Variational Inference for Text Generation: Improving the Posterior

by

Vikash Balasubramanian

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

Waterloo, Ontario, Canada, 2020

© Vikash Balasubramanian 2020

Author Declaration

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

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

Statement of Contributions

Chapter 4 is based on the following paper:

- Vikash Balasubramanian, Ivan Kobyzev, Hareesh Bahuleyan, Ilya Shapiro, and Olga Vechtomova. "Polarized-VAE: Proximity Based Disentangled Representation Learning for Text Generation." arXiv preprint arXiv:2004.10809 (2020).

I have contributed to implementation, experimentation, and preparation of the manuscript of the above mentioned papers.

Abstract

Learning useful representations of data is a crucial task in machine learning with wide ranging applications. In this thesis we explore improving representations of models based on variational inference by improving the posterior. We explore two approaches towards this goal: 1) auxiliary losses to regularize the latent space and enforcing desired properties and 2) normalizing flows to develop more flexible posteriors to be used during variational inference.

We propose a proximity based loss function that helps in disentanglement by regularizing the latent space based on similarity according to a criterion. We evaluate our model on a task of disentangling semantics and syntax in sentences and empirically show that our model successfully manages to learn independent subspaces that learn semantics and syntax respectively. We compare our model to existing approaches using automated metrics and human evaluation to show that our model is competitive.

We also explore the effectiveness of normalizing flows for representation learning and generative modeling. We perform experiments that empirically show that variational inference with normalizing flows beats standard approaches based on simple posteriors across various metrics in text generation and language modeling. We also propose a variant of planar normalizing flows called block planar normalizing flows for use in disentanglement tasks. We perform ablation experiments to empirically show that our proposed block planar flows help in improving disentanglement.

Acknowledgements

I would like to thank all the people that have supported me and made it possible for me to achieve this milestone in my academic career.

I take this opportunity to express my immense gratitude for the support and guidance of my supervisor Prof. Olga Vechtomova. She was patient and kind, and allowed me great flexibility and freedom in pursuing my research interests. I am grateful for the time she spent on reviewing and providing valuable inputs for my thesis.

I would like to thank Hareesh Bahuleyan for being a wonderful mentor. He was patient and always ready to answer my questions and has been a great help in developing my skills as a researcher and as a graduate student.

I thank Dr. Ivan Kobyzev and Dr. Ilya Shapiro for being wonderful co-authors and more importantly amazing mentors that have guided me and provided valuable feedback for my ideas and helped me in broadening the horizons of my research interests.

I am thankful to Borealis AI and Mitacs for their support and opportunities provided for my research during my internship (May 2019 - September 2019).

I am grateful to Professor Pascal Poupart and Professor Yaoliang Yu for taking the time to review my thesis and provide valuable feedback.

I want to thank Gaurav Sahu and Kashif Khan for being wonderful fellow students. The interesting discussions we had about all of our respective research topics helped me greatly in broadening my horizons.

I am also thankful for the friends that I made in the University and the friends that I made in the Columbia Lake Village Community. They played a big part in helping me enjoy my stay in Canada.

Finally, I would like to express my gratitude to my parents and my sister for believing in me and for offering their unwavering support no matter the situation.

Dedication

I dedicate this thesis to my sister for her unconditional love, for being understanding when I was too busy to call her back and for her unbridled excitement about my research that helped inspire me to always do better.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence 1

BPNF Block Planar Normalizing Flow 41

ELBO evidence lower bound 14

LSTM Long Short Term Memory 8, 9, 24, 34, 45

MLP Multilayer Perceptron 5–7

MMD Maximum Mean Discrepancy 16

NLG Natural Language Generation 1, 4

NLP Natural Language Processing 1, 6, 9

NLU Natural Language Understanding 1

PCA Principal Component Analysis 11, 12

PNF Planar Normalizing Flow 41

RNN Recurrent Neural Network 6–8, 32

RNNLM Recurrent Neural Network Language Model 1

Seq2Seq sequence-to-sequence 1

VAE Variational Autoencoder 2, 16, 17, 23, 25, 33

WAE Wasserstein Autoencoder 15–17, 25, 26

Chapter 1

Introduction

1.1 Background

Natural Language Processing (NLP) is a sub field of Artificial Intelligence (AI) that is concerned with the interactions between a computer and human (natural) languages. Some of the prominent areas in NLP include speech recognition, Natural Language Understanding (NLU) and Natural Language Generation (NLG). There has been extensive interest in the area in recent times leading to impressive applications such as machine translation, reading comprehension, dialogue response generation and poetry generation. The emergence of neural networks and deep learning, in particular has been very beneficial to this end and have slowly replaced earlier statistical approaches.

The fields NLU and NLG are complimentary. NLU focuses on understanding natural languages by learning structured representations of natural text, whereas NLG aims to generate natural text from structured representations. Generating text similar to humans is a difficult task, because the generated text has to be coherent, follow the syntax rules of the language and also be meaningful semantically. A lot of recent success in this area can be traced back to the success of the Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2010) and neural sequence-to-sequence (Seq2Seq) models (Sutskever et al., 2014). Neural Seq2Seq models take a sequence of words as input and generate a sequence of words as output using neural networks.

One key drawback of the standard RNNLM is that it does not work from an explicit global sentence representation and as such is not directly useful towards representation learning. Representation learning is an important goal because it allows us to manipulate

the representations using mathematical operators and explore the properties of the induced spaces. The Variational Autoencoder (VAE) introduced in 2014 (Kingma & Welling, 2014) has been crucial to the recent strides in representation learning. S. R. Bowman et al., 2016 showed that the VAE can be used to generate diverse coherent text by adopting the variational inference procedure.

Recently there has been great interest in research seeking disentangled representations of data (R. T. Chen et al., 2018; X. Chen et al., 2016; Higgins et al., 2017). It has been argued that separating the sources of variation in data into distinct factors that explain the variation in data leads to more interpretable representations (Bengio et al., 2013).

In this work, we explore the existing generative models based on variational inference, and seek to develop techniques that lead to learning better representations that improve generation quality and also explore the applications to learning disentangled representations.

1.2 Motivation

In recent times VAE and related models have emerged as leading candidates for generative models for text. However a common problem with such models is the balancing of two usually competing objectives: 1) learning useful representations for reconstructions and other downstream tasks. 2) Generation quality when sampled from the prior. One of the reasons for such issues is the simple form of the posterior distribution learned by the standard VAE.

Typically VAE models the posterior as belonging to the family of factorized Gaussian distribution (covariance matrix is diagonal). It has been argued that such a distribution is not flexible enough to model comlex data and may have undesirable properties that lead to posterior collapse (J. Xu & Durrett, 2018a). In this work, we explore two possible ways to overcome this problem. 1) We seek to develop auxiliary losses that may help with disentanglement and downstream tasks. 2) We demonstrate that using more flexible posteriors leads to improvements across various metrics for both generation and representation learning.

1.3 Contributions

The contributions of this thesis are multi-fold and listed below.

- We propose a proximity based loss function to regularize the latent space based on the pairwise similarity of sentences based on a given criterion. We empirically show that this could be used to disentangle the latent spaces based on the criteria used.
- Our proposed polarized-VAE does not rely on adversarial training or task specific multitask losses.
- We empirically show that our polarized-VAE model can reasonably disentangle semantics and syntax from a sentence even in an unsupervised setting by using readily computed proxies for semantic and syntactic similarity.
- We propose a variant of normalizing flows called block planar normalizing flows to encourage disentanglement into independent subspaces and derive an expression for the variational lower bound while using such flows.
- We also propose to use a subspace entropy loss to enforce independence in subspaces as an alternative to adversarial training. We empirically show that this subspace entropy loss improves performance on disentanglement tasks for text, when compared to simple baselines that do not enforce any independence constraints.
- By performing ablation studies on the amazon sentiment transfer task we empirically show that in conjunction with block planar normalizing flows, the subspace entropy loss performs competitively with adversarial training while being easier to train.

Chapter 2

Background

2.1 Natural Language Generation

Natural Language Generation in general is a broad area that primarily deals with generation of natural language similar to humans. Early NLG techniques were based on rules defined from a predefined grammar, or used statistical techniques to predict next word based on co-occurrence probabilities (Cambria & White, 2014). Probabilistic, iterative approaches such as Markov Chains were also used. In recent years, approaches based on Neural Networks have become popular and have been widely adopted for different applications of NLG.

Some of the applications and research sub-fields of NLG have attracted recent research interest with the success of deep learning for text, they include, but are not limited to:

- **Dialogue Response Generation**: This task is quite common in chatbots and other similar services. The task is to generate response text conditional on a given source text (utterance). Neural network based encoder decoder approaches have been investigated in this domain in recent years. (Gu et al., 2018; Shen et al., 2018).
- **Machine Translation**: In this task the target sequence to be generated should have the same semantic meaning as a source sentence, but should be in a different natural language. (Eg; English to French). Recently Neural Machine Translation systems have become more and more effective in generating excellent translations. (Bahdanau et al., 2014; Wu et al., 2016)

- **Text Summarization**: In this task the target text to be generated is a concise, coherent summary of the source text. Abstractive summarization in particular is closely related to generation tasks. (Nallapati et al., 2016; Rush et al., 2015).

2.2 Neural Networks

In this section we give a general background on the different types of neural networks and related techniques that we use in this work.

2.2.1 Multi Layer Perceptrons

A Multilayer Perceptron (MLP) is a commonly used class of artificial neural network. An MLP consists of 3 or more layers of nodes; an input layer, one or more hidden layers and an output layer. The size of the input and output layers are determined by the size of the input representation and output representation respectively. The size of the hidden layers is typically a hyperparameter that is tuned for a particular experiment.

Figure 2.1 shows the architecture of an MLP with a two hidden layers. Each neuron unit typically also has a non linear activation function. All the neurons in a layer typically use the same activation function for ease of notation and implementation. The operation of the nodes in a layer can be concisely expressed if they are thought of as a non linear function applied to an affine transformation of outputs from a previous layer, with each layer having it's associated set of weights and biases (possibly fixed to zero). For example, the input to the 1st hidden layer in Figure 2.1 can be expressed as $\phi(W_1\boldsymbol{x} + b_1)$ where ϕ is a pointwise nonlinear function.

MLPs and artificial neural networks in general can be thought of as function approximators whose parameters are learned/updated by an optimization algorithm like **Gradient Descent**. Given a set of weights $\boldsymbol{w}$ and a loss function $\mathcal{L}(\boldsymbol{w})$, the gradient descent step can be given by:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \delta_w \mathcal{L}(\boldsymbol{w}) \tag{2.1}$$

Where η is the learning rate and is a tunable hyperparameter. The required gradients can be computed by a technique known as backpropagation (Rumelhart et al., 1986) which makes use of the chain rule of derivatives. For more details about the learning procedure

using gradient descent and backpropagation readers are referred to the deep learning book (Goodfellow et al., 2016).

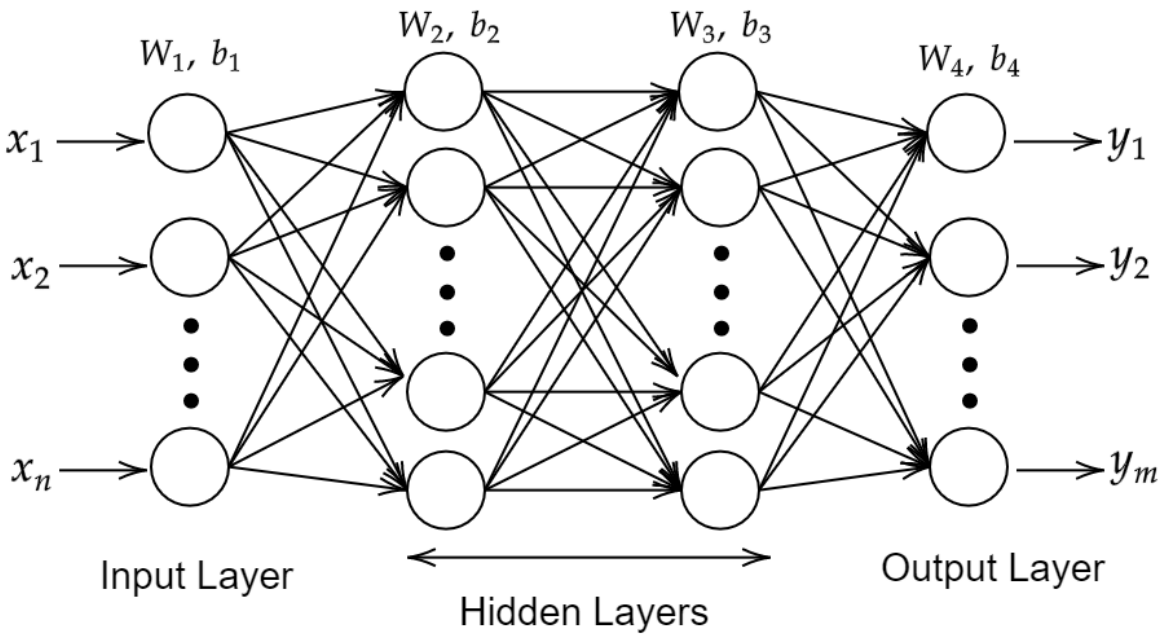


Figure 2.1: Architecture of MLP with two hidden layers.

2.2.2 Recurrent Neural Networks

Recurrent Neural Network (RNN)s (Rumelhart et al., 1986) play a very important role in NLP tasks because they are very good at modeling the sequential relationship between datapoints and can also process sequences of variable lengths pretty easily.

A key idea of RNNs that distinguishes them from MLPs is the sharing of parameters across different parts of the model. This allows the RNN to easily learn shared features across an entire sequence of inputs. For instance consider the two sentences 1) India attained it's independence in 1947. 2) In 1947, India attained independence. A key piece of information; the year occurs in the two sentences at a different position. A RNN would

be able to handle this more easily than a MLP where two neurons in two different position would have to independently learn this information.

A recurrent neural network is equivalent to a network with loops. This equivalence can be thought of as "unrolling the network" and is depicted in Figure 2.2. The input sequence $\boldsymbol{x}$ is fed to the RNN and produces an output sequence $\boldsymbol{o}$ which is compared to the target sequence $\boldsymbol{y}$ using the loss function L (typically the negative log likelihood). Equivalently the unfolded network shows that the output sequence is composed of outputs at each timestep and the losses over each timestep are summed to calculate the total loss.

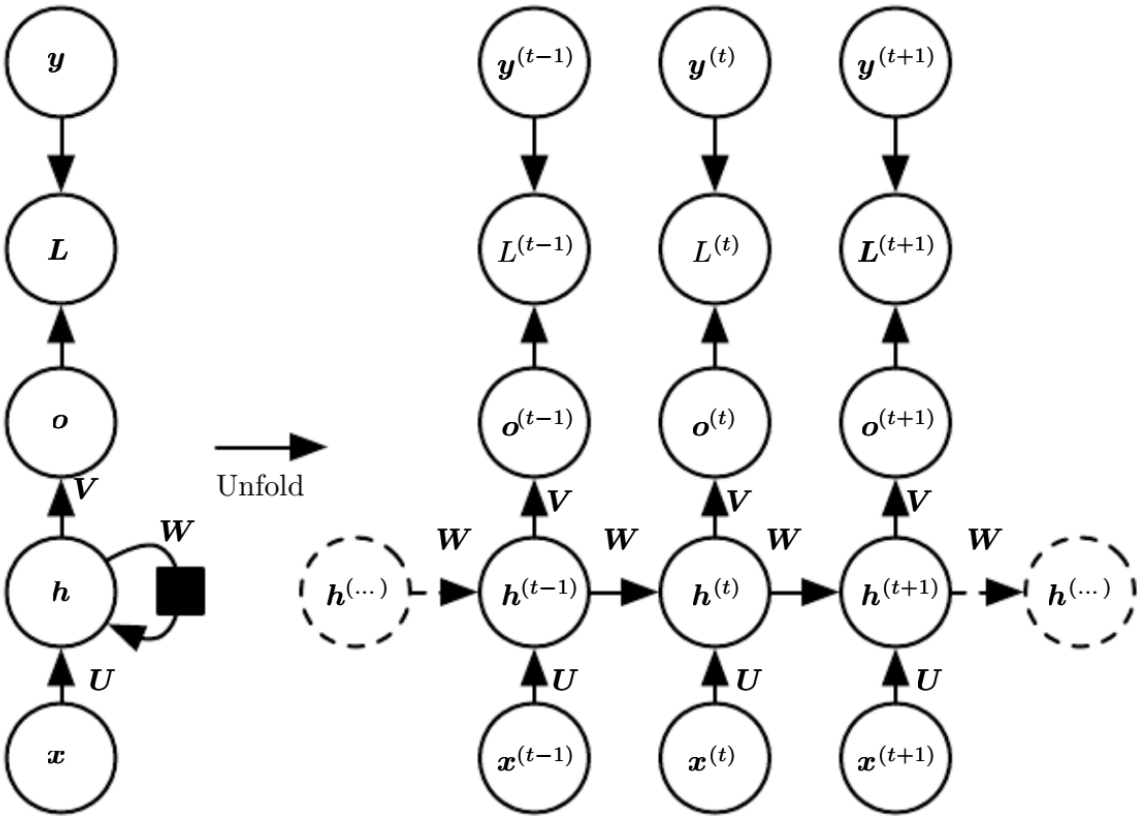


Figure 2.2: Unfolding of the RNN (Goodfellow et al., 2016)

Bidirectional RNNs

RNNs can be extended to take future information into account. A bidirectional RNN is equivalent to two stacked RNNs operating in opposite time directions (forward and backward). The outputs of these RNNs can be pooled in various ways such as averaging and concatenation. Experiments show that bidirectional RNNs often perform better in many tasks (Schuster & Paliwal, 1997).

2.2.3 Long Short Term Memory

RNNs can be difficult to train over long sequences and problems that require learning long term temporal dependencies. They gradients involved in training an RNN are computed using backpropagation through time and they tend to vanish especially over long sequences. This phenomenon termed as the *vanishing gradient* problem lead to research into architectures that can handle long term temporal dependencies. One such architecture is the Long Short Term Memory (LSTM) originally proposed in 1997 (Hochreiter & Schmidhuber, 1997).

The LSTM architecture handles the problem of vanishing gradients by including a "memory cell" that helps with persistence of information. A set of gates are included that help in controlling the storage, retrieval and deletion of information in the memory cell. These gates are the input gate, output gate and the forget gate respectively. The LSTM architecture is shown in Figure 2.3.

The gates can be described by the following equations. It should be noted that all three gates use a sigmoid activation function (σ).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.4}$$

We compute a candidate cell state $\hat{C}_t$ using the following equation:

$$\hat{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.5}$$

The current cell state is a function of the previous cell state and the candidate cell state. They are weighted by the outputs from the forget gate and the input gate respectively by

using the hadamard product $\odot$. The current hidden state is determined by the output gate and the current cell state.

$$C_t = i_t \odot \hat{C}_t + f_t \odot C_{t-1} \tag{2.6}$$
$$h_t = o_t \odot \tanh(C_t) \tag{2.7}$$

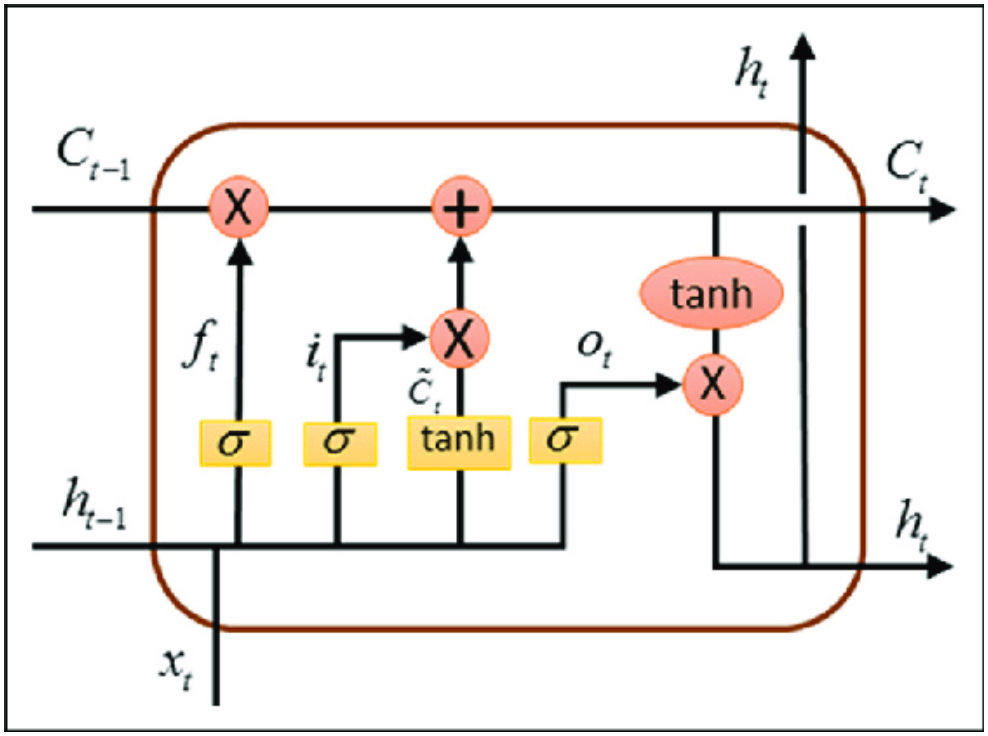


Figure 2.3: LSTM architecture (Tao & Zhou, 2017)

The LSTM architecture has been clearly designed to capture long term dependencies in data. To this end it has been very successful and has been applied to several NLP tasks like sequence tagging (Huang et al., 2015), language modeling (Sundermeyer et al., 2012), text classification (Zhou et al., 2015) and dialogue response generation (Wen et al., 2015).

Similar to RNNs, LSTMs can also be extended to bidirectional for improved performance in several tasks that can make use of future input features from a sequence such as sequence tagging (Huang et al., 2015) and phoneme classification (Graves et al., 2005).

2.2.4 Dropout

Overfitting is a common problem faced by deep neural network architectures. Overfitting refers to the problem of models learning functions that model the training set very well but fail to generalize to the validation/test set. Dropout was proposed as a regularization technique to help mitigate the problem of overfitting (Srivastava et al., 2014).

The motivation behind dropout is quite simple, neurons and their connections are randomly dropped during training. Each neuron can be dropped with a fixed probability that is a tunable hyperparameter. This helps the model prevent overfitting by limiting the development of excessive codependency between units during training. Figure 2.4 illustrates the idea behind dropout. It should be noted that during testing/inference no neurons are dropped.

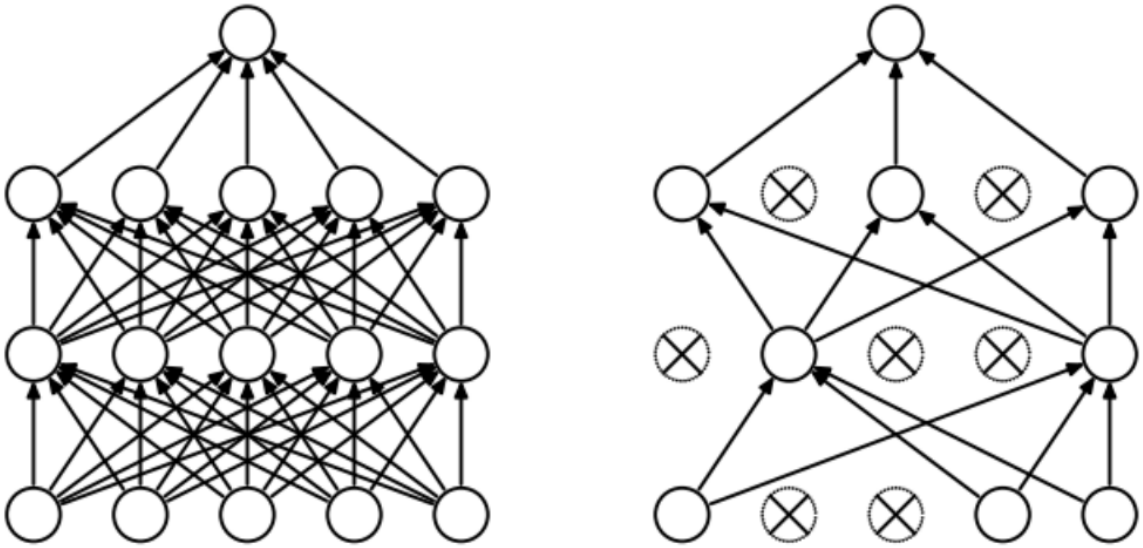


Figure 2.4: Illustration of Dropout (Srivastava et al., 2014)

2.3 Representation Learning

Learning useful representations of data and its features is a crucial problem in machine learning because it signifies a good understanding of the data and its properties. A good representation of data can greatly simplify the learning process and improve results.

For instance in probabilistic models, learning a good posterior that explains the factors of variation in data would be greatly helpful in many tasks including generation and classification.

Bengio et al. (2013) examine several general purpose priors likely to be useful for many AI tasks. One such prior is the learning of manifolds (Section 2.3.1). They also provide several possible factors to consider while examining the usefulness of learned representations. One such factor that we consider in this work is the disentanglement of factors of variation (Section 2.3.6).

In this section we describe the various concepts and techniques that are relevant to our work.

2.3.1 Manifold Learning

A Manifold is a topological space that locally resembles a Euclidean space. The manifold hypothesis states that real world data in high dimensional spaces tend to lie in the vicinity of a low dimensional manifold embedded in a high dimensional space (Fefferman et al., 2016).

This hypothesis provides us with an important perspective of representation learning and forms the backbone of manifold learning.Principal Component Analysis (PCA), which is used extensively for dimensionality reduction is an excellent illustration of manifold learning. In our work we will be mainly focused on specific types of neural network architectures that are useful for manifold learning: autoencoders (Kramer, 1991).

Autoencoders are neural networks that simply learn to reconstruct their inputs. They typically consist of an encoder and a decoder. The encoder learns to map the inputs into a lower dimensional manifold, typically a euclidean latent space with a fixed dimensionality. The decoder learns to reconstruct the input from the latent vectors in the learned manifold. Typically the dimensionality of the latent space learned is fixed to be less than the input size so that only the salient features of the input data may be learned.

If the dimensionality of the latent space is large enough and/or if the encoders and decoders are powerful enough, sometimes the task of reconstruction becomes very simple and the learned representations may become useless for other downstream tasks. Thus, typically many autoencoders employ some form of regularization to improve the quality of representations learned. Some common regularized autoencoders are sparse autoencoders, denoising autoencoders and contractive autoencoders (Goodfellow et al., 2016).

Recently, there has also been great interest in learning non euclidean latent spaces. It has been suggested that for some datasets, generative modeling can be improved by modeling the curvature of the latent space while measuring distances (Arvanitidis et al., 2017).

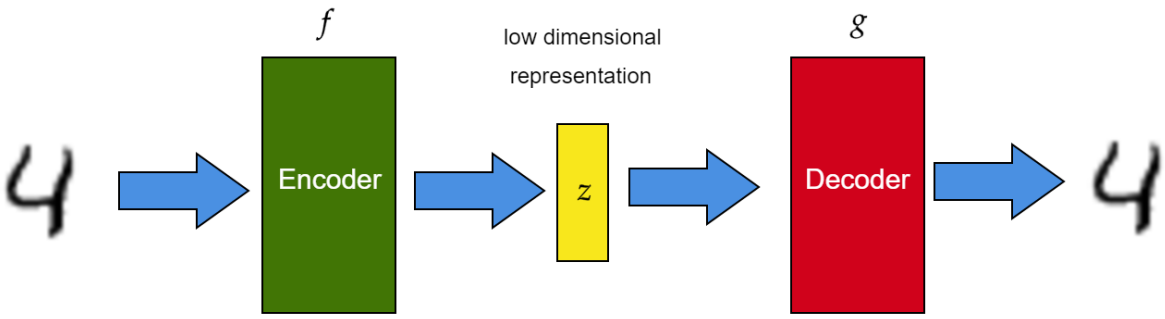


Figure 2.5: Architecture of an autoencoder

The architecture of a typical autoencoder is shown in Figure 2.5. Given an input x The autoencoder tries to minimize the following objective

$$\mathcal{L}(x, g(f(x))) \tag{2.8}$$

Where f and g are the functions learned by the encoder and decoder respectively and $\mathcal{L}$ is a loss function that penalizes $g(f(x))$ for not being similar to x . Typical loss functions for autoencoders include the mean squared error and the cross entropy loss.

Autoencoders are very useful in dimensionality reduction tasks. The approximation power of neural network architectures when combined with non linearity allows autoencoders to perform nonlinear PCA. Autoencoders have also found applications in information retrieval and various areas for learning useful representations of data such as speech enhancement (Lu et al., 2013), binary coding of speech spectrograms (Deng et al., 2010), deep collaborative filtering for recommender systems (S. Li et al., 2015) and learning bilingual word representations (Chandar A P et al., 2014).

2.3.2 Generative Latent Variable Models

Generative latent variable models work on the basic principle of learning representations by maximizing the likelihood of data given the representation. The representations learned

by such models are referred to as latent variables because given just a data point generated by the model, we can't easily determine which setting of the latent variable produced that data point.

Formally, in such models we assume that the data X is generated by a latent random variable Z with a probability density function $P(Z)$. Typically we also have a family of deterministic functions $f(z;\theta)$ parametrized by $\theta \in \Theta$, where $f : Z \times \Theta \to \mathcal{X}$. The generative process works by sampling $z \sim P(Z)$ and computing $f(z;\theta)$. Since z is random and θ is fixed, $f(z;\theta)$ is a random variable in the space $\mathcal{X}$. The goal is to recover the optimal θ such that the likelihood of the data X under this generative process is maximized.

$$P(X) = \int P(X|z;\theta)P(z)dz \tag{2.9}$$

This process is entirely equivalent to the mathematical concept of maximum likelihood estimation. Some of the most common problems faced by such models are the following:

1. The marginal likelihood $P(X)$ computed by the integral in Equation 2.9 is quite often intractable in practice (typically when parametrized by neural networks).

2. The true posterior $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$ is also typically intractable, thus ruling out the applicability of the expectation maximization algorithm.

2.3.3 Variational Autoencoders

The Variational Autoencoder (Kingma & Welling, 2014) is an effective realization of generative latent variable models that try to overcome the problems of intractability described in Section 2.3.2. The key idea is to use a recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ that closely approximates the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ and jointly learn the parameters θ and ϕ that minimize the dissimilarity between the true posterior and the approximate posterior. This is typically done by minimizing notions of distances in the probabilistic space such as the KL divergence D_{KL}.

$$D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})] = E_{\boldsymbol{z}\sim q_\phi}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x})] \tag{2.10}$$

After applying Bayes rule, the above equation can be rewritten as:

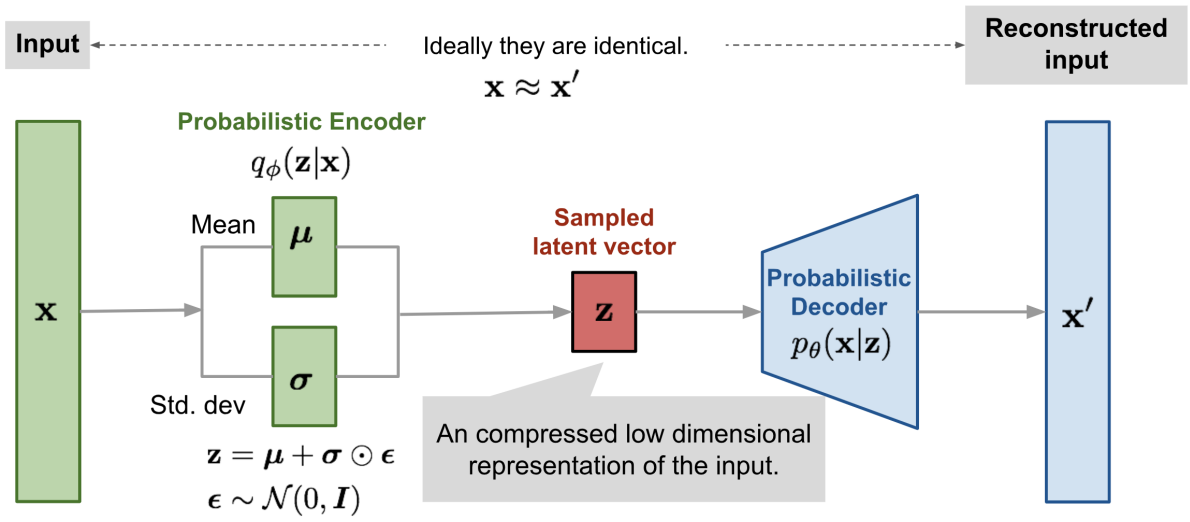


Figure 2.6: Architecture of a variational autoencoder

Source: https://cwkx.github.io/data/teaching/deeplearning/dl-lecture5.pdf

$$\log p_\theta(\boldsymbol{x}) - D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})] = E_{\boldsymbol{z}\sim q_\phi}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})] \quad (2.11)$$

Given that $D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})]$ is always non-negative, variational autoencoders maximize the lower bound on the R.H.S of equation 2.11 also known as the evidence lower bound (ELBO) .

$$\textbf{ELBO} = E_{z\sim q_\phi}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})] \leq \log p_\theta(\boldsymbol{x}) \quad (2.12)$$

From the setup of latent variable models it is clear that the recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is an encoder, because given a data point $\boldsymbol{x}$ it produces a distribution over the latent variable z likely generated the data point. Similarly $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ can be thought of as a probabilistic decoder that can generate data given the latent variable. The encoder and decoder are neural networks parametrized by ϕ and θ respectively. The R.H.S of Equation 2.11 takes the form of an autoencoder where the encoder encodes the data into a distribution and the decoder decodes the data given a sample from the distribution.

Typically in a VAE we assume that q_ϕ takes the form of a Gaussian distribution with diagonal covariance matrix; i.e: $q_\phi \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I})$ and that the prior is the standard gaussian

distribution $p_\theta(\boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{I})$, this allows us to use the reparemetrization trick to propagate the gradients through the sampling procedure. Instead of sampling from the approximate posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ we use the properties of Gaussian random variables to sample from the prior and use the parameters of the posterior to transform it into a posterior sample.

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(0, \mathbf{I}) \\ \boldsymbol{z} &= \boldsymbol{\mu} + \sigma\boldsymbol{\epsilon} \end{aligned} \tag{2.13}$$

The architecture of a standard VAE is shown in Figure 2.6. Since the introducetion of VAEs in 2014, they have emerged as the prime candidate of choice for generative models in various domains and applications such as text generation (S. R. Bowman et al., 2016), speech enhancement (Bando et al., 2018), speech synthesis (Akuzawa et al., 2018) and image generation (Yan et al., 2016). They also have found applications in several downstream tasks that benefit from the smooth latent spaces learned by the VAE such as anomaly detection (Suh et al., 2016), recommender systems and (X. Li & She, 2017), text classification (W. Xu et al., 2017).

2.3.4 Wasserstein Autoencoders

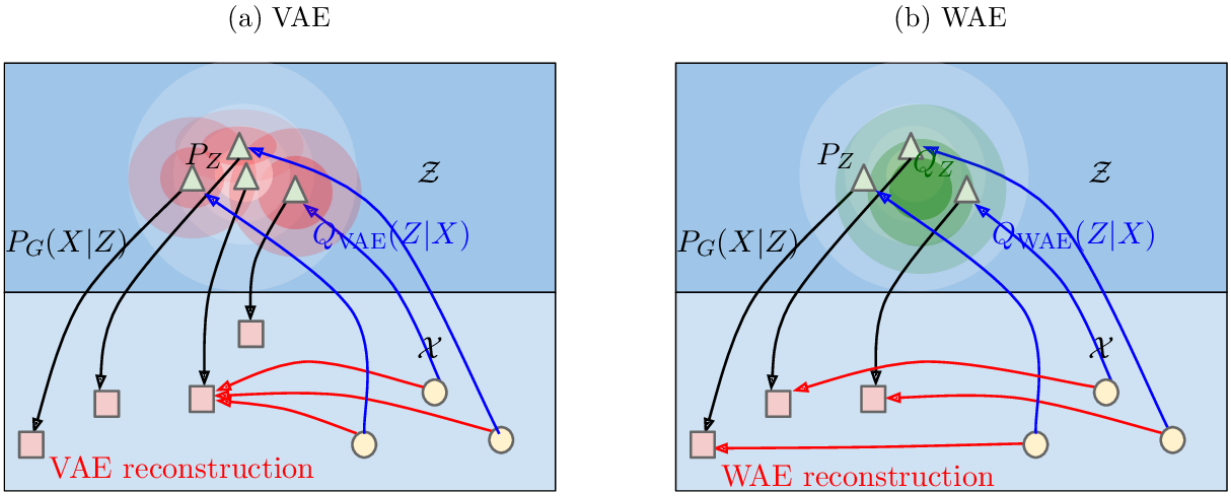


Figure 2.7: Latent space comparison of VAE and WAE. Source: (Tolstikhin et al., 2018)

Wasserstein Autoencoder (WAE)s are an alternative way of training latent variable models. Simplistically they can be thought of as following the framework of variational

autoencoders but with a different divergence measure between the posteriors. The key difference is that a VAE forces the posterior $Q(Z|X = x)$ to match the prior $P(Z)$ for each data point x, this may lead to some overlap between the posteriors induced by the VAE encoder and lead to poor reconstruction as shown in Figure 2.7. To avoid this problem, WAE tries to match the aggregate posterior $q(z) = \int q(z|x)p(x)$ with the prior $p(z)$.

Tolstikhin et al., 2018 relax the above criterion to minimizing the Wasserstein distance between $P(Z)$ and $Q(Z)$ for ease of computation.

The overall objective of WAE is:

$$\mathcal{L}_{\text{WAE}} = \inf_{\phi \in \Phi} E_{P_X} E_{z \in q_\phi}[c(X, p_\theta(X|z))] + \lambda D(Q_Z, P_Z) \tag{2.14}$$

Where λ is a tunable hyperparameter that affects the extend to which the divergence affects the overall objective and c is any non negative cost function. Note that the infimum $\inf_{\phi \in \Phi}$ emphasizes that we minimize the objective overall all possible encoders.

The authors propose two different approaches to measure the required divergence:

1. GAN based divergence: The divergence is set to the Jensen Shannon divergence and the model is trained like a GAN in an adversarial fashion Tolstikhin et al., 2018.
2. Maximum Mean Discrepancy (MMD): For a positive definite reproducing kernel $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ the following expression can be used as the divergence.

$$\text{MMD} = \left\| \int k(z, .)dP(z) - \int k(z, .)dQ(z) \right\|_{\mathcal{H}_k} \tag{2.15}$$

   Where $\mathcal{H}_k$ is the reproducing kernel hilbert space of real valued functions mapping $\mathcal{Z}$ to $\mathbb{R}$.

In this work we will only be working with MMD based WAEs and random encoders q_ϕ although WAEs can also work with deterministic encoders.

Typically implementations use the inverse multi-quadratics kernel because they are well suited for matching high-dimensional Gaussians that are commonly used as the encoding distributions.

$$k(x, y) = \frac{C}{C + \|x - y\|_2^2} \tag{2.16}$$

The advantage of MMD is that it has an unbiased U-Statistic estimator and thus can be computed numerically and used in conjunction with stochastic gradient descent methods.

$$\widehat{\text{MMD}} = \frac{\Sigma_{n \neq m} k(z^{(n)}, z^{(m)})}{N(N-1)} + \frac{\Sigma_{n \neq m} k(\tilde{z}^{(n)}, \tilde{z}^{(m)})}{N(N-1)} - \frac{k(z^{(n)}, \tilde{z}^{(m)})}{N^2} \tag{2.17}$$

where $x^{(n)}$ is a data sample, $z^{(n)}$ is sampled from the aggregated posterior by sampling from $q(z|x^{(n)})$ and $\tilde{z}^{(n)}$ is a sample from the prior $p(z)$.

For text we typically use the negative log likelihood as the cost function, thus the overall training objective for a WAE becomes:

$$\mathcal{L}_{\text{WAE}} = -\Sigma_{n=1}^{N} \Sigma_{t=1}^{|x^{(n)}|} \log p(x_t^{(n)} | z^{(n)}, x_{<t}^{(n)}) + \lambda_{\text{WAE}} \cdot \widehat{\text{MMD}} \tag{2.18}$$

WAEs have emerged as a popular alternative to VAEs due to ease of training. They have been used for various applications such as text generation (Bahuleyan et al., 2019), dialogue response generation (Chan et al., 2019) and collaborative filtering Zhong and Zhang, 2018.

2.3.5 Normalizing Flows

Standard implementations of generative models based on variational inference like VAEs and WAEs typically use the factorized Gaussian as their approximate posterior. In general most approaches to variational inference use only a limited class of approximate posteriors such as mean field approximations. This limits the expressive power of the model and there is evidence to suggest that in general better posterior approximations result in improved performance.

From the variational lower bound in equation 2.12 and the accompanying discussion in section 2.3.3 it is clear that the optimal posterior that allows $D_{KL}(Q||P) = 0$ is the one where $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})$. In other words the optimal case is for the approximate posterior to match the true posterior. This makes it clear that mean field approximations typically used such as the factorized Gaussian will more than likely fail to match the true posterior because they are not flexible enough.

To resolve this issue, Rezende and Mohamed, 2015 propose the idea of normalizing flows, a sequence of invertible transformations applied to probability distributions to obtain a probability distribution with the desired properties.

Consider a smooth invertible mapping $f : \mathbb{R}^d \to \mathbb{R}^d$ where d is the dimensionality of the latent space. Then we can transform the random variable $\boldsymbol{z} \sim q(\boldsymbol{z})$ by using this mapping. This transformation returns the random variable $\boldsymbol{z}' = f(\boldsymbol{z})$ with the following distribution.

$$q(\boldsymbol{z}') = q(\boldsymbol{z})\left|\det \frac{\partial f^{-1}}{\partial \boldsymbol{z}'}\right| = q(\boldsymbol{z})\left|\det \frac{\partial f}{\partial \boldsymbol{z}'}\right|^{-1} \tag{2.19}$$

The above result follows from the chain rule and applies because the function f is invertible which allows us to use the inverse function theorem for the Jacobian of invertible functions.

We can extend this idea to a sequence of K transformations f_k and transform the random variable $\boldsymbol{z}_0 \sim q_0$ to obtain a transformed density $q_K(\boldsymbol{z})$:

$$\boldsymbol{z}_K = f_K \circ ...f_2 \circ f_1 \tag{2.20}$$

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \Sigma_{k=1}^{K} \log \left|\det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}}\right| \tag{2.21}$$

Where $\circ$ denotes function composition.

The key idea that allows us to make use of such transformations for variational inference is that expectations w.r.t the transformed density q_K can be computed without a closed form expression for q_K. The law of the unconscious statistician allows us to rewrite any expectation $E_{q_K}[h(\boldsymbol{z})]$ as an expectation under q_0:

$$E_{q_K}[h(\boldsymbol{z})] = E_{q_0}[h(z_K)] \tag{2.22}$$

There are many interesting classes of powerful transformations that can be considered for normalizing flows, but in this work we will be primarily concerned with one of the simplest transformations; invertible linear time transformations also known as planar normalizing flows. The transformation takes the following form:

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z} + b) \tag{2.23}$$

Where h is a smooth element-wise non-linearity, $\boldsymbol{w} \in \mathbb{R}^d$, $\boldsymbol{u} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are free parameters. The logdet-Jacobian for this mapping can be computed in $O(d)$ time.

$$\psi(\boldsymbol{z}) = h'(\boldsymbol{w}^T\boldsymbol{z} + b)\boldsymbol{w} \tag{2.24}$$

$$\left|\det \frac{\partial f}{\partial \boldsymbol{z}}\right| = \left|\det\left(\boldsymbol{I} + \boldsymbol{u}\psi(\boldsymbol{z})^T\right)\right| = \left|1 + \boldsymbol{u}^T\psi(\boldsymbol{z})\right| \tag{2.25}$$

Given the above transformation, we can apply a sequence of K such transformations to obtain:

$$\begin{aligned} \boldsymbol{z}_K &= f_K \circ ...f_2 \circ f_1 \\ \log q_K(\boldsymbol{z}_K) &= \log q_0(\boldsymbol{z}_0) - \Sigma_{k=1}^K \log\left|1 + \boldsymbol{u}_k^T\psi_k(\boldsymbol{z})\right| \end{aligned} \tag{2.26}$$

The flow defined by equation 2.26 is called a planar normalizing flow, because it transforms the initial density q_0 by applying a series of expansions and contractions perpendicular to the hyperplane $\boldsymbol{w}^T\boldsymbol{z} + b = 0$. This allows us to transform simple distributions into complex multi-modal distributions as shown in Figure 2.8.

We can use the log probability of the transformed distribution computed in equation 2.26 to derive a new variational lower bound.

$$\textbf{ELBO} = E_{q(\boldsymbol{z}_0|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z}_K) + \Sigma_{k=1}^K \log\left|\det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}}\right|] - D_{KL}(q(\boldsymbol{z}_0|x)||p(\boldsymbol{z}_T)) \tag{2.27}$$

Normalizing flows are primarily used in tasks related to density estimation . In recent times generative models using normalizing flows have also been explored (Kumar et al., 2019; Ziegler & Rush, 2019).

2.3.6 Disentanglement

Disentangling the factors of variation is an important aspect of developing useful high quality representations of data. Bengio et al., 2013 suggest that learning disentangled representations help in making the representations more interpretable and would also help with multiple downstream tasks. For instance disentangling object surfaces from their shadows would help in object detection, disentangling stylistic aspects such as sentiment,

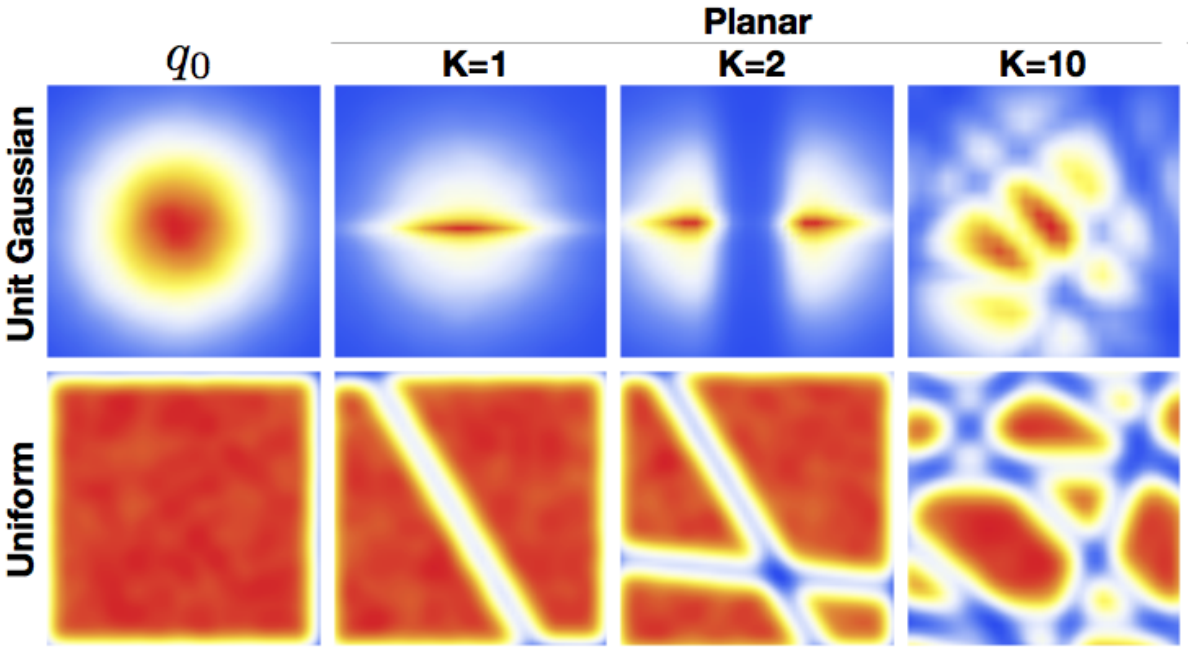


Figure 2.8: Transformation of Unit Gaussian and Uniform distrubtions using planar normalizing flows. Source: (Rezende & Mohamed, 2015)

authorship from content would help in style transfer tasks. Learning disentangled representations of data would also help in controlled generation. Figure 2.9 shows an example of learning disentangled representations and using it for controlled text generation.

Although there is no widely accepted formal definition of disentanglement, most attempts to define it have argued that separating the factors of variation in the data is a key requirement. Thus, many of the proposed ways of evaluating disentanglement in models rely on the statistical relationships between the learned representation and the true factors of variation in the ground truth (X. Chen et al., 2016; Eastwood & Williams, 2018).

There have been various works that attempt to recover the underlying factors of variation in data in a completely unsupervised fashion with varying degrees of success. Many of the approaches are based on variational autoencoders and augmenting their objective. β-VAE (Higgins et al., 2017) increases the weight associated with the KL divergence in the VAE objective to encourage the statistical independence of the underlying factors imposed by the KL divergence when it matches the posterior to a standard Gaussian prior.

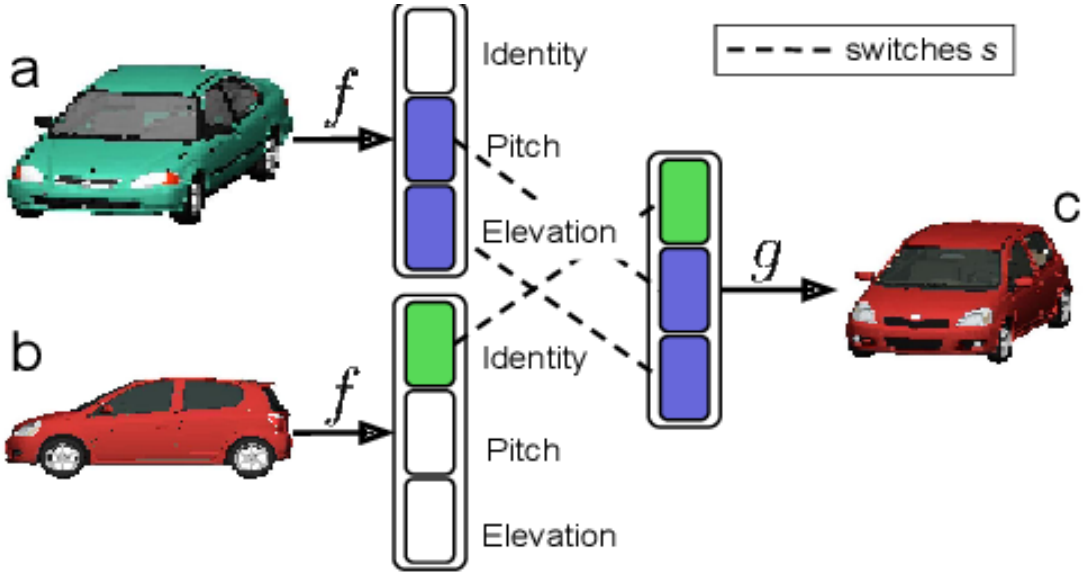


Figure 2.9: Example of disentanglement and it's application to controlled generation. Source: (Reed et al., 2015)

$$\mathcal{L}_{\beta-\text{VAE}} = E_{z\sim q_\phi}[\log p_\theta(x|z)] - \beta D_{KL}[\log q_\phi(z|x)||p_\theta(z)] \tag{2.28}$$

FactorVAE (Kim & Mnih, 2018) and β-TCVAE (R. T. Chen et al., 2018) also follow a similar approach and modify the VAE objective to encourage statistical independence between the various latent dimensions. X. Chen et al., 2016 attempted to learn disentangled latent representations through generative adversarial networks instead of variational autoencoders. Although the above unsupervised approaches to disentanglement show reasonable results in the chosen simple datasets, they are typically difficult to train and reproduce, and do not guarantee disentanglement for even slightly challenging datasets (See Section 3.3).

Chapter 3

Motivation and Related Work

In this chapter, we discuss some of the common limitations and challenges faced by existing models and approaches, the related attempts to resolve them and motivate our work.

3.1 Variational Autoencoders

3.1.1 Reconstruction vs KL tradeoff

From the variational lower bound presented in equation 2.12 it is clear that the VAE objective has two terms: the reconstruction term $E_{\boldsymbol{z}\sim q_\phi}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$ where the objective is to maximize the log likelihood of the decoded sentence and the KL divergence term $D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})]$ where the objective is to match the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to the prior $p_\theta(\boldsymbol{z})$.

From Figure 2.7 it is clear that these two objectives are competing against each other. Optimizing the KL term very well would lead to matching the approximate posterior for each data point closely with the prior which leads to lots of overlap thus hindering reconstruction. This is one of the motivations behind the development of Wasserstein Autoencoders where the aggregate posterior is matched instead of the individual approximate posteriors.

3.1.2 Posterior Collapse

A very common problem faced while training a VAE is that of posterior collapse, especially when powerful decoders are used. It is easy to understand posterior collapse when we see that a trivial optimum of the KL term in the VAE objective occurs when the approximate posterior and the model posterior are exactly the same as the prior i.e $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = p_\theta(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z})$ for all $\boldsymbol{x}$.

He et al., 2019 define two partial collapse states: *model collapse* when the true model posterior matches the prior; $p_\theta(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z})$ and *inference collapse* when the approximate posterior matches the prior; $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{z})$. An illustration is provided in Figure 3.1

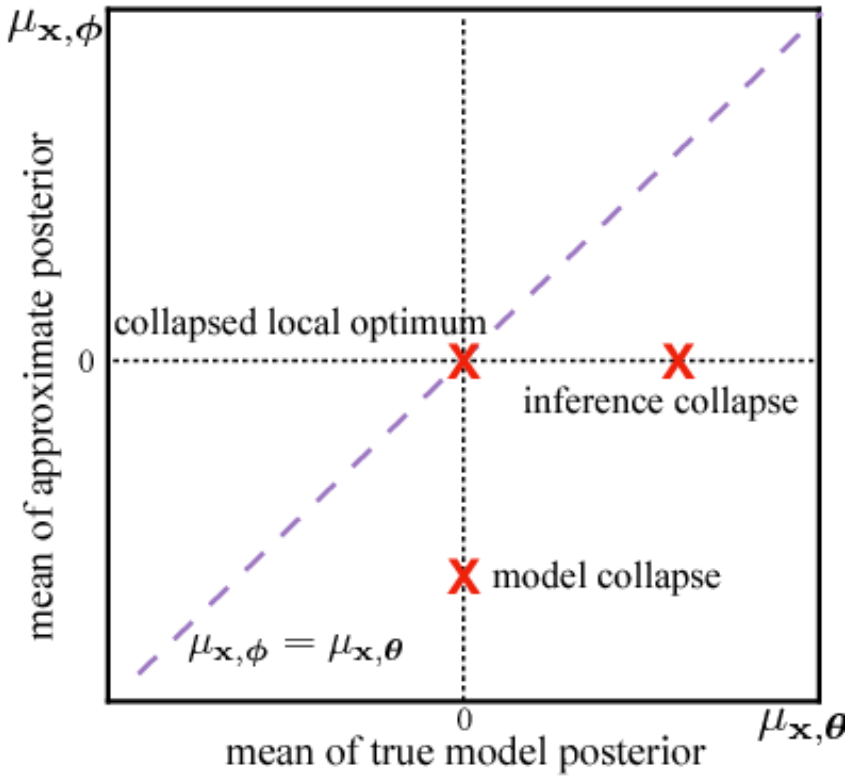


Figure 3.1: Illustration of model and inference collapse in a toy posterior mean space. Source: (He et al., 2019)

Posterior collapse is a concerning problem because it essentially means that the learned representations $\boldsymbol{z}$ are independent of $\boldsymbol{x}$ which makes them not useful. It is easy to see why

powerful decoders exacerbate this problem. Powerful, auto-regressive decoders such as the ones based on LSTMs can learn to generate sentences $\boldsymbol{x}$ similar to the dataset without relying on $\boldsymbol{z}$, thus contributing to the representations $\boldsymbol{z}$ being independent of the data $\boldsymbol{x}$. The use of simple posteriors such as the factorized Gaussian also contribute to the problem because such posteriors are not flexible enough to model complex datasets and thus further causing the decoder to ignore that latent vectors.

Lucas et al., 2019 suggest that powerful decoders are not the only contributing factor towards posterior collapse. They perform experiments to empirically show that optimization issues such as ill-conditioning of stationary points in the log marginal likelihood also can contribute to posterior collapse.

Since it is clear that the KL term in the variational lower bound plays a significant part in posterior collapse A simple and very common way to mitigate posterior collapse is to add a variable weight to the KL term. The weight is initially set to zero and is gradually increased as training progresses. Although very common, this is not a very reliable strategy for preventing posterior collapse.

There has been a lot of research focused on solving the problem of posterior collapse and improve the generative performance of VAEs in general.

He et al., 2019 explore posterior collapse from the perspective of training dynamics. The authors observe that during the initial stages of training, the inference network fails to approximate the true model posterior which causes the decoder to ignore the latent variables and converge towards a local optimum where $\boldsymbol{z}$ and $\boldsymbol{x}$ are independent. They proceed to alleviate this by aggressively training the inference network in the early stages and mitigating the posterior collapse.

J. Xu and Durrett, 2018b propose to use the VonMisesFischer (vMF) distribution in place of factorized Gaussian for preventing posterior collapse. The vMF posterior causes the latent variables to reside in the unit hyper-sphere. The KL divergence term for this choice of prior and posterior is dependent only on the variance of the VMF distribution. They treat it as a tunable hyper-parameter and thus prevent posterior collapse. An illustration of vMF posterior in comparison to the Gaussian is provided in Figure 3.2

Razavi et al., 2019 mitigate posterior collapse by constraining the family of variational posteriors to have a KL divergence from the prior that is bounded below by a constant $\delta > 0$. They show that their approach can prevent posterior collapse even with the most powerful decoders.

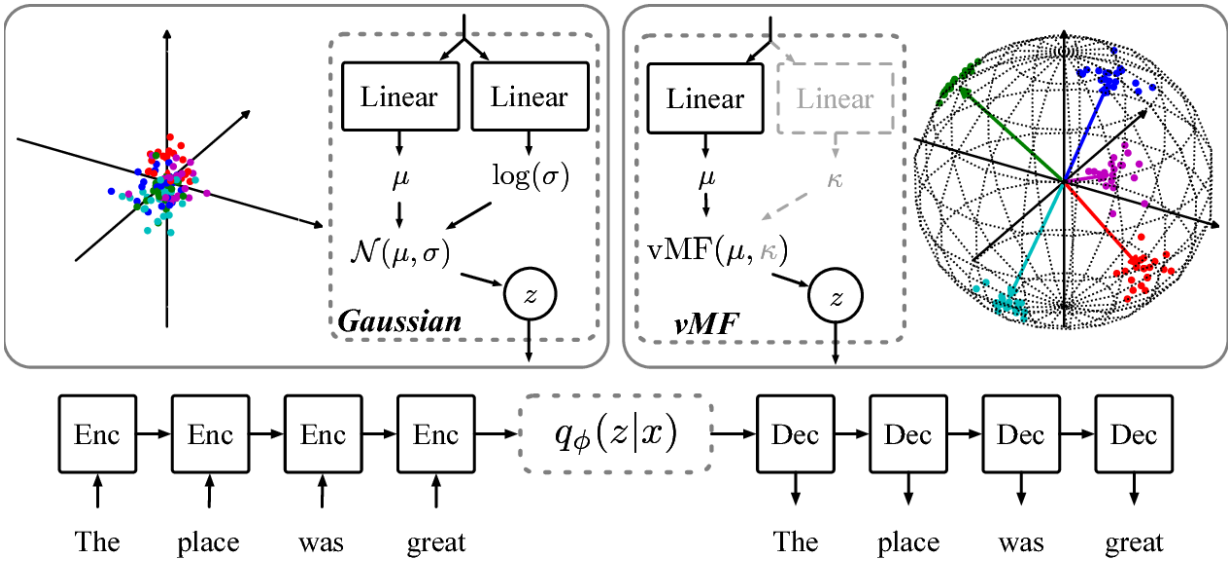


Figure 3.2: Comparison of VAE with Gaussian (left) and vMF (right) priors. Source: (J. Xu & Durrett, 2018b)

3.2 Wasserstein Autoencoders

3.2.1 Dimension Mismatch

Unlike VAEs, WAEs can use deterministic encoders for encoding the data, since they can be viewed as simply regularizing an autoencoder instead of performing variational inference.

If the latent space dimension is set larger than the intrinsic dimensionality of the data then deterministic autoencoders tend to perform very poorly and lead to very poor sample quality when sampling from the latent space. Rubenstein et al., 2018 suggest that this happens because the deterministic encoder is unable to cover the entire latent space. The authors however suggest that this problem is not as prominent in random encoders because they can just fill the rest of the space with noise.

To illustrate this the authors perform an experiment with a fading squares dataset of gray squares in a black background, with the color varying uniformly from black to white in steps of 10^{-3}. It is clear that the intrinsic dimensionality of this data is 1. They train WAEs with latent space dimensionality set to 2, and compare deterministic encoders to random encoders. The results of this experiment is presented in Figure 3.3.

From the figure, especially the upper middle plot with 1000 points, it is clear that de-

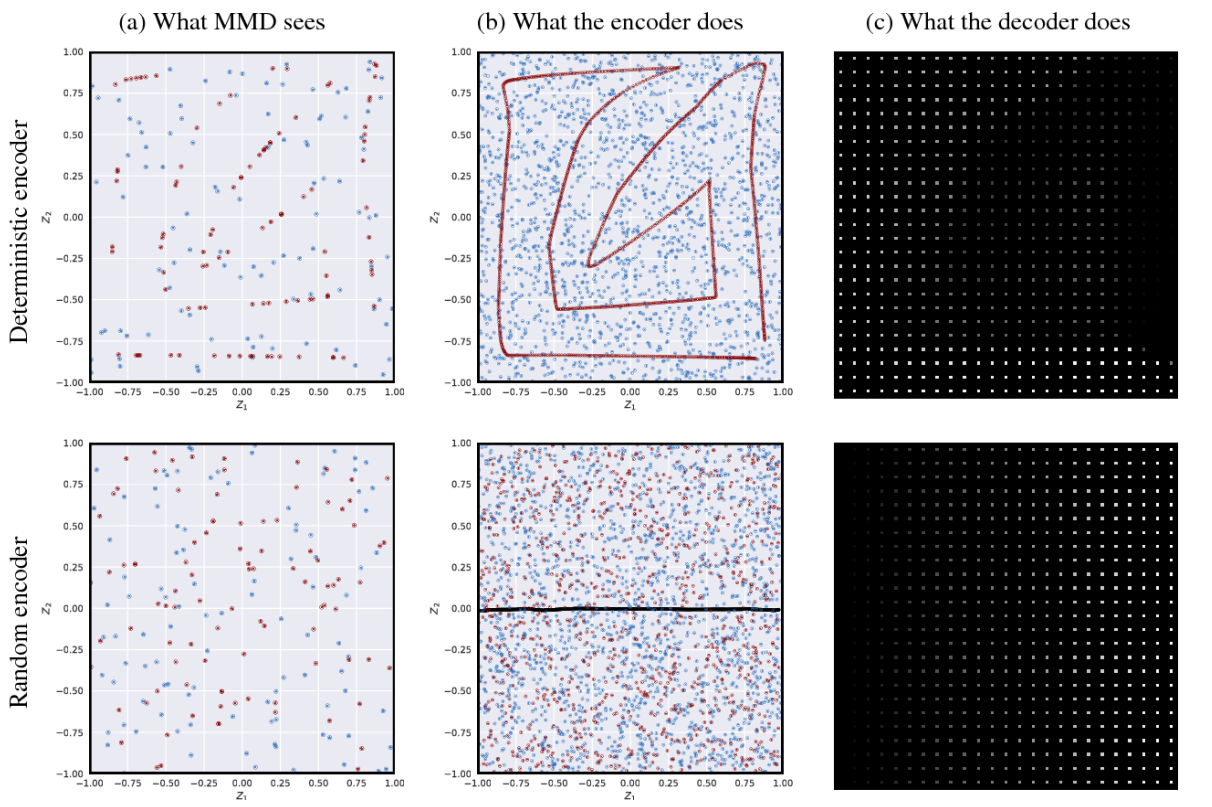


Figure 3.3: a) 100 points sampled from p_z (blue) and q_z (red) (b) Same plot with 1000 points, the black points (lower middle) are the data points mapped to the mean of the random encoder q_ϕ. c) Decoder outputs of points in latent space. Source: (Rubenstein et al., 2018)

terministic encoders fail to match the aggregate posterior to the prior in cases of dimension mismatch. Random encoders can mitigate his problem by filling the rest of the latent space with noise.

3.2.2 Variance Collapse

Although random encoders in WAEs are able to mitigate dimension mismatches reasonably, they often suffer from a phenomenon known as variance collapse especially in high dimensional latent spaces.

For large enough size of latent dimensions the random encoders in WAEs fail to fill

the rest of the latent space with noise to match the aggregate posterior and prior while preserving sample quality. Instead, the variances $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ tend to 0 for almost all dimensions and inputs. Effectively the random encoder has collapsed to a deterministic encoder leading to poor performance. Bahuleyan et al., 2019 study this problem in detail and show that if the variance is small during the initial phase of training, then the factorized Gaussian converges to a dirac-delta function with stochastic gradient descent, thus causing variance collapse. It has been suggested that an auxiliary loss term penalizing the KL divergence between approximate posterior and a factorized Gaussian with covariance $\mathbf{I}$ centered around the predicted mean $\boldsymbol{\mu_x}$ be added to the WAE objective.

$$\tilde{\mathcal{L}}_{\text{WAE}} = \mathcal{L}_{\text{WAE}} + \lambda_{KL}\Sigma_{\boldsymbol{x}\in X} D_{KL}(\mathcal{N}(\boldsymbol{\mu_x}), \sigma^2_{\boldsymbol{x}}\mathbf{I})||\mathcal{N}(\boldsymbol{\mu_x}), \mathbf{I})) \tag{3.1}$$

Where $\boldsymbol{\mu_x}$ and $\sigma^2_{\boldsymbol{x}}$ are the predicted mean and variance for data point $\boldsymbol{x}$, and λ_{KL} is a hyperparameter that controls the effect of the auxiliary loss.

3.3 Challenges in learning disentangled representations

3.3.1 Unsupervised Disentanglement

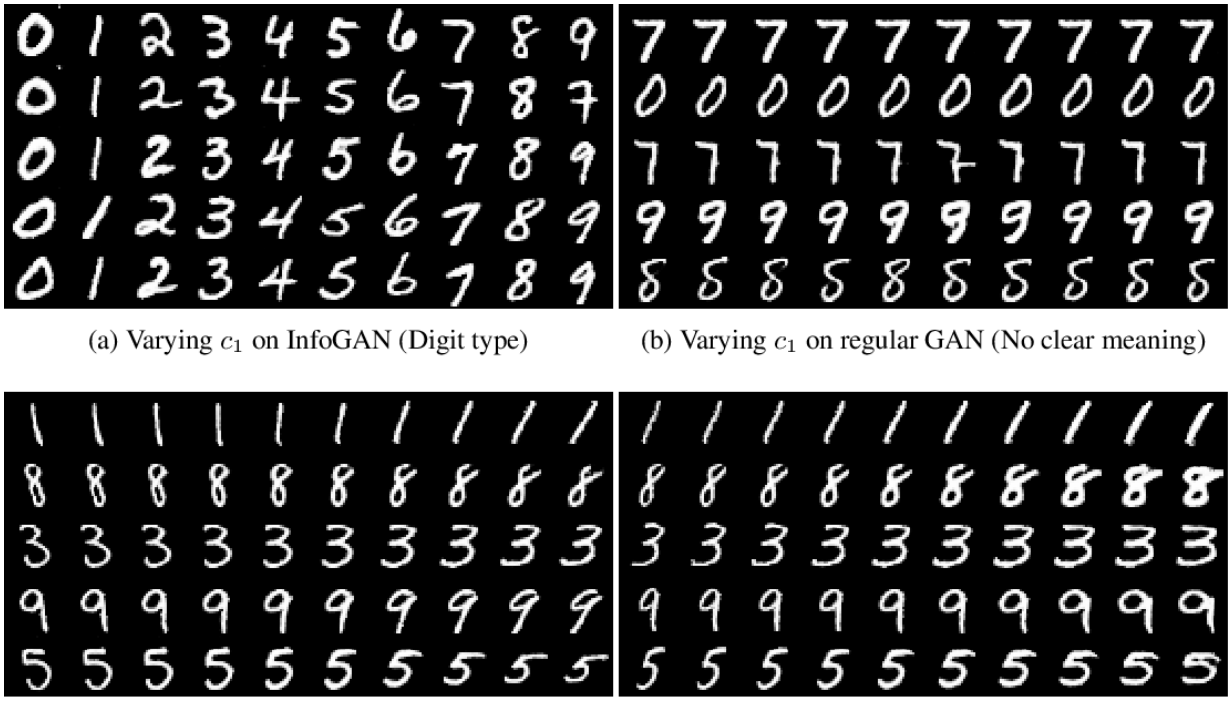
As discussed in Section 2.3.6 there have been several attempts to learn and disentangle the underlying factors of variation in data in an unsupervised manner. Most of these attempts use generative models such as variational autoencoders and generative adversarial networks and modify their objectives to encourage statistical independence between the latent factors (R. T. Chen et al., 2018; X. Chen et al., 2016; Higgins et al., 2017; Kim & Mnih, 2018).

Although models that attempt unsupervised disentanglement have shown moderate success on synthetic datasets or simpler real world datastets, they are typically difficult to train and disentanglement is not guaranteed. There is also a question of identifiability, even if such models manage to enforce independence across the latent factors, how do we guarantee that the required factors to be disentangled are recovered?

Locatello et al., 2019 study this problem extensively and show that unsupervised disentanglement is fundamentally impossible without inductive biases on the learning algorithms and the dataset. They formally prove this impossibility result, but also caution that even with the impossibility result it may be possible for unsupervised disentanglement to work on certain datasets with the right model selection.

3.3.2 Disentanglement over independent subspaces

Many approaches to disentanglement focus on disentangling the different factors of variation over the different dimensions in the latent space. In such models, changing a single latent dimension would change only a single aspect/factor of the data. We provide an example from the InfoGAN (X. Chen et al., 2016) model in Figure 3.4.



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 3.4: Source: (X. Chen et al., 2016)

However an issue with such approaches is that some aspects/factors of variation in data are complex and cannot be easily captured by a single latent dimension. This is especially true in text, where unsupervised approaches to disentanglement have been largely unsuccessful because the aspects of text, such as sentiment, style, formality and syntax are complex. In such cases it makes more sense to reserve a subspace of the latent space for each aspect to be disentangled.

The issue with disentanglement over independent subspaces is that the typical statistical constraints to encourage independence such as mutual information, total correlation, do not extend well to ensure independence over subspaces. The factorized Gaussian posterior

tries to enforce the assumption that the latent dimensions are independent which is too strict and not suitable for disentanglement over independent subspaces.

There has been a lot of work done towards overcoming these issues for disentanglement in text. The most popular approaches typically use adversarial training to enforce the independence of the subspaces (John et al., 2019). However adversarial objectives are typically unstable and hard to train and researchers continue to seek ways to enforce independence across subspaces without the adversarial training.

3.4 Towards better representation learning

In light of the limitations discussed above, this work will focus on improved representation learning in terms of better generative models and downstream tasks for disentanglement. We consider two different approaches to learning improved representations:

1. Adding task specific auxiliary losses to the overall objective that help in learning better representations
2. Use more flexible and more powerful approximate posteriors than the factorized Gaussians typically used.

We briefly discuss and motivate both of these approaches below

3.4.1 Auxiliary Losses

Designing additional auxiliary losses to induce additional desired properties is a common practice in many machine learning problems. Multitask losses that enable the model to learn additional features about the data have been successful. John et al., 2019 use multi-task bag of words loss for improving content preservation and a discriminative cross entropy loss to enable the model to encode information about the sentiment of a sentence given the sentiment labels for the training set.

In this work we propose an auxiliary loss that would enable us to disentangle factors of variation in data based on pairwise similarity given a criteria. We evaluate our approach on the task of disentangling the semantics and syntax of a sentence. Existing work on disentangling semantics and syntax use task specific multitask losses (M. Chen et al., 2019) and adversarial training (Bao et al., 2019).

3.4.2 Flexible Posteriors

As discussed earlier the factorized Gaussian posterior typically used in variational autoencoders is limiting in many ways and contribute to the many limitations faced by standard variational autoencoder.

Normalizing flows (Section 2.3.5) provide us an easy way to use more complex and flexible posteriors while still being able to perform variational inference. Normalizing flows have been extensively used for image generation using generative flows (Hoogeboom et al., 2019; Kingma & Dhariwal, 2018). Normalizing flows have also been used as approximate posterior in variational inference for image generation tasks (Kingma et al., 2016). However the effectiveness of using normalizing flows for text generation tasks has not been extensively studied. Ziegler and Rush, 2019 study the effectiveness of normalizing flows for character level language modeling and polyphonic music generation. Wang and Wang, 2019 propose a Riemannian normalizing flow to model the curvature of the latent space and evalute it on language modeling tasks.

In this work we study the effectiveness of normalizing flows for text generation tasks across various metrics such as fluency and diversity. We also study the effect of using normalizing flows on disentanglement tasks. Finally, we also propose a variation of Planar Normalizing Flows called Block Planar Normalizing Flows for more effective disentanglement into subspaces. We evaluate the effectiveness of normalizing flows for disentanglement on two tasks: 1) disentangling syntax and semantics in sentences 2) disentangling sentiment from content in product reviews.

Chapter 4

Polarized VAE

In this chapter we discuss in detail our proposed model polarized-VAE to disentangle factors of variation in data into different subspaces. This model uses the idea of additional proximity based regularization in the latent space to control the relative location of points based on their similarity according to a criterion. This proximity based regularization encourages similar points to be grouped together and dissimilar points to be distributed farther apart in a given subspace. Effectively, we polarize the latent subspaces learned by a variational autoencoder, thus giving it the name polarized-VAE.

4.1 Approach

4.1.1 Disentanglement into subspaces

Given a collection of criteria $C = \{c_1, ..., c_S\}$ based on which we want to disentangle the latent space learned by a variational autoencoder into S different subspaces $\mathcal{Z} = [\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(S)}]$, where $\boldsymbol{Z}^{(j)}$ is the latent subspace corresponding to the criterion c_j. The vectors corresponding to the different subspaces are concatenated together to give us the representation of a data point. Since we have S different subspaces we have S different encoders parametrized by $\phi_1, \ldots, \phi_S$. We use a single decoder parametrized by θ. Our model architecture is shown in Figure 4.1

Since we evaluate our approach on disentangling semantics and syntax in a sentence, we have $S = 2$, $C = \{c_1, c_2\}$ and $\mathcal{Z} = [\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}]$.

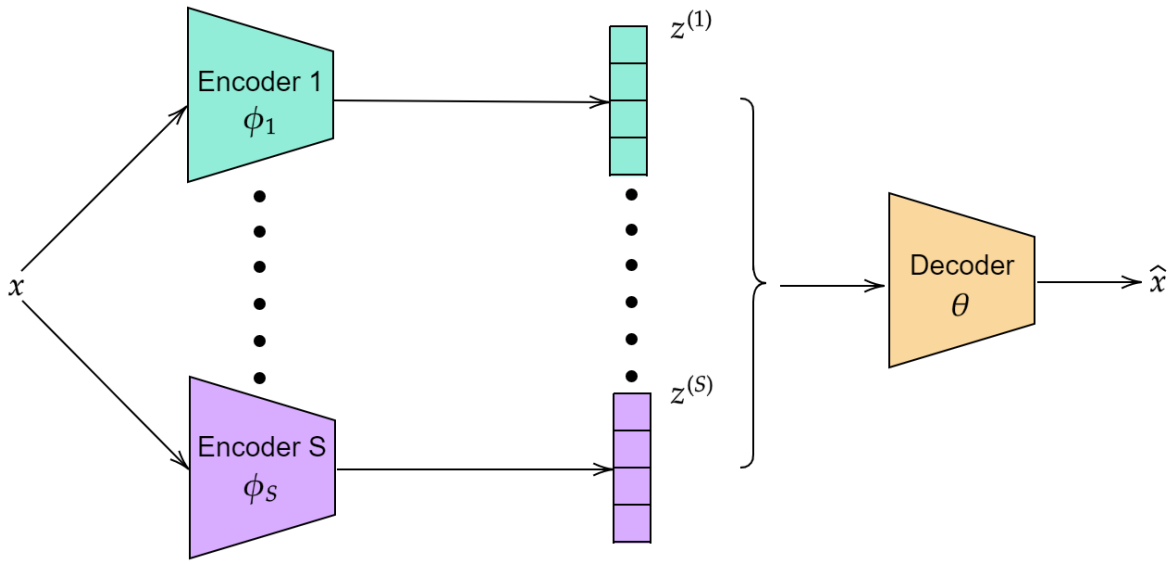


Figure 4.1: polarized-VAE architecture

4.1.2 Supervision based on pairwise similarities

Let us assume that given a pair of sentences, we have a way to determine their similarity according to any given criterion $c \in C$. The pairwise similarity that we determine could potentially be noisy. The notion of similarity used could be a binary label; similar and dissimilar or an integer such as edit distance or even a continuous scalar variable. In this work we will be using binary labels as the notion of similarity.

$$\text{Sim}(\boldsymbol{x}_i, \boldsymbol{x}_j | c) = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar} \\ & \text{w.r.t. the criterion } c \in C \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

We let c_1 be the semantic criterion and c_2 be the syntactic criterion, thus we want $\boldsymbol{z}^{(1)} \in \mathcal{Z}^{(1)}$ to encode the semantic information and $\boldsymbol{z}^{(2)} \in \mathcal{Z}^{(2)}$ to encode the syntactic information given a sentence.

4.1.3 Training objective and proximity function

Extending the standard variational autoencoder we have K RNN based encoders parametrized by $\phi_1, \ldots, \phi_S$. An encoder parametrized by ϕ_c, where $c \in C$, encodes the approximate pos-

terior $q_{\phi_c}(\boldsymbol{z}^{(c)}|\boldsymbol{x})$.

Given two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ we denote the proximity of their encoded representations in the latent space by $D_{\text{prox}}(q_{\phi_c}(\boldsymbol{z}^{(c)}|\boldsymbol{x}_i))||q_{\phi_c}(\boldsymbol{z}^{(c)}|\boldsymbol{x}_j))$.

We experiment with multiple different proximity measures (Section 4.2.6) and found that the cosine distance between the encoded representations performs very well:

$$D_{\text{prox}}(q_{\phi_c}(\boldsymbol{z}|\boldsymbol{x}_i), q_{\phi_c}(\boldsymbol{z}|\boldsymbol{x}_j)) = d_c(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{1}{2}\left(1 - \frac{\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle}{||\boldsymbol{z}_i||||\boldsymbol{z}_j||}\right) \tag{4.2}$$

We add this proximity function to the VAE objective as a regularizer by making use of the similarity information for each criterion. For each example $(\boldsymbol{x}, c)$ we sample a positive example $\boldsymbol{x}_p$ that is similar to $\boldsymbol{x}$ w.r.t criterion c and m negative samples $\boldsymbol{x}_{n_1}, \ldots, \boldsymbol{x}_{n_m}$ that are dissimilar to $\boldsymbol{x}$ w.r.t criterion c. Thus, we have:

$$\text{Sim}(\boldsymbol{x}, \boldsymbol{x}_p|c) = 1 \tag{4.3}$$

$$\text{Sim}(\boldsymbol{x}, \boldsymbol{x}_{n_j}|c) = 0 \quad \forall j \in \{1, \ldots, m\} \tag{4.4}$$

.

We use the max margin loss over the proximity function over positive sample and negative samples as the proximity regularizer:

$$\mathcal{L}_c = \max(0, 1 + d_c(\boldsymbol{z}, \boldsymbol{z}_p) - \frac{1}{m}\sum_{j=1}^{m} d_c(\boldsymbol{z}, \boldsymbol{z}_{n_j})) \tag{4.5}$$

Adding the above regularizer to the VAE objective gives us the overall objective of the polarized-VAE:

$$\mathcal{L}_{\text{polarized-vae}} = \mathcal{L}_{\text{vae}} + \sum_{c \in C} \lambda_c \mathcal{L}_c \tag{4.6}$$

4.2 Experiments

We evaluate our polarized-VAE model on the task of disentangling the semantics and syntax of a sentence into two different subspaces. To this end we carry out semantics syntax transfer experiments on the Stanford Natural Language Inference (SNLI, S. Bowman et al., 2015). We evaluate our model on reconstruction and sample quality, we also follow Bao et al., 2019 and evaluate our model for semantic transfer strength and syntactic transfer

strength. We perform human evaluation to evaluate our model on fluency and semantics syntax transfer. We also evaluate the extend of disentanglement of the latent space using a simple correlation measure and report the performance of alternative proximity functions.

4.2.1 Training Details and Hyperparameters

We use bidirectional LSTMs (Hochreiter & Schmidhuber, 1997) with hidden size of 128 for both the semantic and syntactic encoder. The encoder outputs are transformed using two single hidden layer feedforward neural networks to obtain the parameters for the Gaussian posterior. The semantic encoder has a latent space dimension of 64 and the syntactic encoder has a latent space dimension of 16.

The decoder is a unidirectional LSTM with a hidden size of 128. We adopt standard VAE training tricks for text including word dropout and KL annealing as outlined by S. R. Bowman et al., 2016. The KL weights for both the semantic and syntactic encoders were annealed to 0.3 over 5000 steps following a sigmoid annealing schedule.

The model is trained for 30 epochs using the ADAM optimizer (Kingma & Ba, 2015) with the default parameters and a learning rate of 0.001

4.2.2 Reconstruction and Sample Quality

From the discussion in Section 3.1.1 we know that there exists a tradeoff between the reconstruction quality and the sample quality from the representations learned. Thus, we train our model multiple times with different KL annealing schedules and compare our model with the VAE on two metrics: reconstruction BLEU[1] and the forward perplexity (PPL, Zhao et al., 2018). The reconstruction BLEU determines how well the model can reconstruct a given sentence and the forward PPL evaluates the quality of the sentences generated from sampling from the prior.

Figure 4.2 shows the plot between BLEU and PPL for both polarized-VAE and standard-VAE. Clearly there is a tradeoff between reconstruction and sample quality. To make the trend clear we also show the regression line in the plot. We can see that adding the proximity constraint slightly improves the performance of polarized-VAE indicating that it does not negatively affect reconstruction or sample quality.

[1]We use the BLEU-4 score computed using the NLTK library: https://www.nltk.org/

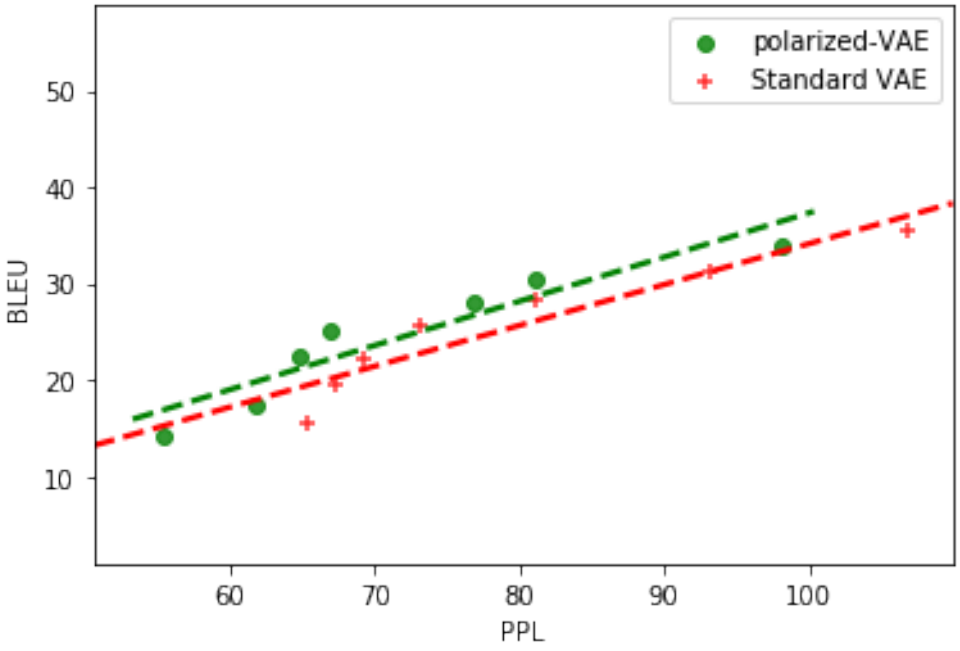


Figure 4.2: Comparing trade-off between BLEU and PPL for Standard VAE and polarized-VAE, with different KL coefficients λ_{KL}.

4.2.3 Controlled generation and Semantics-Syntax Transfer

To evaluate controlled generation and semantics-syntax transfer we follow the same methodology as Bao et al., 2019; M. Chen et al., 2019.

Given two sentences, $\boldsymbol{x}_{\text{sem}}$ and $\boldsymbol{x}_{\text{syn}}$ we wish to generate a sentence that is semantically similar to sentence $\boldsymbol{x}_{\text{sem}}$ and has the syntactic structure of $\boldsymbol{x}_{\text{syn}}$ using the following procedure:

$$\boldsymbol{z}_{\text{sem}} \sim q_{\phi_1}(\boldsymbol{z}^{(1)}|\boldsymbol{x}_{\text{sem}}) \tag{4.7}$$
$$\boldsymbol{z}_{\text{syn}} \sim q_{\phi_2}(\boldsymbol{z}^{(2)}|\boldsymbol{x}_{\text{syn}}) \tag{4.8}$$
$$\boldsymbol{z} = [\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}}] \tag{4.9}$$
$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z}) \tag{4.10}$$

We measure transfer strength based on semantic content preservation using BLEU scores for the semantic subspace and tree edit distance (Zhang & Shasha, 1989) between the syntax trees for the syntactic subspace.

We consider 1000 pairs of sentences from the SNLI corpus for evaluation. Considering equation 4.7 we want the generated sentence to be semantically similar to $\boldsymbol{x}_{\text{sem}}$ and semantically different to $\boldsymbol{x}_{\text{syn}}$. We measure these requirements using BLEU scores. We also measure the differences in these BLEU scores ΔBLEU to indicate semantic transfer strength.

Considering equation 4.8 we want the generated sentence to have a syntactic structure similar to $\boldsymbol{x}_{\text{syn}}$ and have a syntactic structure dissimilar to $\boldsymbol{x}_{\text{sem}}$. We measure these requirements using the tree edit distance (TED) between the syntax trees of the sentences. We also measure the differences in these TED scores ΔTED to indicate syntactic transfer strength.We also report the geometric mean of ΔBLEU and ΔTED; ΔGM to indicate overall transfer strength.

We perform these experiments on four variants of the polarized-VAE. In the default variant polarized-VAE) we use the entailment labels from SNLI dataset as a proxy for semantic similarity. For syntactic similarity we threshold the differences in tree edit distance of the syntax parses provided in the dataset. In the second variant polarized-VAE (wo) we use word overlap as a proxy for semantic similarity, i.e if two sentences have a high word overlap, we consider them to be semantically similar. The proxy for syntactic similarity in polarized-VAE (wo) is the same as default. In the third variant polarized-VAE (len) we use length difference as a proxy for syntactic similarity, i.e if two sentences have a high difference in word count, we consider them to be syntactically dissimilar. The proxy for semantic similarity in polarized-VAE (len) is the same as default. Finally in the last variant polarized-VAE (wo, len) we combine both heuristics. The final variant can be viewed as an unsupervised model that does not make use of any labels or syntax trees.

Model	**BLEU**			**TED**			
	$\boldsymbol{x}_{\text{sem}}{}^{\uparrow}$	$\boldsymbol{x}_{\text{syn}}{}^{\downarrow}$	ΔBLEU$^{\uparrow}$	$\boldsymbol{x}_{\text{sem}}{}^{\uparrow}$	$\boldsymbol{x}_{\text{syn}}{}^{\downarrow}$	ΔTED$^{\uparrow}$	ΔGM$^{\uparrow}$
Standard VAE	4.75	4.67	0.08	13.70	13.60	0.10	0.28
Bao et al., 2019	**13.74**	6.15	7.59	**16.19**	13.11	**3.08**	4.83
polarized-VAE	10.78	0.92	**9.86**	14.09	11.67	2.42	**4.88**
polarized-VAE (`wo`)	9.82	0.84	8.98	14.12	11.65	2.47	4.71
polarized-VAE (`len`)	10.10	**0.76**	9.34	12.68	**11.44**	1.44	3.67
polarized-VAE (`wo`, `len`)	9.41	0.87	8.54	12.65	11.48	1.17	3.16

Table 4.1: Results of syntax transfer generation on SNLI dataset. Bao et al., 2019 report TED after multiplying by 10, we report their score after correcting for it.

From the results presented in Table 4.1 we observe that our model outperforms the VAE baseline on all metrics. Our model is also better at ignoring the semantics of $\boldsymbol{x}_{\text{syn}}$

when compared to Bao et al., 2019 as can be seen from the lower BLEU score w.r.t. $\boldsymbol{x}_{\text{syn}}$. Furthermore, although Bao et al., 2019 outperform our model in matching semantics with $\boldsymbol{x}_{\text{sem}}$ our high ΔBLEU scores indicate better overall semantics transfer strength. For syntactic transfer strength we note that our model scores better in matching the syntactic structure of $\boldsymbol{x}_{\text{syn}}$ as indicated by the lower TED score w.r.t $\boldsymbol{x}_{\text{syn}}$. The overall transfer strength of our model is competitive with Bao et al., 2019 as indicated by our slightly better ΔGM score.

4.2.4 Disentanglement into subspaces

We would like to verify that the two latent subspaces learned by our model do not have significant overlap in information. We also note that this is likely to happen if the attributes themselves are highly correlated such as syntax and length. In the case that the attributes to be disentangled are highly correlated, even existing disentanglement approaches based on adversaria; training John et al., 2019 will fail. However, in case of attributes that are not correlated (or ideally independent) such as syntax and semantics we believe that our approach will disentangle them into different subspaces. We observe that since we apply our proximity loss to each subspace independently our model would encourage the semantic information to be mainly encoded into the semantic subspace and the syntactic information to be mainly encoded in the syntactic subspace.

In the case of independence of individual latent factors and discrete attributes previous work has used mutual information gap as a metric to measure disentanglement (R. T. Chen et al., 2018). However, this is difficult to extend to the case of independent subspaces and complex criterion like semantics and syntax. Thus, we propose to use a much simpler alternative. We empirically compute correlations between the semantic and syntax latent representations for 1000 test sentences, to verify that there is not a signficant overlap in the information learned by the two encoders.

We obtain the $\boldsymbol{z}_{\text{sem}}$ and $\boldsymbol{z}_{\text{syn}}$ latent vectors for 1000 sentences and compute the correlation between them. We report the maximum absolute correlation (max across all pairs of dimensions) and also the mean absolute correlation. A higher correlation would indicate significant overlap between the subspaces learned by the semantic and syntactic encoders.

From Table 4.2 we observe that the correlation values of polarized-VAE are significantly lower than those of a standard variational autoencoder, thus indicating that the subspaces learn reasonably different information.

We would like to note that unfortunately the low correlation values do not necessarily imply that the subspaces are independent.

Model	Max Abs Corr$^{\downarrow}$	Mean Abs Corr$^{\downarrow}$
Baseline-Vae	0.62	0.1
Polarized-Vae	**0.25**	**0.05**

Table 4.2: Maximum Absolute Correlation and Mean Absolute Correlation between the semantic and syntactic latent vectors.

4.2.5 Human Evaluation

We also perform human evaluations to evaluate our model and compare it to the standard variational autoencoder and the approach based on adversarial training proposed by Bao et al., 2019.

Given two sentences, $\boldsymbol{x}_{\text{sem}}$ and $\boldsymbol{x}_{\text{syn}}$ we wish to generate a sentence that is semantically similar to sentence $\boldsymbol{x}_{\text{sem}}$ and has the syntactic structure of $\boldsymbol{x}_{\text{syn}}$ using the procedure described in Section 4.2.3. We then requested 5 human judges to evaluate the outputs produced by 3 models polarized-VAE, a baseline-VAE, and the approach proposed by Bao et al., 2019.

The judges were shown the input sentences $\boldsymbol{x}_{\text{sem}}$, $\boldsymbol{x}_{\text{syn}}$ and the outputs from the 3 models. The outputs from the 3 models were unlabeled and shuffled so that the judges were unaware of the model that generated a given sentence. The judges were then requested to pick one best output according to each of the following criteria 1) Semantic Transfer: (Semantic similarity to $\boldsymbol{x}_{\text{sem}}$ and dissimilarity to $\boldsymbol{x}_{\text{syn}}$) 2) Syntax Transfer (Syntactic similarity to $\boldsymbol{x}_{\text{syn}}$ and dissimilarity to $\boldsymbol{x}_{\text{sem}}$) and 3) Fluency.

We obtain the human evaluation scores for 100 test set examples from the SNLI corpus. We use majority voting and manual tie breaking to aggregate the human evaluation scores and find the best model for each criterion. In Table 4.3 we report the percentage of instances for which a given model was considered a best for each criterion.

Model	Semantics	Syntax	Fluency
Baseline-Vae	11	11	**43**
Bao et al., 2019	24	**58**	19
Polarized-Vae	**65**	31	38

Table 4.3: Human Evaluation scores on Semantics Syntax and Fluency reported as percentages.

We observe that our model performs the best at semantic transfer and is worse at

syntactic transfer in comparison to Bao et al., 2019. This is consistent with the quantitative results reported in Table 4.1 (ΔBLEU and ΔTED). Our model produces more fluent sentences in comparison to Bao et al., 2019 and is competitive with the baseline VAE.

4.2.6 Alternative proximity measures

We also experimented with multiple different proximity measures and report the results in Table 4.4.

Metric	ΔBLEU$^{\uparrow}$	ΔTED$^{\uparrow}$	ΔGM$^{\uparrow}$
Cosine Distance	**9.86**	**2.42**	**4.88**
Hellinger Distance	4.12	0.86	1.42
MMD	5.21	1.17	1.91
KL Divergence	4.32	0.75	1.28
JS Divergence	5.81	1.46	2.33b

Table 4.4: Comparison of different proximity functions we used in our experiments.

The results clearly indicate that the cosine distance performs the best among the proximity measures that we evaluated. Since there is no closed form expression for the Jensen Shannon divergence we use the generalized JS divergence proposed by Nielsen, 2019.

4.2.7 Transfer Examples

We provide qualitative examples of our transfer experiments, where we generate a sentence with the semantics of x_{sem} and the syntactic structure of x_{syn} in Table 4.5. We also provide the sentences generated by a standard-VAE for comparison.

$\boldsymbol{x}_{\mathbf{sem}}$	$\boldsymbol{x}_{\mathbf{syn}}$	**polarized-VAE**	**standard-VAE**
A man works near a vehicle.	A woman showing her face from something to her friend.	A man directing traffic on a bicycle to an emergency vehicle.	A woman works on a loom while sitting outside.
A family in a party preparing food and enjoying a meal.	Man reading a book.	A person enjoying food.	A man plays his guitar.
Two young boys are standing around a camera outdoors.	Three kids are on stage with a vacuum cleaner.	Two young boys are standing around a camera outdoors.	Two people are standing on a snowy hill.
There are a group of people sitting down.	They are outside.	There are people.	They are outside
A shirtless man wearing white shorts.	Two kids smiling.	A shirtless man wearing shorts .	A black dog .
The young girl and a grownup are standing around a table , in front of a fence.	A guy stands with cane outdoors.	The young girl is outside.	The little boy is doing a show.
A young girl washing a lime-green car	two men bask in the sunlight.	A young girl inspects a new car.	A man climbing a rock wall.
The men and women are enjoying a waterfall.	A dog is holding an object.	The man and woman are outdoors.	The two men are working on the roof.
a man dressed in uniform.	There is a man with a horse on it.	A man dressed in black clothing works in a house.	A man dressed in black and white holding a baby.

Table 4.5: Examples of transferred sentences that use the semantics of x_{sem} and syntax of x_{syn}

Chapter 5

Normalizing flows for text generation

In this chapter we discuss in detail our various approaches to improve text generation using normalizing flows.

5.1 Approach

In this work we will only be using Planar Normalizing Flow (PNF)s. We also propose a variant of planar normalizing flows for improving disentanglement which we call Block Planar Normalizing Flow (BPNF)s. We describe their applications and also propose an alternative to adversarial training for encouraging independent subspaces.

5.1.1 Block Planar Normalizing Flows

From equation 2.23 it is clear that when using planar normalizing flows each dimension in $f(\boldsymbol{z})$ potentially depends on all the dimensions of $\boldsymbol{z}$ because of the term $h(\boldsymbol{w}^T\boldsymbol{z} + b)$. This is useful when we simply want to model more complex distributions than the factorized Gaussian where the latant dimensions are no longer independent. However, when we intend to disentangle the representation into S independent subspaces, planar normalizing flows consequently encourage dependency between all dimensions.

To mitigate this we propose to use a variant of planar normalizing flows, where the dependence structure is introduced in a block-wise manner. Say we have S subspaces where the dimension of the ith subspace is d_i and $\Sigma_{s=1}^{S} d_s = d$. We then define the following transform.

$$f(\boldsymbol{z}) = \boldsymbol{z} + \Sigma_{s=1}^{S} \hat{\boldsymbol{u}}_s h(\hat{\boldsymbol{w}}_s^T \boldsymbol{z} + b_s) \tag{5.1}$$

Where $\hat{\boldsymbol{u}}_s$ is of the form $[\boldsymbol{0}^{d_1}, \ldots, \boldsymbol{u}_s, \ldots, \boldsymbol{0}^{d_S}]$ and $\boldsymbol{u}_s \in \mathbb{R}^{d_s}$ and $\hat{\boldsymbol{u}}_s \in \mathbb{R}^d$. Also $\hat{\boldsymbol{w}}_s$ is of the form $[\boldsymbol{0}^{d_1}, \ldots, \boldsymbol{w}_s, \ldots, \boldsymbol{0}^{d_S}]$ and $\boldsymbol{w}_s \in \mathbb{R}^{d_s}$ and $\hat{\boldsymbol{w}}_s \in \mathbb{R}^d$. We also observe that $\hat{\boldsymbol{w}}_s^T \boldsymbol{z} = \boldsymbol{w}_s^T \boldsymbol{z}^{(s)}$, leading to easier computation.

To compute the log-det jacobian we define:

$$\psi_s(\boldsymbol{z}^{(s)}) = h'(\boldsymbol{w}_s^T \boldsymbol{z}^{(s)} + b_s)\boldsymbol{w}_s \tag{5.2}$$

$$\hat{\psi}_s(\boldsymbol{z}) = h'(\hat{\boldsymbol{w}}_s^T \boldsymbol{z} + b_s)\hat{\boldsymbol{w}}_s \tag{5.3}$$

Using the definitions above we can compute $\left|\det \frac{\partial f}{\partial \boldsymbol{z}}\right|$ as follows:

$$\left|\det \frac{\partial f}{\partial \boldsymbol{z}}\right| = \left|\det\Big(\boldsymbol{I} + \Sigma_{s=1}^{S} \hat{\boldsymbol{u}}_s \hat{\psi}_s(\boldsymbol{z})^T\Big)\right| \tag{5.4}$$

We observe that we cannot directly apply the matrix determinant lemma to the determinant in equation 5.4, therefore we define matrices $\boldsymbol{U} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{V} \in \mathbb{R}^{d \times d}$.

$$\boldsymbol{U} = [\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_S] \tag{5.5}$$

$$\boldsymbol{V} = [\hat{\psi}_1(\boldsymbol{z}), \ldots, \hat{\psi}_S(\boldsymbol{z})] \tag{5.6}$$

From this definition we compute the matrix product $\boldsymbol{U}\boldsymbol{V}^T$ using block matrices:

$$\begin{aligned} \boldsymbol{U}\boldsymbol{V}^T &= \begin{bmatrix} \boldsymbol{u}_1 & 0 & \ldots & 0 \\ 0 & \boldsymbol{u}_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & \boldsymbol{u}_S \end{bmatrix} \times \begin{bmatrix} \psi_1^T(\boldsymbol{z}^{(1)}) & 0 & \ldots & 0 \\ 0 & \psi_2^T(\boldsymbol{z}^{(2)}) & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & \psi_S^T(\boldsymbol{z}^{(S)}) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{u}_1\psi_1^T(\boldsymbol{z}^{(1)}) & 0 & \ldots & 0 \\ 0 & \boldsymbol{u}_2\psi_2^T(\boldsymbol{z}^{(2)}) & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & \boldsymbol{u}_S\psi_S^T(\boldsymbol{z}^{(S)}) \end{bmatrix} \end{aligned} \tag{5.7}$$

Now we can rewrite equation 5.4 as follows:

$$\begin{aligned}\left|\det \frac{\partial f}{\partial \boldsymbol{z}}\right| &= \left|\det\left(\boldsymbol{I} + \boldsymbol{U}\boldsymbol{V}^T\right)\right| \\ &= \prod_{s=1}^{S} \left|\det\left(\boldsymbol{I}_{d_s} + \boldsymbol{u}_s \psi_s^T(\boldsymbol{z}^{(s)})\right)\right| \\ &= \prod_{s=1}^{S} \left|1 + \psi_s^T(\boldsymbol{z}^{(s)})\boldsymbol{u}_s\right| \end{aligned} \tag{5.8}$$

Where the second equality is obtained because $\boldsymbol{I} + \boldsymbol{U}\boldsymbol{V}^T$ is a block diagonal matrix and the final equality is obtained by applying the matrix determinant lemma to each product term.

Given the above transformation, we can apply a sequence of K such transformations to obtain:

$$\begin{aligned}\boldsymbol{z}_K &= f_K \circ ... f_2 \circ f_1 \\ \log q_K(\boldsymbol{z}_K) &= \log q_0(\boldsymbol{z}_0) - \Sigma_{k=1}^{K} \Sigma_{s=1}^{S} \log \left|1 + \boldsymbol{u}_{sk}^T \psi_{sk}(\boldsymbol{z}^{(s)})\right| \end{aligned} \tag{5.9}$$

We would like to note that the total number of free parameters in the transformation for a single block planar normalizing flows is only $S-1$ more than that of a planar normalizing flow. The difference arises from having bias parameters unique to each subspace. The rest of the parameters are equivalent in size because many of the elements in the block planar transformation are fixed to be zeroes.

5.1.2 Subspace Entropy Loss

Adversarial losses are commonly used for encouraging disentanglement into independent subspaces (John et al., 2019). However, adversarial training is difficult and not very stable. We propose to use an entropy based loss that can be jointly optimized with the overall objective without using adversarial training.

Earlier works such as the InfoGAN (X. Chen et al., 2016) and β-TCVAE (R. T. Chen et al., 2018) enforced statistical independence objectives for the dimensions of the latent space. They can be viewed as performing Independent Component Analysis (ICA) on the latent space. We seek to extend this to independent subspaces, therefore a natural

extension is to use Independent Subspace Analysis (ISA). Awiszus et al., 2019 successfully apply this idea for disentangling the latent space of images.

For a given batch we obtain the encoded subspaces having dimensions $d_1, \ldots, d_S$, let the maximum dimension be $d_{\max}$. We transform the latent vectors of each subspace into a vector of size $d_{\max}$ by using functions $F_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{\max}}$.

We can now stack the matrices corresponding to different subspaces $\hat{Z} = [\hat{Z}_1, \ldots, \hat{Z}_S]$ and learn to classify each of the vectors as belonging to a given subspace using the cross-entropy loss. We call this the subspace entropy loss $\mathcal{L}_{\text{SE}}$ which can be directly added to any variational inference objective alongside a hyperparameter λ_{SE}.

5.2 Experiments

In this section we perform various experiments that demonstrate the effectiveness of normalizing flows across many tasks including language modeling, text generation and disentanglement. We discuss the training setup, the evaluation metrics and the results.

5.2.1 Datasets

We use the following datasets across various tasks:

- **Penn Tree Bank (PTB)**: We use the Penn Treebank (Marcus et al., 1993) dataset for language modeling tasks. It contains a total of 49198 sentences with the following train/validation/test split (42067/3370/3761).
- **Amazon Product Reviews**: We use the amazon product reviews dataset following Fu et al., 2018. We use this dataset for both language modeling and disentanglement tasks.
- **Stanford Natural Language Inference (SNLI)**: We use the SNLI (S. Bowman et al., 2015) dataset to demonstrate the tradeoff between reconstruction and generation quality. We also use this dataset for disentangling semantics and syntax as described in Section 4.2.

5.2.2 Training details and hyperparameters

The experiments involving normalizing flows always use a sequence of 3 normalizing flows ($K = 3$). The rest of the experimental setup for semantics and syntax transfer remains the same as described in Section 4.2.1.

For the language modeling experiments we use a latent space dimension of 32. The encoders are bidirectional LSTMs with hidden size of 200. The decoder is a unidirectional LSTM with hidden size 200. We train all the models by using the ADAM optimizer Kingma and Ba, 2015 with a learning rate of 0.001. We train all the models on all the datasets for a maximum of 50 epochs, but we use a stopping criterion of ELBO with patience of 5 epochs; i.e if the validation ELBO does not improve for 5 epochs, then we stop training further. The VAE experiments all use a linearly annealing schedule for the KL weight from 0 to 1 over 21 epochs.

The WAE experiments use an MMD weight of 10, and we also add a small KL divergence term to ensure that the individual posteriors do not diverge away from the prior.

The style transfer experiments on the amazon dataset follows the same experimental setup as used by John et al., 2019.

5.2.3 Evaluation Metrics

For the language modeling experiments we measure the performance of the models across various criteria, mainly the optimization performance and quality of sentence generation. The Negative Log Likelihood (NLL) is approximated by the evidence lower bound. We report the NLL and the perplexity computed from the log likelihood. We also report the KL term to show the mitigation of posterior collapse behaviour.

We also evaluate the generation quality of the models by using diversity measures. To measure diversity, we sample 10 responses for each input sentence and measure the following metrics by averaging over the entire test set.

- **distinct-1**: This metric calculates the proportion of unique unigrams in the generated sentences.
- **distinct-2**: This metric calculates the proportion of unique bigrams in the generated sentences.
- **Average Sentence Length (ASL)**: We also report the average sentence length of the generated sentences.

For the semantics and syntax disentanglement experiments, the metrics used are the same as those described in Section 4.2.3.

For the disentanglement of sentiment and style transfer experiments on the amazon dataset, we use the following metrics as used by John et al., 2019.

- **Style Transfer Accuracy (STS)**: This metric measures the accurarcy of a pre-trained classifier in classifying the transferred sentences to their desired target label. Thus, it indicates how successfully a model can transfer sentences to a desired target style.
- **Cosine Similarity (CS)**: We use the content preservation metric used by John et al., 2019 to measure the cosine similarity between the source sentence and the target sentence by using GloVe (Pennington et al., 2014) word embeddings. This indicates whether the source and target sentence have similar content.
- **Perplexity (PPL)**: We measure the fluency of the generated sentences by using a trigram language model (Heafield, 2011) to compute the perplexity of the generated sentences. Lower perplexity indicates higher fluency.

5.2.4 Language Modeling and Generation

We perform experiments that evaluate the effectiveness of planar normalizing flows for language modeling and quality of generation on two datasets: Penn Treebank (PTB, Marcus et al., 1993) and the Stanford Natural Language Inference (SNLI, S. Bowman et al., 2015) datasets.

Model	NLL$^{\downarrow}$	KL$^{\uparrow}$	PPL$^{\downarrow}$	distinct-1$^{\uparrow}$	distinct-2$^{\uparrow}$	ASL$^{\uparrow}$
VAE	106.1	2.42	123.1	0.324	0.588	15.65
VAE-NF	97.1	4.41	84.2	**0.358**	**0.629**	20.32
WAE	105.2	10.45	132.4	0.314	0.561	15.04
WAE-NF	**93.1**	**15.44**	**68.4**	0.351	0.599	**20.55**

Table 5.1: Language modeling and generation results on the PTB dataset

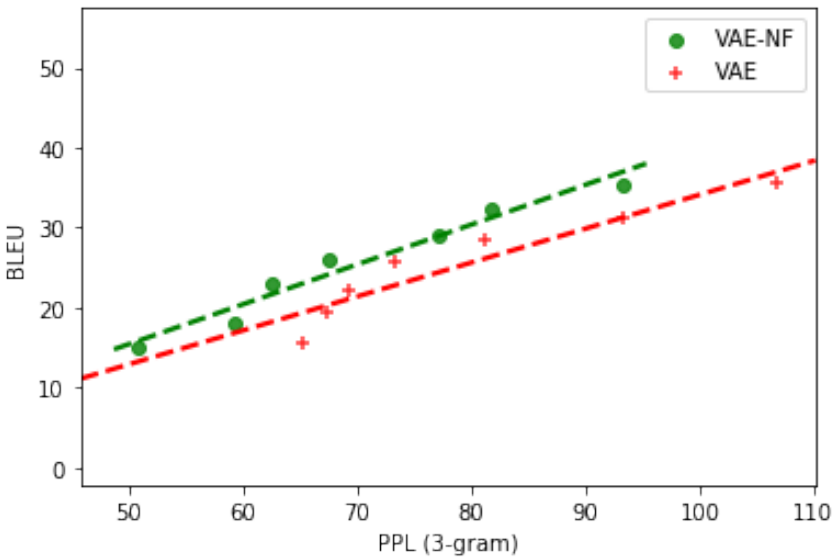
From Table 5.1 and Table 5.2 it is clear that the use of normalizing flows provides various benefits. It mitigates posterior collapse as indicated by the higher KL divergence even while achieving better negative log likelihood. The diversity measures indicate that

Model	NLL$^{\downarrow}$	KL$^{\uparrow}$	PPL$^{\downarrow}$	distinct-1$^{\uparrow}$	distinct-2$^{\uparrow}$	ASL$^{\uparrow}$
VAE	74.42	2.03	61.28	0.342	0.609	19.77
VAE-NF	69.31	4.07	45.42	**0.383**	**0.664**	**20.21**
WAE	73.27	13.42	68.28	0.336	0.587	19.56
WAE-NF	**64.35**	**18.43**	**32.66**	0.371	0.653	19.98

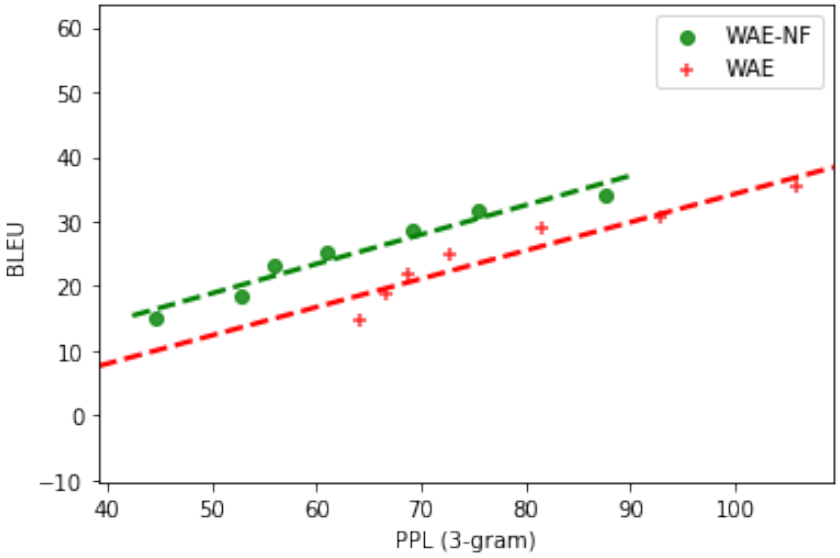
Table 5.2: Language modeling and generation results on the Amazon dataset

normalizing flows improve the diversity of the generated sentences and also produce longer sentences on average.

We also compare the reconstruction and generation quality tradeoff on the SNLI dataset. From Figure 5.1a and Figure 5.1b it is clear that with the usage of normalizing flows we can simultaneously improve both the reconstruction and the generation quality.



(a) Tradeoff for VAE and VAE-NF.



(b) Tradeoff for WAE and WAE-NF

Figure 5.1: Comparing the BLUE vs PPL (3-gram) tradeoff for the standard models and their normalizing flow variants.

5.2.5 Semantics and Syntax Disentanglement

We apply normalizing flows to our proximity-VAE model and evaluate it on the task of semantics syntax transfer as described in Section 4.2.4. We report the ablation results of the different variants in Table 5.3.

Model	BLEU			TED			
	$\boldsymbol{x}_{\text{sem}}{}^{\uparrow}$	$\boldsymbol{x}_{\text{syn}}{}^{\downarrow}$	ΔBLEU$^{\uparrow}$	$\boldsymbol{x}_{\text{sem}}{}^{\uparrow}$	$\boldsymbol{x}_{\text{syn}}{}^{\downarrow}$	ΔTED$^{\uparrow}$	ΔGM$^{\uparrow}$
polarized-VAE	10.78	0.92	9.86	14.09	11.67	2.42	4.88
polarized-VAE (`wo`)	9.82	0.84	8.98	**14.12**	11.65	2.47	4.71
polarized-VAE (`len`)	10.10	**0.76**	9.34	12.68	11.44	1.24	3.40
polarized-VAE (`wo`, `len`)	9.41	0.87	8.54	12.65	11.48	1.17	3.16
polarized-VAE + PNF	10.97	1.08	9.89	14.05	11.58	2.47	4.94
polarized-VAE (wo) + PNF	9.96	0.88	9.08	14.07	11.56	2.51	4.77
polarized-VAE (len) + PNF	10.21	0.82	9.39	12.67	11.42	1.25	3.43
polarized-VAE (wo, len) + PNF	9.53	0.94	8.59	12.63	11.47	1.16	3.16
polarized-VAE + PNF + $\mathcal{L}_{\text{SE}}$	10.98	1.07	9.91	14.06	11.56	2.5	4.98
polarized-VAE (wo) + PNF + $\mathcal{L}_{\text{SE}}$	9.95	0.86	9.09	14.06	11.55	2.51	4.78
polarized-VAE (len) + PNF + $\mathcal{L}_{\text{SE}}$	10.19	0.78	9.41	12.68	11.41	1.27	3.46
polarized-VAE (wo, len) + PNF + $\mathcal{L}_{\text{SE}}$	9.49	0.92	8.57	12.64	11.45	1.19	3.19
polarized-VAE + BPNF	10.97	1.07	9.9	14.04	11.57	2.47	4.94
polarized-VAE (wo) + BPNF	9.97	0.89	9.08	14.08	11.56	2.52	4.78
polarized-VAE (len) + BPNF	10.18	0.79	9.39	12.65	11.40	1.25	3.43
polarized-VAE (wo, len) + BPNF	9.51	0.92	8.59	12.65	11.43	1.22	3.24
polarized-VAE + BPNF + $\mathcal{L}_{\text{SE}}$	**11.05**	1.03	**10.02**	14.10	11.55	2.55	**5.05**
polarized-VAE (wo) + BPNF + $\mathcal{L}_{\text{SE}}$	9.99	0.84	9.15	14.10	11.53	**2.57**	4.85
polarized-VAE (len) + BPNF + $\mathcal{L}_{\text{SE}}$	10.24	0.77	9.47	12.63	**11.38**	1.25	3.44
polarized-VAE (wo, len) + BPNF + $\mathcal{L}_{\text{SE}}$	9.55	0.89	8.66	12.67	11.42	1.25	3.29

Table 5.3: Normalizing flow ablation results of syntax transfer generation on SNLI dataset

The results clearly indicate that using block planar normalizing flows to disentangle into subspaces and enforcing the subspace entropy loss leads to the best results. We do however observe that the block planar normalizing flows do not provide a signficant advantage over the planar normalizing flows without the independence constraint.

5.2.6 Style Transfer on Amazon Reviews

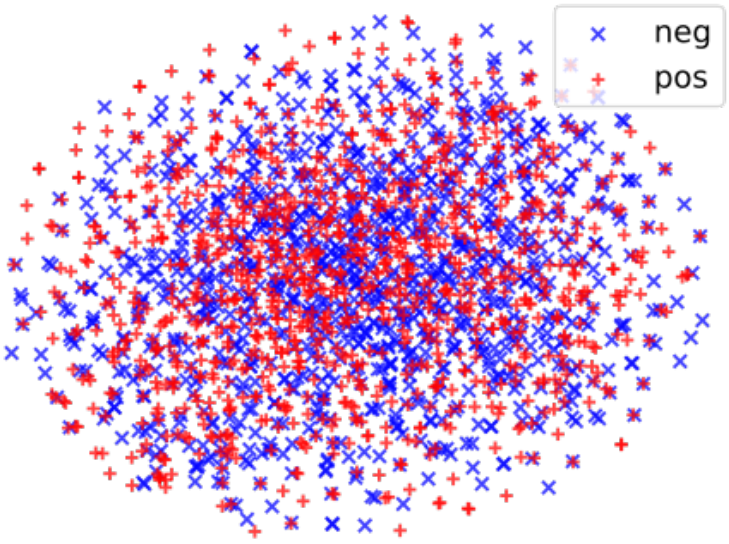
We follow the work of John et al., 2019 and use their model and experimental setup as the basis for the style transfer experiments on the amazon dataset. Their objective uses three main components: the reconstruction loss $\mathcal{L}_{\text{rec}}$, the multitask loss $\mathcal{L}_{\text{mult}}$ and the adversarial loss $\mathcal{L}_{\text{adv}}$.

We use planar normalizing flows and block planar normalizing flows to obtain more flexible posteriors. We also attempt to replace the adversarial training used in the model by our proposed subspace entropy loss (Section 5.1.2) and report the ablation results in Table 5.4.

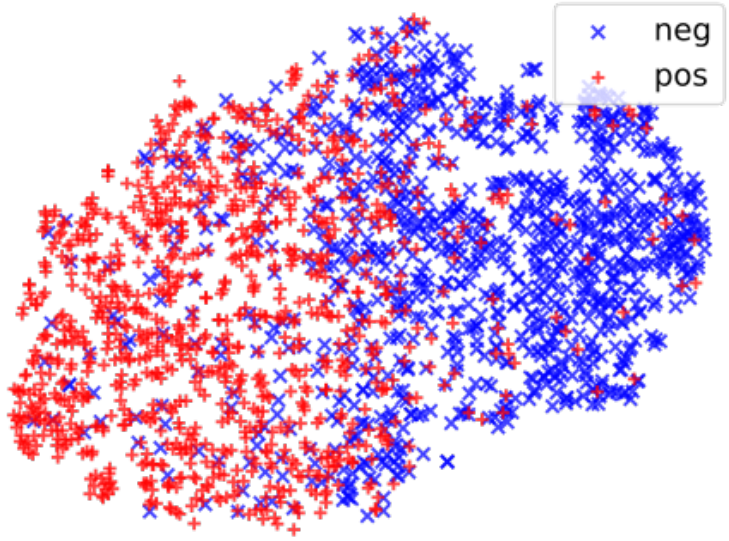
Objectives	**STA**$^{\uparrow}$	**CS**$^{\uparrow}$	**PPL**(3-gram)$^{\downarrow}$
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{adv}}$	0.79	0.8952	63.24
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{SE}}$	0.69	0.8431	68.18
$\mathcal{L}_{\text{rec}}$ + PNF, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{adv}}$	0.80	0.9014	61.94
$\mathcal{L}_{\text{rec}}$ + PNF, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{SE}}$	0.72	0.8615	62.25
$\mathcal{L}_{\text{rec}}$ + BPNF, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{adv}}$	**0.81**	**0.9032**	**59.80**
$\mathcal{L}_{\text{rec}}$ + BPNF, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{SE}}$	0.75	0.8746	60.32

Table 5.4: Style transfer ablation results on the amazon dataset

We observe that adversarial training is still the strongest performing objective to enforce independence between subspaces. However the subspace entropy loss when combined with block planar normalizing flows is competitive with adversarial training. We do note that utilizing normalizing flows does improve performance slightly on style transfer metrics and gives a significant boost to the fluency of the generated sentences.



(a) Distribution of sentences in the content space



(b) Distribution of sentences in the sentiment space

Figure 5.2: Comparing the subspaces learned by the style transfer model with BPNF and the subspace entropy loss by using T-SNE plots

Original (Positve)	**Transferred (Negative)**
this thing works it is affordable permanent hepa filter and quiet	i have had problems with the unit
they both worked very well for hand held tools	i have used it for a few months and it is not a good
the water tank lasts weeks before needing refilling and the steamer produces consistent steam	i have had a problem with the tank and the plastic is not too good
it fit perfectly looks identical to the existing one and holds a charger perfectly	i have had a problem with the fit
i am very happy with this set of measuring cups they are a great addition to my kitchen	the rubber edges are very difficult to clean
Original (Negatve)	**Transferred (Positive)**
the material is flimsy and feels very thin in the hand	material is thin but it is good
i do have thick wavy hair but i have not experienced my hair this bad since i was on a swim team number years ago	i have not experienced any issues with the product and i am not sure
this case looks adorable but it fits my phone very loosely	it is a great case for my iphone
the build quality is ok but it is number plastic feeling number	the quality is very sturdy and is very easy to clean
this new design is thick and too heavy	the design is a great deal for the price

Table 5.5: Sentiment transfer on the amazon dataset performed by VAE + BPNF + $\mathcal{L}_{\text{SE}}$

Chapter 6

Conclusion and Future Work

6.1 Summary

In this thesis, we explored two different directions for improving the posterior of the approximate distribution used by models that use variational inference. We explored adding a proximity based loss to disentangle the subspaces learned by variational autoencoders by using additional information about pairwise similiarity. We evaluate this approach by disentangling the subspaces into encoding semantics and syntactic information about a sentence and perform semantics-syntax transfer experiment on them. We empirically show that our model performs competitively with existing approaches and successfully performs semantics-syntax transfer experiments. We also provide qualitative examples of successful transfer for illustration.

We also explored the effect of using normalizing flows to improve the approximate posterior used by models based on variational inference. We perform experiments on both VAE and WAE to show that normalizing flows improve performance on language modeling tasks and generation quality and diversity. Normalizing flows also help in producing more fluent and longer sentences. We notice that normalizing flows help mitigate the issue of posterior collapse in variational autoencoders, since more complex distributions would allow us to match the true posterior more easily. This in turn encourages the decoder to make use of the more informative latent variables thus mitigating posterior collapse.

We propose a variant of planar normalizing flows called block planar normalizing flows to encourage disentanglement into independent subspaces. We also proposed to use a subspace entropy loss based on independent subspace analysis to enforce independence

between subspaces as an alternative to adversarial losses. We perform ablation experiments on semantics-syntax disentanglement on the SNLI dataset and also perform sentiment transfer experiments on the amazon dataset. We observe that the adversarial losses perform best in terms of enforcing disentanglement between subspaces, but the entropy based losses are competitive, which makes them possibly useful because adversarial losses are difficult to train in a stable manner. Overall our experiments empirically show that normalizing flows and improved posteriors in general improve variational inference and help generative models across in learning better representations.

6.2 Future Work

We experimented with multiple proximity based losses for our polarized-VAE and found the cosine distance to work the best. More loss functions could be further investigated and their properties studied to determine what types of proximity functions are bested suited for this task. In the future we could also investiage the applicability of this approach to multi-attribute disentanglement.

In this work we only investigate simple normalizing flows such as planar flows. In the future we could investigate the performance of more powerful transformations such as autoregressive flows. It is also possible that more powerful flows could help mitigate the shortcomings of the subspace entropy loss when compared to adversarial training, this could be investigated with more experiments. Further investigations could be carried about the power of normalizing flows apart from language modeling and disentanglement including conditional text generation tasks such as dialogue response generation.

References

Akuzawa, K., Iwasawa, Y., & Matsuo, Y. (2018). Expressive speech synthesis via modeling expressions with variational autoencoder. *arXiv preprint arXiv:1804.02135*.

Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: On the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

Awiszus, M., Ackermann, H., & Rosenhahn, B. (2019). Learning disentangled representations via independent subspaces, In *The ieee international conference on computer vision (iccv) workshops*.

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

Bahuleyan, H., Mou, L., Vamaraju, K., & Zhou, H. (2019). Stochastic wasserstein autoencoder for probabilistic sentence generation, In *Naacl-hlt*.

Bando, Y., Mimura, M., Itoyama, K., Yoshii, K., & Kawahara, T. (2018). Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization, In *2018 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE.

Bao, Y., Zhou, H., Huang, S., Li, L., Mou, L., Vechtomova, O., Dai, X., & Chen, J. (2019). Generating sentences from disentangled syntactic and semantic spaces, In *Proceedings of the 57th annual meeting of the association for computational linguistics*.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., & Bengio, S. (2016). Generating sentences from a continuous space, In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, Berlin, Germany, Association for Computational Linguistics. https://doi.org/10.18653/v1/K16-1002

Bowman, S., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference, In *Conference on empirical methods in natural language processing, emnlp 2015*. Association for Computational Linguistics (ACL).

Cambria, E., & White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, *9*(2), 48–57.

Chan, Z., Li, J., Yang, X., Chen, X., Hu, W., Zhao, D., & Yan, R. (2019). Modeling personalization in continuous space for response generation via augmented wasserstein autoencoders, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*.

Chandar A P, S., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., & Saha, A. (2014). An autoencoder approach to learning bilingual word representations (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger, Eds.). In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27*. Curran Associates, Inc. http://papers.nips.cc/paper/5270-an-autoencoder-approach-to-learning-bilingual-word-representations.pdf

Chen, M., Tang, Q., Wiseman, S., & Gimpel, K. (2019). Controllable paraphrase generation with a syntactic exemplar, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1599

Chen, R. T., Li, X., Grosse, R. B., & Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders, In *Advances in neural information processing systems*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets, In *Advances in neural information processing systems*.

Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A.-r., & Hinton, G. (2010). Binary coding of speech spectrograms using a deep auto-encoder, In *Eleventh annual conference of the international speech communication association*.

Eastwood, C., & Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations, In *International conference on learning representations*.

Fefferman, C., Mitter, S., & Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, *29*(4), 983–1049.

Fu, Z., Tan, X., Peng, N., Zhao, D., & Yan, R. (2018). Style transfer in text: Exploration and evaluation, In *Aaai*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Graves, A., Fernández, S., & Schmidhuber, J. (2005). Bidirectional lstm networks for improved phoneme classification and recognition, In *International conference on artificial neural networks*. Springer.

Gu, X., Cho, K., Ha, J.-W., & Kim, S. (2018). Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder. *arXiv preprint arXiv:1805.12352*.

He, J., Spokoyny, D., Neubig, G., & Berg-Kirkpatrick, T. (2019). Lagging inference networks and posterior collapse in variational autoencoders, In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. https://openreview.net/forum?id=rylDfnCqF7

Heafield, K. (2011). KenLM: faster and smaller language model queries, In *Proceedings of the EMNLP 2011 sixth workshop on statistical machine translation*, Edinburgh, Scotland, United Kingdom. https://kheafield.com/papers/avenue/kenlm.pdf

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M. M., Mohamed, S., & Lerchner, A. (2017). Beta-vae: Learning basic visual concepts with a constrained variational framework, In *Iclr*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), https://doi.org/10.1162/neco.1997.9.8.1735, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoogeboom, E., van den Berg, R., & Welling, M. (2019). Emerging convolutions for generative normalizing flows. *ArXiv*, *abs/1901.11137*.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John, V., Mou, L., Bahuleyan, H., & Vechtomova, O. (2019). Disentangled representation learning for non-parallel text style transfer, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1041

Kim, H., & Mnih, A. (2018). Disentangling by factorising. *arXiv preprint arXiv:1802.05983*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *CoRR*, *abs/1312.6114*.

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett, Eds.). In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31*. Curran Associates, Inc. http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.pdf

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow, In *Advances in neural information processing systems*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37*(2), https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.69037 233–243. https://doi.org/10.1002/aic.690370209

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., & Kingma, D. (2019). Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, *2*(5).

Li, S., Kawale, J., & Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder, In *Proceedings of the 24th acm international on conference on information and knowledge management*.

Li, X., & She, J. (2017). Collaborative variational autoencoder for recommender systems, In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations (K. Chaudhuri & R. Salakhutdinov, Eds.). In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning*, Long Beach, California, USA, PMLR. http://proceedings.mlr.press/v97/locatello19a.html

Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013). Speech enhancement based on deep denoising autoencoder., In *Interspeech*.

Lucas, J., Tucker, G., Grosse, R. B., & Norouzi, M. (2019). Don't blame the elbo! a linear vae perspective on posterior collapse, In *Advances in neural information processing systems*.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330. https://www.aclweb.org/anthology/J93-2004

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model, In *Interspeech*.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., Et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Nielsen, F. (2019). On a generalization of the jensen-shannon divergence and the js-symmetrization of distances relying on abstract means. *arXiv preprint arXiv:1904.04017*.

Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial*

Intelligence, *60*, 97–116. https://doi.org/https://doi.org/10.1016/j.engappai.2017.01.013

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation, In *Proceedings of the 40th annual meeting on association for computational linguistics*, Philadelphia, Pennsylvania, Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation, In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Razavi, A., van den Oord, A., Poole, B., & Vinyals, O. (2019). Preventing posterior collapse with delta-VAEs, In *International conference on learning representations*. https://openreview.net/forum?id=BJe0Gn0cY7

Reed, S. E., Zhang, Y., Zhang, Y., & Lee, H. (2015). Deep visual analogy-making, In *Nips*.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows (F. Bach & D. Blei, Eds.). In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning*, Lille, France, PMLR. http://proceedings.mlr.press/v37/rezende15.html

Rubenstein, P. K., Schoelkopf, B., & Tolstikhin, I. (2018). Wasserstein auto-encoders: Latent dimensionality and random encoders. https://openreview.net/forum?id=r157GIJvz

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

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

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Shen, X., Su, H., Niu, S., & Demberg, V. (2018). Improving variational encoder-decoders in dialogue generation, In *Thirty-second aaai conference on artificial intelligence*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Suh, S., Chae, D. H., Kang, H.-G., & Choi, S. (2016). Echo-state conditional variational autoencoder for anomaly detection, In *2016 international joint conference on neural networks (ijcnn)*. IEEE.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). Lstm neural networks for language modeling, In *Thirteenth annual conference of the international speech communication association*.

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

Tao, X., & Zhou, Y. (2017). Fall prediction based on biomechanics equilibrium using kinect. *International Journal of Distributed Sensor Networks*, *13*, 155014771770325. https://doi.org/10.1177/1550147717703257

Tolstikhin, I., Bousquet, O., Gelly, S., & Schoelkopf, B. (2018). Wasserstein auto-encoders, In *International conference on learning representations*. https://openreview.net/forum?id=HkL7n1-0b

Wang, P. Z., & Wang, W. Y. (2019). Riemannian normalizing flow on variational wasserstein autoencoder for text modeling (J. Burstein, C. Doran, & T. Solorio, Eds.). In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019, minneapolis, mn, usa, june 2-7, 2019, volume 1 (long and short papers)*, Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1025

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., & Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, J., & Durrett, G. (2018a). Spherical latent spaces for stable variational autoencoders, In *Emnlp*.

Xu, J., & Durrett, G. (2018b). Spherical latent spaces for stable variational autoencoders, In *Emnlp*.

Xu, W., Sun, H., Deng, C., & Tan, Y. (2017). Variational autoencoder for semi-supervised text classification, In *Thirty-first aaai conference on artificial intelligence*.

Yan, X., Yang, J., Sohn, K., & Lee, H. (2016). Attribute2image: Conditional image generation from visual attributes, In *European conference on computer vision*. Springer.

Zhang, K., & Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, *18*, 1245–1262. https://doi.org/10.1137/0218082

Zhao, J., Kim, Y., Zhang, K., Rush, A. M., & LeCun, Y. (2018). Adversarially regularized autoencoders, In *35th international conference on machine learning, icml 2018*. International Machine Learning Society (IMLS).

Zhong, J., & Zhang, X. (2018). Wasserstein autoencoders for collaborative filtering. *arXiv preprint arXiv:1809.05662*.

Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Ziegler, Z. M., & Rush, A. M. (2019). Latent normalizing flows for discrete sequences. *arXiv preprint arXiv:1901.10548*.