Unsupervised Learning with Regularized Autoencoders

by

Junbo Zhao

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2019

Professor Yann LeCun

Dedication

To my family.

Acknowledgements

To pursue a PhD has been the best decision I made in my life. First and foremost, I must thank my advisor Yann LeCun. To work with Yann, after flying half a globe from China, was indeed an unbelievable luxury. Over the years, Yann has had a huge influence on me. Some of these go far beyond research—the dedication, visionary thinking, problem-solving methodology and many others.

I thank Kyunghyun Cho. We have never stopped communications during my PhD. Every time we chat, Kyunghyun is able to give away awesome advice, whether it is feedback on the research projects, startup and life. I am in his debt. I also must thank Sasha Rush and Kaiming He for the incredible collaboration experience. I greatly thank Zhilin Yang from CMU, Saizheng Zhang from MILA, Yoon Kim and Sam Wiseman from Harvard NLP for the nonstopping collaboration and friendship throughout all these years.

I am grateful for having been around the NYU CILVR lab and the Facebook AI Research crowd. In particular, I want to thank Michael Mathieu, Pablo Sprechmann, Ross Goroshin and Xiang Zhang for teaching me deep learning when I was a newbie in the area. I thank Mikael Henaff, Martin Arjovsky, Cinjon Resnick, Emily Denton, Will Whitney, Sainbayar Sukhbaatar, Ilya Kostrikov, Jason Lee, Ilya Kulikov, Roberta Raileanu, Alex Nowak, Sean Welleck, Ethan Perez, Qi Liu and Jiatao Gu for all the fun, joyful and fruitful discussions. I also want to thank Jinyang Li, Rob Fergus, Douwe Kiela, Joan Bruna, Sam Bowman, Branden Lake, Authur Szlam and Zheng Zhang for keeping giving useful feedback on my research projects.

I thank all my Chinese friends who were very supportive of me: Wenjia Wu, James Zhou, William Li, Minjie Wang, Shane Gu, Jiwei Li and Edison Xiao. Lastly, I thank my parents, Yuting Gui, Nan Zhao and Zeyu Chen for their unconditional loving support.

Jake Zhao

September 2019

Abstract

Deep learning has enjoyed remarkable successes in a variety of domains. These successes often emerge at the cost of large annotated datasets and training computationally heavy neural network models. The learning paradigm for this is called *supervised learning*. However, to reduce the sample complexity while improving the universality of the trained models is a crucial next step that may to artificial intelligence. *Unsupervised Learning*, in contrast to supervised learning, aims to build neural network models with more generic loss objectives requiring little or no labelling effort, and therefore it does not reside in any specific domain-task.

In spite of the brevity of its goal, unsupervised learning is a broad topic that relates or includes several sub-fields, such as density estimation, generative modeling, world model, etc. In this thesis, we primarily adopt an energy-based view unifying these different fields [LeCun et al., 2006]. The desired energy function reflects the data manifold by differentiating the energy assigned to the points on the data manifold against points off the manifold. Basing on this foundation, we first cast the popular autoencoder and adversarial learning framework into an energy-based perspective. Then, we propose several techniques or architectures with a motivation to improve learning the energy function in an unsupervised setting.

The thesis is organized as follows. First, we list out a number of common strategies to shape a good energy function by learning. Among these, we mainly target at two strategies and extend the frontier of them. The resulted models from this thesis demonstrate several applications made possible by using no or few labeled data. It covers a wide spectrum of computer vision and language tasks, such as generation, text summarization, text style-transfer and transfer/semi-supervised

learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The revival of deep learning has influenced many domains in recent years, from computer vision [Krizhevsky et al., 2012], to natural language processing [Sutskever et al., 2014], to game AI [Silver et al., 2017] and a few others [LeCun et al., 2015]. Among all these successes, the paradigm of supervised learning plays the most pivotal role. This means huge costs spent on dataset curation and vast computing powers for achieving each one of the specific tasks.

Unsupervised learning, however, aims to learn relatively more generic models based on loss objectives that require minimal or significantly less labelling efforts. For example, autoencoders are trained to perform self-reconstruction, generative adversarial networks are trained by a minimax game on distinguishing real from generated samples, while the video predictive models aim to capture the dynamics of the world by watching live videos.

1.1 Energy-based learning

Despite many interpretations of unsupervised learning, in this thesis we primarily adopt an energy-based perspective from LeCun et al. [2006]. Formally, let us define two sets of variables, X and Y, with the end goal being to capture the dependency between them. Following a grand introduction by LeCun et al. [2006], one can build an energy-based model $E(X, Y)$ that the energy function $E(\cdot)$ returns a scalar value measuring the *compatibility* between the given values to the variables. At inference time, if to predict Y based on X, we can solve the following optimization problem minimizing an energy:

$$Y^* = \arg\min_{Y \in \mathcal{Y}} E(X, Y), \tag{1.1}$$

where $\mathcal{Y}$ denotes the set of possible values can be assigned to Y.

To give a more concrete example, let us use variable X as a 2-D array of the pixel values in an image and Y is the corresponding annotated label. This is equivalent to a classical image recognition setup. Similarly, when Y represents the next one or few frames in a video sequence and X represents its past frames, modeling the joint dependencies between X and Y is a video prediction problem. Moreover, if we turn this to be a unary variable problem by just setting Y to X, the energy function is essentially parametrized by some of form of *reconstruction*.

One can bridge the energy-based models with other probabilistic models by casting the energy values into probabilities—for example through a Gibbs distribution:

$$P(Y|X) = \frac{\exp^{-\beta E(X,Y)}}{\int_{Y \in \mathcal{Y}} \exp^{-\beta E(X,Y)}}, \tag{1.2}$$

assuming Y is a continuous variable and β tunes the probability distribution as a *temperature* hyperparameter. As it stands, energies can be understood as unnormalized density. This frees us from calculating normalization factors which are often intractable in the probabilistic models.

1.2 Strategies to build a meaningful energy function

How can we characterize good energy functions against the bad ones? In a nutshell, a desirable energy function ought to reflect the shape of the data manifold. By definition from LeCun et al. [2006], energy should be low assigning to the points on-manifold and high to everywhere else.

In this thesis, we mainly focus on extending the horizon of two most popular strategies:

- Push down of the energy of data points while pulling up on the chosen locations;
- Develop a regularizer which limits the volume the space that has low energies.

1.3 Autoencoders as energy-based models

In this section, we examine a simple, flexible but powerful architecture—the autoencoders—and discuss it from an energy-based perspective.

To begin with, an autoencoder is trained to reconstruct a given input:

$$\text{given}(x) \in P_*, \quad g_\psi(f_\theta(x)) \approx x, \tag{1.3}$$

where the P_* denotes the data distribution. The encoder f_θ and the decoder g_ψ are trained jointly to minimize a form of reconstruction error. For continuous data like images, the reconstruction error is often defined by

$$E_{\theta,\psi}(x) = \|g_\psi(f_\theta(x)) - x\|^2. \tag{1.4}$$

Recapping the energy-based learning pipeline, here we essentially have an unary variable objective by setting variable Y to be X.

Although autoencoders are trained to merely "copy" the input, an ideal autoencoder should not just copy everything. It is desirable to only reconstruct the data points on the manifold while leaving the points off-manifold unreconstructed. This corresponds to the desired energy function conditions.

An alternative interpretation is grounded on *representation learning*. Early studies [Ng et al., 2013] of autoencoders pointed out that a two-layer linear autoencoder performs similarly to the Principle Component Analysis algorithm, with the restriction that the number of the hidden layer being smaller than the input size. What is more, a multi-layer autoencoder was proposed by Hinton and Salakhutdinov [2006]. This model consists of multiple Restricted Boltzmann Machine modules and is trained with a layerwise training protocol. One important architectural choice is to reduce the size of the hidden layer progressively from layer to layer. This work pioneered using autoencoders to do dimensionality reduction, or more broadly speaking, representation learning.

The goal of representation learning is to learn a compact, smooth and possibly lower-dimensional code. An ideal representation learning model may contain two components: an encoder performing efficient inference from the input and a decoder mapping the code back to the corresponding input. To let autoencoders learn good representation, what is missing, apart from its reconstruction objective, is an "information bottleneck" being placed in at the code layer. Simply put, this bottleneck can be constructed by a reduced dimensionality, imposing priors like sparsity, etc. We review the literature from this respect in chapter 2. We can also connect the information bottleneck composition to reducing the low-energy area from the energy-based view.

1.4 Generative models

Learning the density function of some complex distribution P_* can be very challenging. Traditional sampling approaches such as MCMC often fail to capture a multimodal distribution, because the sampling may either be too slow or biased. However, the recent progress on deep generative modeling shows the possibility of generating realistic and high-resolution images [Brock et al., 2018], or human-readable text [Bowman et al., 2015c]. The fundamental technique is to rely on a high-capacity neural network as a *generator*, G, and we train it to approximate the real data distribution P_*, such that $P_G \sim P_*$. To introduce the necessary *uncertainty*, G is devised to map a latent vector z, which is obtained from an easy-to-sample distribution, to the real data space. There are two most popular frameworks in this domain—the variational autoencoder [Kingma and Welling, 2013] and generative adversarial network [Goodfellow et al., 2014]. More details about the backgrounds of these models are written in chapter 2.

In chapter 3 of this thesis, we propose the Energy-based Generative Adversarial Network to further embrace generative modeling with the energy-based learning.

1.5 Transfer learning

For years, transfer learning has been used as a paradigm deploying what has been learned from an unsupervisedly trained model to benefit downstream tasks. In computer vision, researchers cache the weights obtained from an unsupervised autoencoder and use them to initialize the neural network performing image classification [Rifai et al., 2011a]. Fine-tuning, which means slightly altering the pretrained weights in favor of the downstream task, is another standardized technique in transfer learning.

More recently, transfer learning has had considerable success in natural language processing. Researchers have shown that training an autoencoder at scale, based on certain forms of maksed reconstrcuction [Devlin et al., 2018], enables significant gain on a variety of language-related tasks such as question-answering [Rajpurkar et al., 2016], natural language inference [Bowman et al., 2015a] and many others. This area is still growing fast with potentials. In this thesis, we propose GLoMo in chapter 7, a structure-oriented transfer learning framework that aims at transferring the first-order dependency between pairwise input units (such as a word in a sentence, or pixels of an image). GLoMo is complementary to the feature-embedding legacy as most of the researches we mentioned in this section target at, and ideally can be used in combination with the feature embeddings.

Finally, it is worth noting that besides having an unsupervised objective in the first training phase, supervised transfer learning is an alternative paradigm, particularly in computer vision. It mostly relies on training a deep convolution-based network, such as ResNet [He et al., 2016], on the ImageNet dataset [Deng et al., 2009]. This pretrained ResNet can be used as a feature extractor afterward to perform downstream tasks such as visual question answering [Agrawal et al., 2017] or image captioning [Chen et al., 2015].

1.6 Thesis Outline

The thesis is organized as follows. In chapter 2, we review the prior related work. Chapter 3, 4, 5 introduce three ways following the strategy (i), that the energy of the points on the manifold is pushed down while some other deliberately chosen locations are being pulled up. These three different strategies involve adversarial training and discretization in the code space. Chapter 6 develops a hierarchical autoencoder model that involves convolution, deconvolution, pooling and unpooling, that limits the volume of the space carrying low energies. Chapter 7 proposes GLoMo, a hierarchical masked-autoencoder model aiming to capture the dependency between atomic units from the input, and we experiment it in a transfer learning scenario. Finally, chapter 8 concludes the thesis and envisions future directions.

Chapter 2

Related Work

In this chapter, we present the prior work that is relevant to the thesis. First, in section 2.1, we cover the literature of training autoencoder framework and all its variants. Then we cover semi-supervised learning and transfer learning paradigms in section 2.2 and section 2.3 respectively.

2.1 Regularized autoencoders

Autoencoders are energy-based models trained with reconstruction energy.

However, an overcapacity and unregularized autoencoder result in learning an identity function on the whole input space. When it does so, this amounts to having zero energy everywhere and the representation learned in the code space would likely be useless. From the energy-based learning perspective, we need to introduce certain regularization mechanism to restrict the low-energy area. This corresponds to preventing the autoencoder from merely copying everything by an identity function.

Information bottleneck in the hidden layer One approach to restrict the low-energy area of an autoencoder is to place an information bottleneck in the hidden layer. Specifically, sparse autoencoder [Boureau et al., 2008] and predictive sparse decomposition [Kavukcuoglu et al., 2010a] utilizes a sparsity prior by penalizing the l_1 norm of the obtained code. This encourages only a few coordinates in the code space to be non-zero. Extension of this framework includes the K-sparse autoencoder [Makhzani and Frey, 2013] where strictly only k hidden elements are active, and the winner-take-all autoencoder [Makhzani and Frey, 2015] as an extreme case that only the largest coordinate are preserved.

Learning encoder as a contraction Other than restricting the code layer, one can also enforce the encoder to learn a contraction function. The constractive autoencoder framework [Rifai et al., 2011a] designs a penalty term of Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input, and by doing so to force the encoder to be locally contractive.

Injecting noise to the code space The popular variational autoencoder (VAE) framework has been proved to be effective in semi-supervised learning, disentanglement of variation of factors, and generating samples [Kingma and Welling, 2013]. As with all the other autoencoder models, VAE can be cast to an energy-based model, trained with reconstruction energy. VAEs differ from regular autoencoders because it minimizes a *variational lowerbound* such that the code c, obtained from its encoder, follows a predefined Gaussian distribution. A VAE instantiation consists of (i)-an encoder that converts input examples into a *posterior* distribution, and (ii)-a decoder trained to reconstruct the input conditioned on the sampled vectors from the posterior. The ELBO bound incorporates another KL term measuring the difference between the posterior distribution and the predefined Gaussian distribution. The end goal of VAE is to both build a latent variable generative model (decoder-output pathway) as well as an inference model (input-encoder pathway).

The output of the VAE encoder has a bi-variable structure: mean μ and covariance matrix σ parametrizing a normal distribution. The reparametrization technique it proposes can be thought of as injecting noise to the code layer. In this case, the encoder has to pull the samples away from each other in the latent space for reconstruction, as later interpreted by WAE [Tolstikhin et al., 2017]. This was proved an effective regularizer in tasks like semi-supervised learning on MNIST [Kingma et al., 2014].

Input/target differentiation Traditional autoencoders are built upon a reconstruction objective using the same input vector as the target. Despite the simplicity, this choice of loss function has made autoencoder tend to copy information over without learning useful representations. Instead of trying to restrict the low-energy area by putting bottleneck in the code space or encoder capacity, an orthogonal way is to differentiate the target from the input. For instance, denoising autoencoders [Vincent et al., 2010] purposely pollute the input image by adding Gaussian noise and let the autoencoder recover the clean image. This framework can be understood as using the noisy images as the negative examples where the model assigns relatively high energy (the model does not map the noisy input back to where they are, but to the data manifold. This results in high reconstruction energy at the noisy examples' area). This technique is also employed in natural language processing, notably in unsupervised neural machine translation [Artetxe et al., 2017; Lample et al., 2018]. However, because texts are discrete data, the noise injected here is also more structural—for example, word deletion, swap and addition.

Masked reconstruction One particular way to set up the autoencoder reconstruction loss objective is through masking. Specifically, the most common factorization is based on an autoregressive prior, by which the model is only allowed to see the input coordinates $t < T$ when trying to reconstruct the timestep T. Autoencoders applying this version of masked reconstruction on text are referred to as "language model" while its generalized version in computer vision includes pixelRNN [Oord et al., 2016], pixelCNN [Van den Oord et al., 2016] etc.

Formally, we can write down this version of masked reconstruction objective by

$$\mathcal{L}(x) = \sum_t \log P(x_t | x_{t'<t}), \tag{2.1}$$

where x denotes the input and t is the considered timestep. Note that the discretization is often conducted in the output space, even for continuous data like images [Oord et al., 2016]. Unlike regular autoencoders, applying this factorization an estimation directly at the likelihood, which is grounded on the autoregressive prior.

More recently, BERT [Devlin et al., 2018] challenges this autoregressive left-to-right or right-to-left factorization. It proposes an alternative version of masked reconstruction by randomly masking out tokens in the input sentence while training a transfomer-based [Vaswani et al., 2017] autoencoder to recover the blanks. Although this compromises the capability of likelihood estimation, the authors show that BERT achieves tremendous results on a variety of benchmarks under the transfer learning paradigm.

2.2 Semi-supervised learning

Semi-supervised learning (SSL) paradigm was proven successful in many standard tasks. For example, Kingma et al. [2014] show that the error rate of MNIST can be reduced to be around 1% in an extreme condition when only using 100 labelled samples. This paradigm builds a connection between unsupervised learning and supervised learning, and mainly targets at the scenarios where labels are lacked. Unlike supervised learning, SSL does not require all data to be labelled. Instead, it makes leverage of the unlabeled data while being capable to train with labels.

General philosophy is to use the unlabeled data to train an energy function, then combine it with the actual supervised loss from labelled data. Such approach harnesses the knowledge from the unlabeled data to discover the data manifold. A few past works can be understood by this philosophy. For example, the ladder network [Rasmus et al., 2015] combining reconstruction energies at several abstraction levels achieves less than 1% error rate on MNIST with only 100 labels. In this thesis, we also explore this domain and propose the stacked what-where autoencoder in chapter 6, and the energy-based adversarial network in chapter 3.

2.3 Transfer learning

There is an overwhelming amount of evidence on the success of transferring pre-trained representation across different tasks in deep learning. Notably in natural language processing, since word2vec [Mikolov et al., 2013a] was proposed, transferring word-level embedding has become a standard step serving many NLP tasks. Some other well-known embeddings include GLoVe [Pennington et al., 2014], ELMo [Peters et al., 2018], and more recently—BERT [Devlin et al., 2018]—thanks to the emerging transformer architecture.

In computer vision, it remains a standard practice to use features learned in supervised learning on the ImageNet dataset for other downstream tasks.

In chapter 7, we introduce GLoMo, a structure-oriented transfer learning approach that is complementary to these aforementioned approaches.

Chapter 3

Energy-based Generative Adversarial Networks

In this chapter, we propose a new framework, Energy-based Generative Adversarial Network (EBGAN). This new framework combines two unsupervised models, Generative Adversarial Networks and Autoencoders. On one side, the resulted model offers a new perspective that views the discriminator in a GAN framework via an energy function. On the other, we show that when using an autoencoder-based energy function in place of the discriminator, the generator can be used as a negative-sample provider leading to regularize the energy function.

3.1 Motivation

We introduce the "Energy-based Generative Adversarial Network" model which views the discriminator as an energy function that attributes low energies to the regions near the data manifold and higher energies to other regions. Similar to the probabilistic GANs, a generator is seen as being trained to produce contrastive samples with minimal energies, while the discriminator is trained to assign high energies to these generated samples. Viewing the discriminator as an energy function allows to use a wide variety of architectures and loss functionals in addition to the usual binary classifier with logistic output. Among them, we show one instantiation of EBGAN framework as using an auto-encoder architecture, with the energy being the reconstruction error, in place of the discriminator. We show that this form of EBGAN exhibits more stable behavior than regular GANs during training. We also show that a single-scale architecture can be trained to generate high-resolution images.

3.2 Introduction

3.2.1 Energy-based model

The essence of the energy-based model [LeCun et al., 2006] is to build a function that maps each point of an input space to a single scalar, which is called "energy". The learning phase is a data-driven process that shapes the energy surface in such a way that the desired configurations get assigned low energies, while the incorrect ones are given high energies. Supervised learning falls into this framework: for each X in the training set, the energy of the pair (X, Y) takes low values when Y is the correct label and higher values for incorrect Y's. Similarly, when modeling X alone within an unsupervised learning setting, lower energy is attributed to the data manifold. The term *contrastive sample* is often used to refer to a data point causing an energy pull-up, such as the incorrect Y's in supervised learning and points from low data density regions in unsupervised learning.

3.2.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] have led to significant improvements in image generation [Denton et al., 2015; Radford et al., 2015; Im et al., 2016; Salimans et al., 2016], video prediction [Mathieu et al., 2015] and a number of other domains. The basic idea of GAN is to simultaneously train a discriminator and a generator. The discriminator is trained to distinguish *real* samples of a dataset from *fake* samples produced by the generator. The generator uses input from an easy-to-sample random source, and is trained to produce fake samples that the discriminator cannot distinguish from real data samples. During training, the generator receives the gradient of the output of the discriminator with respect to the fake sample. In the original formulation of GAN in Goodfellow et al. [2014], the discriminator produces a probability and, under certain conditions, convergence occurs when the distribution produced by the generator matches the data distribution. From a game theory point of view, the convergence of a GAN is reached when the generator and the discriminator reach a Nash equilibrium.

3.2.3 Energy-based Generative Adversarial Networks

In this work, we propose to view the discriminator as an energy function (or a contrast function) without explicit probabilistic interpretation. The energy function computed by the discriminator can be viewed as a trainable cost function for the generator. The discriminator is trained to assign low energy values to the regions of high data density, and higher energy values outside these regions. Conversely, the generator can be viewed as a trainable parameterized function that produces samples in regions of the space to which the discriminator assigns low energy. While it is often possible to convert energies into probabilities through a Gibbs distribution [LeCun et al., 2006], the absence of normalization in this energy-based form of GAN provides greater flexibility in the choice of architecture of the discriminator and the training procedure.

The probabilistic binary discriminator in the original formulation of GAN can be seen as one way among many to define the contrast function and loss functional, as described in LeCun et al. [2006] for the supervised and weakly supervised settings, and Ranzato et al. [2007a] for unsupervised learning. We experimentally demonstrate this concept, in the setting where the discriminator is an auto-encoder architecture, and the energy is the reconstruction error.

Our main contributions are summarized as follows:

- An energy-based formulation for generative adversarial training.
- A proof that under a simple hinge loss, when the system reaches convergence, the generator of EBGAN produces points that follow the underlying data distribution.
- An EBGAN framework with the discriminator using an auto-encoder architecture in which the energy is the reconstruction error.
- A set of systematic experiments to explore hyper-parameters and architectural choices that produce good result for both EBGANs and probabilistic GANs.
- A demonstration that EBGAN framework can be used to generate reasonable high-resolution images from the ImageNet dataset at 256×256 pixel resolution, without a multi-scale approach.

3.3 The EBGAN Model

Let p_{data} be the underlying probability density of the distribution that produces the dataset. The generator G is trained to produce a sample $G(z)$, for instance an image, from a random vector z, which is sampled from a known distribution p_z, for instance $\mathcal{N}(0, 1)$. The discriminator D takes either real or generated images, and estimates the energy value $E \in \mathbb{R}$ accordingly, as explained later. For simplicity, we assume that D produces non-negative values, but the analysis would hold as long as the values are bounded below.

3.3.1 Objective functional

The output of the discriminator goes through an objective functional in order to shape the energy function, attributing low energy to the real data samples and higher energy to the generated ("fake") ones. In this work, we use a margin loss, but many other choices are possible as explained in LeCun et al. [2006]. Similarly to what has been done with the probabilistic GAN [Goodfellow et al., 2014], we use a two different losses, one to train D and the other to train G, in order to get better quality gradients when the generator is far from convergence.
Given a positive margin m, a data sample x and a generated sample $G(z)$, the discriminator loss $\mathcal{L}_D$ and the generator loss $\mathcal{L}_G$ are formally defined by:

$$\mathcal{L}_D(x, z) = D(x) + [m - D\big(G(z)\big)]^+ \tag{3.1}$$

$$\mathcal{L}_G(z) = D\big(G(z)\big) \tag{3.2}$$

where $[\cdot]^+ = max(0, \cdot)$. Minimizing $\mathcal{L}_G$ with respect to the parameters of G is similar to maximizing the second term of $\mathcal{L}_D$. It has the same minimum but non-zero gradients when $D(G(z)) \geq m$.

3.3.2 Optimality of the solution

In this section, we present a theoretical analysis of the system presented in section 3.3.1. We show that if the system reaches a Nash equilibrium, then the generator G produces samples that are indistinguishable from the distribution of the dataset. This section is done in a non-parametric setting, *i.e.* we assume that D and G have infinite capacity.

Given a generator G, let p_G be the density distribution of $G(z)$ where $z \sim p_z$. In other words, p_G is the density distribution of the samples generated by G.
We define:

$$V(G, D) = \int_{x,z} \mathcal{L}_D(x, z) p_{data}(x) p_z(z) \mathrm{d}x \mathrm{d}z \tag{3.3}$$

$$U(G, D) = \int_{z} \mathcal{L}_G(z) p_z(z) \mathrm{d}z \tag{3.4}$$

We train the discriminator D to minimize the quantity V and the generator G to minimize the quantity U.
A Nash equilibrium of the system is a pair (G^*, D^*) that satisfies:

$$V(G^*, D^*) \leq V(G^*, D) \qquad \forall D \tag{3.5}$$

$$U(G^*, D^*) \leq U(G, D^*) \qquad \forall G \tag{3.6}$$

Theorem 1. *If (D^*, G^*) is a Nash equilibrium of the system, then $p_{G^*} = p_{data}$*

almost everywhere, and $V(D^*, G^*) = m$.

Proof. First we observe that

$$\begin{aligned} V(G^*, D) &= \int_x p_{data}(x) D(x) \mathrm{d}x + \int_z p_z(z) \left[m - D(G^*(z))\right]^+ \mathrm{d}z && (3.7) \\ &= \int_x \left(p_{data}(x) D(x) + p_{G^*}(x) \left[m - D(x)\right]^+\right) \mathrm{d}x. && (3.8) \end{aligned}$$

The analysis of the function $\varphi(y) = ay + b(m - y)^+$ (see lemma 1 in appendix 3.7 for details) shows:

(a) $D^*(x) \leq m$ almost everywhere. To verify it, let us assume that there exists a set of measure non-zero such that $D^*(x) > m$. Let $\widetilde{D}(x) = \min(D^*(x), m)$. Then $V(G^*, \widetilde{D}) < V(G^*, D^*)$ which violates equation 3.5.

(b) The function φ reaches its minimum in m if $a < b$ and in 0 otherwise. So $V(G^*, D)$ reaches its minimum when we replace $D^*(x)$ by these values. We obtain

$$\begin{aligned} V(G^*, D^*) &= m \int_x \mathbb{1}_{p_{data}(x) < p_{G^*}(x)} p_{data}(x) \mathrm{d}x + m \int_x \mathbb{1}_{p_{data}(x) \geq p_{G^*}(x)} p_{G^*}(x) \mathrm{d}x && (3.9) \\ &= m \int_x \left(\mathbb{1}_{p_{data}(x) < p_{G^*}(x)} p_{data}(x) + \left(1 - \mathbb{1}_{p_{data}(x) < p_{G^*}(x)}\right) p_{G^*}(x)\right) \mathrm{d}x && (3.10) \\ &= m \int_x p_{G^*}(x) \mathrm{d}x + m \int_x \mathbb{1}_{p_{data}(x) < p_{G^*}(x)} (p_{data}(x) - p_{G^*}(x)) \mathrm{d}x && (3.11) \\ &= m + m \int_x \mathbb{1}_{p_{data}(x) < p_{G^*}(x)} (p_{data}(x) - p_{G^*}(x)) \mathrm{d}x. && (3.12) \end{aligned}$$

The second term in equation 3.12 is non-positive, so $V(G^*, D^*) \leq m$.

By putting the ideal generator that generates p_{data} into the right side of equation 3.6, we get

$$\int_x p_{G^*}(x) D^*(x) \mathrm{d}x \leq \int_x p_{data}(x) D^*(x) \mathrm{d}x. \tag{3.13}$$

Thus by (3.8),
$$\int_x p_{G^*}(x) D^*(x) \mathrm{d}x + \int_x p_{G^*}(x) [m - D^*(x)]^+ \mathrm{d}x \leq V(G^*, D^*) \tag{3.14}$$

and since $D^*(x) \leq m$, we get $m \leq V(G^*, D^*)$.

Thus, $m \leq V(G^*, D^*) \leq m$ *i.e.* $V(G^*, D^*) = m$. Using equation 3.12, we see that can only happen if $\int_x \mathbb{1}_{p_{data}(x) < p_G(x)} \mathrm{d}x = 0$, which is true if and only if $p_G = p_{data}$ almost everywhere (this is because p_{data} and p_G are probabilities densities, see lemma 2 in the appendix 3.7 for details). □

Theorem 2. *A Nash equilibrium of this system exists and is characterized by (a) $p_{G^*} = p_{data}$ (almost everywhere) and (b) there exists a constant $\gamma \in [0, m]$ such that $D^*(x) = \gamma$ (almost everywhere).*[1].

Proof. See appendix 3.7. □

3.3.3 Using auto-encoders

In our experiments, the discriminator D is structured as an auto-encoder:

$$D(x) = ||Dec(Enc(x)) - x||. \tag{3.15}$$

The diagram of the EBGAN model with an auto-encoder discriminator is depicted in figure 3.1. The choice of the auto-encoders for D may seem arbitrary at the first glance, yet we postulate that it is conceptually more attractive than a

[1]This is assuming there is no region where $p_{data}(x) = 0$. If such a region exists, $D^*(x)$ may have any value in $[0, m]$ for x in this region.

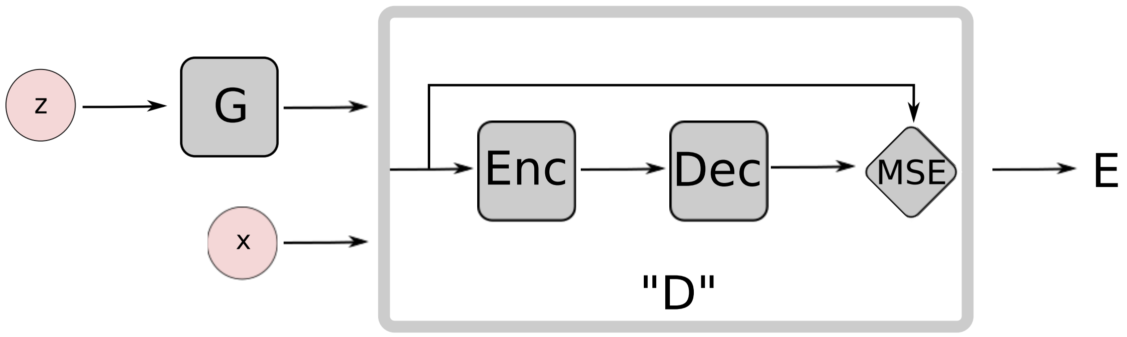

Figure 3.1: EBGAN architecture with an auto-encoder discriminator.

binary logistic network:

- Rather than using a single bit of target information to train the model, the reconstruction-based output offers a diverse targets for the discriminator. With the binary logistic loss, only two targets are possible, so within a minibatch, the gradients corresponding to different samples are most likely far from orthogonal. This leads to inefficient training, and reducing the minibatch sizes is often not an option on current hardware. On the other hand, the reconstruction loss will likely produce very different gradient directions within the minibatch, allowing for larger minibatch size without loss of efficiency.
- Auto-encoders have traditionally been used to represent energy-based model and arise naturally. When trained with some regularization terms (see section 3.3.3.1), auto-encoders have the ability to learn an energy manifold without supervision or negative examples. This means that even when an EBGAN auto-encoding model is trained to reconstruct a *real* sample, the discriminator contributes to discovering the data manifold by itself. To the contrary, without the presence of negative examples from the generator, a discriminator trained with binary logistic loss becomes pointless.

3.3.3.1 Connection to the regularized auto-encoders

One common issue in training auto-encoders is that the model may learn little more than an identity function, meaning that it attributes zero energy to the whole space. In order to avoid this problem, the model must be pushed to give higher energy to points outside the data manifold. Theoretical and experimental results have addressed this issue by regularizing the latent representations [Vincent et al., 2010; Rifai et al., 2011a; Marc'Aurelio Ranzato and Chopra, 2007; Kavukcuoglu et al., 2010c]. Such regularizers aim at restricting the reconstructing power of the auto-encoder so that it can only attribute low energy to a smaller portion of the input points.

We argue that the energy function (the discriminator) in the EBGAN framework is also seen as being regularized by having a generator producing the contrastive samples, to which the discriminator ought to give high reconstruction energies. We further argue that the EBGAN framework allows more flexibility from this perspective, because: (i)-the regularizer (generator) is fully trainable instead of being handcrafted; (ii)-the adversarial training paradigm enables a direct interaction between the duality of producing contrastive sample and learning the energy function.

3.3.4 Repelling regularizer

We propose a "repelling regularizer" which fits well into the EBGAN auto-encoder model, purposely keeping the model from producing samples that are clustered in one or only few modes of p_{data}. Another technique "minibatch discrimination" was developed by Salimans et al. [2016] from the same philosophy.

Implementing the repelling regularizer involves a Pulling-away Term (PT) that runs at a representation level. Formally, let $S \in \mathbb{R}^{s \times N}$ denotes a batch of sample representations taken from the encoder output layer. Let us define PT as:

$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left(\frac{S_i^\intercal S_j}{\|S_i\| \|S_j\|} \right)^2. \tag{3.16}$$

PT operates on a mini-batch and attempts to orthogonalize the pairwise sample representation. It is inspired by the prior work showing the representational power of the encoder in the auto-encoder alike model such as Rasmus et al. [2015] and Zhao et al. [2015]. The rationale for choosing the cosine similarity instead of Euclidean distance is to make the term bounded below and invariant to scale. We use the notation "EBGAN-PT" to refer to the EBGAN auto-encoder model trained with this term. Note the PT is used in the generator loss but not in the discriminator loss.

3.4 Related work

Our work primarily casts GANs into an energy-based model scope. On this direction, the approaches studying contrastive samples are relevant to EBGAN, such as the use of noisy samples [Vincent et al., 2010] and noisy gradient descent methods like contrastive divergence [Carreira-Perpinan and Hinton, 2005]. From the perspective of GANs, several papers were presented to improve the stability of GAN training, [Salimans et al., 2016; Denton et al., 2015; Radford et al., 2015; Im et al., 2016; Mathieu et al., 2015].

Kim and Bengio [2016] propose a probabilistic GAN and cast it into an energy-based density estimator by using the Gibbs distribution. Quite unlike EBGAN, this proposed framework doesn't get rid of the computational challenging partition function, so the choice of the energy function is required to be integratable.

3.5 Experiments

3.5.1 Exhaustive grid search on MNIST

In this section, we study the training stability of EBGANs over GANs on a simple task of MNIST digit generation with fully-connected networks. We run an exhaustive grid search over a set of architectural choices and hyper-parameters for both frameworks.

Formally, we specify the search grid in table 3.1. We impose the following restrictions on EBGAN models: (i)-using learning rate 0.001 and Adam [Kingma and Ba, 2014] for both G and D; (ii)-`nLayerD` represents the total number of layers combining *Enc* and *Dec*. For simplicity, we fix *Dec* to be one layer and only tune the *Enc* #layers; (iii)-the margin is set to 10 and not being tuned. To analyze the results, we use the *inception score* [Salimans et al., 2016] as a numerical means reflecting the generation quality. Some slight modification of the formulation were made to make figure 3.2 visually more approachable while maintaining the score's original meaning, $I' = E_x KL(p(y)||p(y|\mathbf{x}))$[2] Briefly, higher I' score implies better generation quality.

Histograms We plot the histogram of I' scores in figure 3.2. We further separated out the optimization related setting from GAN's grid (`optimD`, `optimG` and `lr`) and plot the histogram of each sub-grid individually, together with the

[2]This form of the "inception score" is only used to better analyze the grid search in the scope of this work, but not to compare with any other published work.

Table 3.1: Grid search specs

Settings	**Description**	**EBGANs**	**GANs**
`nLayerG`	number of layers in G	[2, 3, 4, 5]	[2, 3, 4, 5]
`nLayerD`	number of layers in D	[2, 3, 4, 5]	[2, 3, 4, 5]
`sizeG`	number of neurons in G	[400, 800, 1600, 3200]	[400, 800, 1600, 3200]
`sizeD`	number of neurons in D	[128, 256, 512, 1024]	[128, 256, 512, 1024]
`dropoutD`	if to use dropout in D	[true, false]	[true, false]
`optimD`	to use Adam or SGD for D	adam	[adam, sgd]
`optimG`	to use Adam or SGD for G	adam	[adam, sgd]
`lr`	learning rate	0.001	[0.01, 0.001, 0.0001]
#experiments:	-	**512**	**6144**

EBGAN I' scores as a reference, in figure 3.3. The number of experiments for GANs and EBGANs are both 512 in every subplot. The histograms evidently show that EBGANs are more reliably trained.

Digits generated from the configurations presenting the best inception score are shown in figure 3.4.

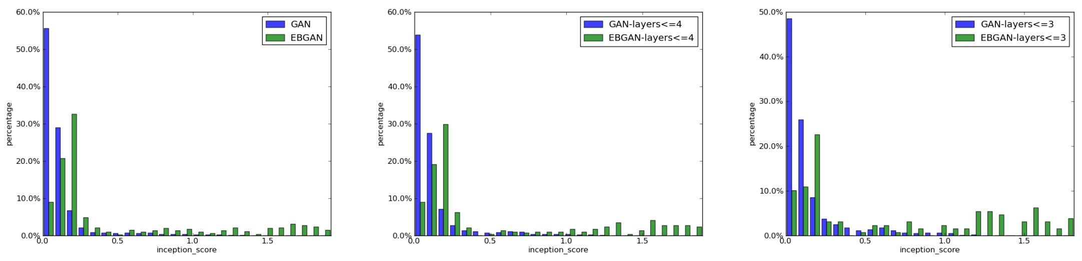

Figure 3.2: **(Zooming in on pdf file is recommended.)** Histogram of the inception scores from the grid search. The x-axis carries the inception score I and y-axis informs the portion of the models (in percentage) falling into certain bins. Left (a): general comparison of EBGANs against GANs; Middle (b): EBGANs and GANs both constrained by `nLayer[GD]<=4`; Right (c): EBGANs and GANs both constrained by `nLayer[GD]<=3`.

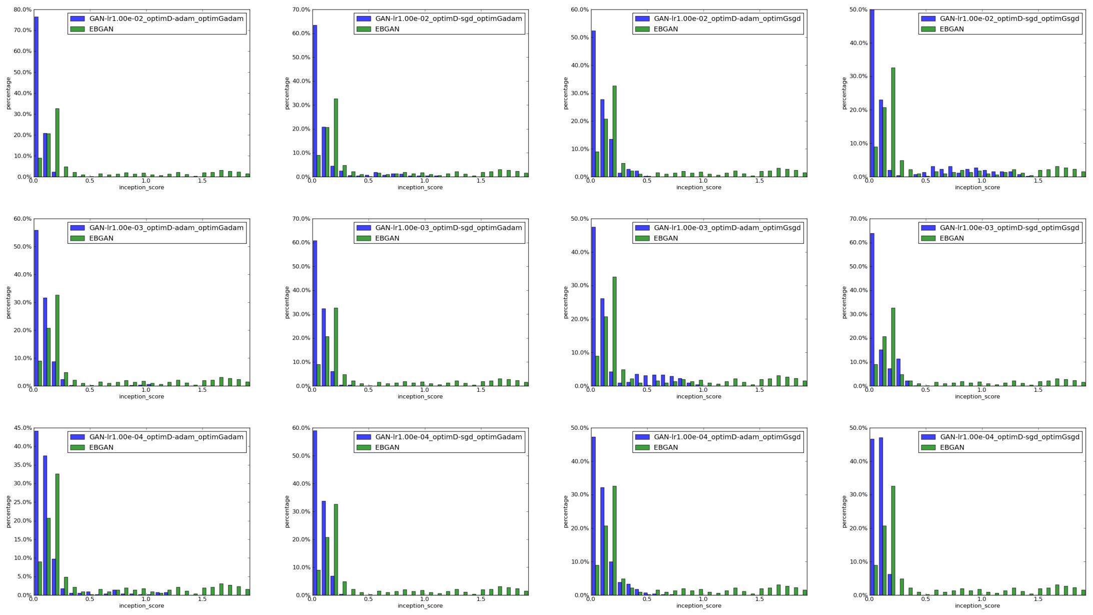

Figure 3.3: **(Zooming in on pdf file is recommended.)** Histogram of the inception scores grouped by different optimization combinations, drawn from `optimD`, `optimG` and `lr` (See text).

3.5.2 Semi-supervised learning on MNIST

We explore the potential of using the EBGAN framework for semi-supervised learning on permutation-invariant MNIST, collectively on using 100, 200 and 1000 labels. We utilized a bottom-layer-cost Ladder Network (LN) [Rasmus et al., 2015] with the EGBAN framework (EBGAN-LN). Ladder Network can be categorized as an energy-based model trained with multiple levels of reconstruction energies.

One crucial technique in enabling EBGAN framework for semi-supervised learning is the margin value m decay, in equation 3.1. The reason is to let the discriminator punish generator **less** when p_G gets closer to the data manifold. Due to that neural networks are continuous functions, we can think of the extreme case where the contrastive samples are exactly pinned on the data manifold. This is when these examples are not contrastive anymore, so the energy function should

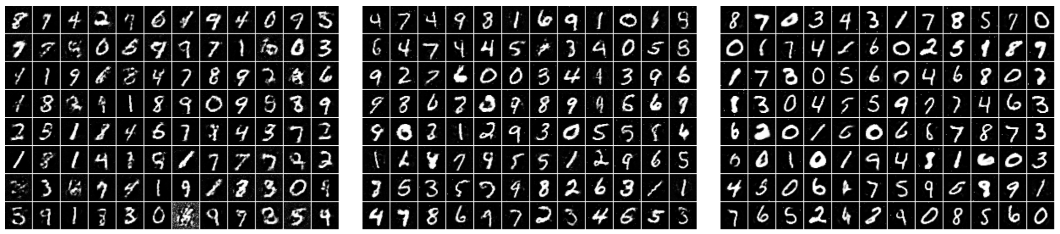

Figure 3.4: Generation from the grid search on MNIST. Left(a): Best GAN model; Middle(b): Best EBGAN model. Right(c): Best EBGAN-PT model.

not pull up their energies afterwards. With the margin decay scheduling, when $m = 0$, the EBGAN-LN model essentially falls back to be a normal Ladder Network. The undesirability of a non-decay dynamics for using the discriminator in the GAN or EBGAN framework is also indicated by Theorem 2: on convergence, the discriminator reflects a flat energy surface. However, through semi-supervised learning experments reported in table 3.2, we argue that the contrastive/negative examples the generative provides allows effective regularization on the discrimintor. In addition, the generated contrastive samples can be thought as an augmentation to the dataset.

Note that due to a discrepancy between the reported results between Rasmus et al. [2015] and Pezeshki et al. [2015], we report both results along with our own implementation of the Ladder Network of the same settings.

Table 3.2: The comparison of LN bottom-layer-cost model and its EBGAN extension on PI-MNIST semi-supervised task. Note the results are error rate (in %) and averaged over 15 different random seeds.

model	**100**	**200**	**1000**
LN reported in Pezeshki et al. [2015]	1.69±0.18	-	1.05±0.02
LN reported in Rasmus et al. [2015]	1.09±0.32	-	0.90±0.05
LN reproduced in this work	1.36±0.21	1.24±0.09	1.04±0.06
EBGAN-LN	**1.04±0.12**	**0.99±0.12**	**0.89±0.04**
Relative percentage improvement	23.5%	20.2%	14.4%

3.5.3 LSUN & CelebA

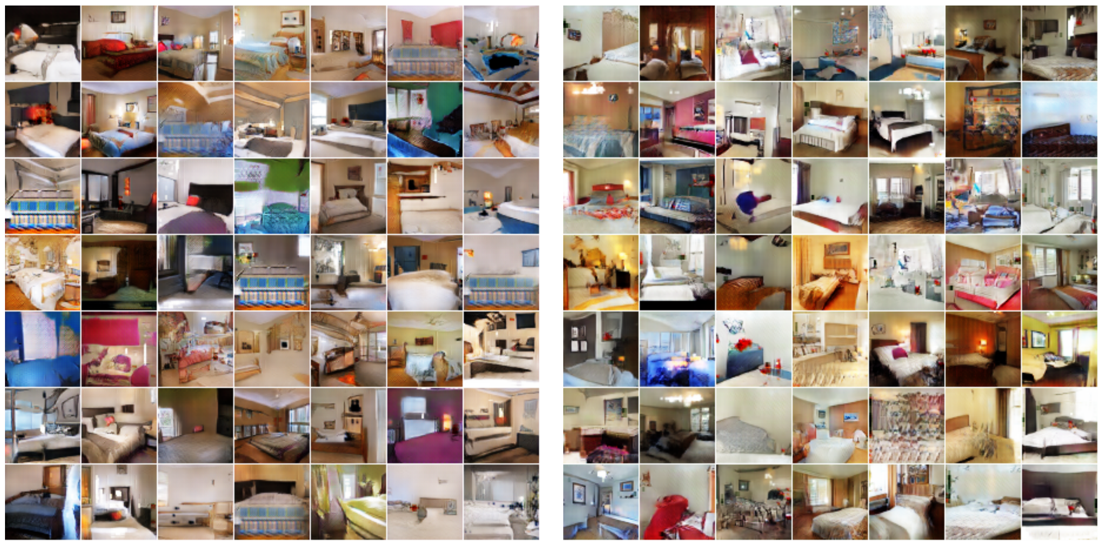

Figure 3.5: Generation from the LSUN bedroom dataset. Left(a): DCGAN generation. Right(b): EBGAN-PT generation.

We apply the EBGAN framework with deep convolutional architecture to generate 64×64 RGB images, a more realistic task, using the LSUN bedroom dataset [Yu et al., 2015] and the large-scale face dataset CelebA under alignment [Liu et al., 2015]. To compare EBGANs with DCGANs [Radford et al., 2015], we train a DCGAN model under the same configuration and show its generation side-by-side with the EBGAN model, in figures 3.5 and 3.6.

3.5.4 ImageNet

Finally, we trained EBGANs to generate high-resolution images on ImageNet [Russakovsky et al., 2015]. Compared with the datasets we have experimented so far, ImageNet presents an extensively larger and wilder space, so modeling the data distribution by a generative model becomes very challenging. We devised an

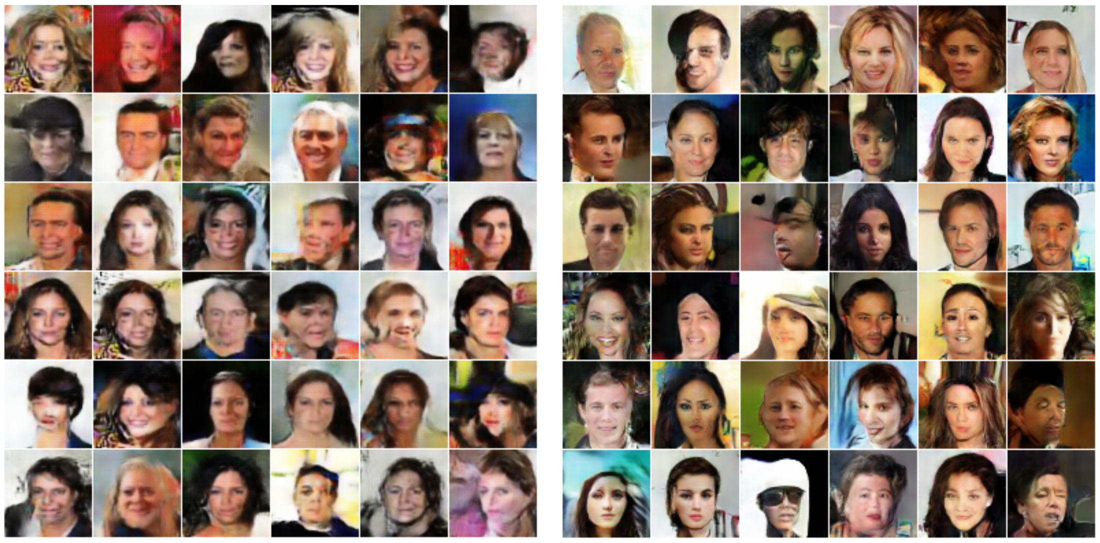

Figure 3.6: Generation from the CelebA dataset. Left(a): DCGAN generation. Right(b): EBGAN-PT generation.

experiment to generate 128 × 128 images, trained on the full ImageNet-1k dataset, which contains roughly 1.3 million images from 1000 different categories. We also trained a network to generate images of size 256 × 256, on a dog-breed subset of ImageNet, using the wordNet IDs provided by Vinyals et al. [2016]. The results are shown in figures 3.7 and 3.8. Despite the difficulty of generating images on a high-resolution level, we observe that EBGANs are able to learn about the fact that objects appear in the foreground, together with various background components resembling grass texture, sea under the horizon, mirrored mountain in the water, buildings, etc. In addition, our 256 × 256 dog-breed generations, although far from realistic, do reflect some knowledge about the appearances of dogs such as their body, furs and eye.

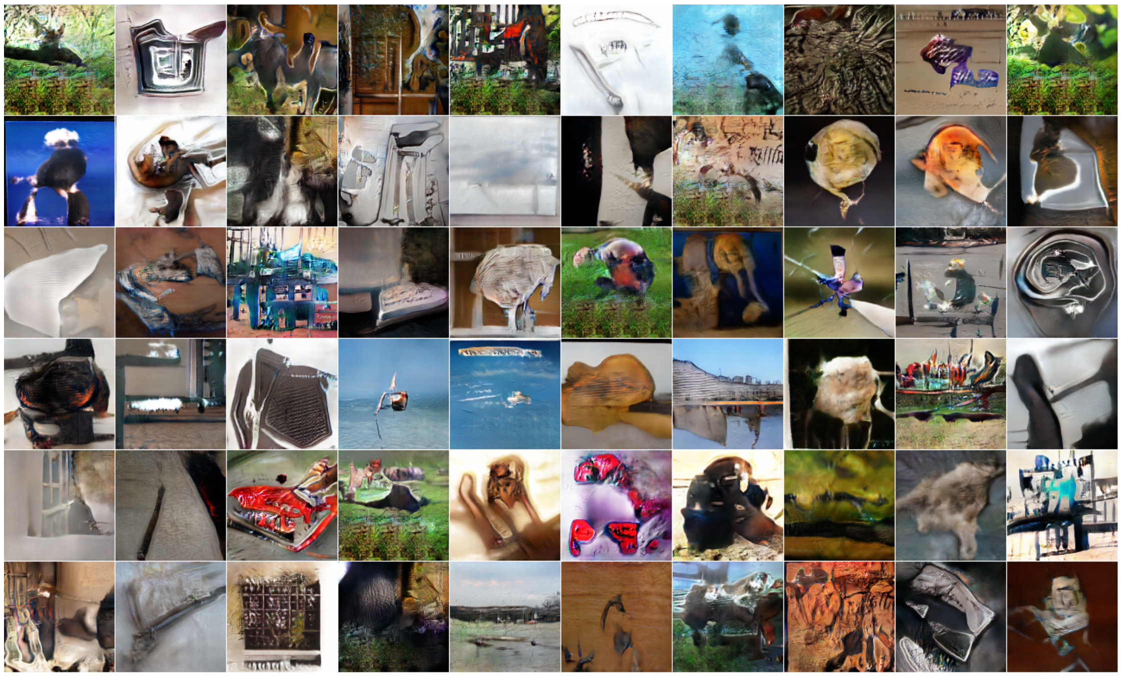

Figure 3.7: ImageNet 128 × 128 generations using an EBGAN-PT.

3.6 Outlook

We bridge two classes of unsupervised learning methods – GANs and auto-encoders – and revisit the GAN framework from an alternative energy-based perspective. EBGANs show better convergence pattern and scalability to generate high-resolution images. A family of energy-based loss functionals presented in LeCun et al. [2006] can easily be incorporated into the EBGAN framework. For the future work, the conditional setting [Denton et al., 2015; Mathieu et al., 2015] is a promising setup to explore. We hope the future research will raise more attention on a broader view of GANs from the energy-based perspective.

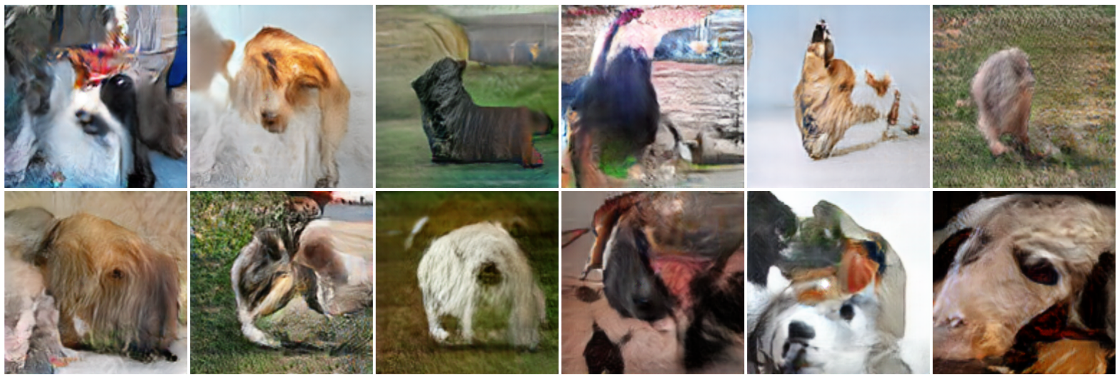
Figure 3.8: ImageNet 256×256 generations using an EBGAN-PT.

3.7 Technical points of section 3.3.2

Lemma 1. *Let $a, b \geq 0$, $\varphi(y) = ay + b\,[m - y]^+$. The minimum of φ on $[0, +\infty)$ exists and is reached in m if $a < b$, and it is reached in 0 otherwise (the minimum may not be unique).*

Proof. φ is defined on $[0, +\infty)$, its derivative is defined on $[0, +\infty)\backslash\{m\}$ and $\varphi'(y) = a - b$ if $y \in [0, m)$ and $\varphi'(y) = a$ if $y \in (m, +\infty)$.

So when $a < b$, the function is decreasing on $[0, m)$ and increasing on $(m, +\infty)$. Since it is continuous, it has a minimum in m. It may not be unique if $a = 0$ or $a - b = 0$.

On the other hand, if $a \geq b$ the function φ is increasing on $[0, +\infty)$, so 0 is a minimum. □

Lemma 2. *If p and q are probability densities, then $\int_x \mathbb{1}_{p(x)<q(x)}\mathrm{d}x = 0$ if and only if $\int_x \mathbb{1}_{p(x)\neq q(x)}\mathrm{d}x = 0$.*

Proof. Let's assume that $\int_x \mathbb{1}_{p(x)<q(x)}\mathrm{d}x = 0$. Then

$$\begin{aligned}
& \int_x \mathbb{1}_{p(x)>q(x)}(p(x)-q(x))\mathrm{d}x && (3.17)\\
= \; & \int_x (1-\mathbb{1}_{p(x)\leq q(x)})(p(x)-q(x))\mathrm{d}x && (3.18)\\
= \; & \int_x p(x)\mathrm{d}x - \int_x q(x)\mathrm{d}x + \int_x \mathbb{1}_{p(x)\leq q(x)}(p(x)-q(x))\mathrm{d}x && (3.19)\\
= \; & 1-1+\int_x \left(\mathbb{1}_{p(x)<q(x)} + \mathbb{1}_{p(x)=q(x)}\right)(p(x)-q(x))\mathrm{d}x && (3.20)\\
= \; & \int_x \mathbb{1}_{p(x)<q(x)}(p(x)-q(x))\mathrm{d}x + \int_x \mathbb{1}_{p(x)=q(x)}(p(x)-q(x))\mathrm{d}x && (3.21)\\
= \; & 0+0=0 && (3.22)
\end{aligned}$$

So $\int_x \mathbb{1}_{p(x)>q(x)}(p(x)-q(x))\mathrm{d}x = 0$ and since the term in the integral is always non-negative, $\mathbb{1}_{p(x)>q(x)}(p(x)-q(x)) = 0$ for almost all x. And $p(x)-q(x)=0$ implies $\mathbb{1}_{p(x)>q(x)} = 0$, so $\mathbb{1}_{p(x)>q(x)} = 0$ almost everywhere. Therefore $\int_x \mathbb{1}_{p(x)>q(x)}\mathrm{d}x = 0$ which completes the proof, given the hypothesis. □

Proof of theorem 2 The sufficient conditions are obvious. The necessary condition on G^* comes from theorem 1, and the necessary condition on $D^*(x) \leq m$ is from the proof of theorem 1.

Let us now assume that $D^*(x)$ is not constant almost everywhere and find a contradiction. If it is not, then there exists a constant C and a set $\mathcal{S}$ of non-zero measure such that $\forall x \in \mathcal{S}, D^*(x) \leq C$ and $\forall x \notin \mathcal{S}, D^*(X) > C$. In addition we can choose $\mathcal{S}$ such that there exists a subset $\mathcal{S}' \subset \mathcal{S}$ of non-zero measure such that $p_{data}(x) > 0$ on $\mathcal{S}'$ (because of the assumption in the footnote). We can build a generator G_0 such that $p_{G_0}(x) \leq p_{data}(x)$ over $\mathcal{S}$ and $p_{G_0}(x) < p_{data}(x)$ over $\mathcal{S}'$. We compute

$$
\begin{aligned}
U(G^*, D^*) - U(G_0, D^*) &= \int_x (p_{data} - p_{G_0}) D^*(x) \mathrm{d}x && (3.23)\\
&= \int_x (p_{data} - p_{G_0})(D^*(x) - C) \mathrm{d}x && (3.24)\\
&= \int_{\mathcal{S}} (p_{data} - p_{G_0})(D^*(x) - C) \mathrm{d}x + && (3.25)\\
&\quad \int_{\mathcal{R}^N \setminus \mathcal{S}} (p_{data} - p_{G_0})(D^*(x) - C) \mathrm{d}x && (3.26)\\
&> 0 && (3.27)
\end{aligned}
$$

which violates equation 3.6.

3.8 Setting a good margin value

It is crucial to set a proper energy margin value m in the framework of EBGAN, from both theoretical and experimental perspective. Here, we provide a few tips:

- Delving into the formulation of the discriminator loss made by equation 3.1, we suggest a numerical balance between its two terms which concern *real* and *fake* sample respectively. The second term is apparently bounded by $[0, m]$ (assuming the energy function $D(x)$ is non-negative). It is desirable to make the first term bounded in a similar range. In theory, the upper bound of the first term is essentially determined by (i)-the capacity of D; (ii)-the complexity of the dataset.
- In practice, for the EBGAN auto-encoder model, one can run D (the auto-encoder) alone on the real sample dataset and monitor the loss. When it converges, the consequential loss implies a rough limit on how well such

setting of D is capable to fit the dataset. This usually suggests a good start for a hyper-parameter searching on m.

- m being overly large results in a training instability/difficulty, while m being too small is prone to the mode-dropping problem. This property of m is depicted in figure 3.9.

Abstracting away from the practical experimental tips, the theoretical understanding of EBGAN in section 3.3.2 also provides some insight for setting a feasible m. For instance, as implied by Theorem 2, setting a large m results in a broader range of γ to which $D^*(x)$ may converge. Instability may come after an overly large γ because it generates two strong gradients pointing to opposite directions, from loss 3.1, which would perhaps demand more careful hyperparamter setting.

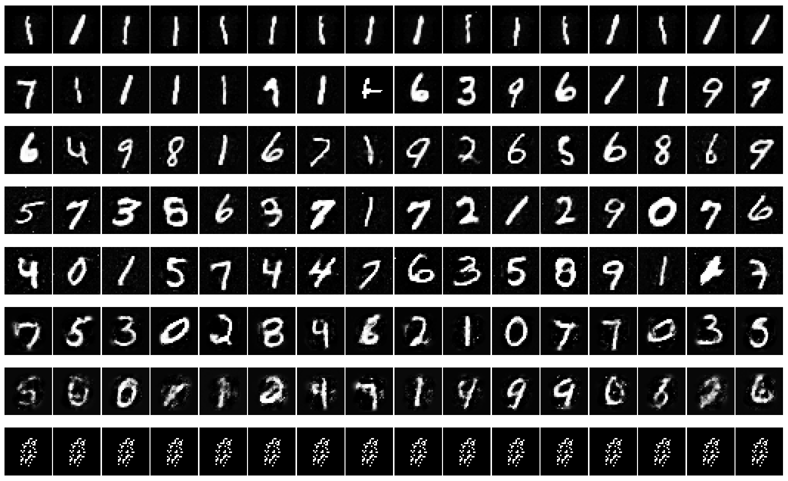

Figure 3.9: Generation from the EBGAN auto-encoder model trained with different m settings. From top to bottom, m is set to 1, 2, 4, 6, 8, 12, 16, 32 respectively. The rest setting is `nLayerG`=5, `nLayerD`=2, `sizeG`=1600, `sizeD`=1024, `dropoutD`=0, `optimD`=ADAM, `optimG`=ADAM, `lr`=0.001.

Chapter 4

Adversarially Regularized Autoencoder

While autoencoders are a key technique in representation learning for continuous structures, such as images or wave forms, developing general-purpose autoencoders for discrete structures, such as text sequence or discretized images, has proven to be more challenging. In particular, discrete inputs make it more difficult to learn a smooth encoder that preserves the complex local relationships in the input space. In this work, we propose an adversarially regularized autoencoder (ARAE) with the goal of learning more robust discrete-space representations. ARAE jointly trains both a rich discrete-space encoder, such as an RNN, and a simpler continuous space generator function, while using generative adversarial network (GAN) training to constrain the distributions to be similar. This method yields a smoother contracted code space that maps similar inputs to nearby codes, and also an implicit latent variable GAN model for generation. Experiments on text and discretized images demonstrate that the GAN model produces clean interpolations and captures the multimodality of the original space, and that the autoencoder produces improvements in semi-supervised learning as well as state-of-the-art results in unaligned text style transfer task using only a shared continuous-space representation.

4.1 Introduction

Recent work on regularized autoencoders, such as variational [Kingma and Welling, 2014; Rezende et al., 2014] and denoising [Vincent et al., 2008] variants, has shown significant progress in learning smooth representations of complex, high-dimensional continuous data such as images. These code-space representations facilitate the ability to apply smoother transformations in latent space in order to produce complex modifications of generated outputs, while still remaining on the data manifold.

Unfortunately, learning similar latent representations of discrete structures, such as text sequences or discretized images, remains a challenging problem. Initial work on VAEs for text has shown that optimization is difficult, as the decoder can easily degenerate into a unconditional language model [Bowman et al., 2015c]. Recent work on generative adversarial networks (GANs) for text has mostly focused on getting around the use of discrete structures either through policy gradient methods [Che et al., 2017; Hjelm et al., 2017; Yu et al., 2017] or with the Gumbel-Softmax distribution [Kusner and Hernandez-Lobato, 2016]. However, neither approach can yet produce robust representations directly.

A major difficulty of discrete autoencoders is mapping a discrete structure to a continuous code vector while also smoothly capturing the complex local relationships of the input space. Inspired by recent work combining pretrained autoencoders with deep latent variable models, we propose to target this issue with an adversarially regularized autoencoder (ARAE). Specifically we jointly train a discrete structure encoder and continuous space generator, while constraining the two models with a discriminator to agree in distribution. This approach allows us to utilize a complex encoder model, such as an RNN, and still constrain it with a very flexible, but more limited generator distribution. The full model can be then used as a smoother discrete structure autoencoder or as a latent variable GAN model where a sample can be decoded, with the same decoder, to a discrete output. Since the system produces a single continuous coded representation—in contrast to methods that act on each RNN state—it can easily be further regularized with problem-specific invariants, for instance to learn to ignore style, sentiment or other attributes for transfer tasks.

Experiments apply ARAE to discretized images and sentences, and demonstrate that the key properties of the model. Using the latent variable model (ARAE-GAN), the model is able to generate varied samples that can be quantitatively shown to cover the input spaces and to generate consistent image and sentence manipulations by moving around in the latent space via interpolation and offset vector arithmetic. Using the discrete encoder, the model can be used in a semi-supervised setting to give improvement in a sentence inference task. When the ARAE model is trained with task-specific adversarial regularization, the model improves the current best results on sentiment transfer reported in Shen et al. [2017] and produces compelling outputs on a topic transfer task using only a single shared code space. All outputs are listed in the Section **??** and code is available at *(removed for review)*.

4.2 Related Work

In practice unregularized autoencoders often learn a degenerate *identity* mapping where the latent code space is free of any structure, so it is necessary to apply some method of regularization. A popular approach is to regularize through an explicit prior on the code space and use a variational approximation to the posterior, leading to a family of models called variational autoencoders (VAE) [Kingma and Welling, 2014; Rezende et al., 2014]. Unfortunately VAEs for discrete text sequences can be challenging to train—for example, if the training procedure is not carefully tuned with techniques like word dropout and KL annealing [Bowman et al., 2015c], the decoder simply becomes a language model and ignores the latent code (although there has been some recent successes with convolutional models [Semeniuta et al., 2017; Yang et al., 2017b]). One possible reason for the difficulty in training VAEs is due to the strictness of the prior (usually a spherical Gaussian) and/or the parameterization of the posterior. There has been some work on making the prior/posterior more flexible through explicit parameterization [Rezende and Mohamed, 2015; Kingma et al., 2016; Chen et al., 2017b]. A notable technique is adversarial autoencoders (AAE) [Makhzani et al., 2015] which attempt to imbue the model with a more flexible prior implicitly through adversarial training. In AAE framework, the discriminator is trained to distinguish between samples from a fixed prior distribution and the input encoding, thereby pushing the code distribution to match the prior. While this adds more flexibility, it has similar issues for modeling text sequences and suffers from mode-collapse in our experiments. Our approach has similar motivation, but notably we do not sample from a fixed prior distribution—our 'prior' is instead parameterized through a flexible generator. Nonetheless, this view (which has been observed by various researchers [Tran et al., 2017; Mescheder et al., 2017; Makhzani and Frey, 2017]) provides an interesting connection between VAEs and GANs.

The success of GANs on images have led many researchers to consider applying GANs to discrete data such as text. Policy gradient methods are a natural way to deal with the resulting non-differentiable generator objective when training directly in discrete space [Glynn, 1987; Williams, 1992a]. When trained on text data however, such methods often require pre-training/co-training with a maximum likelihood (i.e. language modeling) objective [Che et al., 2017; Yu et al., 2017; Li et al., 2017]. This precludes there being a latent encoding of the sentence, and is also a potential disadvantage of existing language models (which can otherwise generate locally-coherent samples). Another direction of work has been through reparameterizing the categorical distribution with the Gumbel-Softmax trick [Jang et al., 2017; Maddison et al., 2017]—while initial experiments were encouraging on a synthetic task [Kusner and Hernandez-Lobato, 2016], scaling them to work on natural language is a challenging open problem. There has also been a flurry of recent, related approaches that work directly with the soft outputs from a generator [Gulrajani et al., 2017; Sai Rajeswar, 2017; Shen et al., 2017; Press et al., 2017]. For example, Shen et al. [Shen et al., 2017] exploits adversarial loss for unaligned style transfer between text by having the discriminator act on the RNN hidden states and using the soft outputs at each step as input to an RNN generator, utilizing the Professor-forcing framework [Lamb et al., 2016]. Our approach instead works entirely in code space and does not require utilizing RNN hidden states directly.

4.3 Background

Discrete Structure Autoencoders Define $\mathcal{X} = \mathcal{V}^n$ to be a set of discrete structures where $\mathcal{V}$ is a vocabulary of symbols and $\mathbb{P}_x$ to be a distribution over this space. For instance, for binarized images $\mathcal{V} = \{0, 1\}$ and n is the number of pixels, while for sentences $\mathcal{V}$ is the vocabulary and n is the sentence length. A discrete autoencoder consists of two parameterized functions: a deterministic encoder function $\text{enc}_\phi : \mathcal{X} \mapsto \mathcal{C}$ with parameters ϕ that maps from input to code space and a conditional decoder distribution $p_\psi(\mathbf{x} \mid \mathbf{c})$ over structures $\mathcal{X}$ with parameters ψ. The parameters are trained on a cross-entropy reconstruction loss:

$$\mathcal{L}_{\text{rec}}(\phi, \psi) = -\log p_\psi(\mathbf{x} \mid \text{enc}_\phi(\mathbf{x}))$$

The choice of the encoder and decoder parameterization is specific to the structure of interest, for example we use RNNs for sequences. We use the notation, $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p_\psi(\mathbf{x} \mid \text{enc}_\phi(\mathbf{x}))$ for the (approximate) decoder mode. When $\mathbf{x} = \hat{\mathbf{x}}$ the autoencoder is said to perfectly reconstruct $\mathbf{x}$.

Generative Adversarial Networks GANs are a class of parameterized implicit generative models [Goodfellow et al., 2014]. The method approximates drawing samples from a true distribution $\mathbf{c} \sim \mathbb{P}_r$ by instead employing a latent variable $\mathbf{z}$ and a parameterized deterministic generator function $\tilde{\mathbf{c}} = g_\theta(\mathbf{z})$ to produce samples $\tilde{\mathbf{c}} \sim \mathbb{P}_g$. Initial work on GANs minimizes the Jensen-Shannon divergence between the distributions. Recent work on Wasserstein GAN (WGAN) [Arjovsky et al., 2017], replaces this with the *Earth-Mover* (Wasserstein-1) distance.

GAN training utilizes two separate models: a *generator* $g_\theta(\mathbf{z})$ maps a latent vector from some easy-to-sample source distribution to a sample and a critic/discriminator $f_w(\mathbf{c})$ aims to distinguish *real* data and *generated* samples from g_θ. Informally, the generator is trained to fool the critic, and the critic to tell real from generated. WGAN training uses the following min-max optimization over generator parameters θ and critic parameters w,

$$\min_{\theta} \max_{w \in \mathcal{W}} \mathbb{E}_{\mathbf{c} \sim \mathbb{P}_r}[f_w(\mathbf{c})] - \mathbb{E}_{\tilde{\mathbf{c}} \sim \mathbb{P}_g}[f_w(\tilde{\mathbf{c}})], \tag{4.1}$$

where $f_w : \mathcal{C} \mapsto \mathbb{R}$ denotes the critic function, $\tilde{\mathbf{c}}$ is obtained from the generator, $\tilde{\mathbf{c}} = g_\theta(\mathbf{z})$, and $\mathbb{P}_r$ and $\mathbb{P}_g$ are real and generated distributions. If the critic parameters w are restricted to an 1-Lipschitz function set $\mathcal{W}$, this term correspond to minimizing Wasserstein-1 distance $W(\mathbb{P}_r, \mathbb{P}_g)$. We use a naive approximation to enforce this property by weight-clipping, i.e. $w = [-\epsilon, \epsilon]^d$ [Arjovsky et al., 2017].

4.4 Model: Adversarially Regularized Autoencoder

Ideally, a discrete autoencoder should be able to *reconstruct* x from c, but also *smoothly* assign similar codes c and c' to similar x and x'. For continuous autoencoders, this property can be enforced directly through explicit regularization. For instance, contractive autoencoders [Rifai et al., 2011b] regularize their loss by the functional smoothness of enc_ϕ. However, this criteria does not apply when inputs are discrete and we lack even a metric on the input space. How can we enforce that similar discrete structures map to nearby codes?

Adversarially regularized autoencoders target this issue by learning a parallel continuous-space generator with a restricted functional form to act as a smoother reference encoding. The joint objective regularizes the autoencoder to constrain the discrete encoder to agree in distribution with its continuous counterpart:

$$\min_{\phi,\psi,\theta} \quad \mathcal{L}_{\text{rec}}(\phi,\psi) + \lambda^{(1)} W(\mathbb{P}_r, \mathbb{P}_g)$$

Above W is the Wasserstein-1 distance between $\mathbb{P}_r$ the distribution of codes from the discrete encoder model ($\text{enc}_\phi(x)$ where $x \sim \mathbb{P}(x)$) and $\mathbb{P}_g$ is the distribution of codes from the continuous generator model ($g_\theta(z)$ for some z, e.g. $z \sim \mathcal{N}(0, I)$). To approximate Wasserstein-1 term, the W function includes an embedded critic function which is optimized adversarially to the encoder and generator as described in the background. The full model is shown in Figure 4.1.

To train the model, we use a block coordinate descent to alternate between optimizing different parts of the model: (1) the encoder and decoder to minimize reconstruction loss, (2) the WGAN critic function to approximate the W term, (3) the encoder and generator to adversarially fool the critic to minimize W:

$$\begin{aligned}
&1) \min_{\phi,\psi} && \mathcal{L}_{\text{rec}}(\phi,\psi) \\
&2) \min_{w\in\mathcal{W}} & \mathcal{L}_{\text{cri}}(w) = & \max_{w\in\mathcal{W}} \quad \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_x}\left[f_w(\text{enc}_\phi(\mathbf{x}))\right] - \mathbb{E}_{\tilde{\mathbf{c}}\sim\mathbb{P}_g}\left[f_w(\tilde{\mathbf{c}})\right] \\
&3) \min_{\phi,\theta} & \mathcal{L}_{\text{encs}}(\phi,\theta) = & \min_{\phi,\theta} \quad \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_x}\left[f_w(\text{enc}_\phi(\mathbf{x}))\right] - \mathbb{E}_{\tilde{\mathbf{c}}\sim\mathbb{P}_g}\left[f_w(\tilde{\mathbf{c}})\right]
\end{aligned}$$

The full training algorithm is shown in Algorithm 1.

Extension: Code Space Transfer One benefit of the ARAE framework is that it compresses the input to a single code vector. This framework makes it ideal

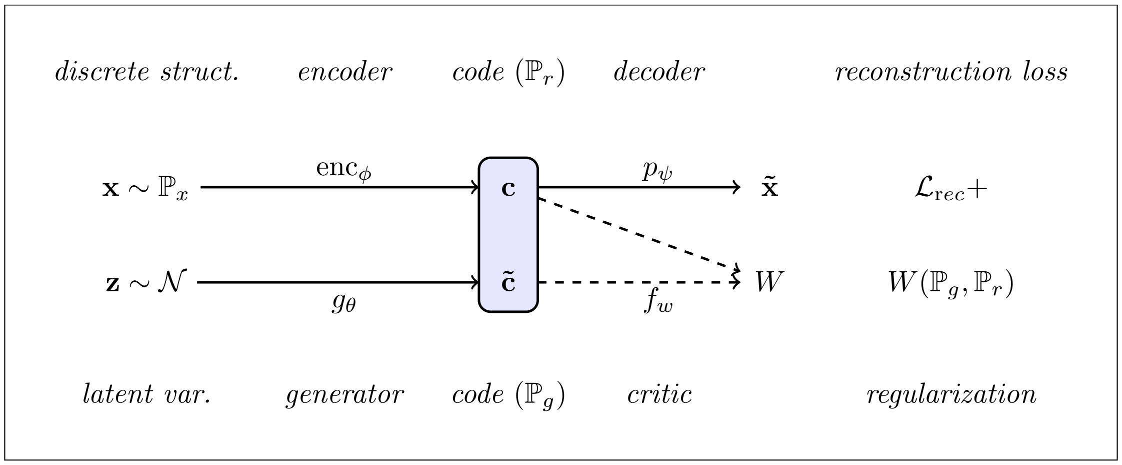

Figure 4.1: ARAE architecture. The model can be used as an autoencoder, where a structure $\mathbf{x}$ is encoded and decoded to produce $\hat{\mathbf{x}}$, and as a GAN (ARAE-GAN), where a sample $\mathbf{z}$ is passed though a generator g_θ to produce a code vector, which is similarly decoded to $\tilde{\mathbf{x}}$. The critic function f_w is only used at training to help approximate W.

for manipulating discrete objects while in continuous code space. For example, consider the problem of unaligned transfer, where we want to change an attribute of a discrete input without supervised examples, e.g. to change the topic or sentiment of a sentence. First, we extend the decoder to condition on a transfer variable denoting this attribute $\mathbf{y}$ which is known during training, to learn $p_\psi(\mathbf{x} \mid \mathbf{c}, y)$. Next, we train the code space to be invariant to this attribute, to force it to be learned fully by the decoder. Specifically, we further regularize the code space to map similar x with different attribute labels y near enough to fool a code space attribute classifier, i.e.:

$$\min_{\phi,\psi,\theta} \quad \mathcal{L}_{\text{rec}}(\phi, \psi) + \lambda^{(1)} W(\mathbb{P}_r, \mathbb{P}_g) - \lambda^{(2)} \mathcal{L}_{\text{class}}(\phi, u)$$

where $\mathcal{L}_{\text{class}}(\phi, u)$ is the loss of a classifier $p_u(y \mid \mathbf{c})$ from code space to labels (in our experiments we always set $\lambda^{(2)} = 1$). To incorporate this additional

Algorithm 1 ARAE Training

```
for number of training iterations do
    (1) Train the autoencoder for reconstruction [$\mathcal{L}_{\text{rec}}(\phi, \psi)$].
        Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$ and compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$.
        Backpropagate reconstruction loss, $\mathcal{L}_{\text{rec}} = -\frac{1}{m}\sum_{i=1}^m \log p_\psi(\mathbf{x}^{(i)} \mid \mathbf{c}^{(i)}, [\mathbf{y}^{(i)}])$, and update.
    (2) Train the critic [$\mathcal{L}_{\text{cri}}(w)$] (Repeat k times)
        Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$ and $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim \mathcal{N}(0, \mathbf{I})$.
        Compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{c}}^{(i)} = g_\theta(\mathbf{z}^{(i)})$.
        Backpropagate loss $-\frac{1}{m}\sum_{i=1}^m f_w(\mathbf{c}^{(i)}) + \frac{1}{m}\sum_{i=1}^m f_w(\tilde{\mathbf{c}}^{(i)})$, update, clip the critic $w$ to $[-\epsilon, \epsilon]^d$.
    (3) Train the generator and encoder adversarially to critic [$\mathcal{L}_{\text{encs}}(\phi, \theta)$]
        Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$ and $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim \mathcal{N}(0, \mathbf{I})$
        Compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{c}}^{(i)} = g_\theta(\mathbf{z}^{(i)})$.
        Backpropagate adversarial loss $\frac{1}{m}\sum_{i=1}^m f_w(\mathbf{c}^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(\tilde{\mathbf{c}}^{(i)})$ and update.
```

regularization, we simply add two more gradient update steps: (2b) training a classifier to discriminate codes, and (3b) adversarially training the encoder to fool this classifier. The algorithm is shown in Algorithm 2. Note that similar technique has been introduced in other domains, notably in images [Lampe et al., 2017] and video modeling [Denton and Birodkar, 2017].

Algorithm 2 ARAE Transfer Extension

```
[Each loop additionally:]
(2b) Train the code classifier [$\min_u \mathcal{L}_{\text{class}}(\phi, u)$]
    Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$, lookup $y^{(i)}$, and compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$.
    Backpropagate loss $-\frac{1}{m}\sum_{i=1}^m \log p_u(y^{(i)}|\mathbf{c}^{(i)})$, update.
(3b) Train the encoder adversarially to code classifier [$\max_\phi \mathcal{L}_{\text{class}}(\phi, u)$]
    Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$, lookup $y^{(i)}$, and compute code-vectors $\mathbf{c}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$.
    Backpropagate adversarial classifier loss $-\frac{1}{m}\sum_{i=1}^m \log p_u(1 - y^{(i)} \mid \mathbf{c}^{(i)})$, update.
```

4.5 Methods and Architectures

We experiment with three different ARAE models: (1) an autoencoder for discretized images trained on the binarized version of MNIST, (2) an autoencoder for text sequences trained using the Stanford Natural Language Inference (SNLI) corpus [Bowman et al., 2015b], and (3) an autoencoder trained for text transfer (Section 4.6.2) based on the Yelp and Yahoo datasets for unaligned sentiment and topic transfer. All three models utilize the same generator architecture, g_θ. The generator architecture uses a low dimensional $\mathbf{z}$ with a Gaussian prior $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$, and maps it to $\mathbf{c}$. Both the critic f_w and the generator g_θ are parameterized as feed-forward MLPs.

The *image* model uses fully-connected NN to autoencode binarized images. Here $\mathcal{X} = \{0,1\}^n$ where n is the image size. The encoder used is a feed-forward MLP network mapping from $\{0,1\}^n \mapsto \mathbb{R}^m$, $\text{enc}_\phi(\mathbf{x}) = \text{MLP}(\mathbf{x}; \phi) = \mathbf{c}$. The decoder predicts each pixel in $\mathbf{x}$ as a parameterized logistic regression, $p_\psi(\mathbf{x} \mid \mathbf{c}) = \prod_{j=1}^{n} \sigma(\mathbf{h})^{x_j}(1 - \sigma(\mathbf{h}))^{1-x_j}$ where $\mathbf{h} = \text{MLP}(\mathbf{c}; \psi)$.

The *text* model uses a recurrent neural network (RNN) for both the encoder and decoder. Here $\mathcal{X} = \mathcal{V}^n$ where n is the sentence length and $\mathcal{V}$ is the vocabulary of the underlying language. Define an RNN as a parameterized recurrent function $\mathbf{h}_j = \text{RNN}(x_j, \mathbf{h}_{j-1}; \phi)$ for $j = 1 \dots n$ (with $\mathbf{h}_0 = \mathbf{0}$) that maps a discrete input structure $\mathbf{x}$ to hidden vectors $\mathbf{h}_1 \dots \mathbf{h}_n$. For the encoder, we define $\text{enc}_\phi(\mathbf{x}) = \mathbf{h}_n = \mathbf{c}$. For decoding we feed $\mathbf{c}$ as an additional input to the decoder RNN at each time step, i.e. $\tilde{\mathbf{h}}_j = \text{RNN}(x_j, \tilde{\mathbf{h}}_{j-1}, \mathbf{c}; \psi)$, and further calculate the distribution over $\mathcal{V}$ at each time step via softmax, $p_\psi(\mathbf{x} \mid \mathbf{c}) = \prod_{j=1}^{n} \text{softmax}(\mathbf{W}\tilde{\mathbf{h}}_j + \mathbf{b})_{x_j}$ where $\mathbf{W}$ and $\mathbf{b}$ are parameters (part of ψ). Finding the most likely sequence $\tilde{\mathbf{x}}$ under this distribution is intractable, but it is possible to approximate it using greedy search or beam search. In our experiments we use an LSTM architecture [Hochreiter and Schmidhuber, 1997] for both the encoder/decoder and decode using greedy search. The *text transfer* model uses the same architecture as the text model but extends it with a code space classifier $p(y|\mathbf{c})$ which is modeled using an MLP and trained to minimize cross-entropy.

Our baselines utilize a standard autoencoder (AE) and the cross-aligned autoencoder [Shen et al., 2017] for transfer. Note that in both our ARAE and standard AE experiments, the encoded code from the encoder is normalized to lie on the unit sphere, and the generated code is bounded to lie in $(-1, 1)^n$ by the tanh function at output layer. We additionally experimented with the sequence VAE introduced by Bowman et al. [2015c] and the adversarial autoencoder (AAE) model [Makhzani et al., 2015] on the SNLI dataset. However despite extensive parameter tuning we found that neither model was able to learn meaningful latent representations—the VAE simply ignored the latent code and the AAE experienced mode-collapse and repeatedly generated the same samples. The Section **??** includes detailed descriptions of the hyperparameters, model architecture, and training regimes.

4.6 Experiments

Our experiments consider three aspects of the model. First we measure the empirical impact of regularization on the autoencoder. Next we apply the discrete autoencoder to two applications, unaligned style transfer and semi-supervised learning. Finally we employ the learned generator network as an implicit latent variable model (ARAE-GAN) over discrete sequences.

4.6.1 Impact of Regularization on Discrete Encoding

Our main goal for ARAE is to regularize the model produce a smoother encoder by requiring the distribution from the encoder to match the distribution from the continuous generator over a simple latent variable. To examine this claim we

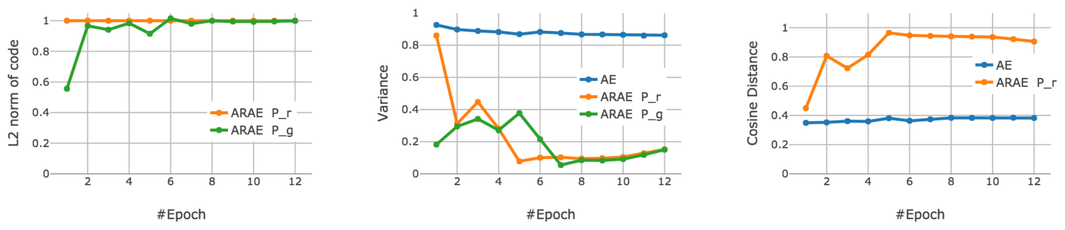

Figure 4.2: Left: $\ell 2$ norm of encoder code $\mathbf{c}$ and generator code $\tilde{\mathbf{c}}$ during ARAE training. The encoder $\mathbf{c}$ is normalized by the model, whereas the generator learns to match this as training progresses. Middle: Sum of the dimension-wise variances of the encoder codes $\mathbb{P}_r$ and generator codes $\mathbb{P}_g$ compared to that of the standard AE. Right: Average cosine similarity of nearby sentences (edit-distance wise) for the ARAE and AE.

consider two basic statistical properties of the code space during training of the text model on SNLI, shown in Figure 4.2. On the left, we see that the $\ell 2$ norm of $\mathbf{c}$ and code $\tilde{\mathbf{c}}$ converge quickly in ARAE training. The encoder code is always restricted to be on the unit sphere, and the generated code $\tilde{\mathbf{c}}$ quickly learns to match it. The middle plot shows the convergence of the trace of the covariance matrix between the generator and the encoder as training progresses. We find that variance of the encoder and the generator match after several epochs. To check the smoothness of the model, for both ARAE/AE, we take a sentence and calculate the average cosine similarity of 100 randomly-selected sentences that had an edit-distance of at most 5 to the original sentence. We do this for 250 sentences and calculate the mean of the average cosine similarity. Figure 2 (right) shows that the cosine similarity of nearby sentences is quite high for the ARAE than in the case for the AE. Edit-distance is not an ideal proxy for similarity in sentences, but it is often a sufficient condition.

Finally an ideal representation should be robust to small changes of the input around the training examples in code space [Rifai et al., 2011b]. We can test this property by feeding a noised input to the encoder and (i) calculating the score given to the original input, and (ii) checking the reconstructions. Table 4.1 (right) shows an experiment for text where we add noise by permuting k words in each sentence. We observe that the ARAE is able to map a noised sentence to a natural sentence, (though not necessarily the denoised sentence). Table 4.1 (left) shows empirical results for these experiments. We obtain the reconstruction error (i.e. negative log likelihood) of the original (non-noised) sentence under the decoder, utilizing the noised code. We find that when $k = 0$ (i.e. no swaps), the regular AE better reconstructs the input as expected. However, as we increase the number of swaps and push the input further away from the data manifold, the ARAE is more likely to produce the original sentence. We note that unlike denoising autoencoders which require a domain-specific noising function [Hill et al., 2016; Vincent et al., 2008], the ARAE is not explicitly trained to denoise an input, but learns to do so as a byproduct of adversarial regularization.

4.6.2 Applications of Discrete Autoencoder

Unaligned Text Transfer A smooth autoencoder combined with low reconstruction error should make it possible to more robustly manipulate discrete objects through code space without dropping off the data manifold. To test this hypothesis, we experimented with two unaligned text transfer tasks. For these tasks, we attempt to change one attribute of a sentence without aligned examples of this change. To perform this transfer, we learn a code space that can represent an input that is agnostic to this attribute, and a decoder that can incorporate the attribute (as described in Section 4.4). We experiment with unaligned transfer of sentiment on the Yelp corpus and topic on the Yahoo corpus [Zhang et al., 2015].

For sentiment we follow the same setup as Shen et al. [2017] and split the Yelp corpus into two sets of unaligned positive and negative reviews. We train an ARAE

k	AE	ARAE
0	1.06	2.19
1	4.51	4.07
2	6.61	5.39
3	9.14	6.86
4	9.97	7.47

Original	A woman wearing sunglasses .
Noised	A woman sunglasses wearing .
AE	A woman sunglasses wearing sunglasses .
ARAE	A woman wearing sunglasses .
Original	Pets galloping down the street .
Noised	Pets down the galloping street .
AE	Pets riding the down galloping .
ARAE	Pets congregate down the street near a ravine .
Original	They have been swimming .
Noised	been have They swimming .
AE	been have been swimming .
ARAE	Children have been swimming .
Original	The child is sleeping .
Noised	child The is sleeping .
AE	The child is sleeping is .
ARAE	The child is sleeping .

Table 4.1: **Top**. Reconstruction error (negative log-likelihood averaged over sentences) of the original sentence from a corrupted sentence. Here k is the number of swaps performed on the original sentence. **Bottom**. Samples generated from AE and ARAE where the input is noised by swapping words.

as an autoencoder with two separate decoders, one for positive and one for negative sentiment, and incorporate adversarial training of the encoder to remove sentiment information from the code space. We test by encoding in sentences of one class and decoding, greedily, with the opposite decoder.

Our evaluation is based on four automatic metrics, shown in Table 4.2: (i) Transfer: measuring how successful the model is at transferring sentiment based on an automatic classifier (we use the `fastText` library [Joulin et al., 2016]). (ii) BLEU: measuring the consistency between the transferred text and the original. We expect the model to maintain as much information as possible and transfer only the style; (iii) Perplexity: measuring the fluency of the generated text; (iv) Reverse Perplexity: measuring the extent to which the generations are representative of the underlying data distribution.[1] Both perplexity numbers are obtained by training an RNN language model.

We additionally perform human evaluations on the cross-aligned AE and our best ARAE model. We randomly select 1000 sentences (500/500 positive/negative), obtain the corresponding transfers from both models, and ask Amazon Mechanical Turkers to evaluate the sentiment (Positive/Neutral/Negative) and naturalness (1-5, 5 being most natural) of the transferred sentences. We create a separate task in which we show the Turkers the original and the transferred sentences, and ask them to evaluate the similarity based on sentence structure (1-5, 5 being most similar). We explicitly ask the Turkers to disregard sentiment in their similarity assessment.

In addition to comparing against the cross-aligned AE of Shen et al. [2017],

[1]This *reverse perplexity* is calculated by training a language model on the generated data and measuring perplexity on held-out, real data (i.e. reverse of regular perplexity). We also found this metric to be helpful for early-stopping based on validation data.

	Automatic Evaluation				Human Evaluation		
Model	Transfer	BLEU	PPL	Reverse PPL	Transfer	Similarity	Naturalness
Cross-Aligned AE	77.1%	17.75	65.9	124.2	57%	3.8	2.7
AE	59.3%	37.28	31.9	68.9	-	-	-
ARAE, $\lambda_a^{(1)}$	73.4%	31.15	29.7	70.1	-	-	-
ARAE, $\lambda_b^{(1)}$	81.8%	20.18	27.7	77.0	74%	3.7	3.8

Table 4.2: Experiments on sentiment transfer. Left shows the automatic metrics (Transfer/BLEU/PPL/Reverse PPL) while right shows human evaluation metrics (Transfer/Similarity/Naturalness). Cross-Aligned AE is from Shen et al. [2017]

we also conduct an ablation study by comparing against an AE trained without the adversarial regularization. For ARAE, we experimented with different $\lambda^{(1)}$ weighting on the adversarial loss (see section 4) with $\lambda_a^{(1)} = 1, \lambda_b^{(1)} = 10$. We generally set $\lambda^{(2)} = 1$ which weights the code classifier component. Experimentally the adversarial regularization enhances transfer and perplexity, but tends to make the transferred text less similar to the original, compared to the AE. Some randomly selected sentences are shown in figure 4.3 and more samples are shown available in Section **??**.

The same method can be applied to other style transfer tasks, for instance the more challenging Yahoo QA data [Zhang et al., 2015]. For Yahoo we chose 3 relatively distinct topic classes for transfer: Science & Math, Entertainment & Music, and Politics & Government. As the dataset contains both questions and answers, we separated our experiments into titles (questions) and replies (answers). The qualitative results are showed in table 4.4. See Section **??** for additional generation examples.

Semi-Supervised Training We further utilize ARAE in a standard AE setup for semi-supervised training. We experiment on a natural language inference task, shown in Table 4.5 (right). We use 22.2%, 10.8% and 5.25% of the original labeled training data, and use the rest of the training set for unlabeled training. The

	Positive ⇒ Negative
	great indoor mall .
ARAE	no smoking mall .
Cross-AE	terrible outdoor urine .
	it has a great atmosphere , with wonderful service .
ARAE	it has no taste , with a complete jerk .
Cross-AE	it has a great horrible food and run out service .
	we came on the recommendation of a bell boy and the food was amazing .
ARAE	we came on the recommendation and the food was a joke .
Cross-AE	we went on the car of the time and the chicken was awful .
	Negative ⇒ Positive
	hell no !
ARAE	hell great !
Cross-AE	incredible pork !
	small , smokey , dark and rude management .
ARAE	small , intimate , and cozy friendly staff .
Cross-AE	great , , , chips and wine .
	the people who ordered off the menu did n't seem to do much better .
ARAE	the people who work there are super friendly and the menu is good .
Cross-AE	the place , one of the office is always worth you do a business .

Table 4.3: Sentiment transfer results. Original sentence and transferred output (from ARAE and the Cross-Aligned AE) of 6 randomly-drawn examples.

labeled set is randomly picked. The full SNLI training set contains 543k sentence pairs, and we use supervised sets of 120k, 59k and 28k sentence pairs respectively for the three settings. As a baseline we use an AE trained on the additional data, similar to the setting explored in Dai and Le [2015]. For ARAE we use the subset of unsupervised data of length < 15, which roughly includes 655k single sentences (due to the length restriction, this is a subset of 715k sentences that were used for AE training). As observed by Dai and Le [2015], training on unlabeled data with an AE objective improves upon a model just trained on labeled data. Training with adversarial regularization provides further gains.

4.6.3 A Latent Variable Model for Discrete Structures

After training, an ARAE can also be used as an implicit latent variable model controlled by $\mathbf{z}$ and the generator g_θ, which we refer to as ARAE-GAN. While models of this form have been widely used for generation in other modalities, they have been less effective for discrete structures. In this section, we attempt to measure the effectiveness of this induced discrete GAN.

A common test for a GANs ability mimic the true distribution $\mathbb{P}_r$ is to train a simple model on generated samples from $\mathbb{P}_g$. While there are pitfalls of this evaluation [Theis et al., 2016], it provides a starting point for text modeling. Here we generate 100k samples from (i) ARAE-GAN, (ii) an AE[2], (iii) a RNN LM trained on the same data, and (iv) the real training set (samples from the models are shown in section 4.7). All models are of the same size to allow for fair comparison. We train an RNN language model on generated samples and evaluate on held-out data to calculate the *reverse perplexity*. As can be seen from Table 4.5, training on real data (understandably) outperforms training on generated data by a large margin. Surprisingly however, we find that a language model trained on ARAE-GAN data performs slightly better than one trained on LM-generated/AE-generated data. We further found that the reverse PPL of an AAE [Makhzani et al., 2015] was quite high (980) due to mode-collapse.

Another property of GANs (and VAEs) is that the Gaussian form of $\mathbf{z}$ induces the ability to smoothly interpolate between outputs by exploiting the structure of the latent space. While language models may provide a better estimate of the underlying probability space, constructing this style of interpolation would

[2]To "sample" from an AE we fit a multivariate Gaussian to the code space after training and generate code vectors from this Gaussian to decode back into sentence space.

A man is on the corner in a sport area .
A man is on corner in a road all .
A lady is on outside a racetrack .
A lady is outside on a racetrack .
A lot of people is outdoors in an urban setting .
A lot of people is outdoors in an urban setting .
A lot of people is outdoors in an urban setting .

A man in a cave is used an escalator .

A man in a cave is used an escalator
A man in a cave is used chairs .
A man in a number is used many equipment
A man in a number is posing so on a big rock .
People are posing in a rural area .
People are posing in a rural area.

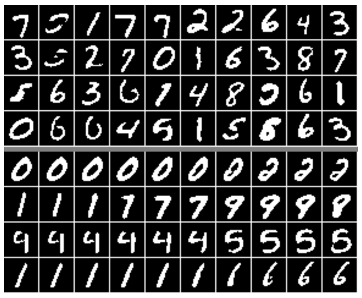

Figure 4.3: Sample interpolations from the ARAE-GAN. Constructed by linearly interpolating in the latent space and decoding to the output space. Word changes are highlighted in black. Results of the ARAE. The top block shows output generation of the decoder taking *fake* hidden codes generated by the GAN; the bottom block shows sample interpolation results.

require combinatorial search, which makes this a useful feature of text GANs. We experiment with this property by sampling two points $\mathbf{z}_0$ and $\mathbf{z}_1$ from $p(\mathbf{z})$ and constructing intermediary points $\mathbf{z}_\lambda = \lambda\mathbf{z}_1 + (1-\lambda)\mathbf{z}_0$. For each we generate the argmax output $\tilde{\mathbf{x}}_\lambda$. The samples are shown in Figure 4.3 (left) for text and in Figure 4.3 (right) for a discretized MNIST ARAE-GAN.

A final intriguing property of image GANs is the ability to move in the latent space via offset vectors (similar to the case with word vectors [Mikolov et al., 2013b]). For example, Radford et al. [Radford et al., 2016] observe that when the mean latent vector for "men with glasses" is subtracted from the mean latent vector for "men without glasses" and applied to an image of a "woman without glasses", the resulting image is that of a "woman with glasses". To experiment with this property we generate 1 million sentences from the ARAE-GAN and compute vector transforms in this space to attempt to change main verbs, subjects and modifier (details in section 4.8). Some examples of successful transformations are shown in Figure 4.5 (right). Quantitative evaluation of the success of the vector transformations is given in Figure 4.5 (left).

Transform	Match %	Prec
walking	85	79.5
man	92	80.2
two	86	74.1
dog	88	77.0
standing	89	79.3
several	70	67.0

Figure 4.4: Quantitative evaluation of transformations. Match % refers to the % of samples where at least one decoder samples (per 100) had the desired transformation in the output, while Prec. measures the average precision of the output against the original sentence.

A man in a tie is sleeping and clapping on balloons .	$\Rightarrow_{\text{walking}}$
A man in a tie is clapping and walking dogs .	
A person is standing in the air beneath a criminal .	$\Rightarrow_{\text{walking}}$
A person is walking in the air beneath a pickup .	
The jewish boy is trying to stay out of his skateboard .	$\Rightarrow_{\text{man}}$
The jewish man is trying to stay out of his horse .	
The people works in a new uniform studio .	$\Rightarrow_{\text{man}}$
A man works in a new studio uniform .	
Some child head a playing plastic with drink .	$\Rightarrow_{\text{Two}}$
Two children playing a head with plastic drink .	
A baby workers is watching steak with the water .	$\Rightarrow_{\text{Two}}$
Two workers watching baby steak with the grass .	
The people shine or looks into an area .	$\Rightarrow_{\text{dog}}$
The dog arrives or looks into an area .	
The boy 's babies is wearing a huge factory .	$\Rightarrow_{\text{dog}}$
The dog 's babies is wearing a huge ears .	
A women are walking outside near a man .	$\Rightarrow_{\text{standing}}$
Three women are standing near a man walking .	
The dogs are sleeping in front of the dinner .	$\Rightarrow_{\text{standing}}$
Two dogs are standing in front of the dinner .	
A side child listening to a piece with steps playing on a table .	$\Rightarrow_{\text{Several}}$
Several child playing a guitar on side with a table .	
Two children are working in red shirt at the cold field .	$\Rightarrow_{\text{Several}}$
Several children working in red shirt are cold at the field .	

Figure 4.5: Examples (out of 100 decoder samples per sentence) where the offset vectors produced successful transformations of the original sentence. See section 4.8 for full methodology.

4.7 Sample Generations

See Figure 4.6.

ARAE-GAN Samples

A woman preparing three fish .
A woman is seeing a man in the river .
There passes a woman near birds in the air .
Some ten people is sitting through their office .
The man got stolen with young dinner bag .
Monks are running in court .
The Two boys in glasses are all girl .
The man is small sitting in two men that tell a children .
The two children are eating the balloon animal .
A woman is trying on a microscope .
The dogs are sleeping in bed .

AE Samples

Two Three woman in a cart tearing over of a tree .
A man is hugging and art .
The fancy skier is starting under the drag cup in .
A dog are ¡unk¿ a
A man is not standing .
The Boys in their swimming .
A surfer and a couple waiting for a show .
A couple is a kids at a barbecue .
The motorcycles is in the ocean loading
I 's bike is on empty
The actor was walking in a a small dog area .
no dog is young their mother

LM Samples

a man walking outside on a dirt road , sitting on the dock .
A large group of people is taking a photo for Christmas and at night .
Someone is avoiding a soccer game .
The man and woman are dressed for a movie .
Person in an empty stadium pointing at a mountain .
Two children and a little boy are ¡unk¿ a man in a blue shirt .
A boy rides a bicycle .
A girl is running another in the forest .
the man is an indian women .

Figure 4.6: Text samples generated from ARAE-GAN, a simple AE, and from a baseline LM trained on the same data. To generate from an AE we fit a multivariate Gaussian to the learned code space and generate code vectors from this Gaussian.

4.8 Vector Arithmetic

We generate 1 million sentences from the ARAE-GAN and parse the sentences to obtain the main verb, subject, and modifier. Then for a given sentence, to change the main verb we subtract the mean latent vector ($\mathbf{t}$) for all other sentences with the same main verb (in the first example in Figure 4.5 this would correspond to all sentences that had "sleeping" as the main verb) and add the mean latent vector for all sentences that have the desired transformation (with the running example this would be all sentences whose main verb was "walking"). We do the same to transform the subject and the modifier. We decode back into sentence space with the transformed latent vector via sampling from $p_\psi(g(\mathbf{z} + \mathbf{t}))$. Some examples of successful transformations are shown in Figure 4.5 (right). Quantitative evaluation of the success of the vector transformations is given in Figure 4.5 (left). For each original vector $\mathbf{z}$ we sample 100 sentences from $p_\psi(g(\mathbf{z}+\mathbf{t}))$ over the transformed new latent vector and consider it a match if *any* of the sentences demonstrate the desired transformation. Match % is proportion of original vectors that yield a match post transformation. As we ideally want the generated samples to only differ in the specified transformation, we also calculate the average word precision against the original sentence (Prec) for any match.

4.9 Optimality Property

One can interpret the ARAE framework as a dual pathway network mapping two distinct distributions into a similar one; enc_ϕ and g_θ both output code vectors that are kept similar in terms of Wasserstein distance as measured by the critic. We provide the following proposition showing that under our parameterization of the encoder and the generator, as the Wasserstein distance converges, the encoder distribution ($\mathbf{c} \sim \mathbb{P}_r$) converges to the generator distribution ($\tilde{\mathbf{c}} \sim \mathbb{P}_g$), and further, their moments converge.

This is ideal since under our setting the generated distribution is simpler than the encoded distribution, because the input to the generator is from a simple distribution (e.g. spherical Gaussian) and the generator possesses less capacity than the encoder. However, it is not so simple that it is overly restrictive (e.g. as in VAEs). Empirically we observe that the first and second moments do indeed converge as training progresses (Section 4.6.1).

Proposition 1. *Let $\mathbb{P}$ be a distribution on a compact set $\mathcal{X}$, and $(\mathbb{P}_n)_{n\in N}$ be a sequence of distributions on $\mathcal{X}$. Further suppose that $W(\mathbb{P}_n, \mathbb{P}) \to 0$. Then the following statements hold:*

(i) $\mathbb{P}_n \rightsquigarrow \mathbb{P}$ *(i.e. convergence in distribution).*

(ii) All moments converge, i.e. for all $k > 1, k \in \mathbb{N}$,

$$\mathbb{E}_{X\sim\mathbb{P}_n}\Big[\prod_{i=1}^{d} X_i^{p_i}\Big] \to \mathbb{E}_{X\sim\mathbb{P}}\Big[\prod_{i=1}^{d} X_i^{p_i}\Big]$$

for all $p_1, \dots, p_d$ *such that* $\sum_{i=1}^{d} p_i = k$

Proof. (i) has been proved in Villani [2008] Theorem 6.9.

For (ii), using *The Portmanteau Theorem*, (i) is equivalent to:

$\mathbb{E}_{X\sim\mathbb{P}_n}[f(X)] \to \mathbb{E}_{X\sim\mathbb{P}}[f(X)]$ for all bounded and continuous function $f\colon \mathbb{R}^d \to \mathbb{R}$, where d is the dimension of the random variable.

The k-th moment of a distribution is given by

$$\mathbb{E}\Big[\prod_{i=1}^{d} X_i^{p_i}\Big] \text{ such that } \sum_{i=1}^{d} p_i = k$$

Our encoded code is bounded as we normalize the encoder output to lie on the unit sphere, and our generated code is also bounded to lie in $(-1, 1)^n$ by the tanh function. Hence $f(X) = \prod_{i=1}^{d} X_i^{q_i}$ is a bounded continuous function for all $q_i > 0$. Therefore,

$$\mathbb{E}_{X\sim\mathbb{P}_n}\Big[\prod_{i=1}^{d} X_i^{p_i}\Big] \to \mathbb{E}_{X\sim\mathbb{P}}\Big[\prod_{i=1}^{d} X_i^{p_i}\Big]$$

where $\sum_{i=1}^{d} p_i = k$ □

	Original Science
	what is an event horizon with regards to black holes ?
Music	what is your favorite sitcom with adam sandler ?
Politics	what is an event with black people ?
	take 1ml of hcl (concentrated) and dilute it to 50ml .
Music	take em to you and shout it to me
Politics	take bribes to islam and it will be punished .
	just multiply the numerator of one fraction by that of the other .
Music	just multiply the fraction of the other one that
Politics	just multiply the same fraction of other countries .
	Original Music
	do you know a website that you can find people who want to join bands ?
Science	do you know a website that can help me with science ?
Politics	do you think that you can find a person who is in prison ?
	all three are fabulous artists , with just incredible talent ! !
Science	all three are genetically bonded with water , but just as many substances , are capable of producing a special case .
Politics	all three are competing with the government , just as far as i can .
	but there are so many more i can
Science	but there are so many more of the number of questions .
Politics	but there are so many more of the can i think of today .
	Original Politics
	republicans : would you vote for a cheney / satan ticket in 2008 ?
Science	guys : how would you solve this question ?
Music	guys : would you rather be a good movie ?
	4 years of an idiot in office + electing the idiot again = ?
Science	4 years of an idiot in the office of science ?
Music	4 ¡unk¿ in an idiot , the idiot is the best of the two points ever !
	anyone who doesnt have a billion dollars for all the publicity cant win .
Science	anyone who doesnt have a decent chance is the same for all the other .
Music	anyone who doesnt have a lot of the show for the publicity .

Table 4.4: Random samples from Yahoo topic transfer. Note the first row is from ARAE trained on titles while the following ones are from replies.

Model	Medium	Small	Tiny
Supervised Encoder	65.9%	62.5%	57.9%
Semi-Supervised AE	68.5%	64.6%	59.9%
Semi-Supervised ARAE	70.9%	66.8%	62.5%

Data for LM	Reverse PPL
Real data	27.4
LM samples	90.6
AE samples	97.3
ARAE-GAN samples	82.2

Table 4.5: **Top.** Semi-Supervised accuracy on the natural language inference (SNLI) test set, respectively using 22.2% (medium), 10.8% (small), 5.25% (tiny) of the supervised labels of the full SNLI training set (rest used for unlabeled AE training). **Bottom.** Perplexity (lower is better) of language models trained on the synthetic samples from a GAN/AE/LM, and evaluated on real data (Reverse PPL).

Chapter 5

Unsupervised Text Summarization by Using Language Models as Regularizers

It has been shown that neural networks can learn to optimize a combinatorial latent space for text summarization with limited supervision. Miao and Blunsom [2016] used semi-supervised learning to train a discrete sequence autoencoder, imposing constraints to make the latent sequence more language-like. Wang and Lee [2018] more recently showed that adversarial learning could make latent sequences more language-like by training a discriminator to distinguish true summaries from generated ones. However, both cases require summary data for the training procedure, making them not fully unsupervised. In this short but focused research contribution, we propose an alternative learning strategy that does not require any summaries for training at all. Experiments on the Gigaword summarization task Rush et al. [2015] reveal that the proposed unsupervised approach is competitive in terms of ROUGE-L, without requiring any target summary data at all.

5.1 Introduction

Unsupervised text summarization can be viewed as training a discrete sequence autoencoder with a combinatorial hidden space [Kaiser et al., 2018]. The hidden space consists of sequences of discrete tokens in a vocabulary that overlaps with the input vocabulary. By constraining the hidden sequence to be shorter than the original sequence, the autoencoder produces a succinct summary of the original sequence. This idea was first discussed and empirically investigated by Miao and Blunsom [2016], who used semi-supervised learning and pointer networks [Vinyals et al., 2015] to make hidden sequences more language-like. More recently, Wang and Lee [2018] proposed to use adversarial learning to encourage hidden sequences to be more language-like. Both approaches did not rely on paired examples, but used two disjoint sets of original text and summaries. Orthogonal to these approaches, Fevry and Phang [2018] constructed a summarization task by adding noise to expand sentences and train a denoising auto-encoder to recover the original unexpanded input.

In this paper, we propose a novel approach to fully unsupervised text summarization. Contrary to previous work, our approach does not require any summary data at all. Motivated by recent observations that well-trained neural language models generate well-formed sentences even when their internal hidden states are perturbed, suggesting that language model effectively shape their combinatorial output space to emphasize language-like sequences [Cho, 2016; Gu et al., 2017; Chen et al., 2018], our algorithm consists of two stages. First, the latent sequence generator is pretrained as an unconditional language model, ignoring the input sequence. Subsequently, all the other parts of the network are trained as an autoencoder while the pretrained latent sequence generator is kept fixed.

We evaluate the proposed approach on the headline generation task built using English Gigaword [Rush et al., 2015]. Our experiments reveal that the summaries generated by this model are competitive with those from the state-of-the-art discrete autoencoder model by Wang and Lee [2018], without requiring any target summary data at all. We analyze the model by investigating the perplexity of generated summaries in various language models.[1]

5.2 Discrete sequence autoencoders

Let $X \in V_x^T$ be a sequence of discrete tokens, where V_x is a set of all unique tokens and T is its length. A *discrete sequence autoencoder* first encodes such a sequence X as a distribution over a hidden set of discrete configurations: $p_{\text{enc}}(Z|X)$. Given a sample $\tilde{Z}$ from this encoder distribution, the discrete autoencoder computes the distribution over the original sequence space V_x^T: $p_{\text{dec}}(X|\tilde{Z})$. Both p_{enc} and p_{dec} are parameterized using differentiable neural networks.

We are interested in a *combinatorial hidden space*, where each configuration Z in the hidden set is constructed as a sequence of T' discrete symbols drawn from a common vocabulary V_z, i.e., $Z \in V_z^{T'}$. The discrete symbols within each hidden configuration Z strongly depend on each other, which must be captured explicitly, i.e., $p_{\text{enc}}(Z|X) \neq \prod_{t'=1}^{T'} p(Z_{t'}|X)$. Hence, we model the encoder distribution in an

[1]All code and data will published at [GITHUB_LINK].

autoregressive manner:

$$p_{\text{enc}}(Z|X) = \prod_{t'=1}^{T'} p_{\text{enc}}(z_{t'}|z_{<t'}, X), \tag{5.1}$$

where $z_{<t'} = (z_1, \ldots, z_{t'-1})$.

Learning For each training example $X \in D_{\text{train}}$, the discrete sequence autoencoder defines the following reconstruction log-probability, its lowerbound and the stochastic approximation to the lower-bound:

$$\log p(X|X) \geq \mathcal{L}(X) \tag{5.2}$$

$$= \mathbb{E}_{Z \sim p_{\text{enc}}(Z|X)} \left[\log p_{\text{dec}}(X|Z)\right]$$

$$\approx \mathcal{L}_K(X) = \frac{1}{K} \sum_{k=1}^{K} \log p_{\text{dec}}(X|Z^k), \tag{5.3}$$

where Z^k is the k-th sample from p_{enc}. We can then maximize the lowerbound using gradient-based optimization.

The gradient of $\mathcal{L}_K$ w.r.t. $\log p_{\text{dec}}(X|Z^k)$ is easily and tractably computed using backpropagation. To compute the gradient w.r.t. p_{enc} we have to rely on REINFORCE [Williams, 1992b], due to the discrete bottleneck:

$$\nabla_{\text{enc}} \mathcal{L} = \mathbb{E}_Z \left[(\log p_{\text{dec}}(X|Z) - V(X)) \times \nabla \log p_{\text{enc}}(Z|X) \right], \tag{5.4}$$

where $V(X)$ is a baseline estimator. We set the baseline according to self-critical sequence training [Rennie et al., 2017]:

$$V(X) = \log p_{\text{dec}}(X|\hat{Z}). \tag{5.5}$$

$\hat{Z} = (\hat{z}_1, \ldots, \hat{z}_{T'})$ is a greedily-decoded sequence, where $\hat{z}_{t'} = \arg\max_{z_{t'}} p_{\text{enc}}(z_{t'}|z_{<t'}, X)$.

5.3 Unsupervised text summarization

Discrete autoencoders can be employed as unsupervised text summarization systems by imposing appropriate constraints on the combinatioral hidden space, namely:

1. Shorter hidden sequence: $T' < T$.
2. Shared vocabulary: $V_z = V_x$.
3. Language-like hidden sequence.

The first constraint can be satisfied by simply terminating inference prematurely by setting $p(z_{\tilde{T}'} = \langle\text{eos}\rangle \, |z_{<\tilde{T}'}, X) = 1$, where $\langle\text{eos}\rangle$ is a special token indicating the end of a sequence. As it stands, $\tilde{T}'$ is less than T' when the model chooses to end the generation early otherwise equals to T' which amounts to a forced stop. Because we regularize the hidden sequence to be language like (see section 5.4), the model trained this way is encouraged to end the generated sentence early in order to meet the $\langle\text{eos}\rangle$ token. The second constraint can be either implemented by simply sharing the vocabulary or using a pointer network Vinyals et al. [2015], of which the latter may severely restrict the fluency and diversity of summaries. The third constraint is where scientific creativity is necessary, as there is no single agreed-upon metric for measuring the language-likeness. There have been two notable approaches in recent years, both of which we describe briefly below.

Semi-Supervised Learning Miao and Blunsom [2016] build a sentence simplification model using a discrete sequence autoencoder with a combinatorial hidden space. They use a pointer network to model the encoder distribution in order to reduce the size of the hidden space and to ensure no word outside the original sequence appears in its summary. To make it work, they rely heavily on semi-supervised learning, in which a small set of annotated examples (sentence-summary pairs) is used to encourage the hidden sequences to be more language-like. Since the source code for this approach was not made publicly available, we do not compare our approach against it.

Adversarial (Transfer) learning Wang and Lee [2018] relaxes the need of paired examples used by Miao and Blunsom [2016] by using adversarial learning [Goodfellow et al., 2014; Ganin et al., 2016]. They introduce a discriminator network that learns to distinguish between real and generated summaries, and use the reversed gradient of the discriminator's output with respect to the encoder to encourage it to generate a language-like hidden sequence. This approach has two weaknesses: first, it relies on the availability of summaries for training the discriminator; and second, the generator is pre-trained using the CNN/Daily Mail dataset, effectively reducing their setup to transfer learning rather than unsupervised learning.

5.4 Language model regularization

We propose a simpler approach for encouraging the hidden combinatorial space to be more language-like. This approach furthermore does not require the availability of even unpaired summaries, although they could be used if available.

The model for p_{enc} can be decomposed as a sequence-to-sequence model [Cho et al., 2014; Sutskever et al., 2014], comprising two parts: "read" and "write". At time t', the first part "read" extracts a vector representation of the input sequence conditioned on the current context $(z_1, \ldots, z_{t'-1})$:

$$\nu_{t'} = f_{\text{read}}(z_1, \ldots, z_{t'-1}) \in \mathbb{R}^d,$$

which can be implemented as an attention mechanism [Bahdanau et al., 2014; Vaswani et al., 2017]. The "write" part then computes the distribution over the next token given $\nu_{t'}$ and the context $z_{<t'}$:

$$p_{\text{enc}}(z_{t'}|z_{<t'}, X) \propto \exp(f_{\text{write}}^{z_{t'}}(\nu_{t'}, z_{<t'})).$$

If $\nu_{t'}$ is fixed to 0, the latent encoder reduces simply to an autoregressive language model with an unbounded context [Mikolov et al., 2010].

Based on this observation, we propose a two-stage algorithm for training a discrete autoencoder as an unsupervised summarizer.

(1) Language Model Pretraining The first stage is thus to train the "write" part of the discrete latent encoder while fixing $\nu_{t'} = 0$ on a large unlabelled corpus as an "unconditional" language model. This shapes the combinatorial hidden space to which the encoder maps input sequences, assigning (much) higher probability to language-like sequences.

(2) Global Training Next, we train the decoder p_{dec} and *only* f_{read} of p_{enc}. f_{write} is held fixed as the pretrained language model. Although it has learned to shape the hidden space to mostly include language-like sequences, we further encourage f_{read} to generate language-like sequences only by introducing two modifications to the learning procedure described in Section 5.2.

First, we replace $\log p_{\text{dec}}(X|Z)$ in Eqs. (5.4)–(5.5), which acts as a reward in policy gradient, with

$$\log p_{\text{dec}}(X|Z) + \lambda \log p_{\text{enc}}(Z|0),$$

where $p_{\text{enc}}(Z|0)$ is the probability assigned by the encoder when $\nu_{t'}$ was held at 0. $\lambda \geq 0$ is a weighting coefficient. This prevents f_{read} from learning to generate a hidden sequence that is unlikely under the pretrained language model.

Second, we down-weight tokens that do not appear in the original sequence during training:

$$p'_{\text{enc}}(z_{t'}|z_{<t'}, X) \propto \begin{cases} p_{\text{enc}}(z_{t'}|z_{<t'}, X), \\ \qquad \text{if } z_{t'} \in X \\ \eta p_{\text{enc}}(z_{t'}|z_{<t'}, X), \\ \qquad \text{otherwise} \end{cases},$$

where $\eta \in [0, 1]$. We start with a small $\eta = 10^{-6}$ and gradually increase it toward 1 over training.

	ROUGE-L
Transfer Learning from CNN/Daily Mail	
W&L	**25.16**
Unsupervised Learning	
URSAE-Major (no pretrain)	15.60
URSAE-Major	**20.77**
URSAE-Minor	20.14

Table 5.1: The quality of generated summaries as measured by ROUGE-L, contrasting our unsupervised approach with the adversarial transfer learning approach of Wang and Lee [2018] (W&L).

5.5 Results

We evaluate the proposed approach on headline generation and use the data prepared by Rush et al. [2015] based on the English Gigaword [Graff et al., 2003]. It contains 119M and 31M words on the article and headline sides respectively.

We build two URSAE—short for Unsupervised Recurrent Sequential Auto-Encoder—models that are distinguished by the corpus that was used to pretrain the encoder as a language model. We use either the longer input sequences (articles) or the shorter output sequences (headlines) to pretrain the encoder, and call the resulting models **URSAE-Major** and **URSAE-Minor**, respectively. These models are trained as described above. Please see the supplementary materials for more details.

5.5.1 Main result

We compare the proposed approach against the adversarial transfer learning approach of Wang and Lee [2018]. We use ROUGE-L [Lin, 2004] to measure the similarity between the generated summary and the ground-truth headline. We

Input	consumer and business confidence in the european union slumped in november to the lowest level in ## years in the face of looming recession , according to an eu survey on thursday
GS	eu consumer business confidence hits ##-year low : survey
URSAE-Major	the european union on thursday slumped to ##
URSAE-Minor	the european union confidence on thursday slumped
W&L	consumer confidence in ## years november
Input	finland scored three goals in a ##-second span of the first period tuesday night for a #-# victory over the czech republic in their world cup of hockey opener
GS	finland routs czech republic at world cup
URSAE-Major	finland scored a #-# victory over the czech republic in world cup first tuesday
URSAE-Minor	finland scored in world cup first tuesday
W&L	finland three goals in world cup opener

Table 5.2: Validation examples together with the associated ground-truth headlines and the summaries generated by the proposed approach (URSAE-Major) and the state-of-the-art approach by Wang and Lee [2018].

	Headlines	GPT2
W&L	380.79	661.34
URSAE-Major	514.10	**506.23**
URSAE-Minor	**192.23**	744.88

Table 5.3: Perplexity results. Alhough comparable in terms of ROUGE-L, the summaries generated by the proposed approach are more language-like.

present our main result in Table 5.1. Although the proposed URSAE-Major had not seen a single ground-truth summary, unlike W&L, the quality of generated summaries from URSAE-Major only slightly lags behind that of W&L in terms of ROUGE-L.

We further did ablation study in Table 5.1 with result for training *without* pretraining. This clearly demonstrates the necessity for pretraining. Without pretraining, we found learning to be unstable, and even when learning was successful, the resulting model significantly underperformed the model trained following the proposed approach.

5.5.2 Perplexity analysis

The ROUGE-L metric measures the quality of summarization, based largely on content overlap, which may not necessarily correspond to how language-like a generated summary is. To that end, we also examine the perplexity of a generated summary by a recurrent language model. We use two language models to better understand what our models have learned, reported in Table 5.3.

In one case, we train a language model from scratch[2] on headlines, ensuring that the language model follows the same distribution, and then use it to calculate the perplexity. In the other case, we used the recently released state-of-the-art GPT2 language model Radford et al. [2019], the perplexity of which serves as an indicator of the language-likeness of the generated summaries.

As can be seen, the summaries from URSAE-Major are more fluent than W&L. We observe that the generated summaries stay similar to the corpus used to pretrain the encoder, as URSAE-Minor more closely mirrors headline data.

5.5.3 Qualitative evaluation

In Table 5.2, we show two randomly selected examples. First, we observe the difference in linguistic styles between URSAE-Major and URSAE-Minor, of which the encoders were pretrained and fixed using two different corpora (articles and headlines, respectively), demonstrating the effectiveness of language model pretraining in URSAEs. Second, URSAE-Major, which never saw a headline during training, creates well-formed summaries of the input sequence, albeit with a few factual mistakes. We believe these examples indicate that the proposed approach

[2]https://github.com/kpu/kenlm

is a promising step forward in fully unsupervised text summarization.

5.6 Conclusion

We introduced a novel method for unsupervised text summarization, using discrete sequential auto-encoders. Contrary to previous approaches, our method does not require any target summary data whatsoever, and can learn to produce summaries using article data only. In future work, we aim to investigate additional reward structures that might produce even better results.

5.7 Technicality points

5.7.1 Additional Experiment Details

Language model We train a 2-layer LSTM language model, with 300 hidden units and dropout at 0.2. We trained the LM with standard SGD optimizer, with learning rate 1 and weight clipping by l_2 norm of 5.

Vocabulary We use a vocabulary size 30,000 throughout all the experiments.

Model-encoder The encoder is parameterized by a word embedding layer followed by a standard attention module, with 300 hidden units. In particular inside the attention module, the query is the hidden state of the LSTM LM from previous time step while the keys are the encoder output. We additionally added the positional-embedding to the keys. We constrain the summary length to be less

Input	vietnam is among the countries most vulnerable to the chronic hepatitis b virus with more than eight percent of vietnamese people contracting the virus , the local newspaper the youth reported tuesday
GS	vietnamese people vulnerable to hepatitis b virus : experts
DSeqAE-IN	more than eight people reported to the virus is among most of vietnam
DSeqAE-OUT	vietnam to UNK with the people of more than eight percent
W&L	vietnam among most than eight percent
Input	a reggae festival in new york which was created to promote peace among cultures is being denounced by gay and lesbian groups for allowing performers with a history of anti-gay lyrics
GS	gay lesbian groups protest ny reggae concert because of anti-gay music
DSeqAE-IN	a new york festival is being created in history
DSeqAE-OUT	a new york festival in peace with history of gay
W&L	new york peace festival in

Table 5.4: More qualitative results

than 15.

Model-decoder The decoder is a convolution based Seq2seq model implemented by Fairseq [3]. The sub-level encoder inside is a 4-layer convolutional network with 300 feature maps and kernel size 4. The sub-level decoder inside is a 3-layer convolutional network with 300 feature maps and kernel size 3. Note that this is the standard configuration `fconv_iwslt_de_en` in the official Fairseq repository.

Model-training Our model is trained using a standard SGD optimizer with Nesterov momentum of 0.9. The learning rate was initially set to 0.25 and decayed by a factor of 10 when training reached a plateau as measured by validation set reconstruction.

Weighing hyperparameter λ is set to 1.

5.7.2 Additional qualitative results

We show three more qualitative results in Table 5.4.

[3]https://github.com/pytorch/fairseq

Chapter 6

Stacked What-Where Autoencoders

In this chapter, we present a novel architecture, the "stacked what-where autoencoders" (SWWAE), which integrates discriminative and generative pathways and provides a unified approach to supervised, semi-supervised and unsupervised learning without requiring sampling. An instantiation of SWWAE is essentially a convolutional net (ConvNet) (LeCun et al. [1998]) coupled with a deconvolutional net (Deconvnet) (Zeiler et al. [2010]). The objective function includes reconstruction terms that penalize the hidden states in the Deconvnet for being different from the hidden state of the ConvNet. Each pooling layer is seen producing two sets of variables: the "what" which are fed to the next layer, and its complementary variable "where" that are fed to the corresponding layer in the generative decoder.

6.1 Motivation: the “what” and “where”

A desirable property of learning models is the ability to be trained in supervised, unsupervised, or semi-supervised mode with a single architecture and a single learning procedure. Another desirable property is the ability to exploit advantageous discriminative and generative models. A popular approach is to pre-train auto-encoders in a layer-wise fashion, and subsequently fine-tune the entire stack of encoders (the feed-forward pathway) in a supervised discriminative manner [Erhan et al., 2010; Gregor and LeCun, 2010; Henaff et al., 2011; Kavukcuoglu et al., 2009, 2008, 2010b; Ranzato et al., 2007b; Ranzato and LeCun, 2007]. This approach fails to provide a unified mechanism to unsupervised and supervised learning. Another approach, which has all the right properties, is the deep boltzmann machine (DBM) model (Hinton et al. [2006]; Larochelle and Bengio [2008]). Each layer in a DBM is a restricted boltzmann machine (RBM), which can be seen as a kind of auto-encoder. Deep RBMs have all the desirable properties, but the learning procedure misuses sampling, which tends to be inefficient. The main issue with stacked auto-encoders is asymmetry. The mapping implemented by the feed-forward pathway is often many-to-one, for example mapping images to invariant features or to class labels. Conversely, the mapping implemented by the feedback (generative) pathway is one-to-many, e.g. mapping class labels to image reconstructions. The common way to deal with this is to view the reconstruction mapping as probabilistic. This is the approach of RBMs and DBMs: the missing information that allows reconstructing an image from a category label is dreamed up by sampling. This tends to be complicated and inefficient.

If the mapping from input to output of the feed-forward pathway were one-to-one, the mappings in both directions would be well-defined functions and there would be no need for sampling while reconstructing. But if the internal representations are to possess good invariance properties, it is desirable that the mapping from one layer to the next being many-to-one. For example, in a ConvNet, invariance is achieved through layers of max-pooling and subsampling.

The main idea of the approach we propose here is very simple: whenever a layer implements a many-to-one mapping, we compute a set of complementary variables that enable reconstruction. In the max-pooling layers of ConvNets, we view the position of the max-pooling "switches" as the complementary information necessary for reconstruction. The model we proposed consists of a feed-forward ConvNet, coupled with a feed-back Deconvnet. Each stage in this architecture is what we call a "what-where auto-encoder" (WWAE).

More specifically, each encoder in the WWAE consists of a max-pooling layer as drawn from the first convolutional network [LeCun et al., 1998]. The output of the max-pooling is the "what" variable, which is fed to the next layer. The complementary variables are the max-pooling "switch" positions, which can be seen as the "where" variables. The "what" variables inform the next layer about the content with incomplete information about the position, while the "where" variables inform the corresponding feedback decoder about where interesting (dominant) features are located. The feedback (generative) decoder reconstructs the input by "unpooling" the "what" using the "where", and running the result through a reconstructing convolutional layer. Such what-where convolutional auto-encoders can be stacked and trained jointly without requiring alternate optimization (Zeiler et al. [2010]). The reconstruction penalty at each layer constrains the hidden states of the feedback pathway to be close to the hidden states of the feed-forward pathway. The system can be trained in purely supervised manner: the bottom input of the feed-forward pathway is given the input, the top layer of the feedback pathway is given the desired output, and the weights are updated to minimize the sum of the reconstruction costs. If only to top-level cost is used, the model reverts to purely supervised backprop. If the hidden layer reconstruction costs are used, the model can be seen as supervised with a reconstruction regularization. In unsupervised mode, the top-layer label output is left unconstrained and simply copied from the output of the feed-forward pathway. The model becomes a stacked what-where auto-encoder. As with boltzmann machines (BM), the underlying learning algorithm doesn't change between the supervised and unsupervised modes. As with BMs the only thing that changes is what is clamped or left unclamped. The model is particularly suitable when one is faced with a large amount of unlabeled data and a relatively small amount of labelled data. The fact that no sampling (or contrastive divergence method) is required gives the model good scaling properties. It's essentially just backprop in a particular architecture.

6.2 SWWAE model architecture

We write down the loss function of SWWAE as follow:

$$L = \lambda_{NLL} L_{NLL} + \lambda_{L2rec} L_{L2rec} + \lambda_{L2M} L_{L2M}, \tag{6.1}$$

where L_{NLL} is a classification loss if available, L_{L2rec} the reconstruction loss at the input level while L_{L2M} penalizes the difference at the intermediate levels between the two pathways. λ_*'s are the weighing terms.

As we mentioned before, the pooling layer from an encoder in the feedforward pathway branches out information into "what" and "where", depicted in figure 6.1(a). For one thing, the "what" component is essentially the output of a conventional max-pooling layer. For another, the "where" variable complements the information by preserving an *argmax* information. A simplest definition of "where" is to store all the "switches" information which indicates the location of the maximally activated pixel inside each pooling region. To connect each separate WWAE modules, the "what" component is fed up to the next layer in the feed-forward pathway, just like a normal feedforward convolutional network. The "where" variables are maintained to be used at the same level at the feedback pathway. More specifically, in each WWAE module in the feedback pathway, a decoder *unpools* the value from the upper layer by using the "where" variable that it obtains from its feedforward counterpart. This pooling and unpooling procedure are depicted in Figure 6.1(a).

The reconstruction energies are normal mean square losses:

$$L_{L2rec} = \|x - \tilde{x}\|_2, \qquad L_{L2M} = \|x_m - \tilde{x}_m\|_2, \tag{6.2}$$

where x_* and $\tilde{x}_*$ are feature maps (or pixels) from the feedforward and feedback pathways respectively. m indexes the layer number. The entire architecture is depicted in figure 6.1(b).

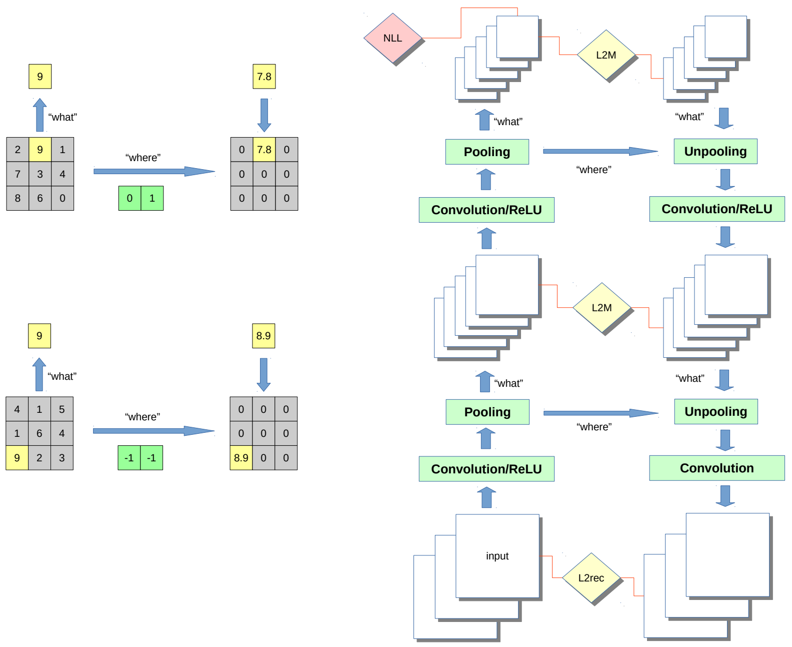

Figure 6.1: Left (a): pooling-unpooling. Right (b): model architecture. For brevity, fully-connected layers are omitted in this figure.

6.2.1 Phase pooling: "what" and "where"

Recently, Goroshin et al. [2015] introduces a soft version of *max* and *argmax* operators within each pooling region:

$$m_k = \sum_{N_k} z(x,y) \frac{e^{\beta z(x,y)}}{\sum_{N_k} e^{\beta z(x,y)}} \approx \max_{N_k} z(x,y) \tag{6.3}$$

$$\mathbf{p_k} = \sum_{N_k} \begin{bmatrix} x \\ y \end{bmatrix} \frac{e^{\beta z(x,y)}}{\sum_{N_k} e^{\beta z(x,y)}} \approx \arg\max_{N_k} z(x,y), \tag{6.4}$$

where $z(x,y)$ is the activation and x, y are spatial locations (for brevity we take normalized value from -1 to 1). N_k is the k^{th} pooling region. We use linera interpolation at the unpooling stage to handle the continuous switches. In addition, β is a non-negative hyper-parameter: larger β behaves like max-pooling while smaller β makes this pooling resemble a mean-pooling.

6.2.2 Training with joint losses and regularization

SWWAE provides a solution to integrate different learning paradigms by a single architecture and single learning algorithm, i.e. stochastic gradient descent and backprop:

- for supervised learning, we can mask out the entire Deconvnet pathway by setting λ_{L2*} to 0 and the SWWAE falls back to vanilla ConvNet.
- for unsupervised learning, we remove the classification facilities on top of the encoding pathway and set λ_{NLL} to 0.
- for semi-supervised learning, we train the model with the interplay of all loss terms.
- for generation-related tasks, we focus on the output from the feedback pathway.

The idea behind using reconstruction as a regularizer was studied in earlier works such as Erhan et al. [2010], and revived by parallel work Rasmus et al. [2015].

6.3 Experiments

6.3.1 Necessity of the "where"

In this section, we demonstrate the necessity of the "where" variables. In particular, we compare a standard SWWAE model with an upsampling-based alternative. Essentially, upsampling is an operation contrary to a WWAE that it copies the value to all locations within each pooling region. This operation is agnostic of any switches information from the feedforward pathway. In Figure 6.2, we show the reconstructed digits that are generated by an SWWAE (left) and an SWWAE-upsample (right) while we progressively increase the pooling region size from 2 to 16. It is obvious that the "where" information is necessary to get good quality reconstruction, especially with the pooling size growing large.

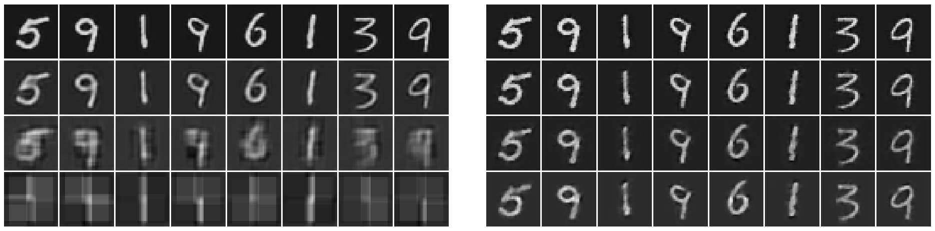

Figure 6.2: Reconstruction of SWWAE-upsample (left) and SWWAE (right). The pooling sizes are respectively 2, 4, 8, 16 up-to-down.

6.3.2 Invariance and Equivariance

In this section, we study the behavior of SWWAE by analyzing the "what" and "where" variables from a perspective given by Hinton et al. [2011]—*invariance* and *equivariance*. We setup the experiements as follows:

(i) we train an SWWAE-unsup model using just horizontally and/or vertically translated MNSIT images;

(ii) during testing, we first feed a batch of untranslated images to the trained SWWAE and we cache both "what" and "where" components on the top level from the feedforward pathway;

(iii) we horizontally translate the same batch of the images then do the same caching as (ii);

(iv) we plot respectively the “what” and “where” components from the untranslated images with respected to their translated counterparts;

(v) repeat (iii)-(iv) for vertically translated images. The results are displayed in Figure 6.3.

From these plots, we can roughly see the “what” and “where” characterize the invariance and equivariance respectively, defined by Hinton et al. [2011]. But unlike the transforming autoencoder [Hinton et al., 2011], we argue that SWWAE achieves this by the nature of its architectural design and without any extra explicit supervision.

6.3.3 Image classification using SWWAE

6.3.3.1 SVHN

We test SWWAEs on the Google Street View Housing Number (SVHN) dataset. The dataset contains 73257 harder training samples and 531131 extra easier samples. To set up a semi-supervised experiment similar to Kingma et al. [2014], we randomly sample 1000 samples from the canonical training set with labels uniformly distributed to 10 classes. Results are reported as the mean accuracy from 10 runs.

We show in Figure 6.4 the validation set performance varying with different λ_{L2*}. The final results on test set are showed in Table 6.1. We additionally evaluate SWWAE on SVHN under a supervised manner with all the labels available—an SWWAE improves upon a ConvNet (same configuration) from **5.89%** to **4.94%**.

Table 6.1: SWWAE against other published results on SVHN with 1000 available labels.

model / N	**error rate (in %)**
KNN	77.93
TSVM (Vapnik and Vapnik [1998])	66.55
M1+KNN (Kingma et al. [2014])	65.63
M1+TSVM (Kingma et al. [2014])	54.33
M1+M2 (Kingma et al. [2014])	36.02
SWWAE without dropout ($\lambda_{L2*} = 0.8$)	27.83
SWWAE with dropout ($\lambda_{L2*} = 0.4$)	**23.56**

Table 6.2: SWWAE against other published results on STL-10.

model	**accuracy**
Multi-task Bayesian Optimization (Swersky et al. [2013])	70.1%
Zero-bias ConvNets + ADCU (Paine et al. [2014])	70.2%
Exemplar ConvNets (Dosovitskiy et al. [2014])	**75.4**%
SWWAE	74.33%
CNN of same configuration	57.45%

6.3.3.2 STL-10

In this section, we assess SWWAEs using the STL-10 dataset. STL-10 is a semi-supervised image classification dataset that has a sheer 100:1 ratio of unlabeled samples to the labelled samples. The result is reported in Table 6.2. Our model achieves competitive results with the state-of-the-art.

6.3.4 Larger scale: CIFAR with 80 Million Tiny Images

As a larger scale experiment, we mix the CIFAR-10 / CIFAR-100 datasets with the 80 million tiny images dataset [Torralba et al., 2008] to make both tasks semi-supervised. In Table 6.3, we compare our result against the best published supervised model. Indeed, incorporating the huge amount of unlabeled data, an SWWAE outperforms the state-of-the-art supervised models on CIFAR-100 by a

Table 6.3: Accuracy of SWWAE on CIFAR-10 and CIFAR-100 in comparison with best published single-model results. Our results are obtained with the common augmentation setup applying contrast normalization, small translation and horizontal mirroring.

model	**CIFAR-10**	**CIFAR-100**
All-CNN (Springenberg et al. [2014])	**92.75**%	66.29%
Highway Network (Srivastava et al. [2015])	92.40%	67.76%
Deeply-supervised nets (Lee et al. [2014])	92.03%	67.76%
SWWAE ($\lambda_{L2rec} = 1, \lambda_{L2M} = 0.2$)	92.23%	**69.12**%

good margin.

6.4 Discussions

The overall system, which can be seen as pairing a Convnet with a Deconvnet, yields good accuracy on a variety of semi-supervised and supervised tasks. We envision that such architecture may also be useful in video related tasks where unlabeled samples abound.

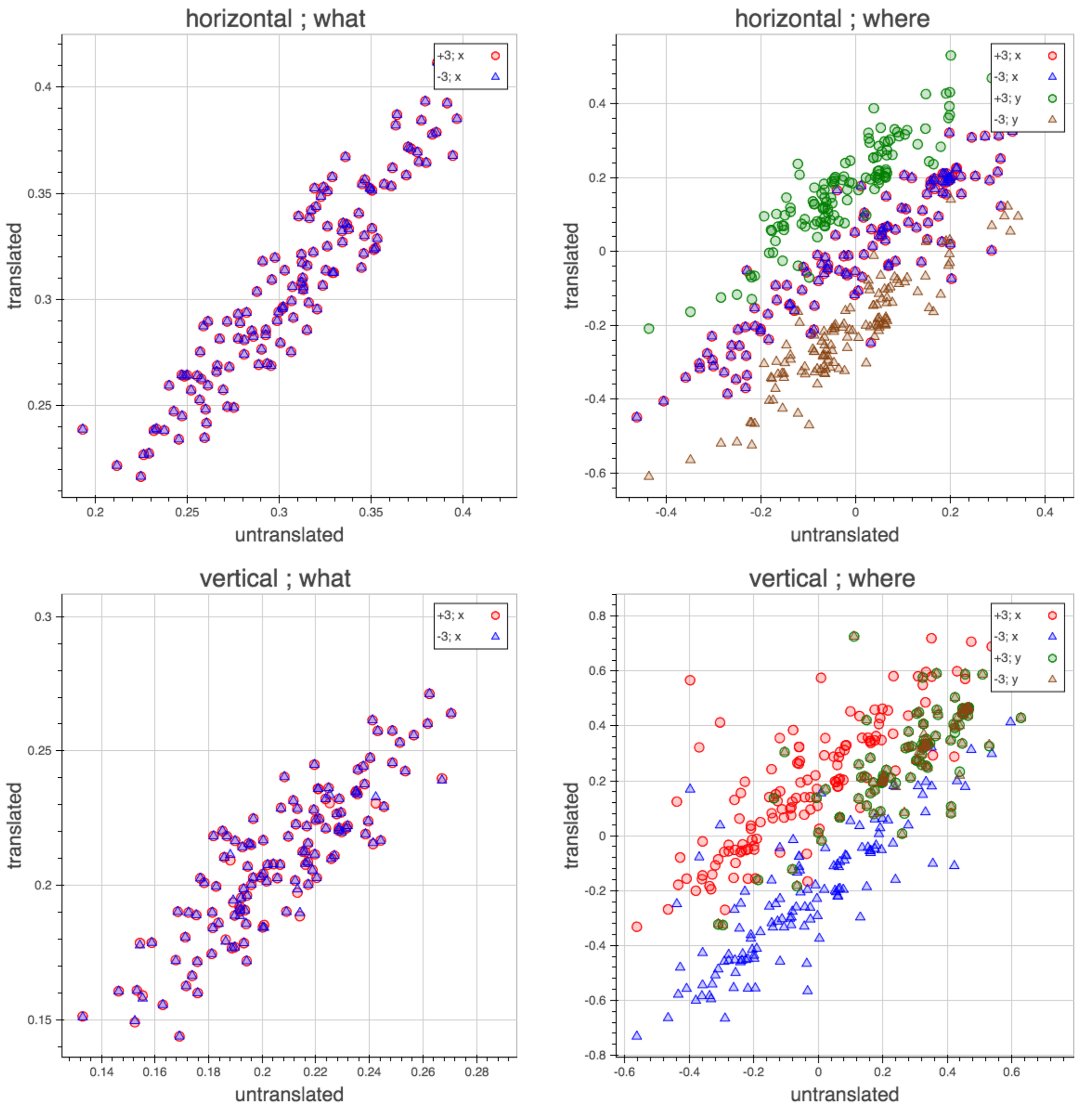

Figure 6.3: Capsule-alike scatter plots. The x-axis represents the "what" or "where" from one feature map for an untranslated digit image while the y-axis is the "what" or "where" output from the counterpart of the same set of translated images. Translation can happen in either horizontal or vertical directions. From left to right, the figures are respectively: first (a): "what" of horizontally translated digits versus original digits; second (b): "where" of horizontally translated digits versus original digits; third (c): "what" of vertically translated digits versus original digits; fourth (d): "where" of vertically translated digits versus original digits. Note that circles are used to feature +3 translation and triangles for -3. In the "where" related plots, x and y denote two dimensions of the "where" respectively.

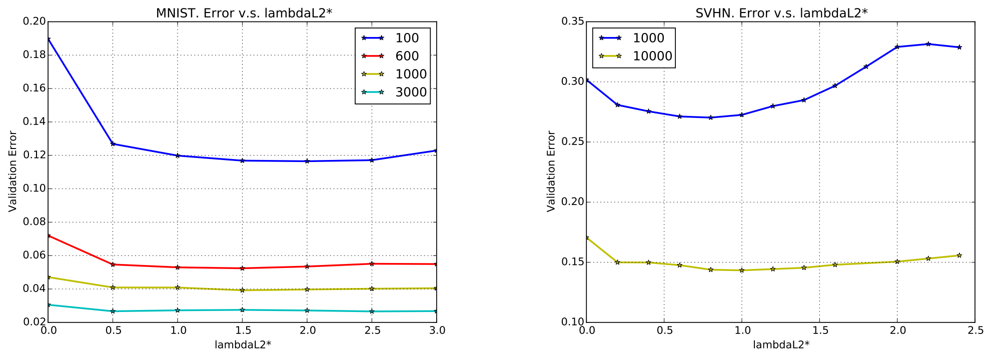

Figure 6.4: Validation-error v.s. λ_{L2*} on a range of datasets for SWWAE semi-supervised experiments. Left (a): MNIST. Right (b): SVHN. Different curves denote different number of labels being used.

Chapter 7

Unsupervised Learning of Transferable Relational Graphs

In this chapter, we propose a new framework that learns a "structural graph" representation based on a masked autoencoder framework. While trained in a complete unsupervised manner, these learned graphs are showed to carry useful dependency information of the input, and further can be transferred to other downstream tasks obtaining enhanced results.

7.1 Motivation

Modern deep transfer learning approaches have mainly focused on learning *generic* feature vectors from one task that are transferable to other tasks, such as word embeddings in language and pretrained convolutional features in vision. However, these approaches usually transfer unary features and largely ignore more structured graphical representations. This work explores the possibility of learning generic *latent relational graphs* that capture dependencies *between* pairs of data units (e.g., words or pixels) from large-scale unlabeled data and transferring the graphs to downstream tasks. Our proposed transfer learning framework improves performance on various tasks including question answering, natural language inference, sentiment analysis, and image classification. We also show that the learned graphs are generic enough to be transferred to different embeddings on which the graphs have not been trained (including GloVe embeddings, ELMo embeddings, and task-specific RNN hidden units), or embedding-free units such as image pixels.

7.2 Introduction

Recent advances in deep learning have largely relied on building blocks such as convolutional networks (CNNs) LeCun et al. [1995] and recurrent networks (RNNs) Hochreiter and Schmidhuber [1997] augmented with attention mechanisms Bahdanau et al. [2014]. While possessing high representational capacity, these architectures primarily operate on grid-like or sequential structures due to their built-in "innate priors". As a result, CNNs and RNNs largely rely on high expressiveness to model complex structural phenomena, compensating the fact that they do not explicitly leverage structural, graphical representations.

This paradigm has led to a standardized norm in transfer learning and pre-training. It is fitting an expressive function on a large dataset with or without supervision, and then applying the function to downstream task data for feature extraction. Notable examples include pretrained ImageNet features He et al. [2016] and pretrained word embeddings Mikolov et al. [2013a]; Pennington et al. [2014].

In contrast, a variety of real-world data exhibit much richer relational *graph* structures than the simple grid-like or sequential structures. This is also emphasized by a parallel work Battaglia et al. [2018]. For example in the language domain, linguists use parse trees to represent syntactic dependency between words; information retrieval systems exploit knowledge graphs to reflect entity relations; and coreference resolution is devised to connect different expressions of the same entity. As such, these exemplified structures are universally present in almost any natural language data regardless of the target tasks, which suggests the possibility of transfer across tasks. These observations also generalize to other domains such as vision, where modeling the relations between pixels is proven useful Parmar et al. [2018]; Zhang et al. [2018]; Wang et al. [2017]. One obstacle remaining, however, is that many of the universal structures are essentially human-curated and expensive to acquire on a large scale, while automatically-induced structures are mostly limited to one task Kipf et al. [2018]; Vaswani et al. [2017]; Wang et al. [2017].

In this chapter, we attempt to address two challenges: 1) to break away from the standardized norm of feature-based deep transfer learning[1], and 2) to learn versatile structures in the data with a data-driven approach. In particular, we are interested in learning transferable *latent relational graphs*, where the nodes of a latent graph are the input *units*, e.g., all the words in a sentence. The task of latent relational graph learning is to learn an *affinity matrix* where the weights (possibly zero) capture the dependencies between any pair of input units.

To achieve the above goals, we propose a novel framework of unsupervised latent graph learning called GLoMo (Graphs from LOw-level unit MOdeling). Specifically, we train a neural network from large-scale unlabeled data to output

[1]Throughout the paper, we use "feature" to refer to unary feature representations, and use "graph" to refer to structural, graphical representations.

latent graphs, and transfer the network to extract graph structures on downstream tasks to augment their training. This approach allows us to separate the features that represent the semantic meaning of each unit and the graphs that reflect how the units may interact. Ideally, the graphs capture task-independent structures underlying the data, and thus become applicable to different sets of features. Figure 7.1 highlights the difference between traditional feature-based transfer learning and our new framework.

Experimental results show that GLoMo improves performance on various language tasks including question answering, natural language inference, and sentiment analysis. We also demonstrate that the learned graphs are generic enough to work with various sets of features on which the graphs have not been trained, including GloVe embeddings Pennington et al. [2014], ELMo embeddings Peters et al. [2018], and task-specific RNN states. We also identify key factors of learning successful generic graphs: decoupling graphs and features, hierarchical graph representations, sparsity, unit-level objectives, and sequence prediction. To demonstrate the generality of our framework, we further show improved results on image classification by applying GLoMo to model the relational dependencies between the pixels.

7.3 Unsupervised Relational Graph Learning

We propose a framework for unsupervised latent graph learning. Given a one-dimensional input $x = (x_1, \cdots, x_T)$, where each x_t denotes an input unit at position t and T is the length of the sequence, the goal of latent graph learning is to learn a $(T \times T)$ affinity matrix $\mathbf{G}$ such that each entry G_{ij} captures the dependency between the unit x_i and the unit x_j. The affinity matrix is asymmetric, representing a

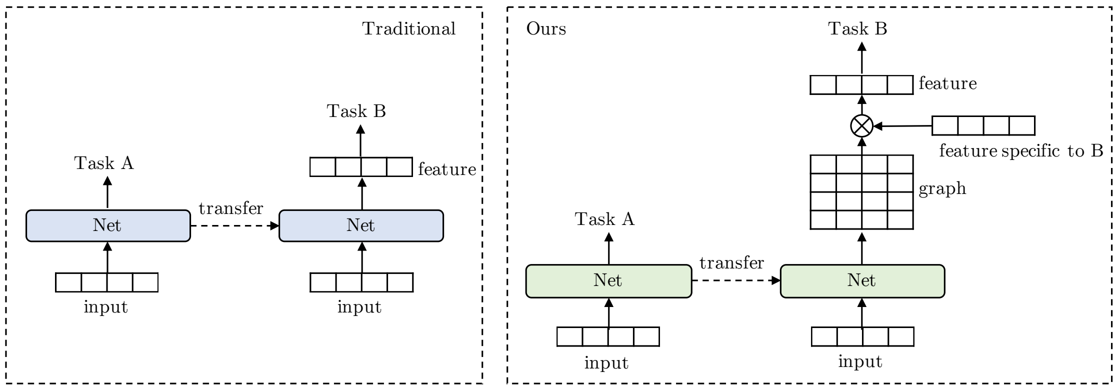

Figure 7.1: Traditional transfer learning versus our new transfer learning framework. Instead of transferring features, we transfer the graphs output by a network. The graphs are multiplied by task-specific features (e.g. embeddings or hidden states) to produce structure-aware features.

directed weighted graph. In particular, in this work we consider the case where each column of the affinity matrix sums to one, for computational convenience. In the following text, with a little abuse of notation, we use $\mathbf{G}$ to denote a set of affinity matrices. We use the terms "affinity matrices" and "graphs" interchangeably.

During the unsupervised learning phase, our framework trains two networks, a *graph predictor* network g and a *feature predictor* network f. Given the input x, our graph predictor g produces a set of graphs $\mathbf{G} = g(x)$. The graphs $\mathbf{G}$ are represented as a 3-d tensor in $\mathbb{R}^{L \times T \times T}$, where L is the number of *layers* that produce graphs. For each layer l , the last two dimensions $\mathbf{G}^l$ define a $(T \times T)$ affinity matrix that captures the dependencies between any pair of input units. The feature predictor network f then takes the graphs $\mathbf{G}$ and the original input x to perform a predictive task.

During the transfer phase, given an input x' from a downstream task, we use the graph predictor g to extract graphs $\mathbf{G} = g(x')$. The extracted graphs $\mathbf{G}$ are then fed as the input to the downstream task network to augment training. Specifically, we multiply $\mathbf{G}$ with task-specific features such as input embeddings and hidden states (see Figure 7.1). The network f is discarded during the transfer phase.

Next, we will introduce the network architectures and objective functions for unsupervised learning, followed by the transfer procedure. An overview of our framework is illustrated in Figure 7.2.

7.3.1 Unsupervised Learning

Graph Predictor The graph predictor g is instantiated as two multi-layer CNNs, a *key* CNN, and a *query* CNN. Given the input x, the key CNN outputs a sequence of convolutional features $(\mathbf{k}_1, \cdots, \mathbf{k}_T)$ and the query CNN similarly outputs $(\mathbf{q}_1, \cdots, \mathbf{q}_T)$. At layer l, based on these convolutional features, we compute the graphs as

$$G_{ij}^l = \frac{\left(\text{ReLU}(\mathbf{k}_i^{l\top}\mathbf{q}_j^l + b)\right)^2}{\sum_{i'}\left(\text{ReLU}(\mathbf{k}_{i'}^{l\top}\mathbf{q}_j^l + b)\right)^2} \tag{7.1}$$

where $\mathbf{k}_i^l = \mathbf{W}_k^l \mathbf{k}_i$ and $\mathbf{q}_j^l = \mathbf{W}_q^l \mathbf{q}_j$. The matrices $\mathbf{W}_k^l$ and $\mathbf{W}_q^l$ are model parameters at layer l, and the bias b is a scalar parameter. This resembles computing the attention weights Bahdanau et al. [2014] from position j to position i except that the exponential activation in the softmax function is replaced with a squared ReLU operation—we use ReLUs to enforce sparsity and the square operations to stabilize training. Moreover, we employ convolutional networks to let the graphs $\mathbf{G}$ be aware of the local order of the input and context, up to the size of each unit's receptive field.

Feature Predictor Now we introduce the feature predictor f. At each layer l, the input to the feature predictor f is a sequence of features $\mathbf{F}^{l-1} = (\mathbf{f}_1^{l-1}, \cdots, \mathbf{f}_t^{l-1})$ and an affinity matrix $\mathbf{G}^l$ extracted by the graph predictor g. The zero-th layer features $\mathbf{F}_0$ are initialized to be the embeddings of x. The affinity matrix $\mathbf{G}^l$ is then combined with the current features to compute the next-layer features at each position t,

$$\mathbf{f}_t^l = v(\sum_j G_{jt}^l \mathbf{f}_j^{l-1}, \mathbf{f}_t^{l-1}) \tag{7.2}$$

where v is a compositional function such as a GRU cell Chung et al. [2014] or a linear layer with residual connections. In other words, the feature at each position is computed as a weighted sum of other features, where the weights are determined by the graph $\mathbf{G}^l$, followed by transformation function v.

Objective Function At the top layer, we obtain the features $\mathbf{F}^L$. At each position t, we use the feature $\mathbf{f}_t^L$ to initialize the hidden states of an RNN decoder, and employ the decoder to predict the units following x_t. Specifically, the RNN decoder maximizes the conditional log probability $\log P(x_{t+1}, \cdots, x_{t+D}|x_t, \mathbf{f}_t^l)$ using an auto-regressive factorization as in standard language modeling Yang et al. [2017a] (also see Figure 7.2). Here D is a hyper-parameter called the *context length*. The overall objective is written as the sum of the objectives at all positions t,

$$\max \sum_t \log P(x_{t+1}, \cdots, x_{t+D}|x_t, \mathbf{f}_t^L) \tag{7.3}$$

Because our objective is context prediction, we mask the convolutional filters and the graph $\mathbf{G}$ (see Eq. 7.1) in the network g to prevent the network from accessing the future, following Salimans et al. [2017].

7.3.1.1 Desiderata

There are several key desiderata of the above unsupervised learning framework, which also highlight the essential differences between our framework and previous work on self-attention and predictive unsupervised learning:

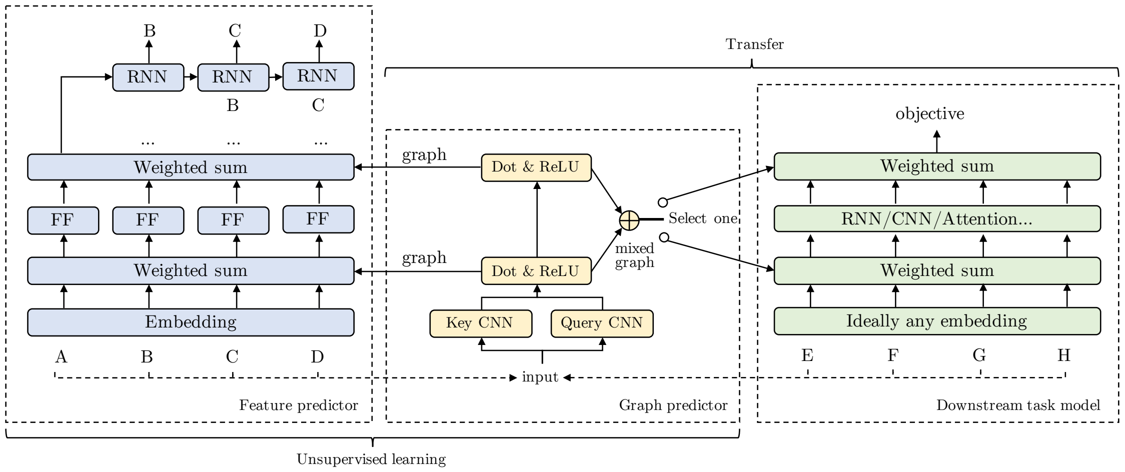

Figure 7.2: Overview of our approach GLoMo. During the unsupervised learning phase, the feature predictor and the graph predictor are jointly trained to perform context prediction. During the transfer phase, the graph predictor is frozen and used to extract graphs for the downstream tasks. An RNN decoder is applied to all positions in the feature predictor, but we only show the one at position "A" for simplicity. "Select one" means the graphs can be transferred to any layer in the downstream task model. "FF" refers to feed-forward networks. The graphs output by the graph predictor are used as the weights in the "weighted sum" operation (see Eq. 7.2).

Decoupling graphs and features Unlike self-attention Vaswani et al. [2017] that fuses the computation of graphs and features into one network, we employ separate networks g and f for learning graphs and features respectively. The features represent the semantic meaning of each unit while the graph reflects how the units may interact. This increases the transferability of the graphs $\mathbf{G}$ because (1) the graph predictor g is freed from encoding task-specific non-structural information, and (2) the decoupled setting is closer to our transfer setting, where the graphs and features are also separated.

Sparsity Instead of using Softmax for attention Bahdanau et al. [2014], we employ a squared ReLU activation in Eq. (7.1) to enforce sparse connections in the graphs. In fact, most of the linguistically meaningful structures are sparse, such as parse trees and coreference links. We believe sparse structures reduce noise and are more transferable.

Hierarchical graph representations We learn multiple layers of graphs, which allows us to model hierarchical structures in the data.

Unit-level objectives In Eq. (7.3), we impose a context prediction objective on each unit x_t. An alternative is to employ a sequence-level objective such as predicting the next sentence Kiros et al. [2015] or translating the input into another language Vaswani et al. [2017]. However, since the weighted sum operation in Eq. (7.2) is permutation invariant, the features in each layer can be randomly shuffled without affecting the objective, which we observed in our preliminary experiments. As a result, the induced graph bears no relation to the structures underlying the input x when a sequence-level objective is employed.

Sequence prediction As opposed to predicting just the immediate next unit Parmar et al. [2018]; Peters et al. [2018], we predict the context of length up to D. This gives stronger training signals to the unsupervised learner.

Later in the experimental section, we will demonstrate that all these factors contribute to successful training of our framework.

7.3.2 Latent Graph Transfer

In this section, we discuss how to transfer the graph predictor g to downstream tasks.

Suppose for a downstream task the model is a deep multi-layer network. Specifically, each layer is denoted as a function h that takes in features $\mathbf{H} = (\mathbf{h}_1, \cdots, \mathbf{h}_T)$ and possibly additional inputs, and outputs features $(\mathbf{h}'_1, \cdots, \mathbf{h}'_T)$. The function h can be instantiated as any neural network component, such as CNNs, RNNs, attention, and feed-forward networks. This setting is general and subsumes the majority of modern neural architectures.

Given an input example x' from the downstream task, we apply the graph predictor to obtain the graphs $\mathbf{G} = g(x')$. Let $\mathbf{\Lambda}^l = \prod_{i=1}^{l} \mathbf{G}^i \in \mathbb{R}^{T \times T}$ denote the product of all affinity matrices from the first layers to the l-th layer. This can be viewed as propagating the connections among multiple layers of graphs, which allows us to model hierarchical structures. We then take a mixture of all the graphs in $\{\mathbf{G}^l\}_{l=1}^L \cup \{\mathbf{\Lambda}^l\}_{l=1}^L$,

$$\mathbf{M} = \sum_{l=1}^{L} m_G^l \mathbf{G}^l + \sum_{l=1}^{L} m_\Lambda^l \mathbf{\Lambda}^l, \quad \text{s.t.} \quad \sum_{l=1}^{L} (m_G^l + m_\Lambda^l) = 1$$

The mixture weights m_G^l and m_Λ^l can be instantiated as Softmax-normalized parameters Peters et al. [2018] or can be conditioned on the features $\mathbf{H}$. To transfer the mixed latent graph, we again adopt the weighted sum operation as in Eq. (7.2). Specifically, we use the weighted sum $\mathbf{HM}$ (see Figures 7.1 and 7.2), in addition to $\mathbf{H}$, as the input to the function h. This can be viewed as performing attention with weights given by the mixed latent graph $\mathbf{M}$. This setup of latent graph transfer is general and easy to be plugged in, as the graphs can be applied to any layer in the network architecture, with either learned or pretrained features $\mathbf{H}$, at variable length.

7.3.3 Extensions and Implementation

So far we have introduced a general framework of unsupervised latent graph learning. This framework can be extended and implemented in various ways.

In our implementation, at position t, in addition to predicting the forward context $(x_{t+1}, \cdots, x_{t+D})$, we also use a separate network to predict the backward context $(x_{t-D}, \cdots, x_{t-1})$, similar to Peters et al. [2018]. This allows the graphs $\mathbf{G}$ to capture both forward and backward dependencies, as graphs learned from one direction are masked on future context. Accordingly, during transfer, we mix the graphs from two directions separately.

In the transfer phase, there are different ways of effectively fusing $\mathbf{H}$ and $\mathbf{HM}$. In practice, we feed the concatenation of $\mathbf{H}$ and a gated output, $\mathbf{W}_1[\mathbf{H}; \mathbf{HM}] \odot \sigma(\mathbf{W}_2[\mathbf{H}; \mathbf{HM}])$, to the function h. Here $\mathbf{W}_1$ and $\mathbf{W}_2$ are parameter matrices, σ denotes the sigmoid function, and $\odot$ denotes element-wise multiplication. We also adopt the multi-head attention Vaswani et al. [2017] to produce multiple graphs per layer. We use a mixture of the graphs from different heads for transfer.

It is also possible to extend our framework to 2-d or 3-d data such as images and videos. The adaptations needed are to adopt high-dimensional attention Wang et al. [2017]; Parmar et al. [2018], and to predict a high-dimensional context (e.g., predicting a grid of future pixels). As an example, in our experiments, we use these adaptations on the task of image classification.

7.4 Experiments

7.4.1 Natural Language Tasks and Setting

Question Answering The stanford question answering dataset Rajpurkar et al. [2016](SQuAD) was recently proposed to advance machine reading comprehension. The dataset consists of more than 100,000+ question-answer pairs from 500+ Wikipedia articles. Each question is associated with a corresponding reading passage in which the answer to the question can be deduced.

Natural Language Inference We chose to use the latest Multi-Genre NLI

Table 7.1: Main results on natural language datasets. Self-attention modules are included in all baseline models. All baseline methods are feature-based transfer learning methods, including ELMo and GloVe. Our methods combine graph-based transfer with feature-based transfer. Our graphs operate on various sets of features, including GloVe embeddings, ELMo embeddings, and RNN states. "mism." refers to the "mismatched" setting.

Transfer method	SQuAD GloVe		SQuAD ELMo	
	EM	*F1*	*EM*	*F1*
transfer feature only (baseline)	69.33	78.73	74.75	82.95
GLoMo on embeddings	70.84	79.90	**76.00**	**84.13**
GLoMo on RNN states	**71.30**	**80.24**	76.20	83.99

Transfer method	MNLI GloVe		IMDB GloVe
	matched	*mism.*	*Accuracy*
transfer feature only (baseline)	77.14	77.40	88.51
GLoMo on embeddings	**78.32**	**78.00**	**89.16**

corpus (MNLI) Williams et al. [2017]. This dataset has collected 433k sentence pairs annotated with textual entailment information. It uses the same modeling protocol as SNLI dataset Bowman et al. [2015a] but covers a 10 different genres of both spoken and formal written text. The evaluation in this dataset can be set up to be in-domain (Matched) or cross-domain (Mismatched). We did not include the SNLI data into our training set.

Sentiment Analysis We use the movie review dataset collected in Maas et al. [2011], with 25,000 training and 25,000 testing samples crawled from IMDB.

Transfer Setting We preprocessed the Wikipedia dump and obtained a corpus of over 700 million tokens after cleaning html tags and removing short paragraphs. We trained the networks g and f on this corpus as discussed in Section 7.3.1. We used randomly initialized embeddings to train both g and f, while the graphs are tested on other embeddings during transfer. We transfer the graph predictor g to a downstream task to extract graphs, which are then used for supervised training, as introduced in Section 7.3.2. We experimented with applying the transferred graphs to various sets of features, including GloVe embeddings, ELMo embeddings, and the first RNN layer's output.

7.4.2 Main results

On SQuAD, we follow the open-sourced implementation Clark and Gardner [2017] except that we dropped weight averaging to rule out ensembling effects. This model employs a self-attention layer following the bi-attention layer, along with multiple layers of RNNs. On MNLI, we adopt the open-sourced implementation Chen et al. [2017a]. Additionally, we add a self-attention layer after the bi-inference component to further model context dependency. For IMDB, our baseline utilizes a feedforward network architecture composed of RNNs, linear layers and self-attention. Note the state-of-the-art (SOTA) models on these datasets are Yu et al. [2018]; Liu et al. [2018b]; Miyato et al. [2016] respectively. However, these SOTA results often rely on data augmentation Yu et al. [2018], semi-supervised learning Miyato et al. [2016], additional training data (SNLI) Liu et al. [2018a], or specialized architectures Liu et al. [2018a]. In this work, we focus on competitive baselines with general architectures that the SOTA models are based on to test the graph transfer performance and exclude independent influence factors.

The main results are reported in Table 7.1. There are three important messages. First, we have purposely incorporated the self-attention module into **all** of our baselines models—indeed having self-attention in the architecture could potentially induce a supervisedly-trained graph, because of which one may argue that this graph could replace its unsupervised counterpart. However, as is shown in Table 7.1, augmenting training with unsupervisedly-learned graphs has further improved

Table 7.2: Ablation study.

Method	SQuAD GloVe		SQuAD ELMo		IMDB GloVe	MNLI GloVe	
	EM	*F1*	*EM*	*F1*	*Accuracy*	*matched*	*mism.*
GLoMo	**70.84**	**79.90**	**76.00**	**84.13**	**89.16**	**78.32**	78.00
- decouple	70.45	79.56	75.89	83.79	-	-	-
- sparse	70.13	79.34	75.61	83.89	88.96	78.07	77.75
- hierarchical	69.92	79.23	75.70	83.72	88.71	77.87	77.85
- unit-level	69.23	78.66	74.84	83.37	88.49	77.58	**78.05**
- sequence	69.92	79.29	75.50	83.70	88.96	78.11	77.76
uniform graph	69.48	78.82	75.14	83.28	88.57	77.26	77.50

performance. Second, as we adopt pretrained embeddings in **all** the models, the baselines establish the performance of feature-based transfer. Our results in Table 7.1 indicate that when combined with feature-based transfer, our graph transfer methods are able to yield further improvement. Third, the learned graphs are generic enough to work with various sets of features, including GloVe embeddings, ELMo embeddings, and RNN output.

7.4.3 Ablation Study

In addition to comparing graph-based transfer against feature-based transfer, we further conducted a series of ablation studies. Here we mainly target at the following components in our framework: decoupling feature and graph networks, sparsity, hierarchical (i.e. multiple layers of) graphs, unit-level objectives, and sequence prediction. Respectively, we experimented with coupling the two networks, removing the ReLU activations, using only a single layer of graphs, using a sentence-level Skip-thought objective Kiros et al. [2015], and reducing the context length to one Peters et al. [2018]. As is shown in Table 7.2, all these factors contribute to better performance of our method, which justifies our desiderata discussed in Section 7.3.1.1. Additionally, we did a sanity check by replacing the trained graphs

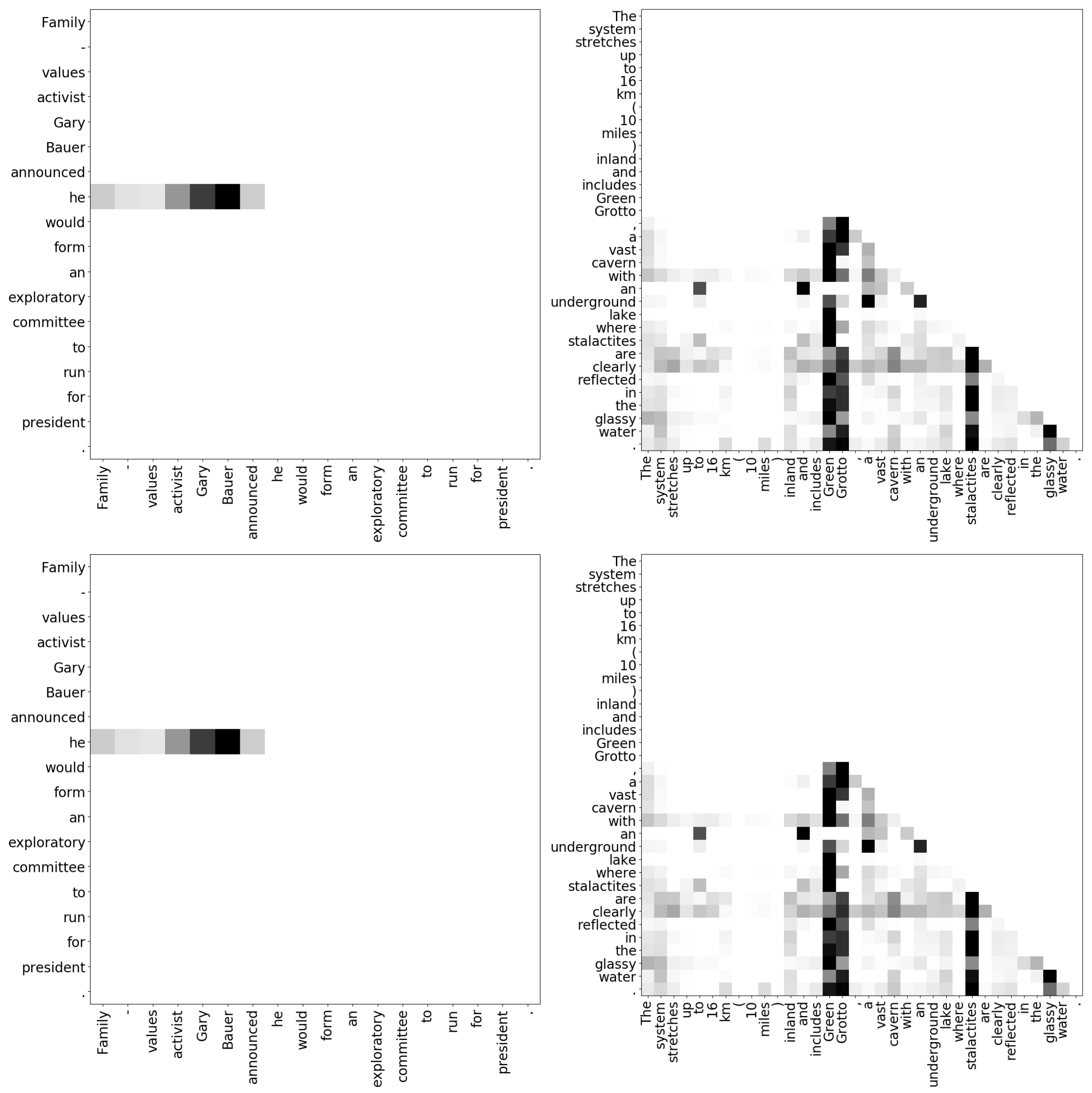

Figure 7.3: Visualization of the graphs on the MNLI dataset. The graph predictor has not been trained on MNLI. The words on the y-axis "attend" to the words on the a-axis; i.e., each row sums to 1.

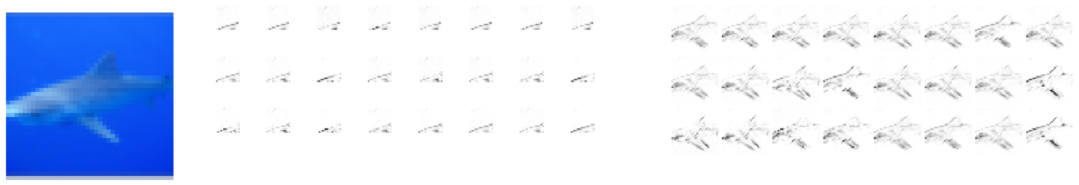

Figure 7.4: Visualization. Left: a shark image as the input. Middle: weights of the edges connected with the *central* pixel, organized into 24 heads (3 layers with 8 heads each). Right: weights of the edges connected with the *bottom-right* pixel. Note the use of masking.

with uniformly sampled affinity matrices (similar to Hu et al. [2017]) during the transfer phase. This result shows that the learned graphs have played a valuable role for transfer.

7.4.4 Visualization and Analysis

We visualize the latent graphs on the MNLI dataset in Figure 7.3. We remove irrelevant rows in the affinity matrices to highlight the key patterns. The graph in Figure 7.3 top-left resembles coreference resolution as “he” is attending to “Gary Bauer”. In Figure 7.3 top-right, the words attend to the objects such as “Green Grotto”, which allows modeling long-term dependency when a clause exists. In Figure 7.3 bottom-left, the words following “not” attend to “not” so that they are aware of the negation; similarly, the predicate “splashed” is attended by the following object and adverbial. Figure 7.3 bottom-right possibly demonstrates a way of topic modeling by attending to informative words in the sentence. Overall, though seemingly different from human-curated structures such as parse trees, these latent graphs display linguistic meanings to some extent. Also note that the graph predictor has not been trained on MNLI, which suggests the transferability of the latent graphs.

Method / Base-model	ResNet-18	ResNet-34
baseline	90.93±0.33	91.42±0.17
GLoMo	**91.55±0.23**	**91.70±0.09**
ablation: uniform graph	91.07±0.24	-

Table 7.3: CIFAR-10 classification results. We adopt a 42,000/8,000 train/validation split—once the best model is selected according to the validation error, we directly forward it to the test set without doing any validation set place-back retraining. We only used horizontal flipping for data augmentation. The results are averaged from 5 rounds of experiments.

7.4.5 Vision Task

Image Classification We are also prompted to extend the scope of our approach from natural language to vision domain. Drawing from natural language graph predictor $g(\cdot)$ leads the unsupervised training phase in vision domain to a PixelCNN-like setup Oord et al. [2016], but with a sequence prediction window of size 3x3 (essentially only predicting the bottom-right quarter under the mask). We leverage the entire ImageNet Deng et al. [2009] dataset and have the images resized to 32x32 Oord et al. [2016]. In the transfer phase, we chose CIFAR-10 classification as our target task. Similar to the language experiments, we augment $\mathbf{H}$ by $\mathbf{HM}$, and obtain the final input through a gating layer. This result is then fed into a ResNet He et al. [2016] to perform regular supervised training. Two architectures, i.e. ResNet-18 and ResNet-34, are experimented here. As shown in Table 7.3, GLoMo improves performance over the baselines, which demonstrates that GLoMo as a general framework also generalizes to images.

In the meantime we display the attention weights we obtain from the graph predictor in Figure 7.4. We can see that g has established the connections from key-point pixels while exhibiting some variation across different attention heads.

There has been similar visualization reported by Zhang et al. [2018] lately, in which a vanilla transformer model is exploited for generative adversarial training. Putting these results together we want to encourage future research to take further exploration into the relational long-term dependency in image modeling.

7.5 Discussion

We present a novel transfer learning scheme based on latent relational graph learning, which is orthogonal to but can be combined with the traditional feature transfer learning framework. Through a variety of experiments in language and vision, this framework is demonstrated to be capable of improving performance and learning generic graphs applicable to various types of features. In the future, we hope to extend the framework to more diverse setups such as knowledge based inference, video modeling, and hierarchical reinforcement learning where rich graph-like structures abound.

Chapter 8

Conclusions and Future Work

Chapter 3 was published as Zhao et al. [2016]. Chapter 4 was published as Zhao et al. [2018], successfully showcasing real-world applications of text generation in style transfer. Chapter 5 is in submission. Chapter 6 was published as Zhao et al. [2015], demonstrating state-of-the-art performance of semi-supervised leraning on datasets like STL-10. Chapter 7 was published as Yang et al. [2018].

Unsupervised learning and its deployment remain as one of the most important missions for artificial intelligence. For any specific tasks, utilizing an unsupervised loss objective to acquire some form of *common sense* would certainly be useful to benefit low-resource tasks, or improve the sample efficiency in other paradigms like reinforcement learning. The current progress of unsupervised transfer learning has only scratched the surface on this regard. In the future, we hope to work on a clearer and more discrete definition of what common sense is. Briefly, it needs to prove the universality of learned features like BERT and ELMo, or models in the meta-learning scenario, or something more comprehensive. Furthermore, *uncertainty* is another important emerging concept in unsupervised learning, particularly in the predive modeling. How to (i)-quantify the uncertainty, (ii)-build efficient inference algorithm and (iii)-to use the uncertainty in a problem such as planning remain as unsolved challenges.

The evaluation is another dimension of the challenge, in particular for generative modeling. Albeit the inception and FID scores, generative modeling in computer vision is still heavily evaluated based on human annotators. Unfortunately, it is even more so in natural language generation tasks. Indeed, there are multiple aspects to be taken care of when evaluating the outcome of a generative model—it may include but not limited to generation quality, diversity, mode-coverage, uncertainty reflection, etc. Depending on the actual generation task, as well as being conditioned on different content, the generative models ought to be evaluated differently and accordingly. In the future, we hope to explore systematic ways to evaluate generative modeling.

Bibliography

Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C. L., Parikh, D., and Batra, D. (2017). Vqa: Visual question answering. *International Journal of Computer Vision*, 123(1):4–31.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv:1701.07875*.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2017). Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Boureau, Y.-l., Cun, Y. L., et al. (2008). Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pages 1185–1192.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015a). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Bowman, S. R., Angeli, G., Potts, C., and Manning., C. D. (2015b). A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015c). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Carreira-Perpinan, M. A. and Hinton, G. (2005). On contrastive divergence learning. In *AISTATS*, volume 10, pages 33–40. Citeseer.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y. (2017). Maximum-Likelihood Augment Discrete Generative Adversarial Networks. *arXiv:1702.07983*.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017a). Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017b). Variational Lossy Autoencoder. In *Proceedings of ICLR*.

Chen, Y., Li, V. O., Cho, K., and Bowman, S. R. (2018). A stable and effective learning strategy for trainable greedy decoding. *arXiv preprint arXiv:1804.07915*.

Cho, K. (2016). Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Clark, C. and Gardner, M. (2017). Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. In *Proceedings of NIPS*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Denton, E. and Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. *arXiv preprint arXiv:1705.10915*.

Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Fevry, T. and Phang, J. (2018). Unsupervised sentence compression using denoising auto-encoders. *arXiv preprint arXiv:1809.02669*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Glynn, P. (1987). Likelihood Ratio Gradient Estimation: An Overview. In *Proceedings of Winter Simulation Conference*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goroshin, R., Mathieu, M., and LeCun, Y. (2015). Learning to linearize under uncertainty. *arXiv preprint arXiv:1506.03011.*

Graff, D., Kong, J., Chen, K., and Maeda, K. (2003). English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proc. International Conference on Machine learning (ICML'10).*

Gu, J., Cho, K., and Li, V. O. (2017). Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429.*

Gulrajani, I., Ahmed, F., Arjovsky, M., and Vincent Dumoulin, A. C. (2017). Improved Training of Wasserstein GANs. *arXiv:1704.00028.*

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Henaff, M., Jarrett, K., Kavukcuoglu, K., and LeCun, Y. (2011). Unsupervised learning of sparse features for scalable audio classification. In *Proceedings of International Symposium on Music Information Retrieval (ISMIR'11).*

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL.*

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

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders.

In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 44–51. Springer.

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

Hjelm, R. D., Jacob, A. P., Che, T., Cho, K., and Bengio, Y. (2017). Boundary-Seeking Generative Adversarial Networks. *arXiv:1702.08431*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*.

Im, D. J., Kim, C. D., Jiang, H., and Memisevic, R. (2016). Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*.

Jang, E., Gu, S., and Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of ICLR*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kaiser, Ł., Roy, A., Vaswani, A., Pamar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Kavukcuoglu, K., Ranzato, M., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE.

Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2008). Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU. Tech Report CBLL-TR-2008-12-01.

Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2010a). Fast inference in sparse coding algorithms with applications to object recognition. *arXiv preprint arXiv:1010.3467*.

Kavukcuoglu, K., Sermanet, P., Boureau, Y., Gregor, K., Mathieu, M., and LeCun, Y. (2010b). Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, volume 23.

Kavukcuoglu, K., Sermanet, P., Boureau, Y.-L., Gregor, K., Mathieu, M., and Cun, Y. L. (2010c). Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pages 1090–1098.

Kim, T. and Bengio, Y. (2016). Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

Kingma, D. P., Salimans, T., and Welling, M. (2016). Improving Variational Inference with Autoregressive Flow. *arXiv:1606.04934*.

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

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kusner, M. and Hernandez-Lobato, J. M. (2016). GANs for Sequences of Discrete Elements with the Gumbel-Softmax Distribution. *arXiv:1611.04051*.

Lamb, A. M., Goyal, A., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Lample, G., Zeghidour, N., Usuniera, N., Bordes, A., Denoyer, L., and Ranzato, M.

(2017). Fader networks: Manipulating images by sliding attributes. In *Proceedings of NIPS*.

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Chopra, S., and Hadsell, R. (2006). A tutorial on energy-based learning.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2014). Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2017). Adversarial Learning for Neural Dialogue Generation. *arXiv:1701.06547*.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018a). Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Liu, X., Duh, K., and Gao, J. (2018b). Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

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

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of ICLR*.

Makhzani, A. and Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*.

Makhzani, A. and Frey, B. (2017). PixelGAN Autoencoders. *arXiv:1706.00531*.

Makhzani, A. and Frey, B. J. (2015). Winner-take-all autoencoders. In *Advances in neural information processing systems*, pages 2791–2799.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial Autoencoders. *arXiv:1511.05644*.

Marc'Aurelio Ranzato, C. P. and Chopra, S. (2007). Efficient learning of sparse representations with an energy-based model.

Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. *arXiv:1701.04722*.

Miao, Y. and Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., tau Yih, S. W., and Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of NAACL*.

Miyato, T., Dai, A. M., and Goodfellow, I. (2016). Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Ng, A. et al. (2013). Sparse autoencoder.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Paine, T. L., Khorrami, P., Han, W., and Huang, T. S. (2014). An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., and Ku, A. (2018). Image transformer. *arXiv preprint arXiv:1802.05751*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pezeshki, M., Fan, L., Brakel, P., Courville, A., and Bengio, Y. (2015). Deconstructing the ladder network architecture. *arXiv preprint arXiv:1511.06430*.

Press, O., Bar, A., Bogin, B., Berant, J., and Wolf, L. (2017). Language Generation with Recurrent Generative Adversarial Networks without Pre-training. *arXiv:1706.01399*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proceedings of ICLR*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ranzato, M., Boureau, Y., Chopra, S., and LeCun, Y. (2007a). A unified energy-based framework for unsupervised learning. In *Proc. Conference on AI and Statistics (AI-Stats)*.

Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. (2007b). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Ranzato, M. and LeCun, Y. (2007). A sparse and locally shift invariant feature extractor applied to document images. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1213–1217. IEEE.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Rezende, D. J. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *Proceedings of ICML*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of ICML*.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011a). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings* *of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011b). Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of ICML*.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sai Rajeswar, Sandeep Subramanian, F. D. C. P. A. C. (2017). Adversarial Generation of Natural Language. *arXiv:1705.10929*.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*.

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

Semeniuta, S., Severyn, A., and Barth, E. (2017). A Hybrid Convolutional Variational Autoencoder for Text Generation. *arXiv:1702.02390*.

Shen, T., Lei, T., Barzilay, R., and Jaakkola, T. (2017). Style Transfer from Non-Parallel Text by Cross-Alignment. *arXiv:1705.09655*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. *arXiv preprint arXiv:1507.06228*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *Proceedings of ICLR*.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017). Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1958–1970.

Tran, D., Ranganath, R., and Blei, D. M. (2017). Deep and Hierarchical Implicit Models. *arXiv:1702.08896*.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798.

Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of ICML*.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wang, X., Girshick, R., Gupta, A., and He, K. (2017). Non-local neural networks. *arXiv preprint arXiv:1711.07971*.

Wang, Y.-S. and Lee, H.-Y. (2018). Learning to encode text as human-readable summaries using generative adversarial networks. *arXiv preprint arXiv:1810.02851*.

Williams, A., Nangia, N., and Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Williams, R. J. (1992a). Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8.

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

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. (2017a). Breaking the softmax bottleneck: a high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

Yang, Z., Dhingra, B., He, K., Cohen, W. W., Salakhutdinov, R., LeCun, Y., et al. (2018). Glomo: Unsupervisedly learned relational graphs as transferable representations. *arXiv preprint arXiv:1806.05662*.

Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. (2017b). Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. In *Proceedings of ICML*.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le,

Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of AAAI*.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhao, J., Kim, Y., Zhang, K., Rush, A., and LeCun, Y. (2018). Adversarially regularized autoencoders. In *International Conference on Machine Learning*, pages 5897–5906.

Zhao, J., Mathieu, M., Goroshin, R., and Lecun, Y. (2015). Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*.

Zhao, J. J., Mathieu, M., and LeCun, Y. (2016). Energy-based generative adversarial network. *CoRR*, abs/1609.03126.