Disentangled Representation Learning for Stylistic Variation in Neural Language Models

by

Vineet John

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, 2018

I hereby declare that I am the sole author of this 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.

Abstract

The neural network has proven to be an effective machine learning method over the past decade, prompting its usage for modelling language, among several other domains. However, the latent representations learned by these neural network function approximators remain uninterpretable, resulting in a new wave of research efforts to improve their explainability, without compromising on their predictive power.

In this work, we tackle the problem of disentangling the latent style and content variables in a language modelling context. This involves splitting the latent representations of documents, by learning which features of a document are discriminative of its style and content, and encoding these features separately using neural network models. To achieve this, we propose a simple, yet effective approach, which incorporates auxiliary objectives: a multi-task classification objective, and dual adversarial objectives for label prediction and bag-of-words prediction, respectively. We show, both qualitatively and quantitatively, that the style and content are indeed disentangled in the latent space, using this approach.

We apply this disentangled latent representation learning method to attribute (e.g. style) transfer in natural language generation. We achieve similar content preservation scores compared to previous state-of-the-art approaches, and considerably better style-transfer strength scores.

Our code is made publicly available for experiment replicability and extensibility.

Acknowledgements

I would like to begin by thanking my supervisor, Dr. Olga Vechtomova, without whom none of this would be possible. Her constant guidance and feedback have been indispensable to my learning and progress, and she has been a continuous source of ideas and inspiration that have been instrumental in the research work presented in this thesis. I'm grateful to Dr. Pascal Poupart and Dr. Mei Nagappan for agreeing to be readers on my thesis committee.

I would also like to thank Dr. Lili Mou, Hareesh Bahuleyan and Ankit Vadehra for the stimulating discussions we have had, and the work done together.

Finally, I extend my gratitude to my family and friends for their unwavering support and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

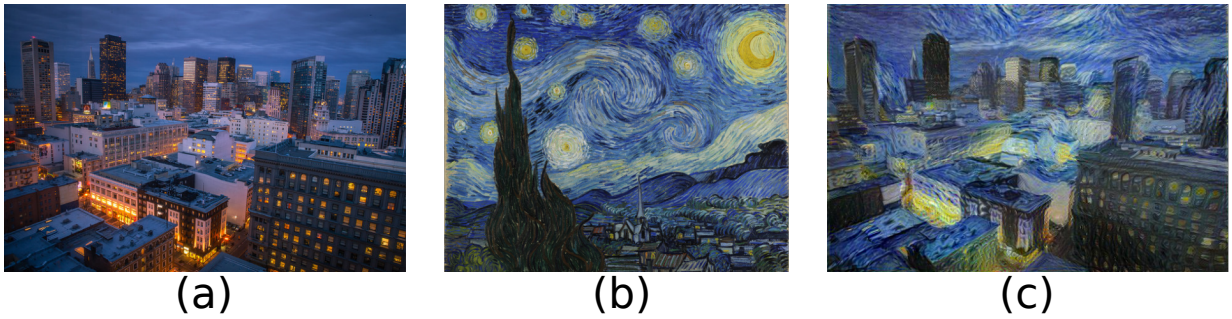
Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI) that deals with the understanding and generation of natural human languages.

In recent times, many of the tasks traditionally addressed by statistical NLP methods have been moving towards the adoption of neural networks, in an effort to learn more expressive models of language. This includes tasks like machine translation, dialogue modelling, abstract summarization, document classification, etc.

In this work, we address the problem of neural disentanglement of style and content in text. We build a unified model that learns separately regularized latent spaces, thereby isolating both style and content. The main benefit of doing this is that the separate latent representations are now interpretable on a macro-level with respect to the style, as we control the flow of style-related information in the model.

As an added benefit, this disentangling model can also be directly applied to the problem of text style transfer. This is analogous to style transfer in computer vision [Gatys et al., 2016]. The requirement of the style transfer task in the vision domain is to transfer the visual style from one image to the other, as illustrated in Figure 1.1. Stylistic transfer in text is based on a similar premise, wherein, given an arbitrary body of text and a predefined style - governed by a set of one or more attributes like sentiment, emotion, tense, authorship, etc. - a new body of text can be generated, such that it incorporates the predefined style its generation is being conditioned on. We evaluate the efficacy of our method by generating text conditioned on any of a given set of discrete 'labels' (used interchangeably with 'attribute' or 'style' henceforth).

Novel text synthesis in multiple styles can be used in application areas including:



Source: https://github.com/fzliu/style-transfer

Figure 1.1: Visual Style Transfer: (a) Content, (b) Style and (c) Synthesized Images

- Personalized styles in conversational agents.
- Converting involved pieces of text into a lucid format.
- Generation of additional high-quality plausible source sentences to augment an existing dataset.

The task of style transfer in context of text was first introduced by Xu et al. [2012] as a statistical model that attempted to paraphrase bodies of text in a different style using a simple phrase replacement strategy. A few examples from this paper are shown in Table 1.1. Although fairly crude, this retrieve and replacement strategy is effective, given a large dataset with enough phrase context overlap. Since the overwhelming adoption of neural network based models in the NLP community, there have been several recent bodies of work that break new ground in this area. These are discussed in the chapter dedicated to related work.

1.1 Problem Statement

The objective of this work is to perform an exploratory analysis of previous methods and test novel hypotheses that tackle the problem of the disentanglement of latent spaces of artificial neural networks and its applications to linguistic style transfer.

We operate under the following constraints and assumptions in our formulation of the problem:

Input	Output
i will bite thee by the ear for that jest .	i ' ll bite you by the ear for that joke .
what further woe conspires against mine age ?	what ' s true despair conspires against my old age ?
how doth my lady ?	how is my lady ?
hast thou slain tybalt ?	have you killed tybalt ?
an i might live to see thee married once , i have my wish .	if i could live to see you married, i ' ve my wish .
benvolio , who began this bloody fray ?	benvolio , who started this bloody fight itself ?
what is your will ?	what do you want ?
call her forth to me .	bring her out to me .

Source: Xu et al. [2012]

Table 1.1: Authorship Style Transfer from Shakespearean Plays

- The model is singular, with no conditional execution branch based on desired attribute i.e. there is only one decoder and the number of decoders does not scale with the number of distinct transferable attributes.
- The corpus of labelled text is non-parallel i.e. for instance, there are no predefined pairs of $(document_1, document_2)$ for style labels $\in (1, 2)$, as would be commonly seen in neural machine translation corpora.
- The corpus is annotated with the current attribute each document possesses e.g. each document has a corresponding 'positive' or 'negative' label if the task is to perform sentiment transfer.
- Optionally, a lexicon of words that are statistically likely to be associated with each distinct class label would be useful, but is not required. If present, it could be used primarily to evaluate how well content has been preserved in a generated sentence, without penalizing a potential change in vocabulary caused by the transferring of style.

1.2 Contributions

Our main contributions in this work can be stated as follows:

- We present a comprehensive review of the current literature on linguistic style transfer. It is a quickly evolving sub-area in NLP and a considerable amount of work has been done in the last year itself.
- We propose a combination of methods that have worked well independently, including the use of adversarial learning and multi-task learning objectives as regularizations on the latent space of neural models.
- We propose the usage of a novel bag-of-words adversary and a style dropout regularization that are aimed at further increasing the potency of latent space disentanglement and auto-encoding accuracy. Our model is novel in the aspect of a dual-adversary regularization as well.
- We empirically show that embeddings in the latent space are regularized like we intend them to be, by training auxiliary classification tasks and plotting embedding samples, a feature that is absent in contemporary approaches that use discrete labels or embedding matrices to represent style. Our model is able to disentangle latent representations of style and content, and also obtains a good separation of classes in the style latent space.
- We show that our model outperforms current state-of-the-art models in neural text attribute style transfer, despite not explicitly training a style transfer objective.
- We devise new metrics to evaluate text content preservation quantitatively, which is still an open research problem, since the corpora used is non-parallel. Most of the contemporary works either do not measure text content preservation, or use human evaluation, which is time consuming and doesn't scale to large corpora.
- Our implementation is open-sourced for inspection, improvement, extension and replicability.

In this chapter, we have provided the motivation for the problem being addressed, and stated our contributions. The next chapter will delve deeper into the background required for an understanding of the models we implement and evaluate.

Chapter 2

Background

2.1 Natural Language Generation

Natural Language Generation (NLG) is a sub-field of Natural Language Processing that attempts to generate sequences of words that resemble natural human languages. Traditionally, this was done either by using production rules of a predefined grammar, or by performing statistical analyses of existing human-written texts to predict sequences of words, based on their occurrence probabilities [Cambria and White, 2014]. Markov-chain text generators are an example of the latter, popularized by their usage in parody text generation [Jelinek, 1985].

This broad classification of problems has multiple applications including, but not limited to:

- Neural machine translation (NMT), in which the generation objective is to produce a semantically similar sentence in a target language, given a sentence in a source language [Bahdanau et al., 2014, Cho et al., 2014a, Luong et al., 2015a, Wu et al., 2016].
- Dialogue generation, in which the objective is to produce a natural and syntactically correct response to a provided utterance [Cavazza and Charles, 2005, Li et al., 2016a,b, 2017].
- Text summarization, which eliminates superfluous and non-pertinent information in a body of text, to express the idea using fewer words [Barzilay and Elhadad, 1999, Gong and Liu, 2001, Conroy and O'leary, 2001].

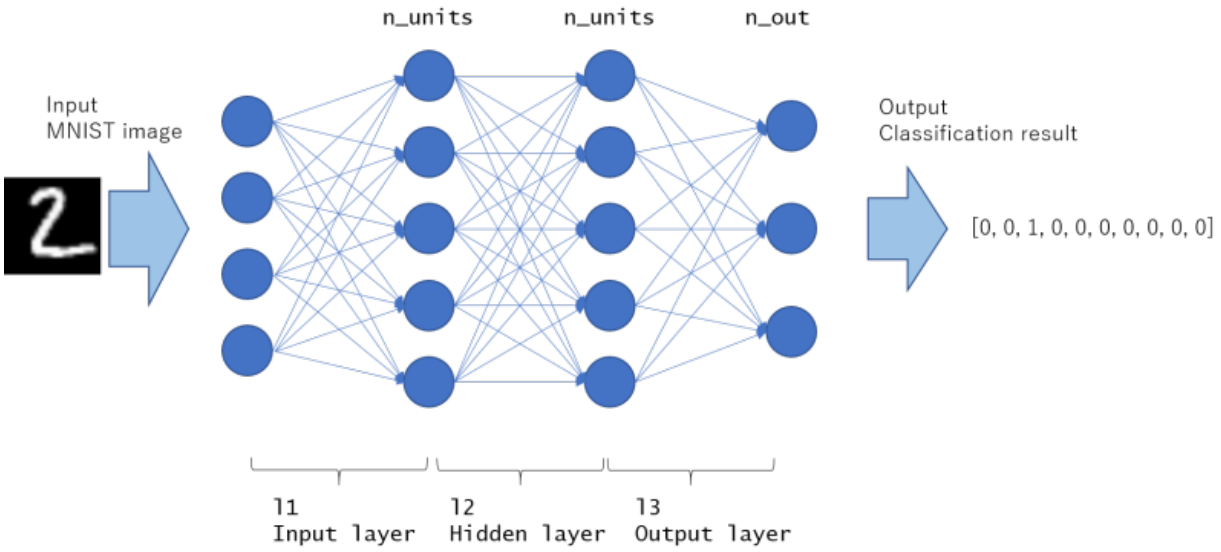
- Data-to-text report generation, which utilizes structured data, sourced from a relational data format, to fill out a text template [Goldberg et al., 1994, Reiter, 2007, Gatt et al., 2009].

2.2 Multi-Layer Perceptrons

Multi-layer perceptrons (MLP) are widely used in the neural network literature as the simplest vanilla artificial neural network. An MLP typically comprises one or more hidden layers that learn intermediate representations before producing an output layer. A simple MLP classification architecture for MNIST digits [LeCun et al., 2010, Deng, 2012] is depicted in Figure 2.1. The network is represented as a directed graph containing nodes and edges, where each node represents a neuron and each edge represents a learned weight. Each weight here is a model parameter. In this example, the weight between two successive layers are densely interconnected. Given a layer of m nodes and a subsequent layer of n nodes that are densely connected, we require $m * n$ edges (or weights) to connect the nodes of these two layers. These $m * n$ edges form a matrix representing the model parameters that transform a vector represented in the first layer to a vector represented in the second layer.

In terms of function approximation, a multilayer perceptron can be thought of as nested function calls on the input x. For example, assume that the intermediate layers approximate the functions f_1 and f_2, and the output layer approximates the function f_3 in the example presented in Figure 2.1. We can then say that the relation of the output distribution y with respect to the features in the input sample x would be described as $y = f_3(f_2(f_1(x)))$. The back-propagation of the objective loss over the weights of the network is achieved by using the chain-rule to differentiate this nested function. The gradients thus obtained are then used to update the weights, thereby changing how the output layer is computed during the next iteration of computation through the directed graph [LeCun et al., 1989].

The activation functions applied at each layer could either be linear or non-linear. The most popularly used non-linear activation functions are sigmoid, tanh, ReLU etc., which are typically used in the intermediate layers of the network. In the case of multi-label prediction problems, the output layer is typically a softmax distribution over the possible output classes. The softmax activation is simply the binary logistic regression classifier that generalizes to multiple classes, since the output of the softmax function can be used to represent a categorical distribution (Equation 2.1).



Source: http://corochann.com/mnist-training-with-multi-layer-perceptron-1149.html

Figure 2.1: Multi-Layer Perceptron

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}} \tag{2.1}$$

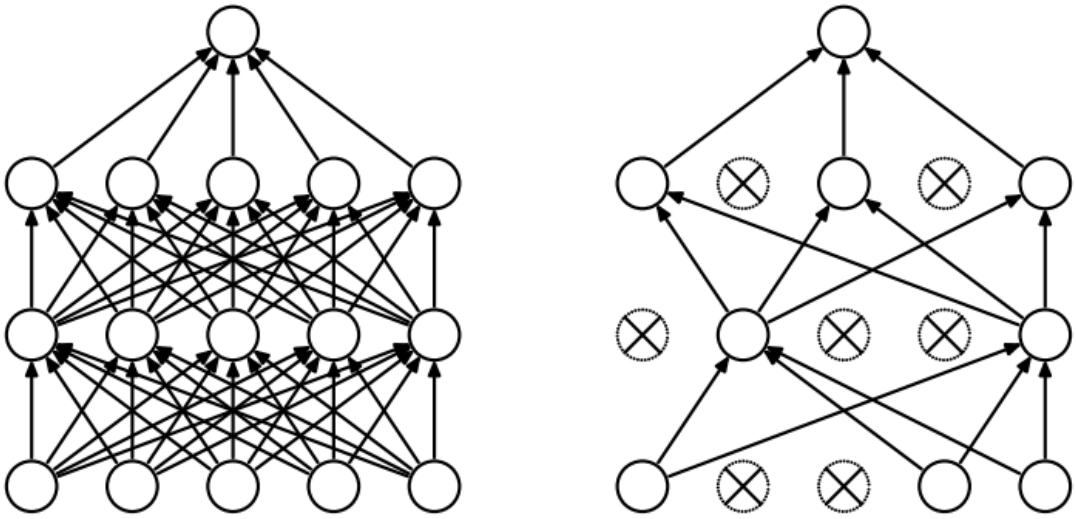
An MLP classifier is trained by minimizing a cross-entropy loss between the predicted softmax distribution y and the true distribution y'

$$\mathcal{H}_{y'}(y) := -\sum_{i}^{K} y'_i \log(y_i) \tag{2.2}$$

where K is the total number of classes.

2.2.1 Dropout

Dropout is a regularization technique used during the training of neural networks. It was proposed by Srivastava et al. [2014] as a simple method to prevent a model from overfitting the approximately learned function on the training data. The idea of dropout is to randomly ignore a set of neural network weights i.e. randomly set function parameters to 0 during each instance of the forward pass, as shown in Figure 2.2.



(a) Standard Neural Net (b) After applying dropout.

Source: Srivastava et al. [2014]

Figure 2.2: Dropout

Each neuron / unit is dropped out with a predefined probability, which can be viewed as a model hyperparameter. This allows for more generalized and robust functions being learned by minimizing the co-dependence of neurons on each other. Dropout has also been empirically shown to improve experiment results on several supervised learning tasks in vision, speech recognition, document classification and computational biology [Srivastava et al., 2014].

2.3 Convolutional Neural Networks

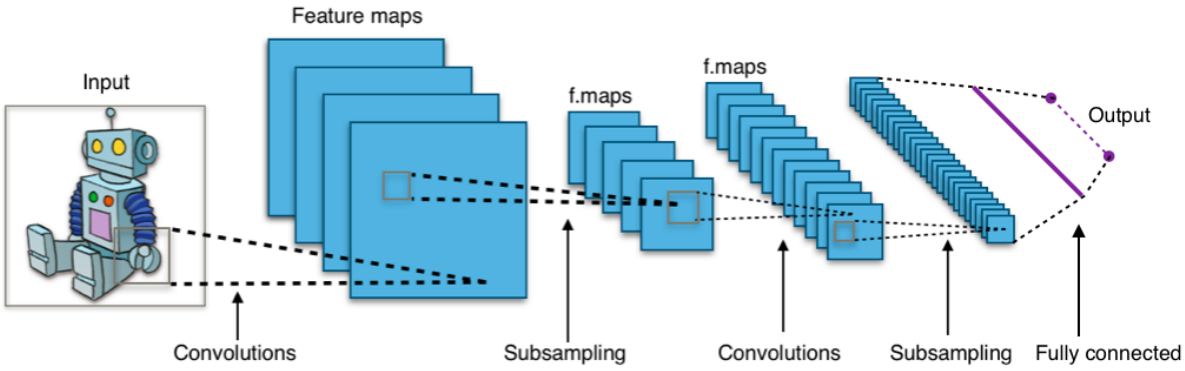
A convolutional neural network (CNN) [LeCun et al., 1995] is a specialized neural network that is optimized for feature extraction from local regions in data, for instance a specific object in an image, or a specific attribute in a body of text. Using densely connected multi-layer perceptrons for feature extraction from all the pixels of an image increases the number of parameters and slows down the training process. Instead, convolutional networks use shared weights over the source data that are called 'filters'.

The 'convolution' within a convolutional neural network is implemented using a filter. A

filter is a learned function that transfers an arbitrary spatial representation of the image, say 3px $*$ 3px, into a single value and this is done for all the similar spatial zones in the image. 2-dimensional convolutions are used to process features in images and video because of their inherent spatial nature. A 1-dimensional equivalent could be used to extract features from contiguous regions of text, usually 3 to 5 words [Kim, 2014]. These features are then sub-sampled to reduce the dimensionality for subsequent layers. This is typically achieved by max-pooling or average-pooling, which simply requires aggregating all the spatially proximal region activations produced by the convolutional layer.

The strategy of stacking convolutional and pooling layers allows the network to learn parameters that are translation invariant. In simpler words, an object can be identified regardless of its location in an image, and an attribute can be identified regardless of its position in a body of text.

An example CNN architecture is presented in Figure 2.3.



Source: https://commons.wikimedia.org/wiki/File%3ATypical_cnn.png

Figure 2.3: Convolutional Neural Network

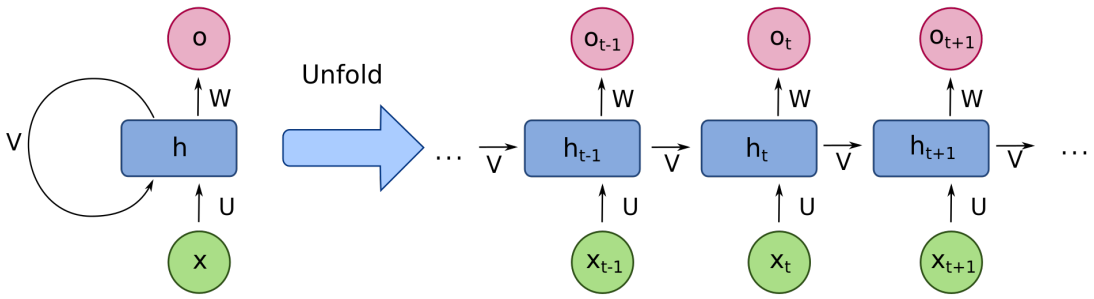
Since their inception, CNNs have been proven to achieve state-of-the-art results in multiple task settings within vision and language, especially with the use of deep convolutional architectures [Lawrence et al., 1997, Krizhevsky et al., 2012, Karpathy et al., 2014, Kalchbrenner et al., 2014, Kim, 2014, Hu et al., 2014].

2.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a sub-class of artificial neural networks that are commonly used for sequence processing. Its units forms a directed graph that operates on a sequence of inputs in a temporally-distributed manner. This makes them a useful tool for extracting features from arbitrary length sequences of input like audio or text. The optimization problem an RNN tries to solve is to predict the next element in a sequence given the historical context, as shown in Equation 2.3.

$$P(w_1, \cdots, w_T) = \prod_{i=1}^{T} P(w_i | w_1, \cdots, w_{i-1}) \tag{2.3}$$

The language model is built in such a way that the features extracted from the sequence at time-step t depend on the features observed during the time-steps $0 \cdots t-1$. A graphical depiction of a temporally-unrolled RNN is shown in Figure 2.4.



Source: https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

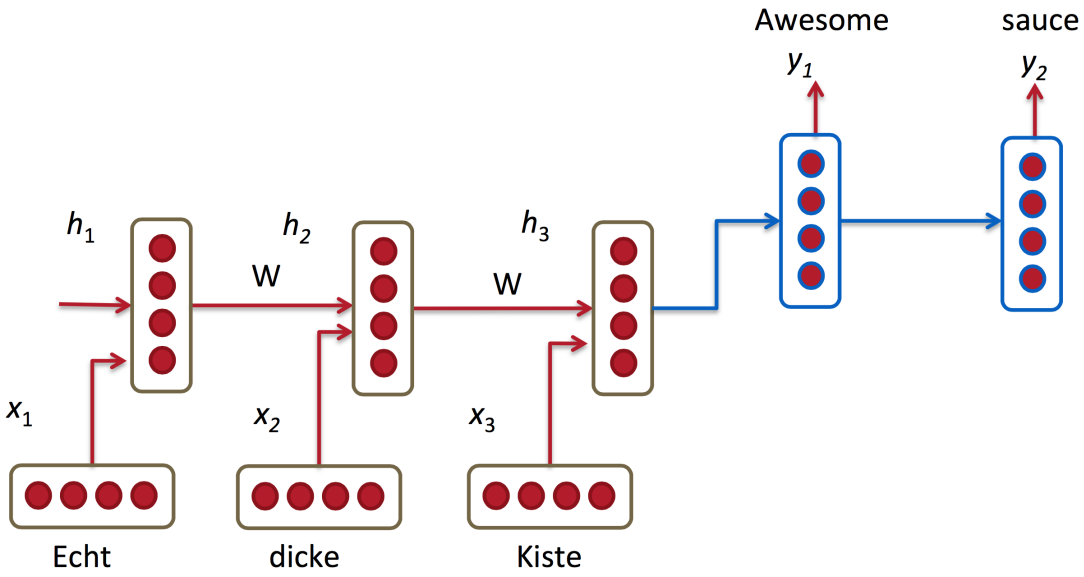
Figure 2.4: Recurrent Neural Network

The most recent and useful variants of recurrent units provide the ability to retain 'memory' of the context in which the current features are to be extracted. In the domain of language processing, this typically implies that the recurrent unit has a stored memory of the previously observed words in a sequence, and this property grants it the ability to learn context as part of the feature space. The prominent variants of recurrent units used in neural networks to extract features from sequences using a memory mechanism, are Long

Short-Term Memory (LSTM) units [Gers and Schmidhuber, 2001] and Gated Recurrent Units (GRU) [Chung et al., 2014].

Recurrent networks in the domain of natural language processing are used frequently for both natural language understanding (NLU) tasks, as well natural language generation (NLG) tasks [Bengio et al., 2003, Morin and Bengio, 2005, Mikolov et al., 2010]. Similar to the manner in which recurrent networks can extract features from arbitrarily long sequences of vectorized text, they can also be used to produce text sequences from a unit vector representation of text, as represented in Figure 2.5.

This is achieved by conditioning the generation of the first word of text either on some latent variable produced by a model, or by sampling from a generative distribution, and conditioning the generation of each subsequently predicted word on the word that was predicted in the previous time-step, until the model predicts an end-of-sentence (EOS) token.



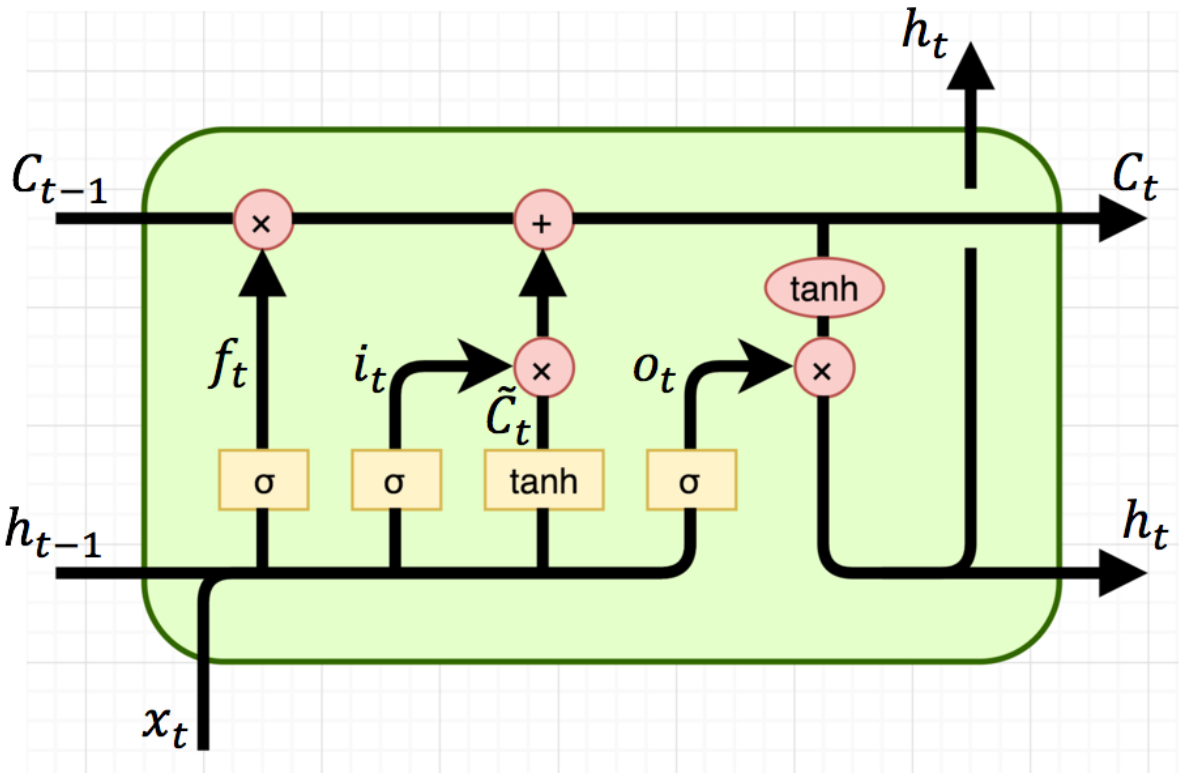
Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns

Figure 2.5: Recurrent Encoder-Decoder Model

2.4.1 Long Short-Term Memory

As described in the previous section, recurrent neural networks (RNN) learn representations of the data over temporal sequences, like text or audio features. However, in practice, RNNs don't seem to be able to assign priorities to which of the past data they choose to ascribe a higher importance to. This is detrimental to their usage in tasks that require processing of long sequences of data like in text, audio or video processing. This lack of ability to learn long-term dependencies in the input features is shown in previous work by Bengio et al. [1994].

Long Short-Term Memory (LSTM) units, as described first by Hochreiter and Schmidhuber [1997], seek to address this issue by explicitly modelling how much to retain and forget at each time-step during the RNN's training procedure.



Source: https://isaacchanghau.github.io/post/lstm-gru-formula

Figure 2.6: Long Short-Term Memory Unit

The structure of an LSTM is depicted in Figure 2.6. An LSTM is comprised of three distinct gating layers internally - namely the forget gate, input gate and output gate - that determine its outputs: the new cell state and hidden state. As can be seen in Figure 2.6, the cell state C_t passes through the LSTM mostly unperturbed, which is intended to address the problem of vanishing gradients [Hochreiter, 1998].

The forget gate uses a sigmoid activation to squash the values of the output of the previous time-step between 0 and 1, which can be understood as being semantically equivalent to deciding how much of the previous cell state to forget.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

The input gate also uses a sigmoid activation on the previous hidden state, which is multiplied by the candidate cell state $\hat{C}_t$. The value thus obtained is then multiplied by the forget-gated previous cell state to form the next cell state.

$$\begin{aligned} i_t &= \sigma(W_i * [h_{t-1}, x_t] + b_i) \\ \hat{C}_t &= \tanh(W_c * [h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * \hat{C}_t \end{aligned}$$

Now that we have obtained the cell state, we propagate that through to the next time-step. To obtain the hidden state, we use an output gate, again with a sigmoid activation, to gate the cell-state and create the hidden-state for the next time-step.

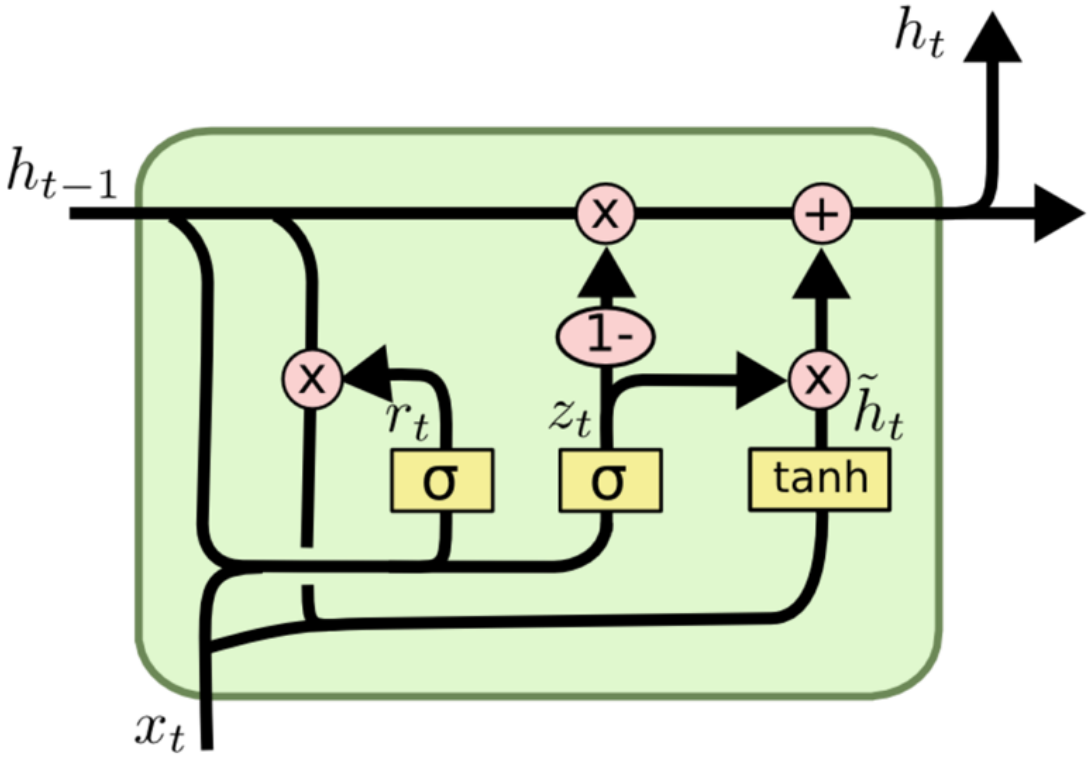
$$\begin{aligned} o_t &= \sigma(W_o * [h_{t-1}, x_t] + b_o) \\ h_t &= \tanh(C_t) \end{aligned}$$

The efficacy of LSTM-backed RNNs is evident from the prevalence of their usage in language modelling literature [Sundermeyer et al., 2012, Gers and Schmidhuber, 2001, Graves and Schmidhuber, 2005] [1].

2.4.2 Gated Recurrent Units

Gated Recurrent Units (GRU) [Cho et al., 2014b] are similar to LSTMs in that they possess gating mechanisms to assist with long-term dependency learning. They differ from LSTMs in a few aspects, like the merging of the input and forget gates into a single update gate. They also merge cell state and hidden state, emitting only a single output vector after the computations within the cell. Figure 2.7 depicts the internal GRU states and gates.

[1] https://karpathy.github.io/2015/05/21/rnn-effectiveness



Source: https://isaacchanghau.github.io/post/lstm-gru-formula

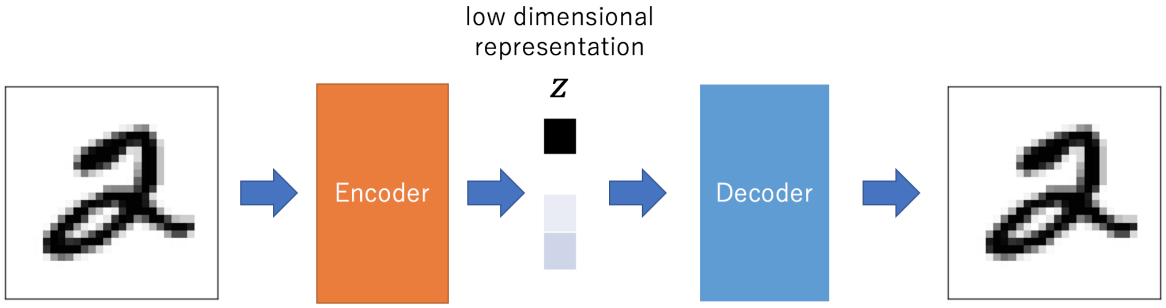
Figure 2.7: Gated Recurrent Unit

GRUs are simpler to work with in practice due to their having only a single output. They have been used interchangeably with LSTMs as memory units for recurrent neural networks [Chung et al., 2015, Fu et al., 2018].

2.5 Autoencoders

Autoencoders are models that are parameterized to convert arbitrary data into a latent representation (encoder), and recover the original data back from the latent representation (decoder). In this setup, the dimensionality of the latent representation is usually much smaller than that of the actual data. A simple autoencoder architecture is depicted in

Figure 2.8.



Source: http://mlexplained.com/2017/12/28/an-intuitive-explanation-of-variational-autoencoders-vaes-part-1

Figure 2.8: Model Architecture: Autoencoder

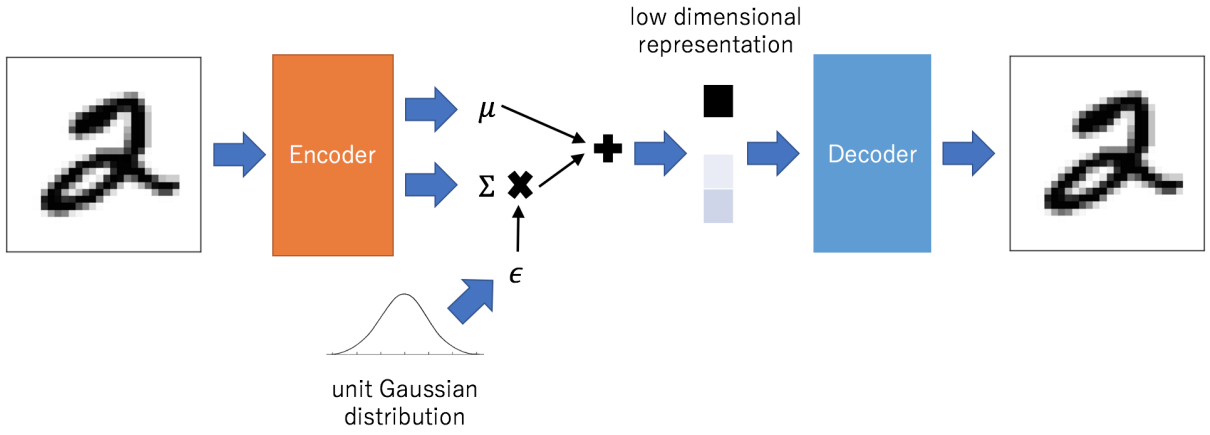
By training a model to reconstruct data that is funnelled through a lower dimensional representation, two objectives can be achieved simultaneously:

- The encoder weights of the model could be used to extract the most salient features of the data in a compressed representation, which is a better input feature format for downstream processing or learning algorithms. [Hinton and Salakhutdinov, 2006]
- The decoder weights of the model could be used as a generator. Given that we can sample from the distribution of the existing latent representations learned, or from a predefined prior (in a variational autoencoder), we can generate plausible novel data.

In the context of natural language, the encoder can be utilized as a sentence-encoding feature-extractor and the decoder can be utilized as a generative model. Autoencoders are also able to de-noise data given pairs of noisy and regular data, by learning a de-noising function, which can then be used for actual noisy data. These properties make autoencoders a good framework to utilize to implement sequence-to-sequence models, wherein both the encoder and decoder weights of the model are parameterized by neural networks that process and generate sequences of data, respectively.

2.5.1 Variational Autoencoders

Variational autoencoders (VAEs) [Kingma and Welling, 2013] are a probabilistic variant of autoencoders. The general structure of a VAE is the same as that of a deterministic autoencoder, with an encoder that learns a compressed latent space and a decoder that learns a function to map the latent space into the original data. In contrast with the deterministic autoencoder, a variational autoencoder parameterizes representations of the latent mean and variance as separate spaces. The method of variational inference requires placing conjugate priors on the mean and variance of the hitherto unknown latent representation. The encoder is then trained to learn the approximate posterior distribution, and the decoder is trained to generate novel data using samples taken from the prior distribution. The family of distributions used for this purpose is Gaussian. A simple VAE architecture is depicted in Figure 2.9.



Source: http://mlexplained.com/2017/12/28/an-intuitive-explanation-of-variational-autoencoders-vaes-part-1

Figure 2.9: Model Architecture: Variational Autoencoder

In addition to the reconstruction loss that a deterministic autoencoder uses, a metric of distance is used to evaluate how closely the learned representations of mean and variance resemble a Gaussian distribution with mean 0 and unit variance i.e. $\mathcal{N}(0, I)$. The co-variance matrix is treated as an identity matrix for the sake of simplicity. The Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951] is used to penalize learned representations that do not resemble the prior. However, we cannot compute the KL di-

vergence directly. Instead we compute an alternative objective that is equivalent to the KL divergence up until an added constant. This is called the evidence lower-bound (ELBO). The ELBO objective maximized in a variational autoencoder model is given in Equation 2.4:

$$ELBO(q) = \mathbb{E}_{q(z)}[(\log(p(x|z))] - \mathbb{KL}(q(z)||p(z)) \tag{2.4}$$

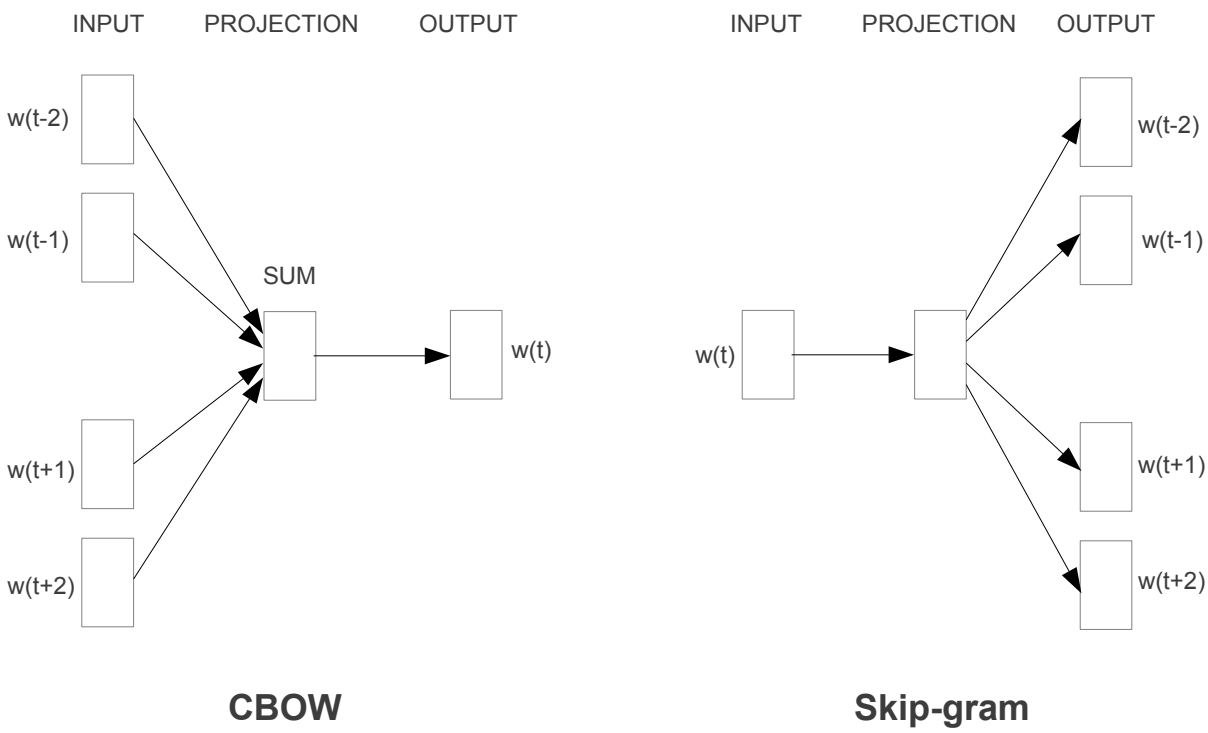
Simultaneously, for the decoding step, we sample from the prior, utilizing the re-parameterization trick to ensure that the model remains differentiable. As a result, the decoder is trained as a generative network that can randomly sample from the prior and generates plausible data that is similar to the input distribution. The decoder being able to act as a generative model independently is what sets a VAE apart from a deterministic autoencoder.

2.6 Word Embeddings

Prior to the usage of matrix factorization and neural models to learn continuous word representations, the numerical representations used for words and documents were one-hot representations, word-ngrams [Brown et al., 1992] or tf-idf weighted statistics. Word-ngrams are useful for expressing probability distributions of word occurrences in a corpus. Higher order ngrams (bigrams, trigrams, four-grams etc.) also take into account the context of nearby words, and can be used to construct language models.

However, these methods use discrete word or character occurrence counts to represent words and documents, and these representations scale with the size of the input corpus vocabulary for word-based models. This makes discrete word representation based language models computationally inefficient for large corpora with diverse vocabularies. Character-based models, on the other hand, are liable to produce invalid words and don't work as well with recurrent memory units as words do [Bojanowski et al., 2015]. These methods are also ineffective at modelling semantic inter-word relationships.

Word2Vec, which aims to alleviate these problems, is a vector space model that embeds words into a continuous space. Also, semantically similar words in Word2Vec are embedded in nearby spaces [Mikolov et al., 2013a,b, Le and Mikolov, 2014]. Word2Vec can model the corpus vocabulary in two different ways, namely CBOW and Skip-gram. The CBOW model predicts a target word from context words, whereas the Skip-gram model does the opposite and predicts target words from a single context word. Both model types are shown in Figure 2.10



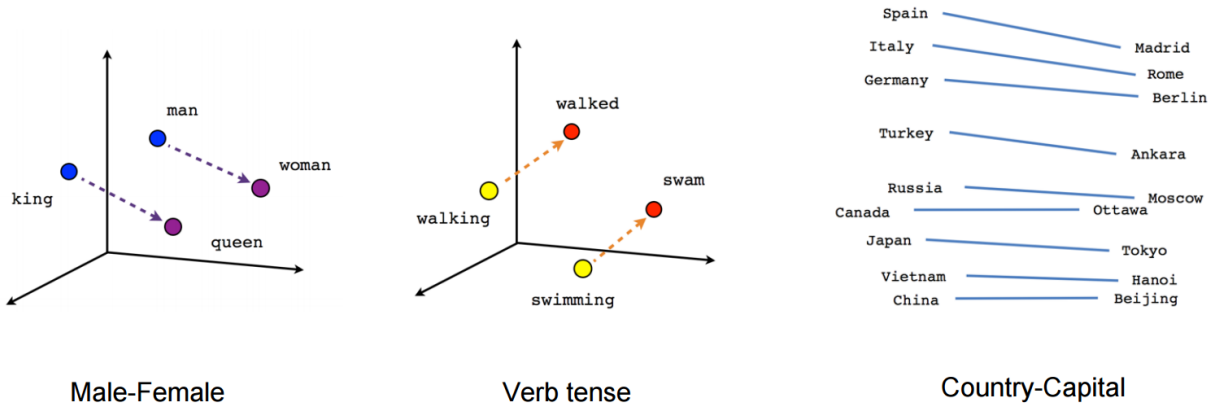
Source: Mikolov et al. [2013c]

Figure 2.10: Word2Vec Model

The Word2Vec model is a simple shallow neural network, with single input, hidden and output layers. The hidden layer is of arbitrary size, called the embedding size. The embedding size is usually much lower that the vocabulary size, which is reminiscent of an autoencoder that learns a compressed salient representation of the words. The one-hot encoded input is transformed into its context or target word, depending on whether the algorithm chosen is CBOW or Skip-gram. This transformation is parameterized by the shallow neural network. After the model converges, the network weights connecting the input and the hidden layer can be used as parameters to obtain dense, continuous representations of words.

A surprising outcome of this embedding model is that translations within the learned

vector space seem to emulate semantic (gender, tense) as well as factual relationships (country-capital), as shown in Figure 2.11.



Source: https://www.tensorflow.org/tutorials/word2vec

Figure 2.11: Word2Vec Relationships

Word embedding neural models are also shown to be implicitly factorizing a word-context matrix by subsequent work in this area [Levy and Goldberg, 2014].

2.7 Sequence-to-Sequence Modelling

Also abbreviated in the literature as Seq2Seq, this is a class of models that learns functions to transform one sequence into another. First introduced by Sutskever et al. [2014], Seq2Seq is a flexible framework for modelling transformations made to arbitrary length sequences, typically using a neural encoder-decoder framework. In the field of natural language processing, the main tasks that benefit from such an encoder-decoder framework are those for which there exists two distinct distributions of text, and the model is trained to learn the mapping from one to the other. Similarly it can also be used for sequence autoencoding tasks, by learning to reconstruct sequences.

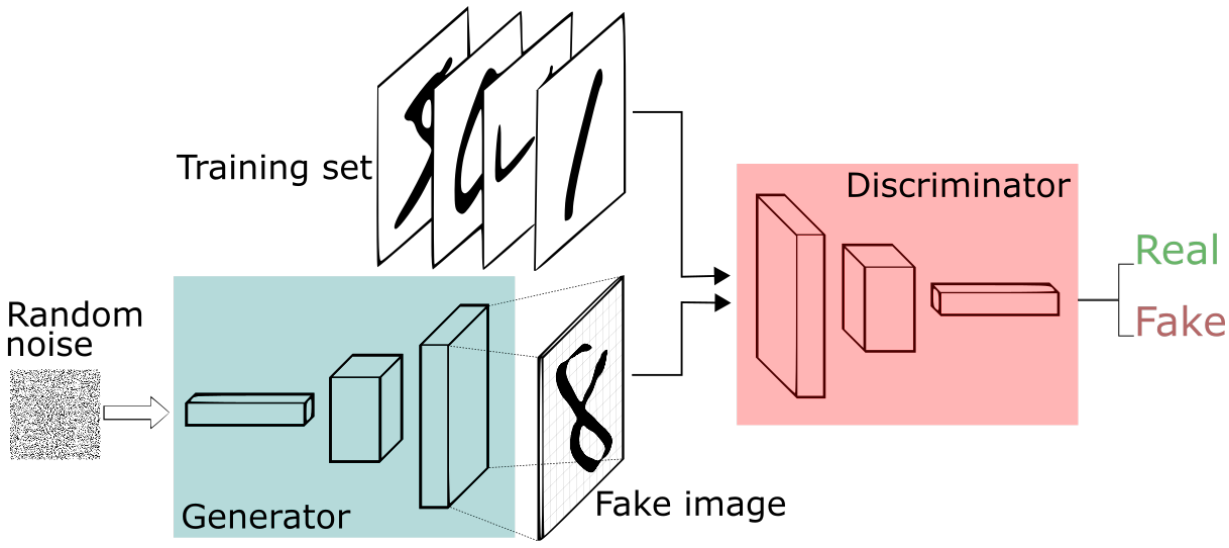
Sutskever et al. [2014] define a sequence-to-sequence model that learns a function (Equation 2.5) to map a sequence of inputs $x_1, \cdots, x_T$ to a sequence of outputs $y_1, \cdots, y_{T'}$, where the initial state h is set to the hidden LSTM representation of $x_1, \cdots, x_T$.

$$p(y_1, \cdots, y_{T'}|x_1, \cdots, x_T) = \prod_{t=1}^{T} p(y_t|h, y_1, \cdots, y_{t-1}) \tag{2.5}$$

This makes the Seq2Seq model an ideal framework using which one can implement solutions to several natural language generation tasks like neural machine translation, dialogue generation, text summarization, etc.

2.8 Generative Adversarial Networks

Goodfellow et al. [2014] presented a novel generative model that utilizes the idea of a game-theoretic competition between a generative model which is a decoder, and a discriminative model which is a classifier. The general idea is to train the discriminative model, also known as the adversary, to be able to discriminate between the synthetic distribution that the generator model produces, and a true distribution that the generator trying to mimic. Figure 2.12 depicts a sample GAN model architecture.



Source: https://deeplearning4j.org/generative-adversarial-network

Figure 2.12: Model Architecture: Generative Adversarial Networks

In the original formulation of adversarial training, which involves using a binary classifier, the generator learns weights to map a randomly initialized latent variable z to a data

distribution x that it is unaware of. The discriminator is trained using the true distribution and samples generated by the generative model, with the objective to discriminate the true samples from the generated samples. The generator on the other hand, attempts to maximize the discriminator's classification error without directly modifying the discriminator's parameters, thereby producing plausible samples that mimic the true distribution. In this manner, both the generator and discriminator are simultaneously trained to both produce better synthetic samples, as well as differentiate between real and synthetic samples respectively. This min-max game is represented by Equation 2.6, where D is the discriminator, G is the generator and V is the value-function or loss-function of the generative model.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.6}$$

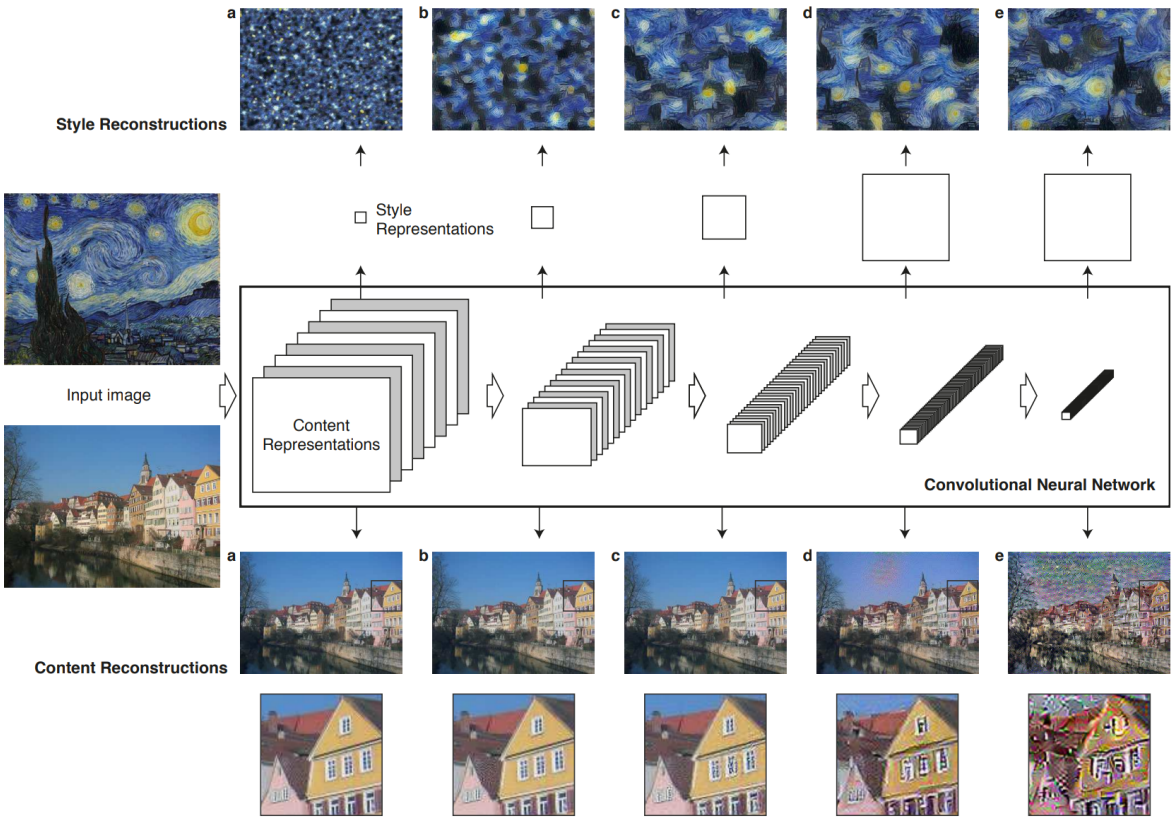
This property the generator model's parameters possess - of translating random noise into plausible data samples - is used in areas of research within computer vision and natural language processing to produce novel data samples. Also, adversarial learning as a principle is an effective tool to ensure that a feature space is devoid of any arbitrarily chosen set of known attributes. We use this property of adversarial learning in our model to assist with the disentanglement of stylistic attributes.

2.9 Visual Style Transfer

The idea of neural style transfer was originally proposed by Gatys et al. [2016]. In the task described in the paper, the authors use two distinct images as input. The first image contributes the content, and the second contributes the style. The objective is to generate a final image that contains all of the physical objects visible in the first image, but depicted in the style (hue, brush strokes, texture, etc.) visible in the second.

The features, as with most of the state-of-the-art image processing techniques, are extracted from the images using Convolutional Neural Networks (CNNs) [LeCun et al., 1998], the deep neural network variants of which have been widely successful at image recognition tasks such as object recognition in the ImageNet dataset [Krizhevsky et al., 2012].

As depicted in Figure 2.13, subsequent layers of the CNN learn incrementally higher level features in both the style image and the content image. These learned features for the style and content images are stored for future use. To generate the final image, the authors begin with a random white noise image. This image is passed through both networks, the parameters of which extract both its style features and its content features.



Source: Gatys et al. [2016]

Figure 2.13: Artistic Style Transfer in Images

This model uses two training objectives:

- Minimize the the element-wise mean squared difference between the style features computed for the style image and the white-noise image.
- Minimize the the mean squared difference between the content features of the content image and the white noise image.

The overall training loss is a linear combination of the above objectives, and is used to iteratively modify the white-noise image until it has the style features of the style image and the content features of the content image.

In this chapter, we discussed the background needed to understand the methods used in our research and other related work. The next chapter builds on the fundamentals established here, and discusses previous attempts at neural disentanglement in the context of linguistic style transfer, as well as methods that do not use disentanglement as the basis of their style transfer model.

Chapter 3

Related Work

In the past year, the works by Hu et al. [2017] and Ficler and Goldberg [2017] expound the applicability of 'attribute-conditioned' generation to linguistic style transfer tasks. Both of these methods - as opposed to the historically used paraphrasing techniques [Xu et al., 2012] - utilize neural network models [LeCun et al., 2015] to parameterize the style transfer function.

Further work presented by Shen et al. [2017] and Fu et al. [2018] propose the idea of removing attribute information from latent representations using adversarial learning. In this section we discuss these works in detail, along with mentions of the other prominent work done in this area.

3.1 Disentangling Factors of Variation

Mathieu et al. [2016] propose the idea of a generative model that learns to separate the factors of variation by splitting the latent space into two codes: one that is relevant to solving a specialized task, and one that is not.

They use a variational autoencoder to learn the generative model, and utilize adversarial training to disentangle the latent space of specific attributes. In their evaluations, they use image datasets like MNIST [LeCun et al., 2004, 2010], Sprites [Reed et al., 2015] and Extended-YaleB [Georghiades et al., 2001].

Handwriting style, body type, hair, position, illumination, etc. are some of the attributes chosen to disentangle from the latent space. This disentanglement is done by

training a discriminator, following Goodfellow et al. [2014], to provide a negative training signal to the generative model if it is able to successfully discriminate training samples based on the given attribute. Once the model is converged and is capable of producing representations devoid of the chosen style or attribute, it can be used to generate new instances of data. Using known representations, and changing the controlled attribute, novel samples can be produced.

3.2 Information Maximizing Generative Adversarial Nets

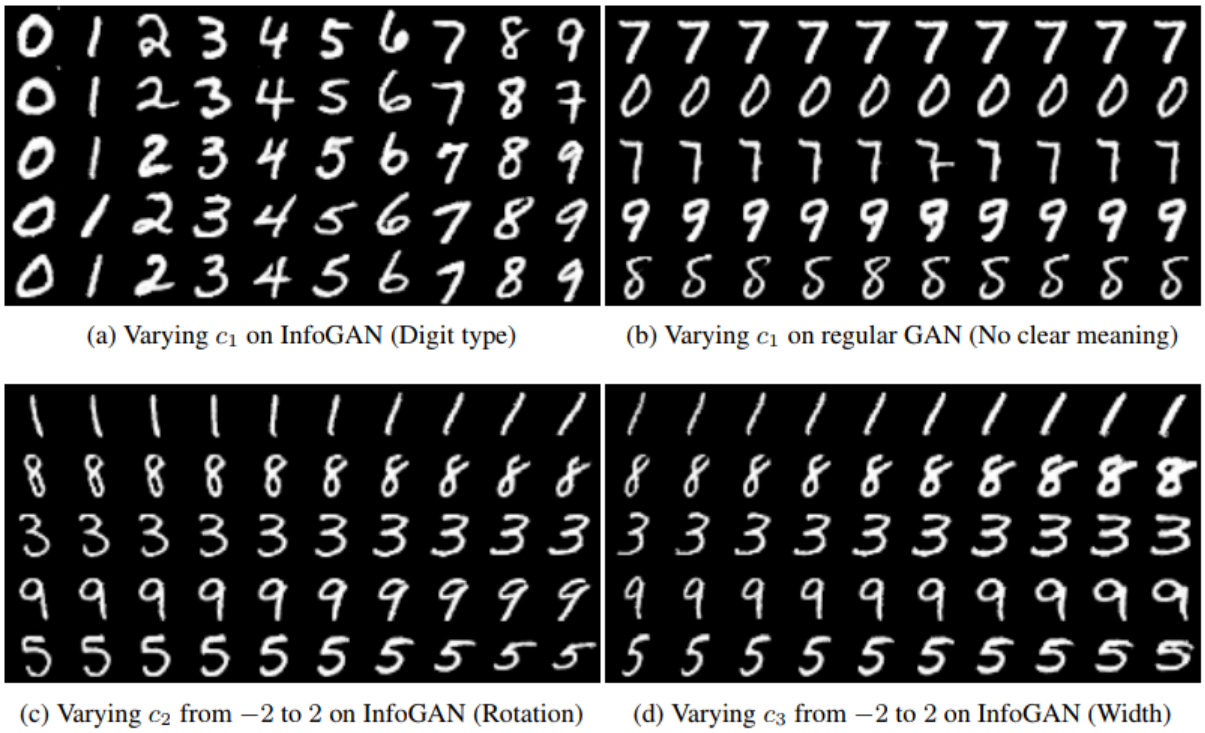
InfoGAN [Chen et al., 2016] is another work that attempts to increase the interpretability of latent representations by utilizing regularizations on the latent space. They propose doing this by maximizing the mutual information between the generated distribution and a subset of variables in the latent space.

The authors present a modification to the GAN objective [Goodfellow et al., 2014], to encourage the learning of interpretable representations. This is done by maximizing the mutual information between a fixed small subset of the GAN's noise variables and the observations. This approach attempts to learn a disentangled latent representation in an unsupervised manner. In addition to the regular noise variables that a GAN generates novel samples from, the authors use additional categorical and continuous variables under certain sampling constraints. For example, on the MNIST dataset [LeCun et al., 2010], the authors use a categorical latent variable of size 10, with the probability of sampling each of those variables discretely as 0.1. These categorical variables are used to represent the 10 distinct digits (labels) present in the MNIST dataset. In addition, they sample two continuous variables sampled from the uniform distribution $\mathcal{U}(-1, 1)$, intended to model the angle and shape of the handwriting style.

In the vanilla formulation of the GAN, the decoder might well chose to ignore these additional latent variables by minimizing the magnitude of the parameters learned between these variables and generated samples. To avoid this, the authors propose the usage of an information maximization objective as an auxiliary training objective. This can be viewed as a regularization intended to force the usage of these additional variables during sample generation.

Using this approach, the authors are able to generate MNIST digits in various handwriting styles by changing the continuous variables, and alter the numerical value represented in the generated image by changing the categorical variable, as shown in Figure 3.1. Here c_1 represents the categorical variable discussed above and c_2, c_3 represent continuous vari-

ables. They also test this method on faces [Liu et al., 2015, Paysan et al., 2009] and chairs [Aubry et al., 2014].



(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)

Source: Chen et al. [2016]

Figure 3.1: InfoGAN: Tweaking Latent Variables

However, this method does not provide a way of explicitly controlling what attribute is represented in each additional latent variable. Samples need to be generated first to understand which features have been represented in the latent variables.

3.3 Controlled Generation of Text

Hu et al. [2017] use a variational autoencoder trained with the reconstruction objective and a KL divergence minimization objective on the latent space with respect to a prior $p(z)$, as described in the original variational auto-encoding paper by Kingma and Welling

[2013]. In addition to the reconstruction objective, the authors use additional discriminative training signals to adapt the desired attributes of the generated text. These training signals are propagated back to the encoder.

Consider a source corpus x and an encoder that is parameterized to generate a latent code z. z is a variational latent space that resembles a Gaussian prior. (This is enforced by a KL divergence loss). The structured code c is a known label of the text (discrete or continuous). The decoder / generator produces the output corpus $\hat{x}$ conditioned on $[z; c]$. It uses greedy decoding, which predicts the word with maximal probability at each step [Langlais et al., 2007].

A classifier or regressor discriminator predicts the structured code c of the generated output $\hat{x}$ to ensure that it is the same as the one the generator was conditioned on i.e. $G(z, c)$. This discriminator is pre-trained on a labelled corpus.

Each decoder step used to predict $\hat{x}$ is a softmax distribution over the corpus vocabulary, scaled by a temperature τ. Higher temperatures flatten the softmax distribution for each word prediction, thereby increasing word diversity. Conversely, setting $\tau = 0$ will resemble a discrete probability distribution over the corpus vocabulary. For their experiments, the authors gradually anneal $\tau \to 0$

The authors describe three objectives that train their model, including the reconstruction objective of a variational autoencoder. The auxiliary objectives are discrete label prediction and content vector prediction by two discriminators. The overall architecture is depicted in Figure 3.2.

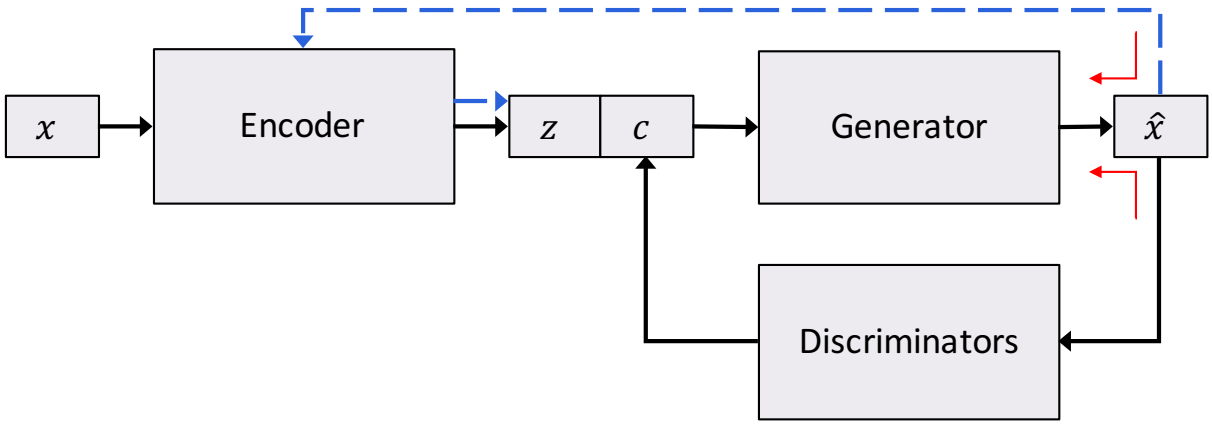
3.3.1 Variational Autoencoder

A reconstruction loss that ensures that the generated sentence $\hat{x}$ is the same as the original sentence x. This is equivalent to minimizing the negative log-likelihood of generating $\hat{x}$. This is shown in Equation 3.1

$$\begin{aligned} \mathcal{L}_{VAE}(\theta_G, \theta_E; x) = &- \mathbb{E}_{q_E(z|x)q_D(c|x)}[\log p_G(x|z, c)] \\ &+ KL(q_E(z|x)||p(z)) \end{aligned} \tag{3.1}$$

3.3.2 Style Discriminator

A discriminator validates if the predicted label for $\hat{x}$ is the same as the corresponding label for x. This is a cross-entropy loss over the probability distribution of the labels. This



Source: Hu et al. [2017]

Figure 3.2: Model Architecture: Toward Controlled Generation of Text

discriminator loss can be further subdivided into two terms, one that maximizes the log likelihood of predicting the correct class as in Equation 3.2, and one that minimizes the empirically observed Shannon entropy of the predicted distribution, thereby incentivizing confident predictions, as in Equation 3.3.

$$\mathcal{L}_s(\theta_D) = -\mathbb{E}_{X_L}[\log q_D(c_L|x_L)] \tag{3.2}$$

$$\mathcal{L}_u(\theta_D) = -\mathbb{E}_{p_G(\hat{x}|z,c)p(z)p(c)}[\log q_D(c|\hat{x}) + \beta\mathcal{H}(q_D(c'|\hat{x}))] \tag{3.3}$$

The expected log likelihood of predicting the correct distribution of the structured code c given the labelled examples X_L, is maximized. This happens before the generator model training.

During the training phase for the generative model, the expected log likelihood of predicting the correct distribution of the structured code c given the generated sentences $\hat{x}$ is also maximized. At the same time, the empirically observed Shannon entropy of the observed discriminator prediction $q_D(c'|\hat{x})$ is minimized. This entropy minimization objective reduces uncertainty and increases confidence of the structured code prediction. A wake-sleep algorithm [Hinton et al., 1995] is used to alternatively train the generator and discriminator.

3.3.3 Content Discriminator

The encoder from the loss equation 3.1, is used to regenerate the latent distribution z devoid of the structured code from the output distribution $\hat{x}$. The authors call this an 'independence constraint'. Regardless of the structured code c that is currently present in either x or $\hat{x}$, processing either through the generator should produce a consistent z. This allows the encoder to encode only latent factors that are independent of the structured code. This is shown in Equation 3.4.

$$\mathcal{L}_{attr,z}(\theta_G) = -\mathbb{E}_{p(z)p(c)}[\log q_E(z|\tilde{G}_\tau(z,c))] \tag{3.4}$$

The model was evaluated on short sentences with length < 15 words from the Stanford Sentiment Treebank [Socher et al., 2013]. The encoder-decoder setup is implemented using single layer LSTMs and the discriminator is implemented using a convolutional neural network (CNN) [Hu et al., 2016]. The KL divergence loss weight is annealed from 0 to 1 during training. Style transfer strength is evaluated using a pre-trained classifier. However, they do not evaluate language quality and content preservation using a quantitative metric.

3.4 Aligned and Cross-Aligned Autoencoders

Shen et al. [2017] also aim to perform style transfer in language using non-parallel corpora by separating content from style. They re-align the latent spaces to perform three tasks: sentiment modification, decipherment of word-substitution ciphers, and recovery of word order. Their method involves learning an encoder that takes a sentence and maps it to a content representation devoid of style. The decoding is then conditioned on the concatenation of the encoded content representation and the target style label.

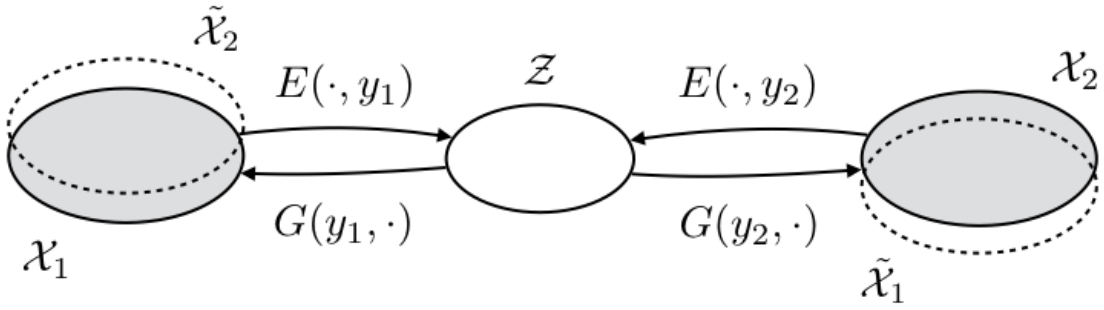
There are two non-parallel corpora $X_1 = x_1^{(1)} \cdots x_1^{(n)}$, which is drawn from the prior $p(x_1|y_1)$, and $X_2 = x_2^{(1)} \cdots x_2^{(n)}$, which is drawn from the prior $p(x_2|y_2)$. The objective is to estimate the style transferred distributions $p(x_1|x_2; y_1, y_2)$ and $p(x_2|x_1; y_1, y_2)$. Here, y_1 and y_2 are class labels.

The authors state a constraint that x_1 and x_2's marginal distributions can only be recovered if for any different styles $y, y' \in Y$, distributions $p(x|y)$ and $p(x|y')$ are different. This is a fair assumption to make because if $p(x|y) = p(x|y')$, then the style changes would be indiscernible to a model.

They also prove that if the content z is sampled from a centred isotropic distribution, the styles cannot be recovered from x, but in the case of z being a more complex distribution

like a Gaussian mixture, then the affine transformation that converts $[y; z]$ into x can be recovered.

A simplified graphical depiction of their model is shown in Figure 3.3.



Source: Shen et al. [2017]

Figure 3.3: Model: Style Transfer from Non-Parallel Text by Cross-Alignment

The base model used is a variational autoencoder. The reconstruction loss is shown in Equation 3.5.

$$\begin{aligned}\mathcal{L}_{rec}(\theta_E, \theta_G) =& \mathbb{E}_{x_1 \sim X_1}[-\log p_G(x_1|y_1, E(x_1, y_1))] + \\ & \mathbb{E}_{x_2 \sim X_2}[-\log p_G(x_2|y_2, E(x_2, y_2))]\end{aligned} \tag{3.5}$$

3.4.1 Aligned Autoencoder

The authors propose aligning the distributions $P_E(z|x_1)$ and $P_E(z|x_2)$ where E is the encoder function. This is done by training an adversarial discriminator to distinguish between the two distributions.

The adversarial objective is shown in Equation 3.6 where $D(\cdot)$ outputs 0 if it predicts the source distribution to be X_1 and 1 if it predicts the source distribution to be X_2.

$$\begin{aligned}\mathcal{L}_{adv}(\theta_E, \theta_D) =& \mathbb{E}_{x_1 \sim X_1}[-\log D(E(x_1, y_1))] + \\ & \mathbb{E}_{x_2 \sim X_2}[-\log(1 - D(E(x_2, y_2)))]\end{aligned} \tag{3.6}$$

The overall optimization objective combining equations 3.5 and 3.6 can be written as

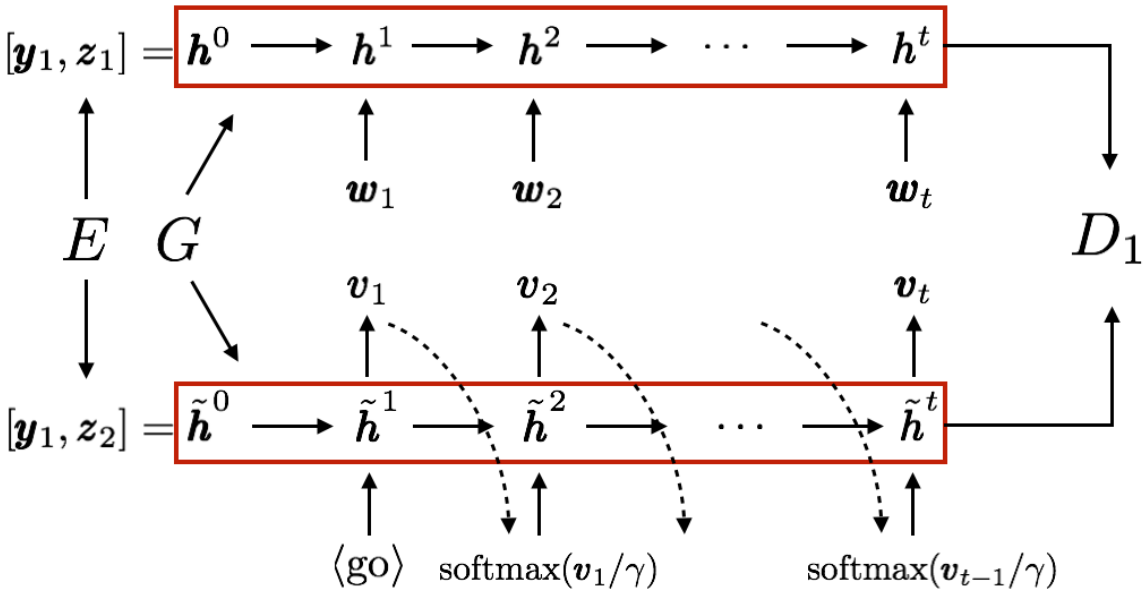
$$\mathcal{L} = \min_{E,G} \max_{D} \mathcal{L}_{rec} - \lambda \mathcal{L}_{adv} \tag{3.7}$$

where λ is a balancing weight for the adversarial term.

3.4.2 Cross-aligned Autoencoder

This is similar to the aligned autoencoder approach, but instead of trying to align $P_E(z|x_1)$ and $P_E(z|x_2)$ using an adversarial discriminator, two distinct adversarial discriminators are used to align a sequence of real and transferred generator hidden states. i.e. D_1 is used to align the distributions $G(y_1, z_1)$ and $G(y_1, z_2)$. Similarly, D_2 is used to align the distributions $G(y_2, z_2)$ and $G(y_2, z_1)$. These discriminators are trained with the objective of identifying the underlying content distributions $P(z_1)$ and $P(z_2)$ from the generated sequences. Professor-forcing [Lamb et al., 2016] is used to train both of these discriminators. Professor forcing uses a discriminator to distinguish if the decoder hidden states are a result of training-time teacher-forcing or inference-time scheduled sampling.

This model is a generalized version of one that simply uses the final encoder state for adversarial discrimination, as was the case in the Aligned Autoencoder approach, and is depicted in Figure 3.4.



Source: Shen et al. [2017]

Figure 3.4: Cross-Aligning Decoder

The overall optimization objective combining the autoencoding loss in equation 3.5 and the two independent discriminators, the losses of which are given in equation 3.6, can be written as:

$$\mathcal{L} = \min_{E,G} \max_{D} \mathcal{L}_{rec} - \lambda(\mathcal{L}_{adv_1} + \mathcal{L}_{adv_2}) \tag{3.8}$$

As opposed to the simple feed-forward classifier used in the aligned autoencoder, the cross-aligned autoencoder uses convolutional nets for text classification [Kim, 2014]. They use Yelp reviews as the dataset with rating > 3 as positive and rating < 3 as negative examples. Reviews with a sentence count > 10 and sentences with a word count > 10 are filtered out. The vocabulary size used is 10000. Style transfer is evaluated using a pre-trained classifier, whereas language fluency and content preservation are qualitatively evaluated using human assessments.

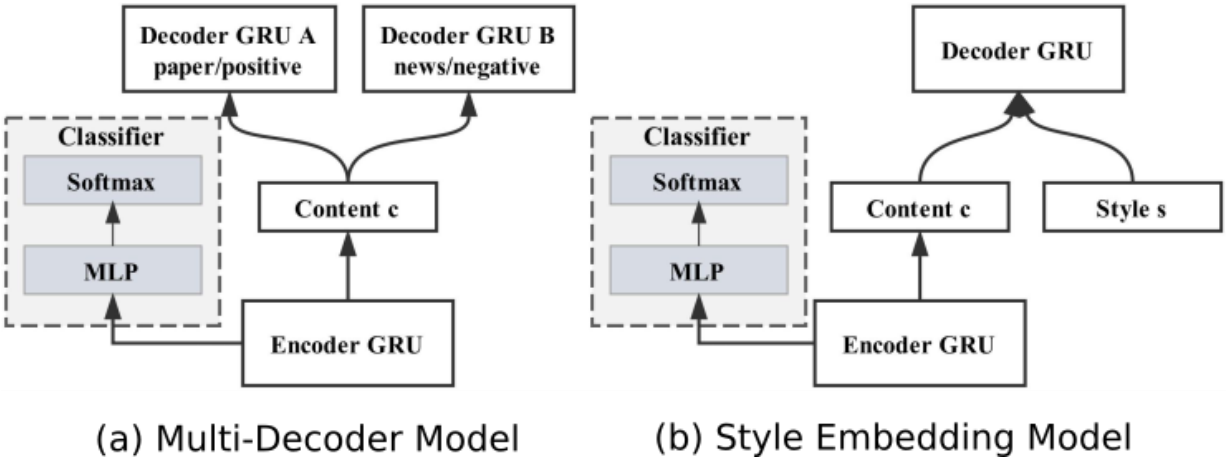
3.5 Style Embedding Autoencoder

Fu et al. [2018] employ two distinct models to perform style transfer

- A **multi-decoder model** that uses distinct decoders to produce text in different styles. This implicitly means that the number of decoders that need to be trained scales with the number of distinct label values to train. The architecture for this model is depicted in Figure 3.5 (a).
- A **style-embedding model** with a single decoder that generate text in different styles. This method utilizes a style embedding matrix, of size $n * m$ where n is the number of distinct label values and m is an arbitrarily chosen embedding size. This matrix is tasked with encoding all of the stylistic content of the encoded sentences. The architecture for this model is depicted in Figure 3.5 (b).

In contrast with the previous models described, this work uses a deterministic autoencoder in lieu of a variational autoencoder. For our purposes, we focus on the style-embedding model, since it requires explicit disentanglement and uses fewer model parameters per added attribute, compared to the multi-decoder model.

The training objectives used for this model include the reconstruction objective for the autoencoder, the style label discriminative objective for a classifier on the latent space, and the adversarial loss propagated back to the autoencoder from the representation learned in the content space. The autoencoder is trained to produce only the salient features that



Source: Fu et al. [2018]

Figure 3.5: Model Architecture: Style Transfer in Text - Exploration and Evaluation

represent the content of the sentences i.e. the content space, and the style embedding matrix is trained via back-propagation of the reconstruction loss.

The reconstruction loss is the same as that of a standard deterministic autoencoder.

$$\mathcal{L}_{seq2seq} = -\sum_{i=1}^{M} \log P(\hat{x}_i | x_i; \theta_e; \theta_d) \tag{3.9}$$

where M is the number of training examples and θ_e and θ_d are the encoder and decoder parameters respectively.

A discriminative classifier is trained on the latent content space learned by the autoencoder, to distinguish between the different possible styles. The following objective is minimized

$$\mathcal{L}_{disc} = -\sum_{i=1}^{M} \log P(l_i | Encoder(x_i; \theta_e); \theta_c) \tag{3.10}$$

where M is the number of training examples and θ_e and θ_c are the encoder and discriminative classifier parameters respectively.

An adversarial objective is applied to the content representation. This objective aims to maximize entropy of the predicted label distribution from the content representation by

minimizing the following objective:

$$\mathcal{L}_{adv} = -\sum_{i=1}^{M}\sum_{j=1}^{N}\mathcal{H}(P(j|Encoder(x_i;\theta_e);\theta_c)) \tag{3.11}$$

where M is the size of the training data and N is the number of distinct styles. Using this adversarial entropy objective, the encoder is penalized if the adversarial discriminator predicts style with a high confidence (low entropy), and is therefore trained to produce a latent content representation which is independent of style.

Similar to the persona-based neural conversation model [Li et al., 2016a], a style embedding is learned for each different style. The conditional generation (decoding) is done using a GRU recurrent neural network. The input to the decoder is a concatenated vector of the hidden state of the encoder and the embedding for the style to apply. The style embeddings matrix is not directly parameterized by the encoder, but the learning algorithm updates its parameters based on how well it combines with the content representation to reconstruct the original text.

The transfer strength is evaluated using a simple LSTM classifier. Content preservation is evaluated by computing the cosine similarity between sentence embeddings of the original and the generated text. We use their content preservation metric in our own work. This is discussed further in Section 5.3.

Several other works have also been published in this domain recently including: using cyclical translation to strip style as a pre-processing step, followed by a multi-decoder model to generate style-transferred sentences [Prabhumoye et al., 2018], conditioned RNN decoding with a set of tunable attributes [Ficler and Goldberg, 2017], using maximum-mean discrepancy metrics to separate common and discriminative factors [Larsson et al., 2017], etc.

In this chapter we discussed the important related works including ones that we directly compare our model's performance against. In the next chapter we describe our approach to learn attribute disentangled representations and provide details of the objectives and regularization methods used in our model.

Chapter 4

Approach

4.1 Challenges

4.1.1 Non-Interpretable Latent Representations

The randomness and inscrutability of the learning processes in neural networks has sparked interest in research pertaining to learning interpretable latent representations [Chen et al., 2016]. Neural networks have proven to be highly capable function approximators in the general machine intelligence literature over the past couple of decades. However, without any explicit additional regularization over the latent variables, they are learned in an arbitrary manner to optimize over a user-defined convex loss function.

Due to this non-explainability of the weights of neural networks, they have been used as a black-box in practice. However, the non-explainability hampers their adoption in use-cases requiring explicit audits for accountability, and for humans to reason about why a system chose a particular outcome over the other options available. We believe that the disentanglement of the learned weights of a neural network is an important step in the direction toward addressing this shortcoming. This is achieved by using available labels to force representation learning to adhere to a predefined separation scheme in the latent space. To avoid the burden of requiring manually labelled data, we can use weakly supervised data. For example, product or service review ratings can be used as indicators of sentiment in text.

4.1.2 Quantitative Evaluation of Language Quality

The evaluation of novel generated text to measure its perceived quality, naturalness, fluency and compliance with predefined attributes, like sentiment, tense etc. is particularly important in tasks like ours where having parallel corpus as a gold set is not possible.

For the presence of certain attributes like sentiment, tense, formality, etc., it is fairly simple to devise a classification model that is trained on a large corpora of samples labelled as such. However, it is much more difficult to unambiguously state what makes a body of text more fluent or natural than another with mathematical rigour using quantitative metrics. Unsurprisingly, contemporary works in this area use human annotators [Hu et al., 2017, Shen et al., 2017, Fu et al., 2018] to gauge model success. However, this is not scalable to large sets of generated sentences, and researchers need to account for subjectivity and inter-annotator agreement to further normalize scores obtained from human assessments [Bermingham and Smeaton, 2009, Nowak and Rüger, 2010].

To offset this problem, while still avoiding the need for human annotators to evaluate the target text, we have used metrics like sentence embedding similarity as proposed by Fu et al. [2018]. In this method, we compute sentence embeddings, augmented by the removal of words that are highly correlated with any specific label, and use it to assess content preservation. In a similar vein, we also report results using a word overlap metric, which can be used interchangeably with content preservation. We also use a Kneser-Ney smoothed [Kneser and Ney, 1995] language model to evaluate fluency. All the quantitative metrics we use are described in greater detail in Section 5.3.

4.2 Model Overview

We provide a novel approach to solve this open problem of non-interpretable latent representations in neural networks. We juxtapose this approach and its experimental results against the current state-of-the-art models. We empirically demonstrate the separation of content and style spaces by visualizing the inferred spaces of labelled sentences in sentiment-transfer tasks, where the positive sentences in the test set are converted to equivalent sentences with a negative sentiment, and vice-versa. This model is also generalizable to problems involving multiple classes, like style transfer by authors, artists, topic, etc. using its standard implementation.

Our model is built upon an autoencoding neural network with a sequence-to-sequence (Seq2Seq) reconstruction objective [Sutskever et al., 2014], as described in Section 4.3.

This is our base model, the objective function of which is to simply reconstruct the input text. We implement two variants of our model, using both a deterministic autoencoder [Baldi, 2012], as well as a variational autoencoder [Kingma and Welling, 2013].

Then, we introduce two auxiliary losses, a multi-task label classification loss and a style discriminative adversarial loss in Sections 4.4 and 4.5, respectively. In addition to these auxiliary losses, we propose a novel bag-of-words adversarial loss in Section 4.6. This makes our model a dual-adversary model. We also propose a novel style dropout regularization to improve the content space encoding in Section 4.7. Section 4.9 presents the approach to transfer style while generating novel text.

Figures 4.1 and 4.2 depict the training and generation processes of our approach.

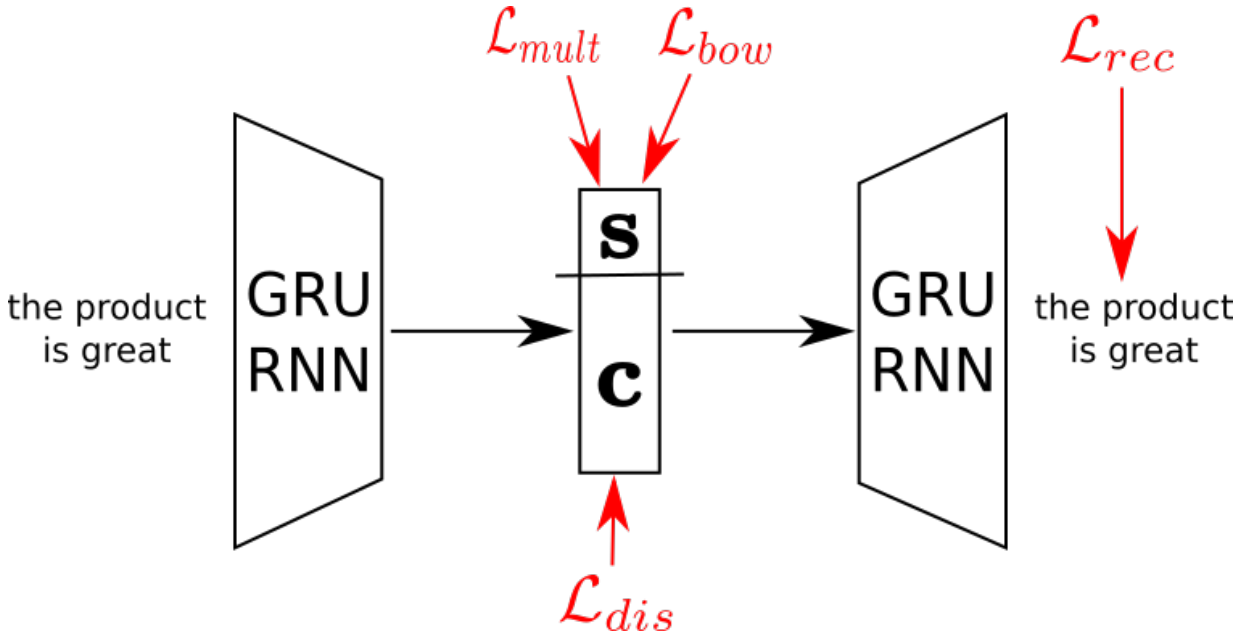


Figure 4.1: Model Training Overview

4.3 Autoencoder

An autoencoder encodes an input to a latent vector space, from which it decodes the input itself. By doing so, the autoencoder learns meaningful representations of data. This serves as our primary learning objective. Besides, we also use the autoencoder weights for text generation in the attribute style-transfer application.

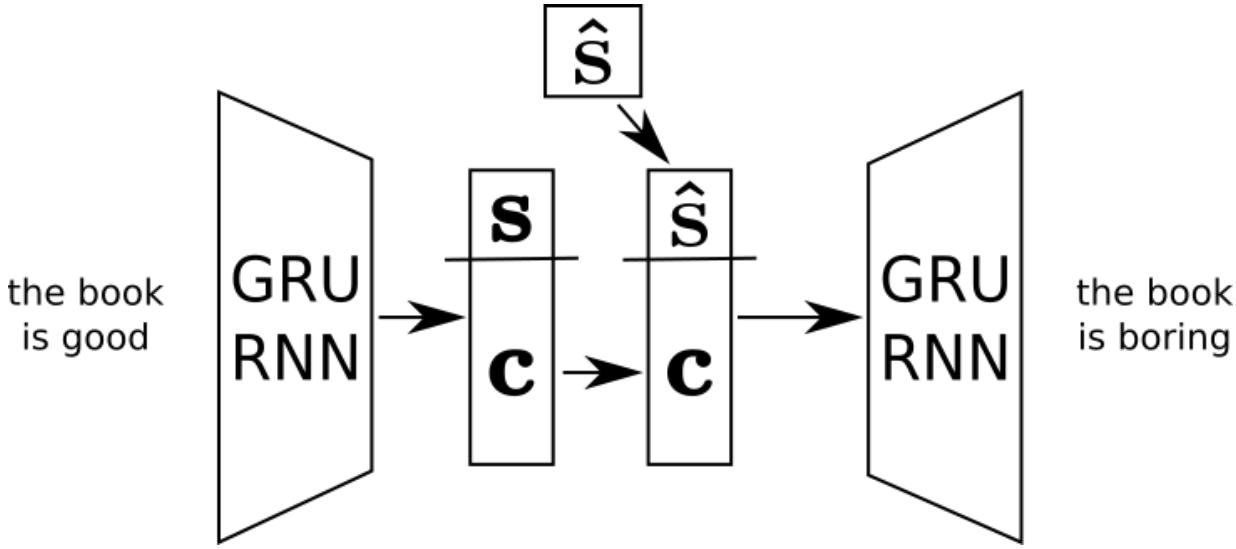


Figure 4.2: Model Inference Overview

Let $x = (x_1, x_2, \cdots x_n)$ be an input sentence. The encoder encodes x using a recurrent neural network (RNN) with gated recurrent units (GRU) [Cho et al., 2014b]. It obtains a hidden state $\boldsymbol{h}$, which in our formulation, is two separate spaces, $\boldsymbol{s}$ for style and, $\boldsymbol{c}$ for content. This variation on how the latent space is parameterized still requires an equal number of parameters when compared to a vanilla RNN encoder. Then a decoder RNN generates a sentence, which ideally should be x itself. Suppose at a time-step t, the decoder RNN predicts the word x_t with probability $p(x_t|\boldsymbol{h}, x_1 \cdots x_{t-1})$, then the autoencoder is trained by minimizing a cross-entropy loss between the input sequence and the generated sequence. The training process is described in more a detailed manner as follows:

The encoder produces the style and content, given that θ_{E_s} represents the subset of encoder parameters responsible for the production of the style vector and θ_{E_c}, similarly, represents the encoder parameters responsible for the production of the content vector.

$$\begin{aligned} \boldsymbol{s} &= \theta_{E_s}(x) \\ \boldsymbol{c} &= \theta_{E_c}(x) \end{aligned}$$

Following which, the decoder attempts to predict the original distribution of x by computing the softmax over the vocabulary size, for as many time-steps as the sequence length, as shown in Equation 4.1

$$\mathcal{L}_{\text{rec}}(\boldsymbol{\theta}_E, \boldsymbol{\theta}_D) = -\mathbb{E}[\log p_D(x|\boldsymbol{s}, \boldsymbol{c})] \tag{4.1}$$

where $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}_D$ are the parameters of the encoder and decoder, respectively. Since this loss trains the autoencoder to reconstruct x, it is also called the reconstruction loss.

4.3.1 Variational Autoencoder

In addition to the deterministic auto-encoding objective presented above, we also implement a variational autoencoder. The variational sampling and KL divergence losses are applied for both the style space $\boldsymbol{s}$ and the content space $\boldsymbol{c}$, separately. The weights applied to each KL divergence loss are tuned independently as a hyperparameter.

Equation 4.2 shows the reconstruction objective that is minimized by the VAE variant of our model.

$$\begin{aligned}\mathcal{L}_{\text{rec}}(\theta_D, \theta_E; x) = &- \mathbb{E}_{q_{E_s}(\boldsymbol{s}|x)q_{E_c}(\boldsymbol{c}|x)}[\log p_D(x|\boldsymbol{s}, \boldsymbol{c})] \\ &+ \lambda_{\text{skl}} KL(q_{E_s}(\boldsymbol{s}|x)||p(z)) \\ &+ \lambda_{\text{ckl}} KL(q_{E_c}(\boldsymbol{c}|x)||p(z))\end{aligned} \tag{4.2}$$

where λ_{skl} and λ_{ckl} are the weights for each of the KL divergence terms.

The motivation for using a variational autoencoder variant to compare the base deterministic autoencoder model to, is the property of VAEs that enable them to learn smooth and continuous latent spaces, without many dead zones in the latent space that cannot be decoded from. Bowman et al. [2016] show this empirically by using the latent space of a text variational autoencoder to interpolate and generate novel sentences from the latent spaces between two valid sentences from a corpus.

Since our objective is to use empirically inferred latent vectors to eventually achieve the attribute style transfer objective, it would be beneficial to compare the experimental performance of a variational autoencoder against a deterministic autoencoder.

Besides the above reconstruction losses, we design auxiliary losses to disentangle the latent space such that $\boldsymbol{h}$ can be separated into two spaces $\boldsymbol{s}$ and $\boldsymbol{c}$, representing style and content respectively, i.e., $\boldsymbol{h} = [\boldsymbol{s}; \boldsymbol{c}]$, where $[\cdot;\cdot]$ denotes concatenation. We also design additional regularizations on the latent space to encourage learning this disentangled representation.

4.4 Multi-Task Classification

Our first auxiliary loss ensures the style space does contain style information. We build a classifier trained using the style space $\boldsymbol{s}$ as its features, predicting the true style (label) y, which is a part of the training data.

This loss can be viewed as a multi-task objective, in addition to the original reconstruction objective, which ensures that the neural network not only decodes the sentence, but also predicts its style (e.g. sentiment) from a sub-space of the latent representation. Similar multi-task losses have been used in previous work for sequence-to-sequence learning [Luong et al., 2015b], sentence representation learning [Jernite et al., 2017] and sentiment analysis [Balikas et al., 2017], among others.

In our application, we follow previous work [Hu et al., 2017, Shen et al., 2017, Fu et al., 2018] and treat the sentiment as the style of interest for our primary evaluation. However, since our model generalizes to more than one class, we use a simple multi-layer perceptron (MLP) followed by a softmax distribution over the possible labels. The softmax function can be expressed as

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{for} j \in 1, \cdots K$$

where z_j denotes an arbitrary label's unweighted probability and K is the total number of labels in the training data. The softmax distribution is re-weighted such that it's constituent values sum up to 1, which is very useful to model predicted probability distributions over an arbitrary number of classes.

The class distribution predicted by the discriminative style classifier is given by

$$y = \sigma(\boldsymbol{w}_{\text{mult}}^{\top} \boldsymbol{s} + b_{\text{mult}}) \tag{4.3}$$

where $\boldsymbol{\theta}_{\text{mult}} = [\boldsymbol{w}_{\text{mult}}; b_{\text{mult}}]$ are the classifier's parameters for multi-task learning.

This is trained using a simple cross-entropy loss

$$\mathcal{L}_{\text{mult}}(\boldsymbol{\theta}_E; \boldsymbol{\theta}_{\text{mult}}) = -\mathbb{E}[\log p(y' | \boldsymbol{s}; \boldsymbol{\theta}_{\text{mult}})] \tag{4.4}$$

where $\boldsymbol{\theta}_E$ are the encoder's parameters, and y' is the true label distribution.

4.5 Adversarial Style Discriminator

The above multi-task loss only operates on the style space, but does not have an effect on the content space $\boldsymbol{c}$. With only the multi-task loss, we cannot guarantee that the

encoder would not also encode stylistic features within the content space. We therefore apply an adversarial loss to disentangle the content space from style information, inspired by adversarial generation [Goodfellow et al., 2014], adversarial domain adaptation [Liu et al., 2017], and adversarial style transfer [Fu et al., 2018].

The main idea is to introduce an adversary that deliberately attempts to discriminate the true label of training samples using only the content vector $\boldsymbol{c}$. Then the autoencoder is trained to learn a content vector space from which its adversary cannot predict style information. Concretely, the adversarial discriminator predicts style y by computing a softmax distribution over a multi-layer perceptron's output layer.

$$y = \sigma(\boldsymbol{w}_{\text{dis}}^{\top}\boldsymbol{c} + b_{\text{dis}}) \tag{4.5}$$

where $\boldsymbol{\theta}_{\text{dis}} = [\boldsymbol{w}_{\text{dis}}; b_{\text{dis}}]$ are the parameters of the adversary.

It is trained by minimizing the following objective, using a cross-entropy loss.

$$\mathcal{L}_{\text{dis}}(\boldsymbol{\theta}_{\text{dis}}) = -\mathbb{E}[\log p(y'|\boldsymbol{c}; \boldsymbol{\theta}_{\text{dis}})] \tag{4.6}$$

where y' is the true label distribution

The adversarial loss appears similar to the multi-task loss as in Equation 4.4. However, it should be emphasized that, for the adversary, the gradient is not propagated back to the autoencoder, i.e. the contents of $\boldsymbol{c}$ are treated as shallow features, as is evident from the omission of θ_E from the training parameters. The network parameters are mutually exclusive for the autoencoder and the adversarial discriminator.

Having trained an adversary, we would like the autoencoder to learn latent representations, such that $\boldsymbol{c}$ is not discriminative in terms of the style or attribute we want to transfer. In other words, we penalize the entropy of the adversary's prediction, given by

$$\mathcal{L}_{\text{adv}}(\boldsymbol{\theta}_E) = \mathcal{H}(y|\boldsymbol{c}; \boldsymbol{\theta}_{\text{dis}}) \tag{4.7}$$

where $\mathcal{H} = -\sum_{i \in \text{labels}} p_i \log p_i$ is the empirical Shannon entropy of the distribution and y is the predicted distribution of style. The adversarial objective is maximized, in this phase, with respect to the encoder. The motivation behind this objective is to increase the uncertainty of the discriminative classifier with respect to predicting style from the content vector. The maximum entropy value attainable over the distribution of attributes is the point at which each label is predicted with equal probability i.e. the highest uncertainty.

4.6 Adversarial Bag-of-Words Discriminator

In addition to the auxiliary losses used above, we also propose a novel bag-of-words discriminator on the style space. The motivation for having this objective is to try and emulate the adversarial signal provided by the style discriminator, and do the same for the content. Here, we define the content of the sentence as the words from the original sentence without any words that are discriminative of style. An input sentence x is represented as a vector of the same size as the corpus vocabulary, with each index of the vector denoting a discrete probability of a word's presence in the sentence. Therefore, this bag-of-words representation is comprised of only 0s and 1s.

The bag-of-words discriminator uses the style vector $\boldsymbol{s}$ produced by the autoencoder model and tries to predict the true bag-of-words distribution using a set of parameters that are mutually exclusive from those of the autoencoder. The discriminator uses a logistic regression to predict a probability of each word's occurrence in the original sentence, between 0 and 1.

$$b = \sigma(\boldsymbol{w}_{\text{bow}}^{\top}\boldsymbol{s} + b_{\text{bow}}) \tag{4.8}$$

where $\boldsymbol{\theta}_{\text{bow}} = [\boldsymbol{w}_{\text{bow}}; b_{\text{bow}}]$ are the classifier's parameters for bag-of-words prediction.

This objective is trained in a similar method to the style adversary, using a cross-entropy loss

$$\mathcal{L}_{\text{bow}}(\boldsymbol{\theta}_{\text{bow}}) = -\mathbb{E}[\log p(b'|\boldsymbol{s}; \boldsymbol{\theta}_{\text{bow}})] \tag{4.9}$$

where b' is the true word distribution.

We also refrain from propagating any updates of this discriminator loss $\mathcal{L}_{\text{bow}}$ to the encoder parameters.

Similar to the style adversary, the empirical Shannon entropy of the predicted distribution is provided as a training signal for the autoencoder model to maximize.

$$\mathcal{L}_{\text{badv}}(\boldsymbol{\theta}_E) = \mathcal{H}(b|\boldsymbol{s}; \boldsymbol{\theta}_{\text{bow}}) \tag{4.10}$$

where b is the predicted word distribution.

The motivation for this adversarial loss is similar to the one used in the context of the style discriminator. We want to encourage the encoder to learn a representation of style that the bag-of-words discriminator cannot predict most of the original words from.

4.6.1 Lexicon Augmented Discriminator

An improvement to the bag-of-words discriminator could be made assuming the availability of a lexicon of words that are associated with a high probability with each of the styles that we are trying to disentangle. A bag-of-words representation that excludes these words can now be created, such that the size of each representation is the set difference given by #((vocab words) \ (lexicon words)).

Our motivation for this enhancement is to avoid penalizing the autoencoder model for encoding words that are pertinent and discriminative for style that may be encoded in the style space. For the sentiment task in our experiments, we use a lexicon of positive and negative sentiment words curated by Hu and Liu [2004].

4.7 Style Dropout Regularization

In addition to the regular training objectives, we also describe a novel regularization technique for disentanglement tasks. This method can also be viewed as analogous to a training objective like the bag-of-words loss.

In our model, we describe the main adversarial loss for style disentanglement by training a discriminator on the content space. However, there is a possibility of the autoencoder learning to ignore the content vector during generation entirely and using only the style space for encoding and later decoding the salient features of the corpus. This scenario is depicted in Figure 4.3 and can be thought of as the model bypassing the content space due to the presence of an adversary that penalizes any style signal in the content space $\boldsymbol{c}$.

To address this issue, we propose a customized version of Dropout [Srivastava et al., 2014]. Similar to how each neuron in a network is dropped out with a predefined probability in the general version of dropout, in this version, we drop out entire style vectors in the mini-batch randomly with a predefined probability. The motivation for doing this is to reinforce the fact that the content latent vector is not to be ignored for the auto-encoding task despite the negative training signal from the adversary.

4.8 Training Process

The overall loss $\mathcal{L}_{\text{ovr}}$ used for the autoencoder is thus comprised of four different objectives: the reconstruction objective, the multi-task objective and the style and content

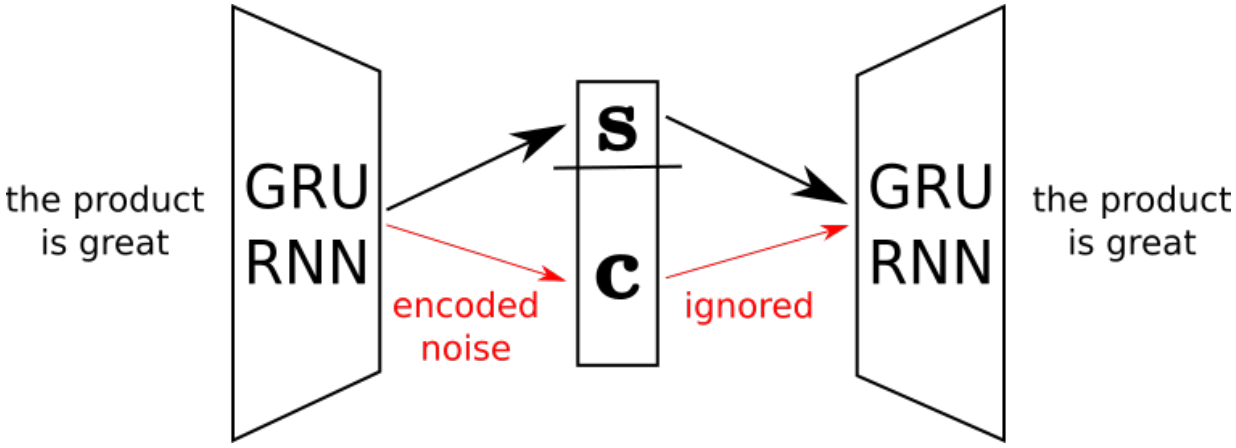


Figure 4.3: Content Space Bypassing

adversaries.

$$\begin{aligned}\mathcal{L}_{\text{ovr}} =& \mathcal{L}_{\text{rec}}(\boldsymbol{\theta}_E, \boldsymbol{\theta}_D) - \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}(\theta_E) + \\ & \lambda_{\text{mult}}\mathcal{L}_{\text{mult}}(\boldsymbol{\theta}_E, \boldsymbol{\theta}_{\text{mult}}) - \lambda_{\text{badv}}\mathcal{L}_{\text{badv}}(\theta_E)\end{aligned}$$

where λ_{mult}, λ_{adv} and λ_{badv} balance the model's auxiliary losses.

To summarize the model training setup, our training process is a loop of the minimization objectives described in Algorithm 1.

```
while epochs remaining do
    minimize L_dis w.r.t. θ_dis;
    minimize L_bow w.r.t. θ_bow;
    minimize L_ovr w.r.t. θ_E, θ_D, θ_mult;
end
```

Algorithm 1: Training Algorithm

4.9 Generating Style-Transferred Sentences

A direct application of our model that disentangles latent space is style transfer for natural language generation. For example, we can generate a sentence with the same general meaning (content) but a different sentiment.

Let x be an input sentence with $\boldsymbol{s}$ and $\boldsymbol{c}$ being the encoded, disentangled style and content vectors, respectively. If we would like to pair its content with a different style, we compute an empirical estimate of the target style's vector $\hat{\boldsymbol{s}}$ by

$$\hat{\boldsymbol{s}} = \frac{\sum_{i \in \text{target style}} \boldsymbol{s}_i}{\#\text{ target style samples}}$$

The inferred target style $\hat{\boldsymbol{s}}$ is concatenated with the encoded content $\boldsymbol{c}$ and fed to the decoder as shown in Figure 4.2.

4.9.1 Nearest-Neighbour Approach for Sentence Generation

The inference approach described above uses the empirical mean of learned embeddings in the style space to transfer new sentences to the style it represents. Though simple and intuitive, this method is fairly rigid and doesn't take into account the fact that the new sentences' content could differ from the content embeddings that could plausibly be coupled with the mean style embedding. This incompatibility between the style and content vectors could hinder the decoder from producing meaningful sentences. In other words, although the mean style vector could produce meaningful sentences when concatenated with almost any content vector, an outlier content vector that is on the periphery of the distribution of the content space might not be decoded into a valid sentence when concatenated with the mean style vector.

Hence, we want to supply a style embedding that might be a closer counterpart to each new content vector seen, by comparing the inference-time content vector to other content vectors seen during training, and identifying a more suitable style vector using a cosine proximity search of the embeddings learned during training. To address this, we propose a nearest neighbour approach to identify better candidates for the style embedding than

the empirical mean of training time style embeddings. Algorithm 2 describes our method.

Data: $S \Rightarrow$ empirical style embeddings from the training data
Data: $C \Rightarrow$ empirical content embeddings from the training data
Data: $s \Rightarrow$ inferred content embedding (singular) of a test sample
Data: $c \Rightarrow$ inferred style embedding (singular) of a test sample
Data: $N \Rightarrow$ number of neighbours (hyperparameter)
Result: new sentence generation style vector for transferring style
$(S, C) \Rightarrow$ pairs of (style, content) embeddings;
(cosine-distance, style-vector) $\Rightarrow$ empty list of tuples;
for s_t, c_t *in* (S, C) **do**
    Compute cd, the cosine distance between c and c_t;
    Add tuple (cd, s_t) to (cosine-distance, style-vector);
end
best-tuples $\Rightarrow$ N best tuples with minimum cosine distance in (cosine-distance, style-vector);
define aggregate-style-vector;
for cd, s_t *in best-tuples* **do**
    aggregate-style-vector = aggregate-style-vector + s_t;
end
return $\frac{\textit{aggregate-style-vector}}{N}$;

Algorithm 2: Nearest-Neighbour Algorithm

However, this method is computationally more expensive, because, for each test sentence, we need to iterate over the entire list of empirical content embeddings whose count is the same as the count of training data samples.

In this chapter, we described our model, including its objectives and regularizations. The next chapter describes the tasks undertaken, details of the datasets, model implementation details and hyperparameters, and the subsequent model evaluation results.

Chapter 5

Experiments

5.1 Datasets and Tasks

We conduct experiments on two datasets, the details for which are given below.

5.1.1 Yelp Service Reviews

We use a Yelp review dataset [Challenge, 2013], which has been sourced from the code repository accompanying the implementation of the paper by Shen et al. [2017], open-sourced by the authors [1]. It contains 444101, 126670 and 63483 sentences for train, validation, and test, respectively, each sampling accompanied by binary sentiment labels. The maximum sentence length is 15, and the vocabulary size is about 9200.

5.1.2 Amazon Product Reviews

We also use an Amazon product reviews dataset [2], following Fu et al. [2018]. The reviews were sourced from the code repository accompanying the paper [3]. It contains 559142, 2000 and 2000 sentences for train, validation, and test, respectively, each sampling accompanied by binary sentiment labels. The maximum sentence length is 20, and the vocabulary size is about 58000.

[1] https://github.com/shentianxiao/language-style-transfer
[2] http://jmcauley.ucsd.edu/data/amazon/
[3] https://github.com/fuzhenxin/text_style_transfer

5.2 Implementation Details and Hyperparameters

We implement the neural-network model in Tensorflow [Abadi et al., 2016], and use Scikit-learn [Pedregosa et al., 2011], NumPy and SciPy for evaluation metrics.

One of the most challenging aspects of training a variational autoencoder (VAE) is avoiding posterior collapse. In previous work [Yang et al., 2017, Bowman et al., 2016, Bahuleyan et al., 2017], this has been done using manually crafted annealing schedules for weight of the KL divergence term. We utilize the tanh annealing schedule proposed by Bahuleyan et al. [2017], and also use their technique of appending the latent vector ($[\boldsymbol{s}; \boldsymbol{c}]$) to the hidden state of the recurrent decoder at each time-step of decoding.

We initialize the word embeddings matrix using pre-trained Word2Vec word embeddings [Mikolov et al., 2013c,a,b] on each of the training corpora, which is then used for the autoencoder model. Greedy decoding is used to predict the next word at each time-step of the decoder RNN [Germann, 2003].

The model uses style embeddings of size 8 and content embeddings of size 128 by default. The encoder and decoder are both GRU-RNNs with a size of 256. The recurrent dropout probability for units in the encoder and decoder is 0.2, and the same dropout probability is used for units in fully-connected layers as well.

We use the Adam optimizer [Kingma and Ba, 2014] for the autoencoder and the RMSProp optimizer [Tieleman and Hinton, 2012] for the discriminators, with an initial learning rate of 10^{-3}, and train the model for 20 epochs. Both the autoencoder and the discriminators are trained once per epoch with $\lambda_{\text{mult}} = 1$, $\lambda_{\text{adv}} = 0.3$ and $\lambda_{\text{badv}} = 0.001$. For the VAE model, we set $\lambda_{\text{skl}} = 0.03$ and $\lambda_{\text{ckl}} = 0.03$.

5.3 Evaluation Metrics

5.3.1 Transfer Strength

Transfer strength can be defined as a measure of how successful the model has been in generating sentences that are consistent with the attribute that needs to be transferred into them. This can be formulated as a Natural Language Understanding (NLU) task that takes in a sentence as input and predicts the probability of the presence of an attribute that we wish to evaluate the presence of.

Consistent with the approach taken by Hu et al. [2017], Shen et al. [2017], Fu et al. [2018], we train a separate model that learns to predict the label among the different

labels to which transfer is possible. We use an open source implementation of the work presented by Kim [2014], which is a convolutional neural network (CNN) model used for text classification [4]. This model has frequently been used as a text classification baseline to compare specialized models against [Tai et al., 2015, Kiros et al., 2015, Zhang et al., 2015]. This CNN classifier is trained on the entire corpus described in the previous section, with a held out validation dataset to assess improvements in accuracy over time. The test dataset for this classifier consists of the style-transferred sentences generated from the autoencoder model.

The style transfer strength is the ratio of sentences successfully transferred to the target style, to the total number of test sentences.

$$\text{transfer-strength} = \frac{count(\text{generated sentences with target style attribute})}{count(\text{generated sentences})}$$

Therefore, the optimal style transfer strength score is 1 and the worst is 0. Here, we choose to ignore the classification errors by our automated classification model, since a good model (with a high classification accuracy) would mis-classify an approximately equal number of samples for each label, and a biased classifier would negatively impact the style transfer strength score. This metric measures the sentence-level style transfer i.e. if 90 sentences are transferred out of 100, the transfer strength is 0.9.

The classifier we train achieves a classification accuracy of approximately 0.97 on the Yelp validation data, and 0.84 on the Amazon validation data.

5.3.2 Content Preservation

The content preservation metric used by Fu et al. [2018] is also used in our work to judge content preservation of the generated sentences compared to the original. This metric is required to ensure that the generated sentences don't contain content, subjects or topics entirely different compared to the original sentences. It measures the sentence-level content preservation.

This method uses 100-dimensional GloVe embeddings [Pennington et al., 2014] pre-trained on the Wikipedia + GigaWord corpora [5]. The min, mean and max GloVe word embeddings for each sentence are concatenated together to create a sentence representation.

[4] https://github.com/dennybritz/cnn-text-classification-tf
[5] https://nlp.stanford.edu/projects/glove/

The cosine similarity between the source sentence and target sentence is used as a measure of content preservation.

Let W be the set of word embeddings in any sentence s. Then the cosine similarity between two sentences s_1 and s_2 is obtained by

$$\begin{aligned} \text{emb} &= [min(W); max(W); mean(W)] \\ \text{cosine-similarity} &= 1 - \cos(\text{emb}_1, \text{emb}_2) \end{aligned}$$

where emb_1 and emb_2 are the sentence embeddings for s_1 and s_2 respectively. The mean cosine similarity of all the sentences in the test dataset is the content preservation that is reported for the experiment.

Original (Positive)	**Transferred (Negative)**
we had the shrimp with vegetables and shrimp fried rice both lovely	we had the bacon cheeseburger and it was cold
both dishes prepared with quality veggies	eggs benedict with no flavor
my appetizer was also very good and unique	my steak was very dry and flavorless
both times i have eaten the lunch buffet and it was outstanding	yes i have had the worst pizza i have ever had
the new york eggrolls are outstanding and the beef dishes we ordered were flavorful	the main issue was our server was extremely rude
Original (Negative)	**Transferred (Positive)**
the chicken " strip were paper thin oddly flavored strips	the bread was wonderful as well
the big chicken sandwich should be called the big mayonnaise sandwich	the salsa is the best i have ever had
fries are not worth coming back	prices are good
but honestly the worst hookah in las vegas	very authentic food in the east valley
second the service was terribly slow	the sushi was delicious

Table 5.1: Examples of Poor Content Preservation

As seen in the examples in Table 5.1, it is easy to optimize for a style-transfer objective as long as there are no constraints on the content preservation regularization. However, this

results in a disconnect in terms of content and the generated sentence no longer resembles the original. To ensure this does not happen, we typically use a much larger content latent space than style latent space, as well as add the necessary regularizations discussed in the previous chapter. Similar to Fu et al. [2018], for the sentiment task, we exclude words from a sentiment lexicon [Hu and Liu, 2004] while evaluating the content preservation score.

However, a slight drawback of this content preservation metric devised by Fu et al. [2018] is that it takes a small range of possible values, and is therefore less sensitive to model performance changes. In theory, the maximum content preservation value achievable is 1 and the minimum is 0. However, from experiments performed by matching random pairs of Yelp corpus sentences, the content preservation we observe is 0.7591 ± 0.0004. This can be treated as an approximate lower bound on the metric for model assessment.

5.3.3 Word Overlap

In addition to the content preservation metric described above, we also utilize a simpler unigram overlap metric. Given a source sentence x and an attribute style transferred sentence y, let w_x and w_y be the set of unique words present in x and y respectively. Then, the word overlap score can be calculated using

$$\text{word-overlap} = \frac{count(w_x \cap w_y)}{count(w_x \cup w_y)}$$

which is simply a normalized measure of overlapping unigrams in the source and target sentences, aggregated over all the sentences being evaluated.

Theoretically, the word overlap score ranges from 0 to 1. However, in practice, since the expectation is that some words will be changed due to the style-transfer process, we wish to strike a good balance with respect to the style transfer strength and original content preservation. Using experiments on random pairs of sentences, the word overlap score computed is 0.0098 ± 0.0006, which can again be treated as a more realistic approximate lower-bound on this metric.

Word overlap can be used interchangeably with the 'Content Preservation' metric described above. For the experiments in this work, we report both.

5.3.4 Language Fluency

Previous works in this area have either utilized human evaluations [Shen et al., 2017, Fu et al., 2018] to evaluate language fluency and content preservation, or do not evaluate

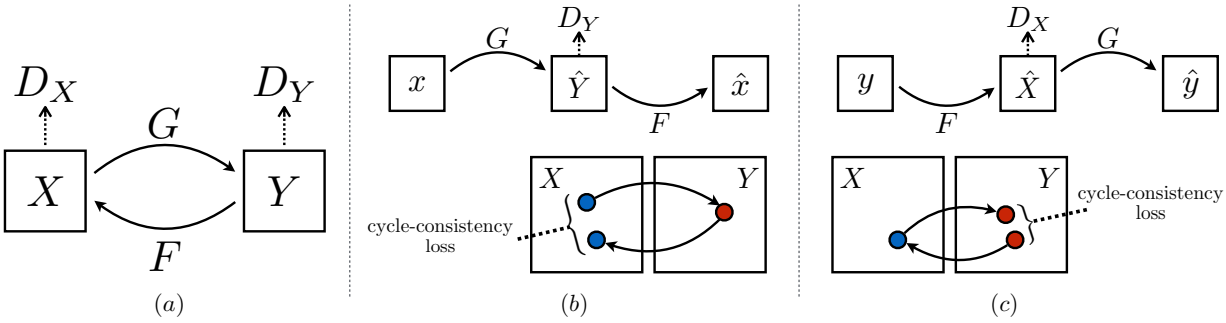
it at all [Hu et al., 2017].

We use a trigram based Kneser–Ney Smoothed language model [Kneser and Ney, 1995] as a quantifiable and automated scoring metric by which to assess the quality of generated sentences. It calculates the probability distribution of trigrams in a document based on their occurrence statistics to build a language model. We train this language model on the complete corpus that we evaluate our style transfer models with. This metric is a word-level evaluator of the fluency of the generated sentences, aggregated over all the sentences.

After the generation of style transferred sentences, we parse them out into trigrams, and calculate the probabilities of each of the trigrams being from the same corpora as the original model was trained on. The log-likelihood scores, as predicted by the Kneser-Ney language model and aggregated over the generated corpus, are reported as the indicator for language fluency.

5.3.5 A Note about Cycle Consistency Loss

The cycle consistency loss is described in the image domain-adaptation work done by Zhu et al. [2017]. Given domains X and Y, they aim to learn a mapping similar to ours, $G : X \rightarrow Y$ which transfers attributes of X to Y. At the same time, they also learn an inverse mapping $F : Y \rightarrow X$. The cycle consistency loss is the step that pushes $F(G(X)) \approx X$ and $G(F(Y)) \approx Y$, as shown in Figure 5.1.



Source: Zhu et al. [2017]

Figure 5.1: Cycle Consistency Loss

This training routine could also be used as an evaluation metric for other domain adaptation tasks, like text attribute style transfer. The advantage of using a cycle consistency

loss is that no other content preservation metric is needed, and no domain or style specific lexicons would be needed.

In the case of our model, assume that, for a sample sentence x from the domain X and the ideal equivalent sentence transferred to the domain Y is y. The transferred sentence generated by our model is $\hat{y}$. In the previous approaches, we compare $\hat{y}$ with the original x using a content similarity metric. However, if the cycle consistency loss were to be used, we can further transfer the style of the sentence $\hat{y}$ back to the domain X, obtaining $\hat{x}$. Now that we have two sentences in the same domain i.e. x and $\hat{x}$, it is a lot simpler to compare them, and this can be achieved by traditional parallel corpora evaluation metrics like BLEU [Papineni et al., 2002].

However, in our problem, a scenario in which both functions F and G do not successfully disentangle style and content, like the examples presented in Table 5.1, is a possibility. This results in the undesirable transfer of content, in addition to the desired transfer of style at inference time. In this case, the transformation could look like

$$x \xrightarrow{G} y^* \xrightarrow{F} x$$

where y^* is a sentence with the requisite transferred style, as well as undesirable transferred content. Such examples would score highly on both the style transfer strength metric and the cycle consistency metric, while still producing generated sentences of poor quality i.e. generated sentences with changed content.

Hence, this metric would not be able to tell the difference between a model with poor content preservation and a good model, and we refrain from using it in our evaluation.

5.4 Experiment Results

5.4.1 Disentangling Latent Space

We first analyse how the style (sentiment) and content of the latent space are disentangled. We train softmax classifiers that use each set of latent space units as features. These classifiers are trained alongside the autoencoder model - without updating the encoder parameters - to predict the true class of an input sample. The results reported here are for the Yelp dataset and are shown in Table 5.2.

The latent space can also be visualized with our method since we are able to infer a pair of style and content embeddings for every training data sample. We use t-SNE

	DAE	VAE
Random/Majority guess	0.6018	
Content latent space ($\boldsymbol{c}$)	0.6137	0.6567
Style latent space ($\boldsymbol{s}$)	0.7927	0.7911
Complete latent space ($[\boldsymbol{s}; \boldsymbol{c}]$)	0.7918	0.7918

Table 5.2: Results - Style Classification Accuracy

plots [Maaten and Hinton, 2008] and use a set of random samples from each label to plot both the style and content embeddings inferred for them during a given epoch of the training procedure. Our hypothesis is that while the points for different labels plotted in the content embedding space would be mixed and indistinguishable in terms of plot co-ordinates, the style embedding space would display the opposite characteristic and show a clean separation of representative samples for each label.

We show t-SNE plots of both the deterministic autoencoder (DAE) and the variational autoencoder (VAE) models in Figure 5.2 and Figure 5.3, respectively.

5.4.2 Style-Transferred Text Generation

We apply our model to a style-transfer sentence generation task, where the goal is to generate a sentence with different sentiment (interpreted as style in this task). We use the metrics discussed in Section 5.3 to evaluate our models.

We compare our approach with the current state-of-the-art work in automated text attribute style transfer [Shen et al., 2017, Fu et al., 2018]. We re-conduct the experiments using their publicly available code and the datasets described in Section 5.1. The results are shown in Table 5.3 (Yelp dataset) and Table 5.4 (Amazon dataset). The models used for evaluations on both datasets are identical.

Table 5.5 presents the results of ablation tests done by evaluating our model using all possible combinations of our training objectives. We see that both, the adversarial and multi-task losses, play a role in the strength of style transfer. It also shows that their usage in a combination can further boost performance of the style-transfer strength.

Some qualitative examples of style-transferred sentence generation are presented in Table 5.6. We see that, with the empirically estimated style vector, we can flexibly control the sentiment of generated sentences.

Model	Transfer Strength	Content Preservation	Word Overlap	Language Fluency
Cross-Alignment [Shen et al., 2017]	0.8087	0.8920	0.2087	-23.3886
Style Embedding [Fu et al., 2018]	0.1819	0.9586	0.6661	-16.1711
Ours (DAE)	0.8425	0.8924	0.2552	-16.4808
Ours (VAE)	0.8903	0.8824	0.2105	-14.4099

Table 5.3: Results - Yelp Dataset Sentiment Transfer

Model	Transfer Strength	Content Preservation	Word Overlap	Language Fluency
Cross-Alignment [Shen et al., 2017]	0.6063	0.8933	0.0241	-26.3093
Style Embedding [Fu et al., 2018]	0.4165	0.9332	0.3588	-28.1346
Ours (DAE)	0.7032	0.9178	0.1305	-32.4184
Ours (VAE)	0.7259	0.9090	0.0814	-28.4953

Table 5.4: Results - Amazon Dataset Sentiment Transfer

In this chapter, we have described the datasets and task, and presented quantitative results for the disentangled space learning and its application to a style (sentiment) transfer task. We also display qualitative results of sentences generated by our model. In the next chapter, we perform a deeper analysis for each result presented here. We also show trade-offs between model scores on different metrics and discuss methods whose results do not fit our original hypotheses.

Objectives	**Transfer Strength**	**Content Preservation**	**Word Overlap**	**Language Fluency**
$\mathcal{L}_{\text{rec}}$	0.1436	**0.9154**	**0.3288**	-14.2781
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{adv}}$	0.7274	0.8800	0.2037	-14.1567
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{mult}}$	0.7894	0.8976	0.2589	-14.5607
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{badv}}$	0.1677	0.9147	0.3282	-14.4486
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{adv}}$, $\mathcal{L}_{\text{mult}}$	**0.8903**	0.8824	0.2105	-14.4099
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{adv}}$, $\mathcal{L}_{\text{badv}}$	0.7491	0.8827	0.2022	-14.3568
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{badv}}$	0.7832	0.8958	0.2567	-14.3373
$\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{adv}}$, $\mathcal{L}_{\text{mult}}$, $\mathcal{L}_{\text{badv}}$	0.8846	0.8782	0.1970	**-14.0479**

Table 5.5: Results - Ablation Tests

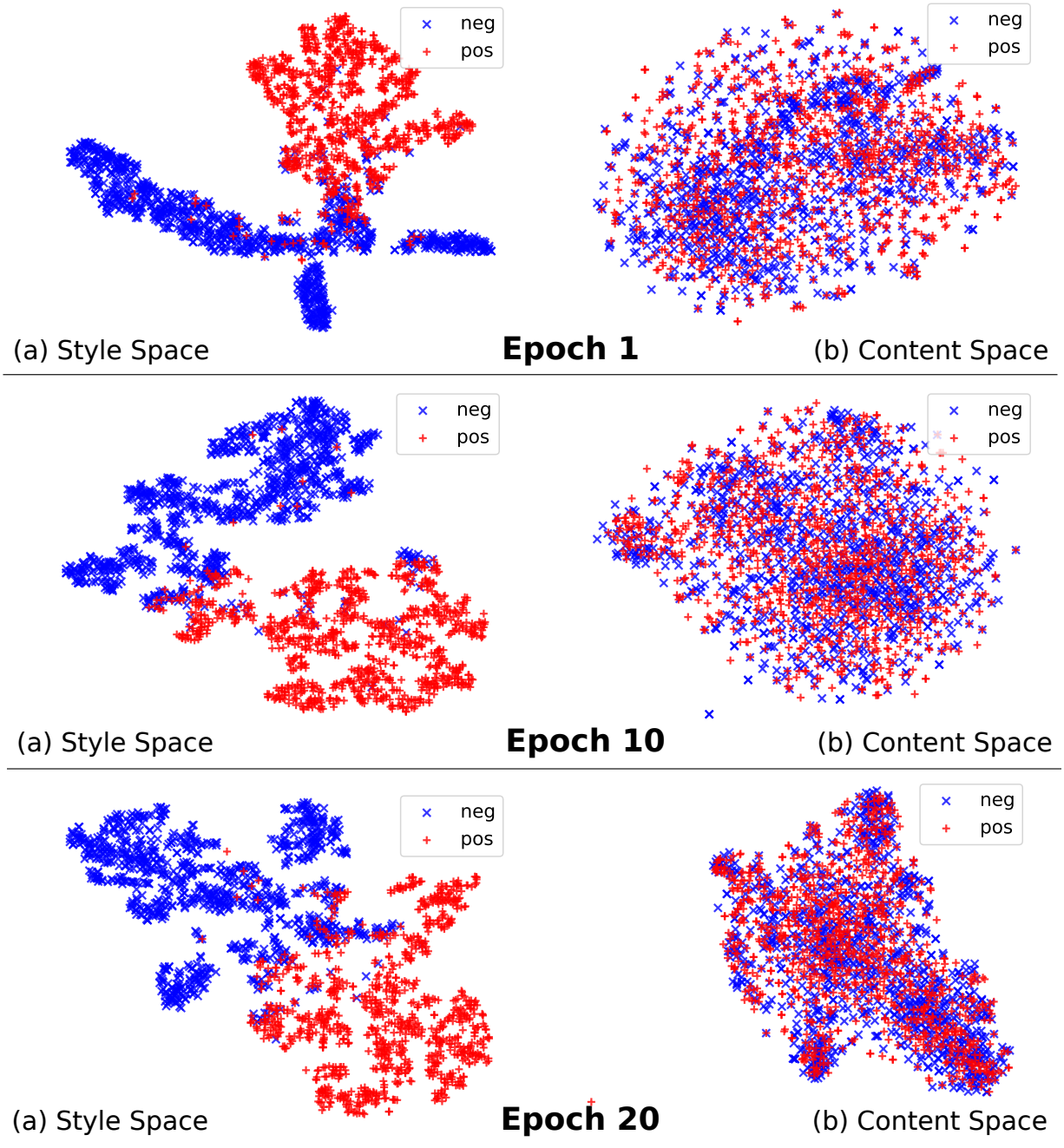


Figure 5.2: T-SNE Plots: (a) Style and (b) Content Spaces - DAE Model

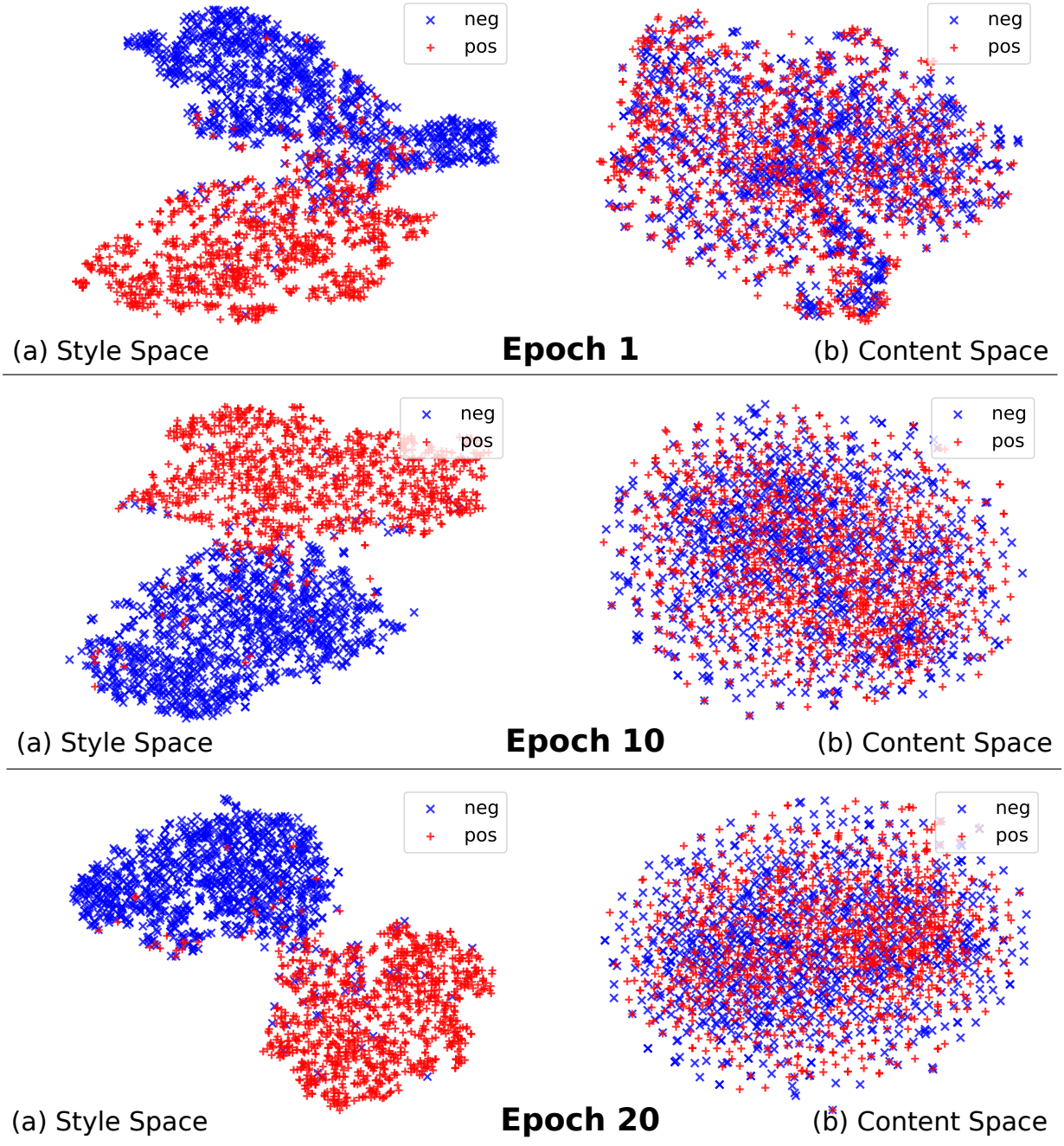


Figure 5.3: T-SNE Plots: (a) Style and (b) Content Spaces - VAE Model

Original (Positive)	**DAE Transferred (Negative)**	**VAE Transferred (Negative)**
i would recommend a visit here	i would not recommend this place again	i would not recommend this place for my experience
the restaurant itself is romantic and quiet	the restaurant itself is soooo quiet	the restaurant itself was dirty
my experience was brief but very good	my experience was very loud and very expensive	my experience was ok but not very much
the food is excellent and the service is exceptional	the food is by the worst part is the horrible costumer service	the food was bland and i am not thrilled with this
the food is very very amazing like beef and fish	the food is very horrible i have ever had mostly fish	the food is very bland and just not fresh
we will definitely come back here	we will not come back here again	we will never come back here
both were very good	everything was very bland	both were very bad
Original (Negative)	**DAE Transferred (Positive)**	**VAE Transferred (Positive)**
so nasty	so helpful	so fabulous
consistently slow	consistently awesome	fast service
crap fries hard hamburger buns burger tasted like crap	cheap and yummy sandwiches really something different	yummy hamburgers and blue cheese bagels are classic italian
oh and terrible tea	oh and awesome tea	oh and great tea
the interior is old and generally falling apart	the interior is clean and orderly as entertaining	the interior is old and noble
front office customer service does not exist here	front office is very professional does you	kudos to customer service is very professional
the crust was kinda gooey like	the crust is kinda traditional	the crust is soooooo worth it

Table 5.6: Style-Transferred Text Examples

Chapter 6

Analysis

6.1 Comparison to State-of-the-Art Approaches

The results in Table 5.3 show that, our approach achieves comparable content-preservation scores to previous work, but considerably better style-transfer strength.

In comparison to Shen et al. [2017], our models obtain better scores in all the evaluation metrics on the Yelp dataset, and on all metrics except language fluency on the Amazon dataset.

Unfortunately, we find that the model described by Fu et al. [2018] performs extremely poorly for both sentiment transfer tasks. A qualitative inspection of the actual sentences generated by their model shows that it frequently resorts to simply reconstructing the source sentence, thereby producing high content preservation, word overlap and language fluency scores and correspondingly low style transfer strength scores. We also tweak the latent space size hyperparameters as described in their paper, but obtain no further improvement.

Our deterministic model is easier to train than a variational model because of the fewer hyperparameters and the absence of KL-weight annealing. It outperforms the previous state-of-the-art model [Shen et al., 2017] in all respects. The variational model achieves comparable performance, while yielding even better style transfer strength and language fluency scores.

This indicates that the disentangled latent space approach we describe can be used for better style-transfer sentence generation, despite not being explicitly trained for the purpose, by simply using empirically encoded vectors from the training process.

6.2 Disentangled Representation Learning

As can be seen from the t-SNE plots in Figure 5.2 and Figure 5.3, sentences with different styles are noticeably separated in a cleaner manner in the style space (LHS), but are indistinguishable in the content space (RHS). It is also evident that the latent space learned by the variational autoencoder is considerably smoother and continuous than the one learned by the deterministic autoencoder.

Since we essentially give the decoder a previously unseen combination of style and latent space at inference time, our hypothesis is that using a variational autoencoder would lead to more fluent generated sentences that are able to preserve content better. In the evaluations performed, the deterministic autoencoder proves equally potent at preserving content, and even exceeds the variational autoencoder's performance in terms of content preservation. However, the fluency of the variational autoencoder is better on average, as measure by a pre-trained language model, and the deterministic autoencoder tends to produce ungrammatical sentences as seen in the Results Table 5.3 and the qualitative examples presented in Table 5.6.

6.3 Latent Space Style Signal

As observed in the results in Table 5.2, the 128-dimensional content vector is not particularly discriminative for style. It achieves a classification accuracy that is slightly better than random/majority guess.

However, the 8-dimensional style vector $\boldsymbol{s}$, despite its low dimensionality, achieves considerably higher style classification accuracy. On combining content and style vectors, we achieve no further improvement. These results verify the effectiveness of our disentangling approach, because the content space doesn't contain style information, opposed to the style space.

This result is consistent with our original hypothesis of restricting the style signal of each encoded sentence into a contained sub-space of the latent representation.

6.4 Transfer Strength vs. Content Preservation

We observe, over the course of experimentation, that there is a visible trade-off between style transfer strength and content preservation produced by our model. For the VAE

model, we tune the KL divergence weight hyperparameters for both the style and content spaces, i.e. λ_{skl} and λ_{ckl} from Equation 4.2, and present results from these experiments in Table 6.1.

λ_{skl}	λ_{ckl}	**Transfer Strength**	**Content Preservation**	**Word Overlap**
0.01	0.01	0.8232	0.8696	0.2118
0.03	0.01	0.7646	0.8769	0.2314
0.03	0.03	0.8123	0.8724	0.2150
0.1	0.03	0.7707	0.8817	0.2334
0.3	0.03	0.7254	0.8783	0.2357
0.03	0.1	0.9052	0.8546	0.1617
0.1	0.1	0.8474	0.8609	0.1941
0.3	0.1	0.7682	0.8645	0.2166
0.03	0.3	0.9434	0.7538	0.0516
0.1	0.3	0.9523	0.7533	0.0707
0.3	0.3	0.9122	0.8069	0.1326

Table 6.1: Results - KL Divergence Weight Hyperparameter Optimisation

We observe that as a general rule, imposing heavier KL divergence constraints on the content embedding space results in improvements in style transfer-strength. However, this also gradually results in posterior collapse when the content KL divergence weight is too high. This results in random noise being encoded in the content space, and the decoder produces random sentences, albeit with a considerably higher transfer strength. For the content KL-weight value 0.3, we obtain content preservation scores ≈ 0.75, which, as shown in Section 5.3, is close to our approximate lower bound for the content preservation metric. This implies that the content of generated sentences for that model is vastly different from the original sentences. Conversely, when the KL divergence constraints are relaxed and we approach a deterministic autoencoder, the content preservation improves and the style transfer strength drops.

In our final VAE model, we set both style and content KL-weights to 0.03, but also append the latent vector $[\boldsymbol{s}; \boldsymbol{c}]$ to the hidden state of the decoder at each time step, which improves the performance of both the style transfer strength and content preservation to better scores than those reported in Table 6.1.

6.5 Negative Results

6.5.1 Style Dropout

Although the idea of regularization by dropout [Srivastava et al., 2014] has been effectively used across neural network models and time-distributed word dropout has been used similarly for text generation tasks [Dai and Le, 2015, Bowman et al., 2016], the empirical results for style dropout prove considerably less effective than expected, in comparison to the other model variants tested in this work.

We perform a simple ablation test on the VAE model with style KL-weight as 0.1 and content KL-weight as 0.03 to test the effect of the addition of the style-dropout regularization, and the results are presented in Table 6.2.

Model Variant	**Transfer Strength**	**Content Preservation**	**Word Overlap**
Without Style Dropout	0.7707	0.8817	0.2334
With Style Dropout	0.5726	0.8816	0.2404

Table 6.2: Results - Style Dropout Effect

As is clear from the results, style dropout only offers a marginally better word overlap score and is comparable on the content preservation metric. However, it performs considerably worse in terms of style transfer strength.

A possible reason for this is that the spaces are already well regularized by the existing objectives and the fact that the content vector is an order of magnitude larger in size compared to the style vector makes it difficult for the model to ignore the content vector and encode garbage into it. Hence, this negates the need for additional regularization to ensure that the content vector is used by the autoencoding model.

6.5.2 Nearest Neighbour Algorithm

We also test the VAE model using the nearest neighbour inference algorithm described in Section 4.9.1. We utilize a VAE model with style KL-weight as 0.03 and content KL-weight as 0.03 for these tests, in which we vary the number of neighbour embeddings considered while selecting a style embedding with which to generate the new sentence. We use a VAE model in which the latent vector is used only for the initial state of the decoder.

The results are presented in Table 6.3, where N is the number of neighbours considered. 'All' neighbours simply means the empirical mean style embedding is used, as in the results shown in the previous tables.

N	**Transfer Strength**	**Content Preservation**	**Word Overlap**
1	0.8040	0.8703	0.1823
10	0.7909	0.8762	0.1859
100	0.8089	0.8792	0.1977
All	0.8123	0.8724	0.2150

Table 6.3: Results - Nearest Neighbour Inference

We observe that the empirical mean embedding that considers all training examples performs marginally better than even a specialized nearest-neighbour algorithm. In addition, it is easier to implement and more computationally efficient, as it does not require neighbour similarity to be computed at inference time.

In this chapter, we have presented an analysis of our experimental results. In the next chapter, we summarize our contributions and conclude this work by presenting avenues for extension and improvement on the current model.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this thesis, we have presented a model that utilizes previously proposed representation learning objectives like multi-task classification and adversarial learning to disentangle the latent space of a language model. We also propose a novel multi-adversary setup using a bag-of-words classification objective. We empirically show the successful disentanglement of the latent spaces, both by using auxiliary classification objectives on the learned latent spaces, and also by visualizing empirical samples using t-SNE plots.

We show that the model maintains a balance between style transfer strength and content preservation, and outperforms state-of-the-art models with respect to the style transfer strength, while performing comparably on content preservation and word overlap. In terms of fluency as measured by the Kneser-Ney language model, our variational autoencoder model outperforms all other models, while simultaneously obtaining considerably better style transfer strength scores. The deterministic autoencoder model performs slightly worse in terms of style transfer strength and fluency, but still eclipses the previous state-of-the-art, in terms of style transfer strength, content preservation and word overlap. We also quantitatively demonstrate, using ablation experiments, that the addition of each of the training objectives we propose individually improves the style transfer capabilities of the model.

We open-source our implementation under a permissive license [1] so that the general framework can be applied to other problems that are thematically similar, and can be

[1] https://github.com/vineetjohn/linguistic-style-transfer

re-purposed for extensions to the model, described below.

7.2 Future Direction

7.2.1 Model Improvements

Albeit effective at the task it sets out to do, there are several improvements to the model that could be evaluated to improve its efficacy, including:

- Unsupervisedly building a lexicon based on word association with each label from the training corpora, such that a hand-crafted lexicon like Hu and Liu [2004] for sentiment is not needed during evaluation.
- The existing model supports transfer between an arbitrary number of discrete labels. However, it can also be easily extended, by training a regressor in place of a classifier, to continuous labels. For instance, ratings on a scale from 1-5 in Amazon/Yelp reviews, controlling generation on a continuous scale for emotion valence etc.
- Although the adversarial learning process is quite effective in the current model, many others have found the adversarial objective hard to train because the family of f-divergences tend to max-out and provide no meaningful gradients. Using an optimal transport metric like the Wasserstein-1 (Earth Mover's Distance) metric seems to address this problem and stabilize training [Arjovsky et al., 2017, Gulrajani et al., 2017]. This could improve the content space disentanglement of our model.
- Since a variational autoencoder is hard to train due to the requirement of manually crafting KL divergence annealing procedures, an alternative is to use a Wasserstein autoencoder [Tolstikhin et al., 2017], that uses empirical sampling from the encoded distribution and a Gaussian distribution, and can effectively replace the KL divergence objective for VAEs.

7.2.2 Other Domains of Application

Although this thesis focuses attention mainly on sentiment transfer as a proxy for general style transfer, this model can also be used for modifying latent attributes in text, like controlling the style of an author or an artist and modelling attributes like generated sentence length. Discourse style could also be modelled and controlled by changing the autoencoder framework to an encoder-decoder framework.

References

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. Paraphrasing for style. *Proceedings of COLING 2012*, pages 2899–2914, 2012.

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

Frederick Jelinek. Markov source modeling of text generation. In *The Impact of Processing Techniques on Communications*, pages 569–591. Springer, 1985.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014a.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015a.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Marc Cavazza and Fred Charles. Dialogue generation in character-based interactive storytelling. In *AIIDE*, pages 21–26, 2005.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016b.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

Yihong Gong and Xin Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25. ACM, 2001.

John M Conroy and Dianne P O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM, 2001.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Intelligent Systems*, pages 45–53, 1994.

Ehud Reiter. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 97–104. Association for Computational Linguistics, 2007.

Albert Gatt, Francois Portet, Ehud Reiter, Jim Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *Ai Communications*, 22(3):153–186, 2009.

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

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1): 98–113, 1997.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

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

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014b.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. In *AAAI*, pages 663–670, 2018.

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

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

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4): 467–479, 1992.

Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. Alternative structures for character-level rnns. *arXiv preprint arXiv:1511.06303*, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013b.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013c.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596, 2017.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6833–6844, 2017.

Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–104. IEEE, 2004.

Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in neural information processing systems*, pages 1252–1260, 2015.

Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced video*

and signal based surveillance, 2009. AVSS'09. Sixth IEEE International Conference on, pages 296–301. Ieee, 2009.

Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3762–3769, 2014.

Philippe Langlais, Alexandre Patry, and Fabrizio Gotti. A greedy decoder for phrase-based statistical machine translation. *Proc. of TMI*, 2007.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*, 2018.

Maria Larsson, Amanda Nilsson, and Mikael Kaageback. Disentangled representations for manipulation of sentiment in text. *arXiv preprint arXiv:1712.10066*, 2017.

Adam Bermingham and Alan F Smeaton. A study of inter-annotator agreement for opinion retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 784–785. ACM, 2009.

Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566. ACM, 2010.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *icassp*, volume 1, page 181e4, 1995.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015b.

Yacine Jernite, Samuel R Bowman, and David Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*, 2017.

Georgios Balikas, Simon Moura, and Massih-Reza Amini. Multitask learning for fine-grained twitter sentiment analysis. In *SIGIR*, pages 1005–1008, 2017.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1–10, 2017.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD*, pages 168–177, 2004.

Yelp Dataset Challenge. Yelp dataset challenge, 2013.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, pages 3881–3890, 2017.

Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. Variational attention for sequence-to-sequence models. *arXiv preprint arXiv:1712.08207*, 2017.

Ulrich Germann. Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 1–8. Association for Computational Linguistics, 2003.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

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

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2242–2251. IEEE, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 9: 2579–2605, 2008.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

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

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.