \log Z_\theta \tag{2.5}$$

When taking the gradient of $KL(p_G||p_\theta)$ with respect to the parameters $\boldsymbol{w}$ of the generator, the log-partition function $\log Z_\theta$ disappears and we can optimize $\boldsymbol{w}$ by minimizing

$$\mathcal{L}_G = -H[p_G] + \mathbb{E}_{\boldsymbol{z}\sim p_z} E_\theta(G(\boldsymbol{z})) \tag{2.6}$$

where p_z is the prior distribution of the latent variable of the generator.

In order to approximately maximize the entropy $H[p_G]$ at the output of the generator, we use one recently proposed nonparametric mutual information maximization techniques [Belghazi et al., 2018, Oord et al., 2018, Hjelm et al., 2018]. Poole et al. [2018] show that these techniques can be unified into a single framework derived from the variational bound of Barber and Agakov [2003]. Since the generator is deterministic, mutual information between inputs and outputs reduces to simply entropy of the outputs, since the conditional entropy of a deterministic function is zero:

$$I(X,Z) = H(X) - H(X|Z) = H(G(Z)) - \cancelto{0}{H(G(Z)|Z)}$$

In particular, we use the estimator from Hjelm et al. [2018], which estimates the Jensen-Shannon divergence between the joint distribution ($p(\boldsymbol{x},\boldsymbol{z})$) and the product of marginals ($p(\boldsymbol{x})p(\boldsymbol{z})$). We refer to this information measure as $I_{JSD}(X,Z)$. We found that the JSD-based estimator works better in practice than the KL-based estimator (which corresponds to the mutual information).

The estimator of Hjelm et al. [2018] is given by

$$\mathcal{I}_{JSD}(X,Z) = \sup_{T\in\mathcal{T}} \mathbb{E}_{p(X,Z)}[-\mathrm{sp}(-T(X,Z))] - \mathbb{E}_{p(X)p(Z)}[\mathrm{sp}(T(X,Z))] \tag{2.7}$$

where $\mathrm{sp}(a) = \log(1+e^a)$ is the softplus function. The supremum is approximated

using gradient descent on the parameters of the discriminator T.

With $X = G(Z)$ the output of the generator, $\mathcal{I}_{JSD}(G(Z), Z)$ is one of the terms to be maximized in the objective function for training G, which would maximize the generator's output entropy $H(G(Z))$.

Thus the final training objective to be minimized for the generator G and the energy function E is

$$\mathcal{L}_G = -\mathcal{I}_{JSD}(G(Z), Z) + \mathbb{E}_{\boldsymbol{z}\sim p_z} E_\theta(G(\boldsymbol{z})) \tag{2.8}$$

$$\mathcal{L}_E = \mathbb{E}_{\boldsymbol{x}\sim p_D} E_\theta(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_z} E_\theta(G(\boldsymbol{z})) \tag{2.9}$$

where $Z \sim p_z$, the latent prior (typically a $N(0, I)$ Gaussian).

2.3.1 Improving training stability

As can be seen from the above equations, the generator and the energy function are in an adversarial game, similar to generative adversarial networks [Goodfellow et al., 2014b]. This makes optimization via simultaneous gradient descent challenging since the gradient vector field of such an optimization problem is non-conservative as noted by Mescheder et al. [2017]. This is particularly accentuated by the use of deep neural networks for the generator and the energy function. In particular, we noticed that during training the magnitude of the energy function values would diverge.

To help alleviate this issue we look towards another technique for learning energy-based models called *score matching* proposed by Hyvärinen [2005]. Score matching estimates the energy function by matching the score functions of the data density and the model density, where the score function ψ is the gradient of the log density with respect to the sample $\psi(\boldsymbol{x}) = \frac{\partial \log p(\boldsymbol{x})}{\partial \boldsymbol{x}}$. If $\psi_D(\boldsymbol{x})$ and $\psi_E(\boldsymbol{x})$ are the score functions under the data distribution and model distribution respectively, the score matching objective is given by

$$\mathcal{J}_{SM} = \mathbb{E}_{\boldsymbol{x}\sim P_\mathrm{D}} \left[\|\psi_D(\boldsymbol{x}) - \psi_E(\boldsymbol{x})\|_2^2 \right].$$

While the score function for the data distribution is typically unknown and would

require estimation, Theorem 1 in Hyvärinen [2005] shows that with partial integrations, the score matching objective can be reduced to the following objective which does not depend on the score function under the data distribution:

$$\begin{aligned}\mathcal{J}_{SM} &= \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{D}}}\left[\sum_i \partial_i \psi_i(\boldsymbol{x}) + \frac{1}{2}\psi_i(\boldsymbol{x})^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{D}}}\left[\sum_i -\frac{\partial^2 E(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i} + \frac{1}{2}\left(\frac{-\partial E(\boldsymbol{x})}{\partial \boldsymbol{x}_i}\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{D}}}\left[\frac{1}{2}\left\|\frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{x}}\right\|_2^2 + \sum_i -\frac{\partial^2 E(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i}\right]\end{aligned} \tag{2.10}$$

The above objective is hard to optimize when using deep neural networks because of the difficulty in estimating the gradient of the Hessian diagonal, so we use the first term in our objective, i.e. the zero-centered gradient penalty, pushing the data points to sit near critical points (generally a local minimum) of the energy function.

This term is also similar to the gradient penalty regularization proposed by Gulrajani et al. [2017] which however is one-centered and applied on interpolations of the data and model samples, and is derived from the Lipschitz continuity requirements of Wasserstein GANs [Arjovsky et al., 2017b].

2.3.2 Improving sample quality via latent space MCMC

Since MEG simultaneously trains a generator and a valid energy function, we can improve the quality of samples by biasing sampling towards high density regions. Furthermore, doing the MCMC walk in the latent space should be easier than in data space because the transformed data manifold (in latent space) is flatter than in the original observed data space, as initially discussed by Bengio et al. [2013]. The motivation is also similar to that of the "truncation trick" used successfully by Brock et al. [2018]. However, we use an MCMC-based approach for this which is applicable to arbitrary latent distributions.

We use the Metropolis-adjusted Langevin algorithm (MALA, Girolami and Calderhead [2011]), with Langevin dynamics producing a proposal distribution in the latent space as follows:

$$\tilde{\mathbf{z}}_{t+1} = \boldsymbol{z}_t - \alpha \frac{\partial E_\theta(G_\omega(\boldsymbol{z}_t))}{\partial \boldsymbol{z}_t} + \epsilon\sqrt{2 * \alpha}, \text{ where } \epsilon \sim \mathcal{N}(0, I_d)$$

Next, the proposed $\tilde{z}_{t+1}$ is accepted or rejected using the Metropolis Hastings algorithm, by computing the acceptance ratio:

$$r = \frac{p(\tilde{\mathbf{z}}_{t+1})q(\mathbf{z}_t|\tilde{\mathbf{z}}_{t+1})}{p(\mathbf{z}_t)q(\tilde{\mathbf{z}}_{t+1}|\mathbf{z}_t)} \tag{2.11}$$

$$\frac{p(\tilde{\mathbf{z}}_{t+1})}{p(\mathbf{z}_t)} = \exp\big\{ - E_\theta(G_\omega(\tilde{\mathbf{z}}_{t+1})) + E_\theta(G_\omega(\mathbf{z}_t))\big\} \tag{2.12}$$

$$q(\tilde{\mathbf{z}}_{t+1}|\mathbf{z}_t) \propto \exp\left(\frac{-1}{4\alpha}\left\|\tilde{\mathbf{z}}_{t+1} - \mathbf{z}_t + \alpha\frac{\partial E_\theta(G_\omega(\mathbf{z}_t))}{\partial \mathbf{z}_t}\right\|_2^2\right) \tag{2.13}$$

and accepting (setting $\boldsymbol{z}_{t+1} = \tilde{\boldsymbol{z}}_{t+1}$) with probability r.

The overall training procedure for MEG is detailed in Algorithm 1.

2.4 Related Work

Early work on deep learning relied on unsupervised learning [Hinton et al., 2006, Bengio et al., 2007, Larochelle et al., 2009] to train energy-based models [LeCun et al., 2006], in particular Restricted Boltzmann Machines, or RBMs. Hinton [2000] proposed k-step Contrastive Divergence (CD-k), to efficiently approximate the negative phase log-likelihood gradient. Subsequent work have improved on CD-k such as Persistent CD [Salakhutdinov and Hinton, 2009, Tieleman, 2008b]. Hyvärinen [2005] proposed an alternative method to train non-normalized graphical models using Score Matching, which does not require computation of the partition function.

Kim and Bengio [2016] and Dai et al. [2017] also learn a generator that approximates samples from an energy-based model. However, their approach for entropy maximization is different from our own. Kim and Bengio [2016] argue that batch normalization [Ioffe and Szegedy, 2015] makes the hidden activations of the generator network approximately Gaussian distributed and thus maximize the log-variance for

Algorithm 1 MEG Training Procedure Default values: Adam parameters $\alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9; \lambda = 0.1\,;\, n_\varphi = 5$

Require: Score penalty coefficient λ, # of θ updates per generator update n_φ , # of training iterations T, Adam hyperparameters α, β_1 and β_2.

Require: Energy function E_θ with parameters θ, entropy statistics function T_ϕ with parameters ϕ, generator function G_ω with parameters ω, minibatch size m,

for $t = 1, ..., T$ **do**

for $1, ..., n_\varphi$ **do**

Sample minibatch of real data $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)}\} \sim P_D$.

Sample minibatch of latent $\{\mathbf{z}_0^{(1)}, ..., \mathbf{z}_0^{(m)}\} \sim P_z$.

$\tilde{\mathbf{x}} \leftarrow G_\omega(\mathbf{z})$

$$\mathcal{L}_E \leftarrow \frac{1}{m}\left[\sum_i^m E_\theta(\mathbf{x}^{(i)}) - \sum_i^m E_\theta(\tilde{\mathbf{x}}^{(i)}) + \lambda \sum_i^m \left|\left|\nabla_{\mathbf{x}^{(i)}} E_\theta(\mathbf{x}^{(i)})\right|\right|^2\right]$$

$\theta \leftarrow \text{Adam}(\mathcal{L}_E, \theta, \alpha, \beta_1, \beta_2)$

end for

Sample minibatch of latent $\mathbf{z} = \{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\} \sim P_z$.

Per-dimension shuffle of $\mathbf{z}$, yielding $\{\tilde{\mathbf{z}}^{(1)}, ..., \tilde{\mathbf{z}}^{(m)}\}$.

$\tilde{\mathbf{x}} \leftarrow G_\omega(\mathbf{z})$

$$\mathcal{L}_H \leftarrow \frac{1}{m}\sum_i^m - \left[\log \sigma(T_\phi(\tilde{\mathbf{x}}^{(i)}, \mathbf{z}^{(i)})) - \log\left(1 - \sigma(T_\phi(\tilde{\mathbf{x}}^{(i)}, \tilde{\mathbf{z}}^{(i)}))\right)\right]$$

$$\mathcal{L}_G \leftarrow \frac{1}{m}\left[\sum_i^m E_\theta(\tilde{\mathbf{x}}^{(i)})\right] + \mathcal{L}_H$$

$\omega \leftarrow \text{Adam}(\mathcal{L}_G, \omega, \alpha, \beta_1, \beta_2)$

$\phi \leftarrow \text{Adam}(\mathcal{L}_H, \phi, \alpha, \beta_1, \beta_2)$

end for

each hidden activation of the network. Dai et al. [2017] propose two approaches to entropy maximization. One which minimizes entropy of the inverse model ($p_{gen}(z|x)$) which is approximated using an amortized inverse model similar to ALI [Dumoulin et al., 2016], and another which makes isotropic Gaussian assumptions for the data. In our work, we perform entropy maximization using a tight mutual information estimator which does not make any assumptions about the data distribution.

Zhao et al. [2016] use an autoencoder as the discriminator and use the reconstruction loss as a signal to classify between real and fake samples. The autoencoder is highly regularized to allow its interpretation as an energy function. However Dai et al. [2017] prove that the EBGAN objective does not guarantee the discriminator to recover the true energy function. The generator diverges from the true data

distribution after matching it, since it would continue to receive training signal from the discriminator. The discriminator signal does not vanish even at optimality (when $P_G = P_D$) if it retains density information, since some samples would be considered "more real" than others.

2.5 Experiments

To understand the benefits of MEG, we first visualize the energy densities learnt by our generative model on toy data. Next, we evaluate the efficacy of our entropy maximizer by running discrete mode collapse experiments to verify that we learn all modes and the corresponding mode count (frequency) distribution. Furthermore, we evaluate the performance of MEG on sharp image generation, since this is a common failure mode of models trained with maximum likelihood which tend to generate blurry samples [Theis et al., 2015]. We also compare MCMC samples in visible space and our proposed sampling from the latent space of the composed energy function. Finally, we run anomaly detection experiments to test the application of the learnt energy function.

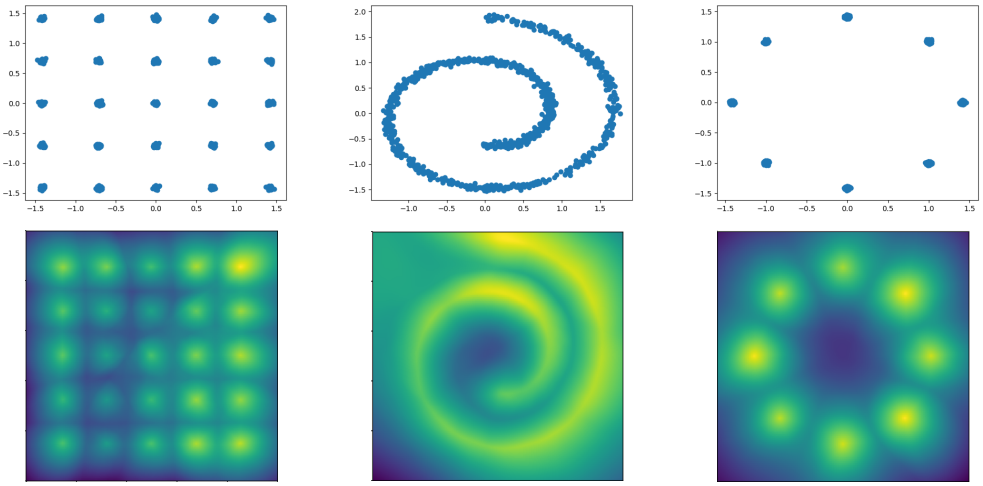
We've released open-source code [1] for all the experiments.

2.5.1 Visualizing the learned energy function

Generative models trained with maximum likelihood often suffer from the problem of spurious modes and excessive entropy of the trained distribution, where the model incorrectly assigns high probability mass to regions not present in the data manifold. Typical energy-based models such as RBMs suffer from this problem partly because of the poor approximation of the negative phase gradient, as discussed above, and the large price paid in terms of log-likelihood for not putting enough probability mass near data points (i.e. for missing modes).

To check if MEG suffers from spurious modes, we train the energy-based model on synthetic 2D datasets (swissroll, 25gaussians and 8gaussians) and visualize the

1. https://github.com/ritheshkumar95/energy_based_generative_models



(a) **(b)** **(c)**

Figure 2.2 – **Top**: True data points for three popular toy dataset (a) 25-gaussians, (b) swiss roll, and (c) 8-gaussians. **Bottom:** Corresponding probability density visualizations using the learned energy function. Density was estimated using a sample based approximation of the partition function.

energy function. From the probability density plots on Figure 2.2, we can see that the energy model doesn't suffer from spurious modes and learns a sharp distribution.

2.5.2 Investigating Mode Collapse

GANs are notorious for having mode collapse issues wherein certain modes of the data distribution are not represented by the generated distribution. Since the generator is trained to minimize its KL divergence with the energy model distribution (which is trained via maximum likelihood), we expect the generator to faithfully capture all the modes of the data distribution. Our theory requires we maximize entropy of the generated distribution, which we believe is instrumental in ensuring full mode capture.

To empirically verify MEG captures all the modes of the data distribution, we follow the same experimental setup as [Metz et al., 2016] and [Srivastava et al., 2017]. We train our generative model on the StackedMNIST dataset, which is a synthetic dataset created by stacking MNIST on different channels. The number of modes are counted using a pretrained MNIST classifier, and the KL divergence

is calculated empirically between the generated mode distribution and the data distribution.

Table 2.1 – Number of captured modes and Kullback-Leibler divergence between the training and samples distributions for ALI [Dumoulin et al., 2016], Unrolled GAN [Metz et al., 2016], VeeGAN [Srivastava et al., 2017], WGAN-GP [Gulrajani et al., 2017]. Numbers except MEG and WGAN-GP are borrowed from Belghazi et al. [2018]

(Max 10^3)	Modes	KL
Unrolled GAN	48.7	4.32
VEEGAN	150.0	2.95
WGAN-GP	959.0	0.7276
MEG (ours)	1000.0	0.0313

(Max 10^4)	Modes	KL
WGAN-GP	9538.0	0.9144
MEG (ours)	10000.0	0.0480

From Table 1, we can see that MEG naturally covers all the modes in that data, without dropping a single mode. Apart from just representing all the modes of the data distribution, MEG also better matches the data distribution as evidenced by the significantly smaller KL divergence score compared to the baseline WGAN-GP.

Apart from the standard 3-StackMNIST, we also evaluate MEG on a new dataset with 10^4 modes (4 stacks) [2] which is evidence that MEG does not suffer from mode collapse issues unlike state-of-the-art GANs like WGAN-GP.

2.5.3 Modeling Natural Images

While the energy landscapes in Figure 2.2 provide evidence that MEG trains energy models with sharp distributions, we next investigate if this also holds when learning a distribution over high-dimensional natural images. Energy-based models trained with existing techniques produce blurry samples due to the energy function not learning a sharp distribution.

2. The 4-StackedMNIST was created in a way analogous to the original 3-StackedMNIST dataset. We randomly sample and fix 128×10^4 images to train the generative model and take 26×10^4 samples for evaluations.

We train MEG on the standard benchmark 32x32 CIFAR10 [Krizhevsky et al., 2009] dataset for image modeling. We additionally train MEG on the 64x64 cropped CelebA - celebrity faces dataset [Liu et al., 2015] to report qualitative samples from MEG. Similar to recent GAN works [Miyato et al., 2018], we report both Inception Score (IS) and Fréchet Inception Distance (FID) scores on the CIFAR10 dataset and compare it with a competitive WGAN-GP baseline.

Table 2.2 – Inception scores and FIDs with unsupervised image generation on CIFAR-10. We used 50000 sample estimates to compute Inception Score and FID.

Method	**Inception score**	**FID**
Real data	11.24±.12	7.8
WGAN-GP	6.81 ± .08	30.95
MEG (Generator)	6.49 ± .05	35.02
MEG (MCMC)	**7.31 ± .06**	33.18

From Table 2.2, we can see that in addition to learning an energy function, MEG-trained generative model produces samples comparable to recent GAN methods such as WGAN-GP [Gulrajani et al., 2017]. Note that the perceptual quality of the samples improves by using the proposed MCMC sampler in the latent space. See also Figure 2.3 for an ablation study which shows that MCMC on the visible space does not perform as well as MCMC on the latent space.

2.5.4 Anomaly Detection

Apart from the usefulness of energy estimates for relative density estimation (up to the normalization constant), energy functions can also be useful to perform unsupervised anomaly detection. Unsupervised anomaly detection is a fundamental problem in machine learning, with critical applications in many areas, such as cyber-security, complex system management, medical care, etc. Density estimation is at the core of anomaly detection since anomalies are data points residing in low probability density areas. We test the efficacy of our energy-based density model for anomaly detection using two popular benchmark datasets: KDDCUP and MNIST.

KDDCUP We first test our generative model on the KDDCUP99 10 percent dataset from the UCI repository [Lichman et al., 2013]. Our baseline for this task is

Deep Structured Energy-based Model for Anomaly Detection (DSEBM) [Zhai et al., 2016], which trains deep energy models such as Convolutional and Recurrent EBMs using denoising score matching [Vincent, 2011] instead of maximum likelihood, for performing anomaly detection. We also report scores on the state of the art DAGMM [Zong et al., 2018], which learns a Gaussian Mixture density model (GMM) over a low dimensional latent space produced by a deep autoencoder. We train MEG on the KDD99 data and use the score norm $||\nabla_x E_\theta(x)||_2^2$ as the decision function, similar to Zhai et al. [2016].

Table 2.3 – Performance on the KDD99 dataset. Values for OC-SVM, DSEBM values were obtained from Zong et al. [2018]. Values for MEG are derived from 5 runs. For each individual run, the metrics are averaged over the last 10 epochs.

Model	Precision	Recall	F1
Kernel PCA	0.8627	0.6319	0.7352
OC-SVM	0.7457	0.8523	0.7954
DSEBM-e	0.8619	0.6446	0.7399
DAGMM	0.9297	0.9442	0.9369
MEG (ours)	**0.9354 ± 0.016**	**0.9521 ± 0.014**	**0.9441 ± 0.015**

From Table 2.3, we can see that the MEG energy function outperforms the previous SOTA energy-based model (DSEBM) by a large margin (+0.1990 F1 score) and is comparable to the current SOTA model (DAGMM). Note that DAGMM is specially designed for anomaly detection, while MEG is a general-purpose energy-based model.

MNIST Next we evaluate our generative model on anomaly detection of high dimensional image data. We follow the same experiment setup as [Zenati et al., 2018] and make each digit class an anomaly and treat the remaining 9 digits as normal examples. We also use the area under the precision-recall curve (AUPRC) as the metric to compare models. From Table 4, it can be seen that our energy model outperforms VAEs for outlier detection and is comparable to the SOTA BiGAN-based anomaly detection methods for this dataset [Zenati et al., 2018] which train bidirectional GANs to learn both an encoder and decoder (generator) simultaneously and use a combination of the reconstruction error in output space

Table 2.4 – Performance on the unsupervised anomaly detection task on MNIST measured by area under precision recall curve. Numbers except ours are obtained from [Zenati et al., 2018]. Results for MEG are averaged over the last 10 epochs to account for the variance in scores.

Heldout Digit	VAE	MEG	BiGAN-σ
1	0.063	0.281 ± 0.035	0.287 ± 0.023
4	0.337	0.401 ± 0.061	0.443 ± 0.029
5	0.325	0.402 ± 0.062	0.514 ± 0.029
7	0.148	0.29 ± 0.040	0.347 ± 0.017
9	0.104	0.342 ± 0.034	0.307 ± 0.028

as well as the discriminator's cross entropy loss as the decision function.

Our aim here is not to claim state-of-the-art on the task of anomaly detection but to demonstrate the quality of the energy functions learned by our technique, as judged by its competitive performance on anomaly detection.

2.5.5 MCMC Sampling in visible vs latent space

To show that the Metropolis-Adjusted Langevin Algorithm (MALA) performed in latent space produces good samples in observed space, we attach samples from the beginning (with z sampled from a Gaussian) and end of the chain for visual inspection 2.3. From the attached samples, it can be seen that the MCMC sampler appears to perform a smooth walk on the image manifold, with the initial and final images only differing in a few latent attributes such as hairstyle, background color, face orientation, etc. Note that the MALA sampler run on E_θ in visible space 2.3 did not work well and tends to get attracted to spurious modes (which G eliminates, hence the advantage of the proposed p_{EG} sampling scheme).

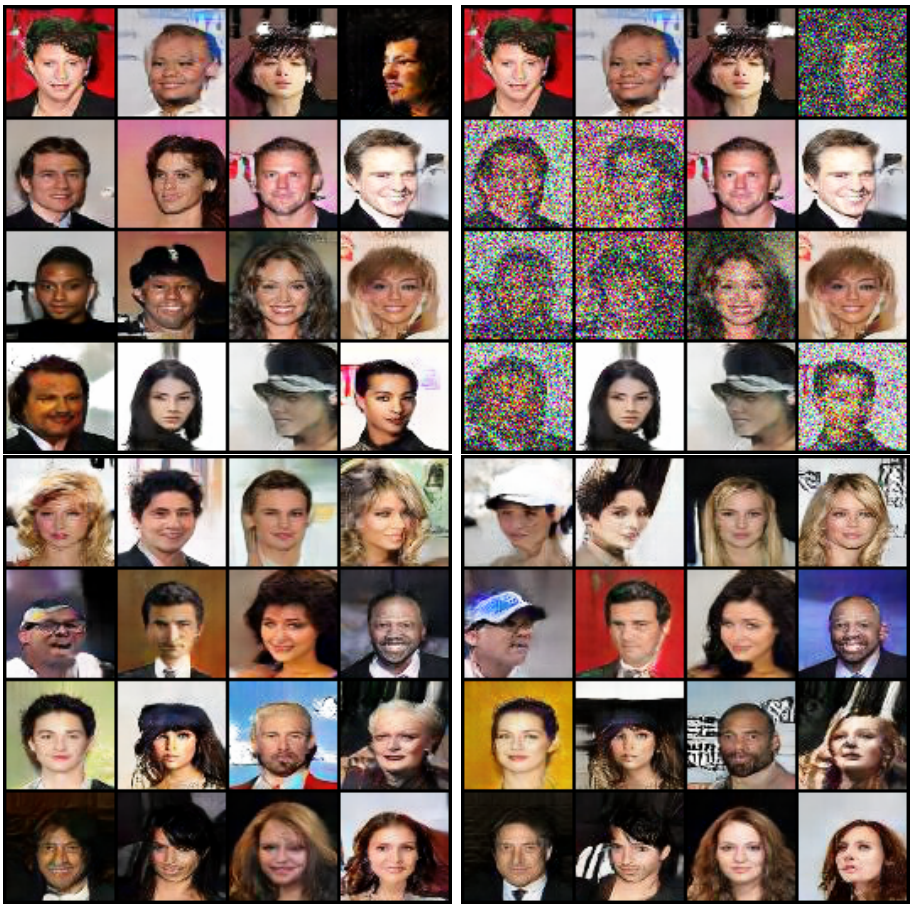


Figure 2.3 – Samples from the beginning and end of the MCMC in visible space (**top**) and latent space (**bottom**) using the MALA proposal and acceptance criteria. MCMC in visible space has poor mixing and gets attracted to spurious modes, while MCMC in latent space seems to change semantic attributes of the image, while not producing spurious modes.

3 Conclusion

We proposed MEG, an energy-based generative model that produces energy estimates using an energy model and a generator that produces fast approximate samples. This takes advantage of novel methods to maximize the entropy at the output of the generator using a nonparametric mutual information lower bound estimator. We have shown that our energy model learns good energy estimates using visualizations in toy 2D datasets and through performance in unsupervised anomaly detection. We have also shown that our generator produces samples of high perceptual quality by measuring Inception Scores and Fréchet Inception Distance and shown that MEG is robust to the respective weaknesses of GAN models (mode dropping) and maximum-likelihood energy-based models (spurious modes).

This work has made an important step towards the training of energy-based models. The future work in this direction involves more careful understanding of the explosion of the temperature of the energy-function during training, which seems closely related to the Lipschitz constraints in the Wasserstein GAN literature. This work also notices that MCMC performed in visible space gets attracted to spurious modes and also doesn't mix between modes very well. This is an important problem that still needs to be addressed. More effort devoted towards better entropy maximizers in neural networks might be worth the effort as well, since entropy maximization is fundamental to prevent mode collapse in generative models as well as maximum entropy policies in Reinforcement Learning, etc.

Bibliographie

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

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

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

David Barber and Felix Agakov. The im algorithm: A variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 201–208, Cambridge, MA, USA, 2003. MIT Press. URL http://dl.acm.org/citation.cfm?id=2981345.2981371.

Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062, ICML'2018*, 2018.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS'2006*, 2007.

Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers, 2009.

Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *ICML'2013*, 2013.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

Y-Lan Boureau, Sumit Chopra, Yann LeCun, et al. A unified energy-based framework for unsupervised learning. In *Artificial Intelligence and Statistics*, pages 371–379, 2007.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1): 2096–2030, 2016.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Ian Goodfellow. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515*, 2014.

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

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NIPS'2014*, 2014b.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *NIPS'2017*, pages 5767–5777, 2017.

M. Gutmann and A. Hyvarinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS'2010*, 2010.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local

nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Geoffrey E Hinton. Products of experts. 1999.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Unit, University College London, 2000.

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

Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

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

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Justin B Kinney and Gurinder S Atwal. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Hugo Larochelle, Yoshua Bengio, Jerome Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *J. Machine Learning Res.*, 10:1–40, 2009.

Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *AIStats*, volume 6, page 34, 2005.

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

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc-Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Scholkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*, pages 191–246. MIT Press, 2006.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial

network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Moshe Lichman et al. Uci machine learning repository, 2013.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

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

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

Ben Poole, Sherjil Ozair, Aaron van den Oord, Alex Alemi, and George Tucker. On variational lower bounds of mutual information. 2018.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

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

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML'2008*, pages 1064–1071, 2008a.

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

D Michael Titterington, Adrian FM Smith, and Udi E Makov. *Statistical analysis of finite mixture distributions*. Wiley,, 1985.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

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

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

Wikipedia. Latent variable — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Latent%20variable&oldid=906955761, 2019a. [Online ; accessed 22-July-2019].

Wikipedia. Metropolis-adjusted Langevin algorithm — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Metropolis-adjusted%20Langevin%20algorithm&oldid=884997886, 2019b. [Online ; accessed 23-July-2019].

Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017.

Laurent Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228, 1998.

Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*, 2016.

He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

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

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

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. 2018.