Incorporating Structural Bias into Neural Networks for Natural Language Processing

Zichao Yang

CMU-CS-19-103

February 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant number IIS-1409802, the Microsoft Corporation under the Large Scale Computation project and the Bloomberg Philanthropies. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Natural Language Processing, Neural Networks, Structure Bias, Attention Mechanism, Visual Question Answering, Document Classification, VAE, GAN

Dedicated to my parents for their unconditional love.

Abstract

Neural networks in recent years have achieved great breakthroughs in natural language processing. Though powerful, neural networks are often statistically inefficient and require large quantities of labeled data to train. One potential reason is that natural language has rich latent structure and general purpose neural architectures have difficulty learning underlying patterns from limited data. In this thesis, we aim to improve the efficiency of neural networks by exploring structural properties of natural language in designing neural model architectures. We accomplish this by embedding prior knowledge into the model itself as a type of inductive bias.

In the first half of this thesis, we explore supervised tasks related to natural language—for example, visual question answering and document classification. We find in those tasks, the inputs have salient features that provide clues to the answers. The salient regions of inputs, however, is not directly annotated and cannot be directly leveraged for training. Moreover, the salient features must be reasoned about and discovered according to context in a step by step manner. By building a specific neural network module using iterative attention mechanism, we are able to localize the most important parts from inputs gradually and use them for prediction. The resulting systems not only achieve the state-of-the-art results, but also provide interpretations for their predictions.

In the second half of this thesis, we explore several unsupervised modeling tasks related to nature language—specifically, variational auto-encoders (VAEs) [59] and generative adversarial networks (GANs) [33]. We find those model designed for continuous inputs such as images do not perform well with natural languages as inputs. The main challenges lie in that the existing neural network modules in VAEs and GANs are not good at dealing with discrete and sequential inputs. To overcome the limitations, we designed network modules with input structure taken into consideration. Specifically, we proposed to use dilated CNNs as decoders for VAEs to control the contextual capacity. For GANs, we proposed to use more structured discriminators to replace the binary classifiers to provide better feedback to generators. Altogether, we have shown that by modifying architectural properties of component modules, we can constrain unsupervised learning problems in a way that makes them more feasible and leads to improved practical results.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors Alex Smola, Eric Xing and Taylor Berg-Kirkpatrick for their guidance, vision and continuing support. I am fortunate to have them as advisors and have learned extensively from them about research, writing and presentation. Their care and encouragement also helped me through some difficult situations in the past five years.

It's my honor to have Ruslan Salakhutdinov and Li Deng in my thesis committee. They provided many insightful comments on my thesis. I also thank them for many invaluable discussion and feedback on many research projects.

I am extremely grateful to have collaborated with many intelligent and amazing people. Special thanks go to Zhiting Hu and Diyi Yang, for not only being great collaborators, but also being best friends. They are profound thinkers and are always great resources for inspirations. I learned a lot from many other collaborators: Chris Dyer, Xiaodong He, Jianfeng Gao, Andrew Wilson, Wang Ling, Nando de Freitas, Mu Li and others. Thanks for many creative discussions and insightful comments.

I am also very thankful for my former research advisor John C.S. Lui for his guidance and support during my M.Phil at Hong Kong. He is the reason why I decided to pursue a PhD at CMU. Thanks also go to the elder ANSR Lab members—Weijie Wu, Minghong Lin and Bin Fan—for their help and advice.

Many very good friends have helped and supported me throughout my PhD, including Xiao Liu, Peng Zhang, Liwei Wang, Wendy Wang and many others. Thanks for the endless hours of chatting with me when I met with difficulties. It's your support, advice and comfort that helped me through the bad days.

My family have supported me along the way and I want to thank them deeply for this. Thanks my parents and sisters for their constant care and unconditional love. I could not have finished the PhD without you. It was you that taught me love, value and meaning of life. You protected and supported me all the way along.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, neural networks have achieved great breakthroughs in many areas of natural language processing (NLP), including language modeling [90], machine translation [110], question answering [41], dialogues systems [120]. With neural networks, we can nowadays easily build models that translate between various pairs of languages–though, translation quality is limited by available training data for each pair. We can also build chat bots that are able to carry on conversations with humans—albeit only for a short duration before losing credibility. There are two main factors underlying the success of neural networks: the unprecedented *scale of datasets*, as well as unprecedented *scale of model*. With large scale supervised data, we can train an extremely large neural network model with billions of parameters. When given sufficient supervised training data, many such models are capable of learning the complicated structures of natural language and generalize to new examples without over-fitting. For example, the state-of-the-art neural machine translation models contain dozens of layers and billions of parameters and are trained with millions of paired sequences. With the unprecedented data and model scale, minimum human efforts are required to pre-process the natural language as inputs and neural network models are directly resorted to optimize learning the feature representations.

Though powerful, neural networks are often *statistically inefficient* and require many labeled examples to train. For some scenarios, it is very expensive to get large amount of supervised data. Training neural networks with limited supervised data can induce over-fitting and make the model generalize poorly on the test set. Further, for many of the most interesting tasks, fully supervised data simply cannot be collected due to the nature of the problem. Suppose we would like to build models for chat bots, it is very difficult to get high quality dialogues to train a model. Human dialogues can be dramatically different in various aspects. It can be of different types such as greeting, chit-chat, information seeking, debating, question answering and so on. For each utterance, there are numerous ways to respond to it. It is hard to get large amount of high quality training data to cover the diverse types of dialogues. Data scarcity problem is also quite common in other areas such as self-driving cars. It is easy to get human experts driving experience in normal situations while the data for emergent situations is quite rare. Training a neural network only on safe traces can be catastrophic. The model is agnostic to emergent situations and it is unpredictable what actions the model will take in contingency. Deploying such models in self-driving cars poses serious safety concerns.

When fully supervised data can be collected, it must be collected in large quantities for current neural architectures to lead to successful results on NLP tasks. This is due to the inherent complexity of language. For example, suppose you want to build a system to translate from German to English. There are many levels of linguistic structure that are relevant for successful generalization: syntactic correspondence between the two languages, lexical correspondence and sparser phenomena such as idiomatic expressions. Suppose a system that needs to process and classify long documents, not only must it be able to process and represent individual sentences, it must capture how the sequence of sentences in the document compose to define the overall meaning. A neural network trained with limited examples can hardly capture such patterns in a way that lets it generalize successful to new data. For limited data scenarios where complex neural models are applied, over-fitting is a common outcome.

Thesis statement

In this thesis, we aim to overcome some of the statistical efficiency concerns about neural networks when applied to NLP data. Specifically, we want to improve the efficiency of neural networks for NLP with limited training data. We propose to constrain aspects of neural architectures to reflect known structural aspects of various type of natural language data. In particular, this is accomplished by embedding *prior knowledge of data structure* into the model itself as a type of inductive bias. By directly incorporating the structural properties of data into the model, we bias the model to more easily capture underlying patterns with limited training data. In some sense, we are trading variance for bias—as long as the constraints we incorporate generalize, we can reduce variance and learn from less data without hurting performance. The key to this approach is in the nature of the constraints we add. The goal is to bias the model in the right direction without making it inflexible.

For example, a common property of language data is that it is hierarchical in nature. Generic neural models might treat language data as a simple sequence, require complex parameter setting to capture any sort of generalizable hierarchy. However, as we will show in Chapter 3, it is possible to add simple attentional structures to such models in order to cause them to treat language as hierarchical data from the beginning. This is just one example—we will go on to see additional types of bias that can be added in supervised and unsupervised systems to make learning more successful.

Incorporating structural properties of natural language into designing neural network models has several advantages. First, as we will demonstrate in this thesis, it improves the modeling efficiency of neural networks. The model we designed achieved the state-of-the-art results on a series of tasks. Since the characteristic of the neural network reflects the structure of data, it is easier for the models to learn from data and generalize to new examples. Second, incorporating structural bias into neural networks make predictions more interpretable. Because such models encode certain assumptions about language structure, their learned parameters can be inspected and interpreted more easily. We will demonstrate this across several models with structural attention.

1.2 Outline and contributions

Designing the model according to structure of natural language comes with its own challenges. First of all, we have to figure out what type of structure to model. The type of inductive bias we will investigate is mainly focused on controlling to flow of information in the model. In some sense, these are *structural* bias, rather than low-level bias via manual feature engineering or lexical constraints. Our goal is for these constraints to be highly generalizable across domains of linguistic data, while leaving room for flexibility to fit specific aspects of data.

Part I: Supervised Learning Many supervised natural language problems require the model to learn to identify salient aspects of input text in order to predict the correct output. The salient regions of inputs, however, is not directly annotated and cannot be directly leveraged for training. Moreover, in many problems, the salient features must be reasoned and discovered according context in a step by step manner. By building a specific neural network module using iterative attention mechanism, we are able to localize the most important parts from inputs gradually and use them for prediction. The resulting systems not only achieved the state-of-the-art results, but also provided interpretations for their predictions. This part consists of the following two chapters:

Stacked Attention Networks We propose a stacked attention model to answer natural language questions based on a reference image. Previously, a commonly used approach was to extract a global image feature vector using a convolution neural network (CNN) and encode the corresponding question as a feature vector using a long short-term memory network (LSTM) and then combine them to infer the answer. Though impressive results have been reported, these models often ignore an important structure of the data–the correspondence between image object and entities names in questions. By examining the image QA data sets, we find that it is often that case that answering a question from an image requires multi-step reasoning–identifying the entities in images and then action based on the question. We propose stacked attention networks (SANs) that captures of the structure of the input image and question pairs. The SAN first uses the question vector to query the image vectors in the first attention layer, then combine the question vector and the retrieved image vectors to form a refined query vector to query the image vectors again in the second attention layer. The higher-level attention layer gives a sharper attention distribution focusing on the regions that are more relevant to the answer. In experiments, we find our model with structure into consideration improve upon previous baseline by more than 4 percent in accuracy. This chapter is largely based on previous work published in [131].

Hierarchical Attention Networks Recent approaches used deep learning, such as convolutional neural networks and recurrent neural networks based on long short-term memory (LSTM) to learn text representations. In this chapter we test the hypothesis that better representations can be obtained by incorporating knowledge of document structure in the model architecture. The intuition underlying our model is that not all parts of a document are equally relevant for answering a query and that determining the relevant sections involves modeling the interactions of the words, not just their presence in isolation. Our primary contribution is a new neural architecture, the Hierarchical Attention Network (HAN) that is designed to capture two basic insights about document structure. First, since documents have a hierarchical structure (words form sentences,

sentences form a document), we likewise construct a document representation by first building representations of sentences and then aggregating those into a document representation. Second, it is observed that different words and sentences in a documents are differentially informative. We test our model with several large scale documents classification task and find our model outperforms previous state-of-the-art model by 4 percent. This chapter is largely based on previous work published in [132].

Part II: Unsupervised Learning We find unsupervised learning models—including variational auto-encoders (VAEs) [59] and generative adversarial networks (GANs) [33] —designed for continuous inputs such as images do not perform well with natural languages as inputs. The main challenges lie in that the existing neural network modules in VAEs and GANs are not good at dealing with discrete and sequential inputs. One view of these issues is that the unsupervised learning objectives are under-constrained. Because these models are so complex, many deficient local optima are available to the learning optimization. Here, we again use targeted architectural changes in order to bias and *constrain* the learning problem so that unsupervised learning becomes more feasible. Specifically, we proposed to use dilated CNNs as decoders for VAEs to control the contextual capacity. For GANs, we proposed to use more structured discriminators to replace the binary classifiers to provide better feedback to generators. The improved models achieved state-of-the-art results on text modeling and task of unsupervised text style transfer. This part consists of the following two chapters:

Dilated CNN Decoders for VAEs The obvious choice for decoding architecture for a textual VAE is an LSTM. However, previous work found that using an LSTM-VAE for text modeling yields higher perplexity on held-out data than using an LSTM language model. In particular, they observe that the LSTM decoder in VAE does not make effective use of the latent representation during training. We hypothesize that the contextual capacity of the decoder plays an important role in whether VAEs effectively condition on the latent representation when trained on text data. We propose the use of a dilated CNN as a decoder in VAE. In the two extremes, depending on the choice of dilation, the CNN decoder can reproduce a simple MLP using a bags of words representation of text, or can reproduce the long-range dependence of recurrent architectures (like an LSTM) by conditioning on the entire history. Thus, by choosing a dilated CNN as the decoder, we are able to conduct experiments where we vary contextual capacity, finding a sweet spot where the decoder can accurately model text but does not yet overpower the latent representation. We conduct experiments on both language modeling and semi-supervised classification and find our carefully chosen decoder architecture output perform vanilla RNN decoder. This chapter is largely based on previous work published in [133].

Language Model Discriminator for GANs In adversarial training, a binary discriminator is used to evaluate whether a generated sentence is real or fake. However, in practice, the error signal from a binary classifier is sometimes insufficient to train the generator to produce fluent language, and optimization can be unstable as a result of the adversarial training step. We propose to use an implicitly trained language model as a new type of discriminator, replacing the more conventional binary classifier. The language model calculates a sentence's likelihood, which decomposes into a product of token-level conditional probabilities. We find empirically that when using the language model as a structured discriminator, it is possible to eliminate adversarial training steps that use negative samples—a critical part of traditional adversarial training. We show that our approach, which uses only a language model as the discriminator, outperforms a broad set of state-of-the-art approaches on the three tasks. This chapter is largely based on previous work published in [134].

Part I

Structural Bias for Supervised Learning

Chapter 2

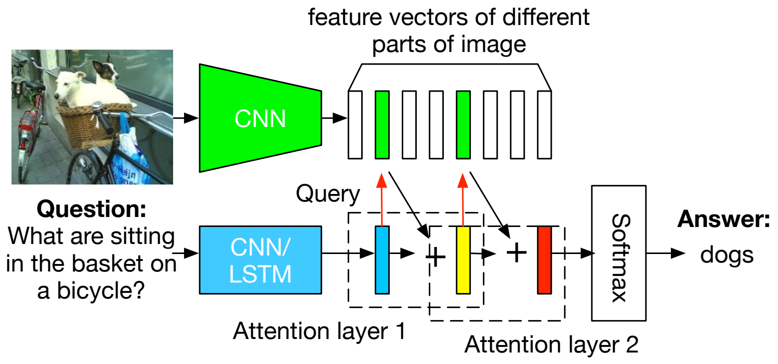
Stacked Attention Networks

2.1 Overview

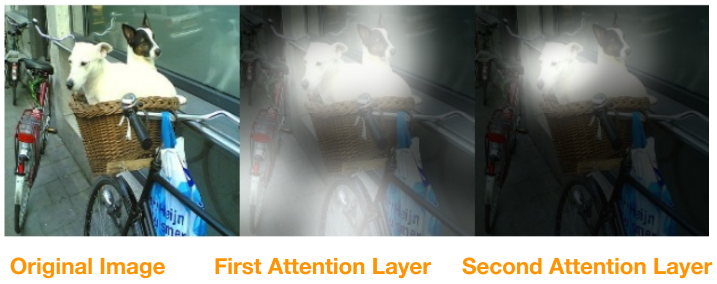
In this chapter, we propose an attention model to answer natural language questions based on a reference image. Most of the recently proposed image QA models are based on neural networks [4, 29, 85, 86, 98]. A commonly used approach was to extract a global image feature vector using a convolution neural network (CNN) [72] and encode the corresponding question as a feature vector using a long short-term memory network (LSTM) [42] and then combine them to infer the answer. Though impressive results have been reported, these models often fail to give precise answers when such answers are related to a set of *fine-grained* regions in an image.

By examining the image QA data sets, we find that it is often that case that answering a question from an image requires multi-step reasoning. Take the question and image in Fig. 2.1 as an example. There are several objects in the image: `bicycles, window, street, baskets` and `dogs`. To answer the question `what are sitting in the basket on a bicycle`, we need to first locate those objects (e.g. `basket, bicycle`) and concepts (e.g., `sitting in`) referred in the question, then gradually rule out irrelevant objects, and finally pinpoint to the region that are most indicative to infer the answer (i.e., `dogs` in the example).

In this chapter, we propose stacked attention networks (SANs) that allow multi-step reasoning for image QA. SANs can be viewed as an extension of the attention mechanism that has been successfully applied in image captioning [127] and machine translation [8]. The overall architecture of SAN is illustrated in Fig. 2.1a. The SAN consists of three major components: (1) the image model, which uses a CNN to extract high level image representations, e.g. one vector for each region of the image; (2) the question model, which uses a CNN or a LSTM to extract a semantic vector of the question and (3) the stacked attention model, which locates, via multi-step reasoning, the image regions that are relevant to the question for answer prediction. As illustrated in Fig. 2.1a, the SAN first uses the question vector to query the image vectors in the first visual attention layer, then combine the question vector and the retrieved image vectors to form a refined query vector to query the image vectors again in the second attention layer. The higher-level attention layer gives a sharper attention distribution focusing on the regions that are more relevant to the answer. Finally, we combine the image features from the highest attention


feature vectors of different parts of image
CNN
Question:
What are sitting in the basket on a bicycle?
CNN/ LSTM
Query
Attention layer 1
Attention layer 2
Softmax
Answer:
dogs

(a) Stacked Attention Network for Image QA



(b) Visualization of the learned multiple attention layers. The stacked attention network first focuses on all referred concepts, e.g., `bicycle, basket` and objects in the basket (`dogs`) in the first attention layer and then further narrows down the focus in the second layer and finds out the answer `dog`.

Figure 2.1: Model architecture and visualization of stacked attention networks

layer with the last query vector to predict the answer.

We perform comprehensive evaluations of our novel stacked attention networks on four image QA benchmarks, demonstrating that the proposed multiple-layer SAN outperforms previous state-of-the-art approaches by a substantial margin. We also perform a detailed analysis where we visualize the outputs of different attention layers of the SAN and demonstrate the process that the SAN takes multiple steps to progressively focus the attention on the relevant visual clues that lead to the answer.

2.2 Related Work

Image QA is closely related to image captioning [18, 26, 55, 63, 88, 122, 127]. In [122], the system first extracted a high level image feature vector from GoogleNet and then fed it into a LSTM to generate captions. The method proposed in [127] went one step further to use an attention mechanism in the caption generation process. Different from [122, 127], the approach proposed in [26] first used a CNN to detect words given the images, then used a maximum entropy language model to generate a list of caption candidates, and finally used a deep multimodal similarity model (DMSM) to re-rank the candidates. Instead of using a RNN or a LSTM, the DMSM uses a CNN to model the semantics of captions.

Unlike image captioning, in image QA, the question is given and the task is to learn the relevant visual and text representation to infer the answer. In order to facilitate the research of image QA, several data sets have been constructed in [4, 29, 86, 98] either through automatic generation based on image caption data or by human labeling of questions and answers given images. Among them, the image QA data set in [98] is generated based on the COCO caption data set. Given a sentence that describes an image, the authors first used a parser to parse the sentence, then replaced the key word in the sentence using question words and the key word became the answer. [29] created an image QA data set through human labeling. The initial version was in Chinese and then was translated to English. [4] also created an image QA data set through human labeling. They collected questions and answers not only for real images, but also for abstract scenes.

Several image QA models were proposed in the literature. [85] used semantic parsers and image segmentation methods to predict answers based on images and questions. [29, 86] both used encoder-decoder framework to generate answers given images and questions. They first used a LSTM to encoder the images and questions and then used another LSTM to decode the answers. They both fed the image feature to every LSTM cell. [98] proposed several neural network based models, including the encoder-decoder based models that use single direction LSTMs and bi-direction LSTMs, respectively. However, the authors found the concatenation of image features and bag of words features worked the best. [4] first encoded questions with LSTMs and then combined question vectors with image vectors by element wise multiplication. [82] used a CNN for question modeling and used convolution operations to combine question vectors and image feature vectors. We compare the SAN with these models in Sec. 3.4.

2.3 Stacked Attention Networks (SANs)

The overall architecture of the SAN is shown in Fig. 2.1a. We describe the three major components of SAN in this section: the image model, the question model, and the stacked attention model.

2.3.1 Image Model

The image model uses a CNN [65, 106, 112] to get the representation of images. Specifically, the VGGNet [106] is used to extract the image feature map f_I from a raw image I:

$$f_I = \text{CNN}_{vgg}(I). \tag{2.1}$$

Unlike previous studies [29, 82, 98] that use features from the last inner product layer, we choose the features f_I from the last pooling layer, which retains spatial information of the original images. We first rescale the images to be 448×448 pixels, and then take the features from the last pooling layer, which therefore have a dimension of $512 \times 14 \times 14$, as shown in Fig. 2.2. 14×14 is the number of regions in the image and 512 is the dimension of the feature vector for each

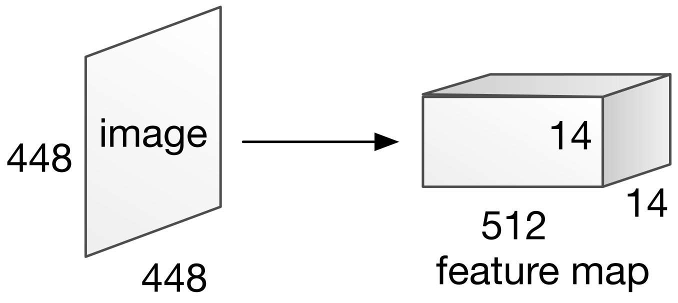


Figure 2.2: CNN based image model for SANs

region. Accordingly, each feature vector in f_I corresponds to a 32×32 pixel region of the input images. We denote by $f_i, i \in [0, 195]$ the feature vector of each image region.

Then for modeling convenience, we use a single layer perceptron to transform each feature vector to a new vector that has the same dimension as the question vector (described in Sec. 2.3.2):

$$v_I = \tanh(W_I f_I + b_I), \tag{2.2}$$

where v_I is a matrix and its i-th column v_i is the visual feature vector for the region indexed by i.

2.3.2 Question Model

As [26, 105, 110] show that LSTMs and CNNs are powerful to capture the semantic meaning of texts, we explore both models for question representations in this study.

LSTM based question model

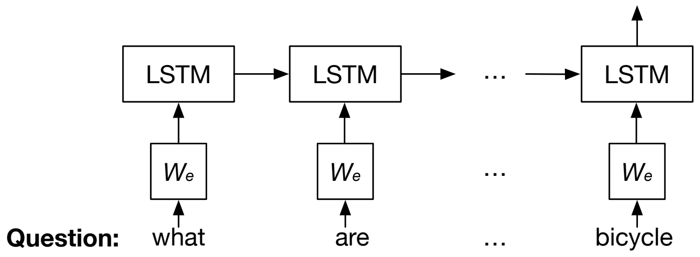


Figure 2.3: LSTM based question model for SANs

The essential structure of a LSTM unit is a memory cell c_t which reserves the state of a sequence. At each step, the LSTM unit takes one input vector (word vector in our case) x_t and updates the memory cell c_t, then output a hidden state h_t. The update process uses the gate mechanism. A forget gate f_t controls how much information from past state c_{t-1} is preserved.

An input gate i_t controls how much the current input x_t updates the memory cell. An output gate o_t controls how much information of the memory is fed to the output as hidden state. The detailed update process is as follows:

$$\begin{aligned}
i_t =& \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), && (2.3)\\
f_t =& \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), && (2.4)\\
o_t =& \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), && (2.5)\\
c_t =& f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), && (2.6)\\
h_t =& o_t \tanh(c_t), && (2.7)
\end{aligned}$$

where i, f, o, c are input gate, forget gate, output gate and memory cell, respectively. The weight matrix and bias are parameters of the LSTM and are learned on training data.

Given the question $q = [q_1, ...q_T]$, where q_t is the one hot vector representation of word at position t, we first embed the words to a vector space through an embedding matrix $x_t = W_e q_t$. Then for every time step, we feed the embedding vector of words in the question to LSTM:

$$\begin{aligned}
x_t =& W_e q_t, t \in \{1, 2, ...T\}, && (2.8)\\
h_t =& \text{LSTM}(x_t), t \in \{1, 2, ...T\}. && (2.9)
\end{aligned}$$

As shown in Fig. 2.3, the question `what are sitting in the basket on a bicycle` is fed into the LSTM. Then the final hidden layer is taken as the representation vector for the question, i.e., $v_Q = h_T$.

CNN based question model

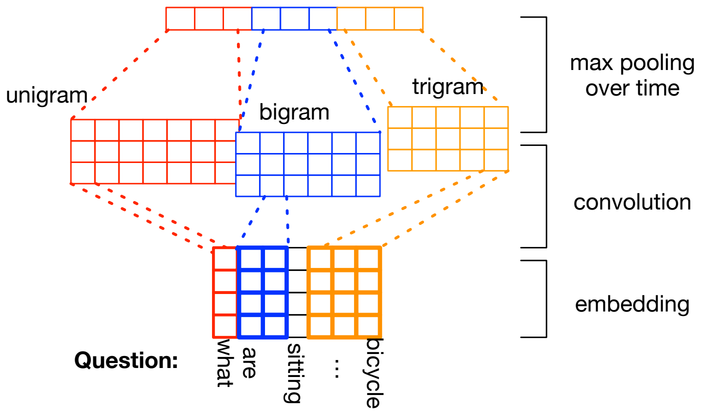


Figure 2.4: CNN based question model for SANs

In this study, we also explore to use a CNN similar to [56] for question representation. Similar to the LSTM-based question model, we first embed words to vectors $x_t = W_e q_t$ and get the question vector by concatenating the word vectors:

$$x_{1:T} = [x_1, x_2, ..., x_T]. \qquad (2.10)$$

Then we apply convolution operation on the word embedding vectors. We use three convolution filters, which have the size of one (unigram), two (bigram) and three (trigram) respectively. The t-th convolution output using window size c is given by:

$$h_{c,t} = \tanh(W_c x_{t:t+c-1} + b_c). \tag{2.11}$$

The filter is applied only to window $t : t + c - 1$ of size c. W_c is the convolution weight and b_c is the bias. The feature map of the filter with convolution size c is given by:

$$h_c = [h_{c,1}, h_{c,2}, ..., h_{c,T-c+1}]. \tag{2.12}$$

Then we apply max-pooling over the feature maps of the convolution size c and denote it as

$$\tilde{h}_c = \max_t [h_{c,1}, h_{c,2}, ..., h_{c,T-c+1}]. \tag{2.13}$$

The max-pooling over these vectors is a coordinate-wise max operation. For convolution feature maps of different sizes $c = 1, 2, 3$, we concatenate them to form the feature representation vector of the whole question sentence:

$$h = [\tilde{h}_1, \tilde{h}_2, \tilde{h}_3], \tag{2.14}$$

hence $v_Q = h$ is the CNN based question vector.

The diagram of CNN model for question is shown in Fig. 4.1b. The convolutional and pooling layers for unigrams, bigrams and trigrams are drawn in red, blue and orange, respectively.

2.3.3 Stacked Attention Networks

Given the image feature matrix v_I and the question feature vector v_Q, SAN predicts the answer via multi-step reasoning.

In many cases, an answer only related to a small region of an image. For example, in Fig. 2.1b, although there are multiple objects in the image: `bicycles`, `baskets`, `window`, `street` and `dogs` and the answer to the question only relates to `dogs`. Therefore, using the one global image feature vector to predict the answer could lead to sub-optimal results due to the noises introduced from regions that are irrelevant to the potential answer. Instead, reasoning via multiple attention layers progressively, the SAN are able to gradually filter out noises and pinpoint the regions that are highly relevant to the answer.

Given the image feature matrix v_I and the question vector v_Q, we first feed them through a single layer neural network and then a softmax function to generate the attention distribution over the regions of the image:

$$h_A = \tanh(W_{I,A} v_I \oplus (W_{Q,A} v_Q + b_A)), \tag{2.15}$$
$$p_I = \text{softmax}(W_P h_A + b_P), \tag{2.16}$$

where $v_I \in \mathbf{R}^{d \times m}$, d is the image representation dimension and m is the number of image regions, $v_Q \in \mathbf{R}^d$ is a d dimensional vector. Suppose $W_{I,A}, W_{Q,A} \in \mathbf{R}^{k \times d}$ and $W_P \in \mathbf{R}^{1 \times k}$, then $p_I \in \mathbf{R}^m$ is an m dimensional vector, which corresponds to the attention probability of each image region given v_Q. Note that we denote by $\oplus$ the addition of a matrix and a vector. Since $W_{I,A}v_I \in \mathbf{R}^{k\times m}$ and both $W_{Q,A}v_Q, b_A \in \mathbf{R}^k$ are vectors, the addition between a matrix and a vector is performed by adding each column of the matrix by the vector.

Based on the attention distribution, we calculate the weighted sum of the image vectors, each from a region, $\tilde{v}_i$ as in Eq. 2.17. We then combine $\tilde{v}_i$ with the question vector v_Q to form a refined query vector u as in Eq. 2.18. u is regarded as a refined query since it encodes both question information and the visual information that is relevant to the potential answer:

$$\tilde{v}_I = \sum_i p_i v_i, \tag{2.17}$$

$$u = \tilde{v}_I + v_Q. \tag{2.18}$$

Compared to models that simply combine the question vector and the global image vector, attention models construct a more informative u since higher weights are put on the visual regions that are more relevant to the question. However, for complicated questions, a single attention layer is not sufficient to locate the correct region for answer prediction. For example, the question in Fig. 2.1 `what are sitting in the basket on a bicycle` refers to some subtle relationships among multiple objects in an image. Therefore, we iterate the above query-attention process using multiple attention layers, each extracting more fine-grained visual attention information for answer prediction. Formally, the SANs take the following formula: for the k-th attention layer, we compute:

$$h_A^k = \tanh(W_{I,A}^k v_I \oplus (W_{Q,A}^k u^{k-1} + b_A^k)), \tag{2.19}$$

$$p_I^k = \text{softmax}(W_P^k h_A^k + b_P^k). \tag{2.20}$$

where u^0 is initialized to be v_Q. Then the aggregated image feature vector is added to the previous query vector to form a new query vector:

$$\tilde{v}_I^k = \sum_i p_i^k v_i, \tag{2.21}$$

$$u^k = \tilde{v}_I^k + u^{k-1}. \tag{2.22}$$

That is, in every layer, we use the combined question and image vector u^{k-1} as the query for the image. After the image region is picked, we update the new query vector as $u^k = \tilde{v}_I^k + u^{k-1}$. We repeat this K times and then use the final u^K to infer the answer:

$$p_{\text{ans}} = \text{softmax}(W_u u^K + b_u). \tag{2.23}$$

Fig. 2.1b illustrates the reasoning process by an example. In the first attention layer, the model identifies roughly the area that are relevant to `basket`, `bicycle`, and `sitting in`. In the second attention layer, the model focuses more sharply on the region that corresponds to the answer `dogs`. More examples can be found in Sec. 3.4.

2.4 Experiments

2.4.1 Data sets

We evaluate the SAN on four image QA data sets.

DAQUAR-ALL is proposed in [85]. There are $6,795$ training questions and $5,673$ test questions. These questions are generated on 795 and 654 images respectively. The images are mainly indoor scenes. The questions are categorized into three types including *Object*, *Color* and *Number*. Most of the answers are single words. Following the setting in [82, 86, 98], we exclude data samples that have multiple words answers. The remaining data set covers 90% of the original data set.

DAQUAR-REDUCED is a reduced version of DAQUAR-ALL. There are $3,876$ training samples and 297 test samples. This data set is constrained to 37 object categories and uses only 25 test images. The single word answers data set covers 98% of the original data set.

COCO-QA is proposed in [98]. Based on the Microsoft COCO data set, the authors first parse the caption of the image with an off-the-shelf parser, then replace the key components in the caption with question words for form questions. There are 78736 training samples and 38948 test samples in the data set. These questions are based on $8,000$ and $4,000$ images respectively. There are four types of questions including *Object*, *Number*, *Color*, and *Location*. Each type takes $70\%, 7\%, 17\%$, and 6% of the whole data set, respectively. All answers in this data set are single word.

VQA is created through human labeling [4]. The data set uses images in the COCO image caption data set [78]. Unlike the other data sets, for each image, there are three questions and for each question, there are ten answers labeled by human annotators. There are $248,349$ training questions and $121,512$ validation questions in the data set. Following [4], we use the top 1000 most frequent answer as possible outputs and this set of answers covers 82.67% of all answers. We first studied the performance of the proposed model on the validation set. Following [26], we split the validation data set into two halves, val1 and val2. We use training set and val1 to train and validate and val2 to test locally. The results on the val2 set are reported in Table. 2.6. We also evaluated the best model, SAN(2, CNN), on the standard test server as provided in [4] and report the results in Table. 2.5.

2.4.2 Baselines and evaluation methods

We compare our models with a set of baselines proposed recently [4, 82, 85, 86, 98] on image QA. Since the results of these baselines are reported on different data sets in different literature, we present the experimental results on different data sets in different tables.

For all four data sets, we formulate image QA as a classification problem since most of answers are single words. We evaluate the model using classification accuracy as reported in [4, 86, 98]. The reference models also report the Wu-Palmer similarity (WUPS) measure [124]. The WUPS measure calculates the similarity between two words based on their longest common subsequence in the taxonomy tree. We can set a threshold for WUPS, if the similarity is less than the threshold, then it is zeroed out. Following the reference models, we use WUPS0.9 and WUPS0.0 as evaluation metrics besides the classification accuracy. The evaluation on the VQA data set is different from other three data sets, since for each question there are ten answer labels that may or may not be the same. We follow [4] to use the following metric: $min(\#$ human labels that match that answer$/3, 1)$, which basically gives full credit to the answer when three or more of the ten human labels match the answer and gives partial credit if there are less matches.

2.4.3 Model configuration and training

For the image model, we use the VGGNet to extract features. When training the SAN, the parameter set of the CNN of the VGGNet is fixed. We take the output from the last pooling layer as our image feature which has a dimension of $512 \times 14 \times 14$.

For DAQUAR and COCO-QA, we set the word embedding dimension and LSTM's dimension to be 500 in the question model. For the CNN based question model, we set the unigram, bigram and trigram convolution filter size to be 128, 256, 256 respectively. The combination of these filters makes the question vector size to be 640. For VQA dataset, since it is larger than other data sets, we double the model size of the LSTM and the CNN to accommodate the large data set and the large number of classes. In evaluation, we experiment with SAN with one and two attention layers. We find that using three or more attention layers does not further improve the performance.

In our experiments[1], all the models are trained using stochastic gradient descent with momentum 0.9. The batch size is fixed to be 100. The best learning rate is picked using grid search. Gradient clipping technique [34] and dropout [108] are used.

2.4.4 Results and analysis

The experimental results on DAQUAR-ALL, DAQUAR-REDUCED, COCO-QA and VQA are presented in Table. 2.1 to 2.6 respectively. Our model names explain their settings: SAN is short for the proposed stacked attention networks, the value 1 or 2 in the brackets refer to using one or two attention layers, respectively. The keyword LSTM or CNN refers to the question model that SANs use.

The experimental results in Table. 2.1 to 2.6 show that the two-layer SAN gives the best results across all data sets and the two kinds of question models in the SAN, LSTM and CNN, give similar performance. For example, on DAQUAR-ALL (Table. 2.1), both of the proposed two-layer SANs outperform the two best baselines, the IMG-CNN in [82] and the Ask-Your-Neuron in [86], by 5.9% and 7.6% absolute in accuracy, respectively. Similar range of improvements are observed in metrics of WUPS0.9 and WUPS0.0. We also observe significant improvements on DAQUAR-REDUCED (Table. 2.2), i.e., our SAN(2, LSTM) outperforms the IMG-CNN [82], the 2-VIS+BLSTM [98], the Ask-Your-Neurons approach [86] and the Multi-World [85] by 6.5%, 10.4%, 11.5% and 33.5% absolute in accuracy, respectively. On the larger COCO-QA data set, the proposed two-layer SANs significantly outperform the best baselines from [82] (IMG-CNN) and [98] (IMG+BOW and 2-VIS+BLSTM) by 5.1% and 6.6% in accuracy (Table. 2.3). Table. 2.5 summarizes the performance of various models on VQA, which is the largest among

[1]Our code is publicly available at `https://github.com/zcyang/imageqa-san`

Methods	Accuracy	WUPS0.9	WUPS0.0
`Multi-World:` [85]			
Multi-World	7.9	11.9	38.8
`Ask-Your-Neurons:` [86]			
Language	19.1	25.2	65.1
Language + IMG	21.7	28.0	65.0
`CNN:` [82]			
IMG-CNN	23.4	29.6	63.0
`Ours:`			
SAN(1, LSTM)	28.9	34.7	68.5
SAN(1, CNN)	29.2	35.1	67.8
SAN(2, LSTM)	**29.3**	34.9	68.1
SAN(2, CNN)	**29.3**	**35.1**	**68.6**
`Human :`[85]			
Human	50.2	50.8	67.3

Table 2.1: DAQUAR-ALL results, in percentage

the four data sets. The overall results show that our best model, SAN(2, CNN), outperforms the LSTM Q+I model, the best baseline from [4], by 4.8% absolute. The superior performance of the SANs across all four benchmarks demonstrate the effectiveness of using multiple layers of attention.

In order to study the strength and weakness of the SAN in detail, we report performance at the question-type level on the two large data sets, COCO-QA and VQA, in Table. 2.4 and 2.5, respectively. We observe that on COCO-QA, compared to the two best baselines, IMG+BOW and 2-VIS+BLSTM, out best model SAN(2, CNN) improves 7.2% in the question type of *Color*, followed by 6.1% in *Objects*, 5.7% in *Location* and 4.2% in *Number*. We observe similar trend of improvements on VQA. As shown in Table. 2.5, compared to the best baseline LSTM Q+I, the biggest improvement of SAN(2, CNN) is in the *Other* type, 9.7%, followed by the 1.4% improvement in *Number* and 0.4% improvement in *Yes/No*. Note that the *Other* type in VQA refers to questions that usually have the form of "what color, what kind, what are, what type, where" etc., which are similar to question types of *Color*, *Objects* and *Location* in COCO-QA. The VQA data set has a special *Yes/No* type of questions. The SAN only improves the performance of this type of questions slightly. This could due to that the answer for a *Yes/No* question is very question dependent, so better modeling of the visual information does not provide much additional gains. This also confirms the similar observation reported in [4], e.g., using additional image information only slightly improves the performance in *Yes/No*, as shown in Table. 2.5, Q+I vs Question, and LSTM Q+I vs LSTM Q.

Our results demonstrate clearly the positive impact of using multiple attention layers. In all four data sets, two-layer SANs always perform better than the one-layer SAN. Specifically, on COCO-QA, on average the two-layer SANs outperform the one-layer SANs by 2.2% in the type

Methods	Accuracy	WUPS0.9	WUPS0.0
Multi-World: [85]			
Multi-World	12.7	18.2	51.5
Ask-Your-Neurons: [86]			
Language	31.7	38.4	80.1
Language + IMG	34.7	40.8	79.5
VSE: [98]			
GUESS	18.2	29.7	77.6
BOW	32.7	43.2	81.3
LSTM	32.7	43.5	81.6
IMG+BOW	34.2	45.0	81.5
VIS+LSTM	34.4	46.1	82.2
2-VIS+BLSTM	35.8	46.8	82.2
CNN: [82]			
IMG-CNN	39.7	44.9	83.1
Ours:			
SAN(1, LSTM)	45.2	49.6	84.0
SAN(1, CNN)	45.2	49.6	83.7
SAN(2, LSTM)	**46.2**	**51.2**	**85.1**
SAN(2, CNN)	45.5	50.2	83.6
Human :[85]			
Human	60.3	61.0	79.0

Table 2.2: DAQUAR-REDUCED results, in percentage

Methods	Accuracy	WUPS0.9	WUPS0.0
`VSE:` [98]			
GUESS	6.7	17.4	73.4
BOW	37.5	48.5	82.8
LSTM	36.8	47.6	82.3
IMG	43.0	58.6	85.9
IMG+BOW	55.9	66.8	89.0
VIS+LSTM	53.3	63.9	88.3
2-VIS+BLSTM	55.1	65.3	88.6
`CNN:` [82]			
IMG-CNN	55.0	65.4	88.6
CNN	32.7	44.3	80.9
`Ours:`			
SAN(1, LSTM)	59.6	69.6	90.1
SAN(1, CNN)	60.7	70.6	90.5
SAN(2, LSTM)	61.0	71.0	90.7
SAN(2, CNN)	**61.6**	**71.6**	**90.9**

Table 2.3: COCO-QA results, in percentage

Methods	Objects	Number	Color	Location
`VSE:` [98]				
GUESS	2.1	35.8	13.9	8.9
BOW	37.3	43.6	34.8	40.8
LSTM	35.9	45.3	36.3	38.4
IMG	40.4	29.3	42.7	44.2
IMG+BOW	58.7	44.1	52.0	49.4
VIS+LSTM	56.5	46.1	45.9	45.5
2-VIS+BLSTM	58.2	44.8	49.5	47.3
`Ours:`				
SAN(1, LSTM)	62.5	49.0	54.8	51.6
SAN(1, CNN)	63.6	48.7	56.7	52.7
SAN(2, LSTM)	63.6	**49.8**	57.9	52.8
SAN(2, CNN)	**64.5**	48.6	**57.9**	**54.0**

Table 2.4: COCO-QA accuracy per class, in percentage

	test-dev				test-std
Methods	All	Yes/No	Number	Other	All
VQA: [4]					
Question	48.1	75.7	36.7	27.1	-
Image	28.1	64.0	0.4	3.8	-
Q+I	52.6	75.6	33.7	37.4	-
LSTM Q	48.8	78.2	35.7	26.6	-
LSTM Q+I	53.7	78.9	35.2	36.4	54.1
SAN(2, CNN)	**58.7**	**79.3**	**36.6**	**46.1**	**58.9**

Table 2.5: VQA results on the official server, in percentage

Methods	All	Yes/No 36%	Number 10%	Other 54%
SAN(1, LSTM)	56.6	78.1	41.6	44.8
SAN(1, CNN)	56.9	78.8	42.0	45.0
SAN(2, LSTM)	57.3	78.3	**42.2**	45.9
SAN(2, CNN)	**57.6**	78.6	41.8	**46.4**

Table 2.6: VQA results on our partition, in percentage

of *Color*, followed by 1.3% and 1.0% in the *Location* and *Objects* categories, and then 0.4% in *Number*. This aligns to the order of the improvements of the SAN over baselines. Similar trends are observed on VQA (Table. 2.6), e.g., the two-layer SAN improve over the one-layer SAN by 1.4% for the *Other* type of question, followed by 0.2% improvement for *Number*, and flat for *Yes/No*.

2.4.5 Visualization of attention layers

In this section, we present analysis to demonstrate that using multiple attention layers to perform multi-step reasoning leads to more fine-grained attention layer-by-layer in locating the regions that are relevant to the potential answers. We do so by visualizing the outputs of the attention layers of a sample set of images from the COCO-QA test set. Note the attention probability distribution is of size 14×14 and the original image is 448×448, we up-sample the attention probability distribution and apply a Gaussian filter to make it the same size as the original image.

Fig. 2.5 presents six examples. More examples are presented in the appendix. They cover types as broad as *Object*, *Numbers*, *Color* and *Location*. For each example, the three images from left to right are the original image, the output of the first attention layer and the output of the second attention layer, respectively. The bright part of the image is the detected attention. Across all those examples, we see that in the first attention layer, the attention is scattered on many objects in the image, largely corresponds to the objects and concepts referred in the question, whereas in the second layer, the attention is far more focused on the regions that lead to the correct answer. For example, consider the question `what is the color of the horns`, which asks the color of the horn on the woman's head in Fig. 2.5(f). In the output of the first attention layer, the model first recognizes a woman in the image. In the output of the second attention layer, the attention is focused on the head of the woman, which leads to the answer of the question: the color of the horn is `red`.

2.4.6 Errors analysis

We randomly sample 100 images from the COCO-QA test set that the SAN make mistakes. We group the errors into four categories: (i) the SANs focus the attention on the wrong regions (22%), e.g., the example in Fig. 2.6(a); (ii) the SANs focus on the right region but predict a wrong answer (42%), e.g., the examples in Fig. 2.6(b)(c)(d); (iii) the answer is ambiguous, the SANs give answers that are different from labels, but might be acceptable (31%). E.g., in Fig. 2.6(e), the answer label is `pot`, but out model predicts `vase`, which is also visually reasonable; (iv) the labels are clearly wrong (5%). E.g., in Fig. 2.6(f), our model gives the correct answer `trains` while the label `cars` is wrong.

2.5 Recent development

Since its publication, this work has attracted a lot of attention and been frequently cited and expanded upon. These works are divided into two main categories. The first category of works embedded more prior knowledge to design the attention models. [81] improved upon our work by using attention both on the image and question. [125] used dynamic memory network for this task, in which a RNN was used to model the iterative attention steps. [142] added Conditional Random Field over the attention maps as regularization to ensure attention maps in different layers are different. The other category of work used knowledge from other data and model sources. For example, [3] used semantic parsing to get the tree structure of the question. They then defined operations based on parsing tree results to build modular network for each question and then executed the network on the image input. Similarly, [2] used an object detection model to segment the objects from the images. They then applied our model on the object outputs and found significant improvement.

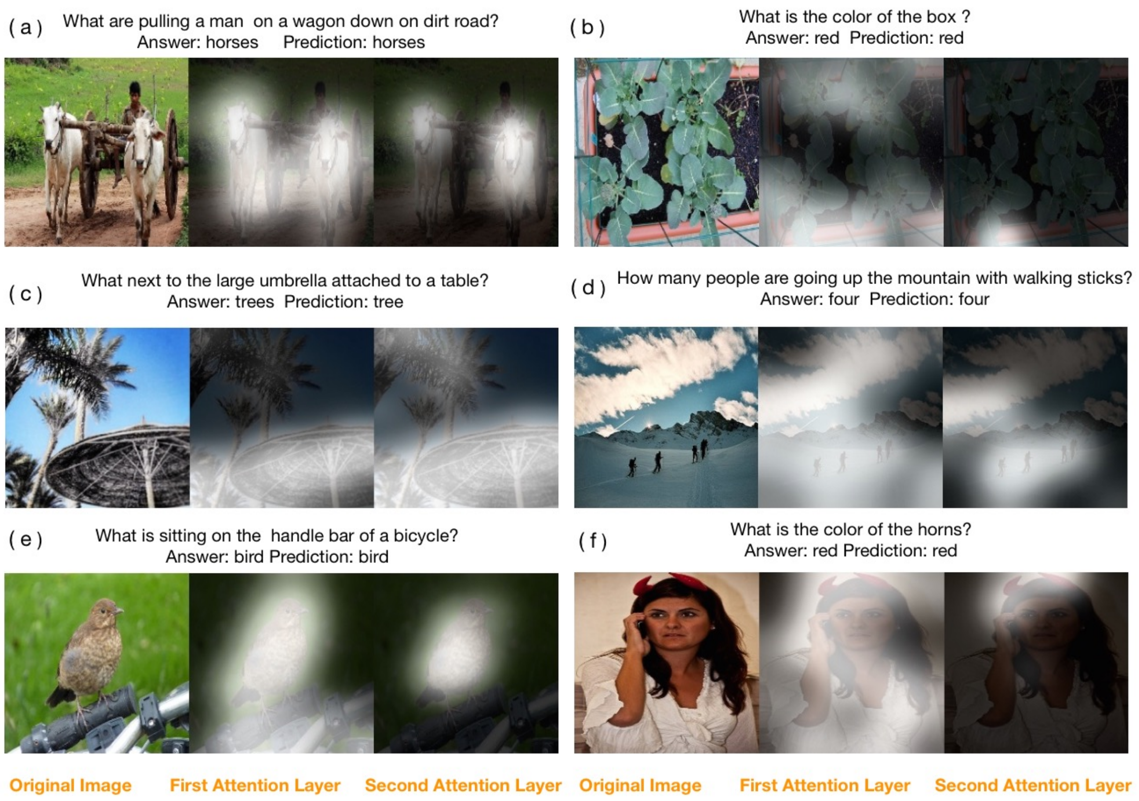

(a) What are pulling a man on a wagon down on dirt road?
Answer: horses Prediction: horses
(b) What is the color of the box ?
Answer: red Prediction: red
(c) What next to the large umbrella attached to a table?
Answer: trees Prediction: tree
(d) How many people are going up the mountain with walking sticks?
Answer: four Prediction: four
(e) What is sitting on the handle bar of a bicycle?
Answer: bird Prediction: bird
(f) What is the color of the horns?
Answer: red Prediction: red
Original Image
First Attention Layer
Second Attention Layer
Original Image
First Attention Layer
Second Attention Layer

Figure 2.5: Visualization of two attention layers

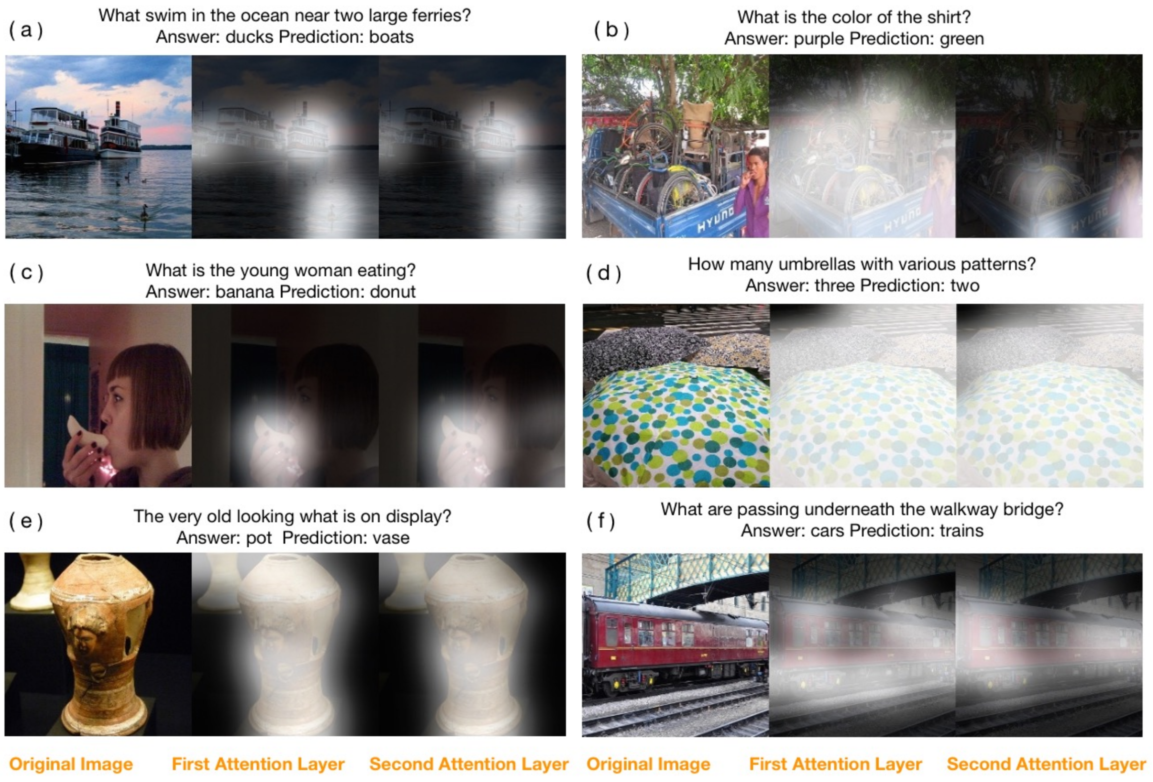

(a) What swim in the ocean near two large ferries?
Answer: ducks Prediction: boats
(b) What is the color of the shirt?
Answer: purple Prediction: green
(c) What is the young woman eating?
Answer: banana Prediction: donut
(d) How many umbrellas with various patterns?
Answer: three Prediction: two
(e) The very old looking what is on display?
Answer: pot Prediction: vase
(f) What are passing underneath the walkway bridge?
Answer: cars Prediction: trains
Original Image
First Attention Layer
Second Attention Layer
Original Image
First Attention Layer
Second Attention Layer

Figure 2.6: Examples of mistakes

Chapter 3

Hierarchical Attention Networks

3.1 Overview

The idea from previous chapter can be generalized to many other supervised tasks, such as document classification. Documents are quite long with very rich semantic structure. However, there are some key sentences or words that deliver the main meaning of the document. The key parts have to be discovered through reasoning according to the document context. To achieve that goal, we propose an attention model for document classification in this chapter. We build a model with attention mechanism that can automatically extract the most relevant words and sentences out of documents.

Traditional approaches of text classification represent documents with sparse lexical features, such as n-grams, and then use a linear model or kernel methods on this representation [51, 123]. More recent approaches used deep learning, such as convolutional neural networks [10] and recurrent neural networks based on long short-term memory (LSTM) [42] to learn text representations.

pork belly = delicious . || scallops? || I don't even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Figure 3.1: A simple example review from Yelp 2013 that consists of five sentences, delimited by period, question mark. The first and third sentence delivers stronger meaning and inside, the word *delicious, a-m-a-z-i-n-g* contributes the most in defining sentiment of the two sentences.

Although neural-network–based approaches to text classification have been quite effective [52, 56, 115, 139], in this chapter we test the hypothesis that better representations can be obtained by incorporating knowledge of document structure in the model architecture. The intuition underlying our model is that not all parts of a document are equally relevant for answering a query and that determining the relevant sections involves modeling the interactions of the words, not just their presence in isolation.

Our primary contribution is a new neural architecture (§3.3), the Hierarchical Attention Network (HAN) that is designed to capture two basic insights about document structure. First, since documents have a hierarchical structure (words form sentences, sentences form a document), we likewise construct a document representation by first building representations of sentences and then aggregating those into a document representation. Second, it is observed that different words and sentences in a documents are differentially informative. Moreover, the importance of words and sentences are highly context dependent, i.e. the same word or sentence may be differentially important in different context (§3.4.5). To include sensitivity to this fact, our model includes two levels of attention mechanisms [8, 127] — one at the word level and one at the sentence level — that let the model to pay more or less attention to individual words and sentences when constructing the representation of the document. To illustrate, consider the example in Fig. 3.1, which is a short Yelp review where the task is to predict the rating on a scale from 1–5. Intuitively, the first and third sentence have stronger information in assisting the prediction of the rating; within these sentences, the word `delicious, a-m-a-z-i-n-g` contributes more in implying the positive attitude contained in this review. Attention serves two benefits: not only does it often result in better performance, but it also provides insight into which words and sentences contribute to the classification decision which can be of value in applications and analysis [30, 105].

The key difference to previous work is that our system uses *context* to discover *when* a sequence of tokens is relevant rather than simply filtering for (sequences of) tokens, taken out of context. To evaluate the performance of our model in comparison to other common classification architectures, we look at six data sets (§3.4). Our model outperforms previous approaches by a significant margin.

3.2 Related Work

[56] use neural networks for text classification. The architecture is a direct application of CNNs, as used in computer vision [72], albeit with NLP interpretations. [52] explores the case of directly using a high-dimensional one hot vector as input. They find that it performs well. Unlike word level modelings, [139] apply a character-level CNN for text classification and achieve competitive results. [107] use recursive neural networks for text classification. [113] explore the structure of a sentence and use a tree-structured LSTMs for classification. There are also some works that combine LSTM and CNN structure to for sentence classification [67, 141]. [115] use hierarchical structure in sentiment classification. They first use a CNN or LSTM to get a sentence vector and then a bi-directional gated recurrent neural network to compose the sentence vectors to get a document vectors. There are some other works that use hierarchical structure in sequence generation [73] and language modeling [77].

The attention mechanism was proposed by [8] in machine translation. The encoder decoder framework is used and an attention mechanism is used to select the reference words in original language for words in foreign language before translation. [127] uses the attention mechanism in image caption generation to select the relevant image regions when generating words in the captions. Further uses of the attention mechanism include parsing [121], natural language question answering [41, 66, 109], and image question answering [130]. Unlike these works, we explore a *hierarchical* attention mechanism (to the best of our knowledge this is the first such instance).

3.3 Hierarchical Attention Networks

The overall architecture of the Hierarchical Attention Network (HAN) is shown in Fig. 3.2. It consists of several parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. We describe the details of different components in the following sections.

3.3.1 GRU-based sequence encoder

The GRU [8] uses a gating mechanism to track the state of sequences without using separate memory cells. There are two types of gates: the reset gate r_t and the update gate z_t. They together control how information is updated to the state. At time t, the GRU computes the new state as

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{3.1}$$

This is a linear interpolation between the previous state h_{t-1} and the current new state $\tilde{h}_t$ computed with new sequence information. The gate z_t decides how much past information is kept and how much new information is added. z_t is updated as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{3.2}$$

where x_t is the sequence vector at time t. The candidate state $\tilde{h}_t$ is computed in a way similar to a traditional recurrent neural network (RNN):

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h), \tag{3.3}$$

Here r_t is the reset gate which controls how much the past state contributes to the candidate state. If r_t is zero, then it forgets the previous state. The reset gate is updated as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.4}$$

3.3.2 Hierarchical Attention

We focus on document-level classification in this work. Assume that a document has L sentences s_i and each sentence contains T_i words. w_{it} with $t \in [1, T]$ represents the words in the ith sentence. The proposed model projects the raw document into a vector representation, on which we build a classifier to perform document classification. In the following, we will present how we build the document level vector progressively from word vectors by using the hierarchical structure.

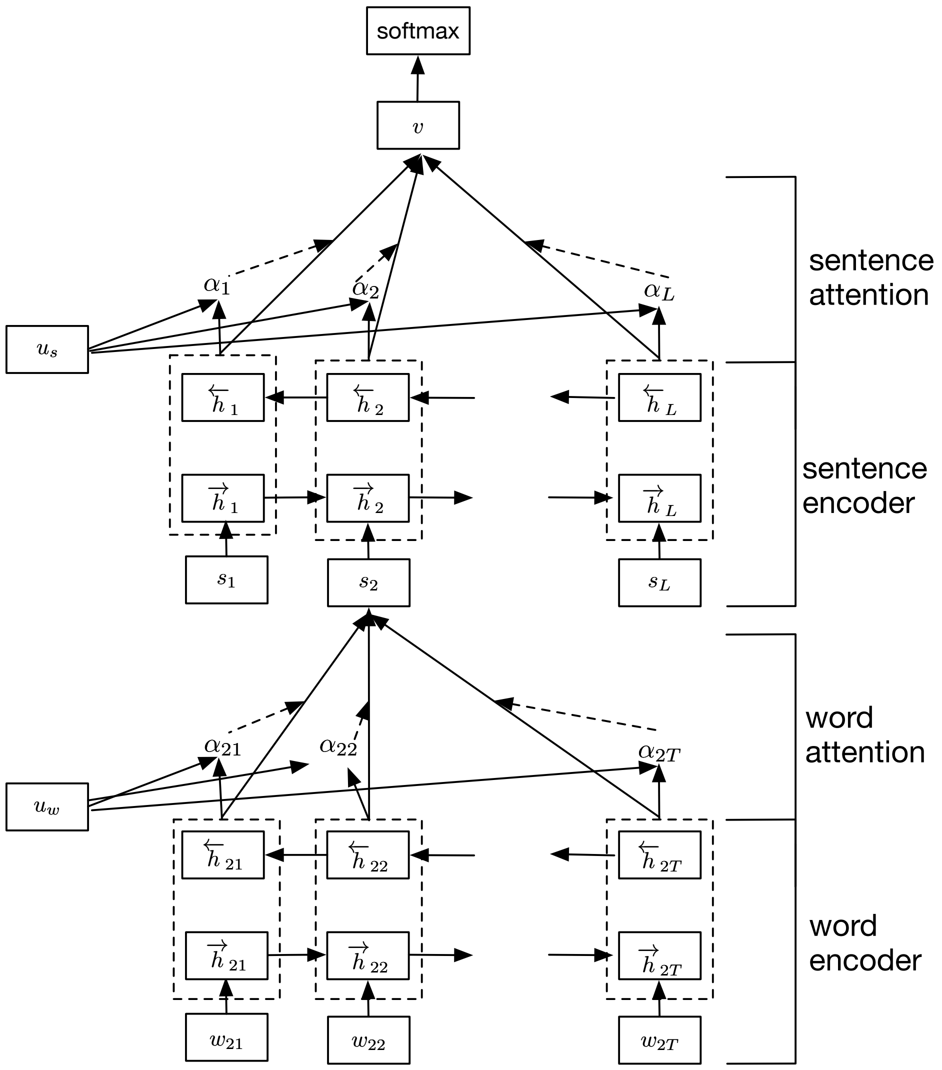


Figure 3.2: Hierarchical Attention Network.

Word Encoder Given a sentence with words $w_{it}, t \in [0, T]$, we first embed the words to vectors through an embedding matrix W_e, $x_{ij} = W_e w_{ij}$. We use a bidirectional GRU [8] to get annotations of words by summarizing information from both directions for words, and therefore incorporate the contextual information in the annotation. The bidirectional GRU contains the forward GRU $\overrightarrow{f}$ which reads the sentence s_i from w_{i1} to w_{iT} and a backward GRU $\overleftarrow{f}$ which reads from w_{iT} to w_{i1}:

$$\begin{aligned} x_{it} =& W_e w_{it}, t \in [1, T], \\ \overrightarrow{h}_{it} =& \overrightarrow{\mathrm{GRU}}(x_{it}), t \in [1, T], \\ \overleftarrow{h}_{it} =& \overleftarrow{\mathrm{GRU}}(x_{it}), t \in [T, 1]. \end{aligned}$$

We obtain an annotation for a given word w_{it} by concatenating the forward hidden state $\overrightarrow{h}_{it}$ and backward hidden state $\overleftarrow{h}_{it}$, i.e., $h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$, which summarizes the information of the whole sentence centered around w_{it}.

Note that we directly use word embeddings. For a more complete model we could use a GRU to get word vectors directly from characters, similarly to [79]. We omitted this for simplicity.

Word Attention Not all words contribute equally to the representation of the sentence meaning. Hence, we introduce attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. Specifically,

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{3.5}$$

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)} \tag{3.6}$$

$$s_i = \sum_t \alpha_{it} h_{it}. \tag{3.7}$$

That is, we first feed the word annotation h_{it} through a one-layer MLP to get u_{it} as a hidden representation of h_{it}, then we measure the importance of the word as the similarity of u_{it} with a word level context vector u_w and get a normalized importance weight α_{it} through a softmax function. After that, we compute the sentence vector s_i (we abuse the notation here) as a weighted sum of the word annotations based on the weights. The context vector u_w can be seen as a high level representation of a fixed query "what is the informative word" over the words like that used in memory networks [66, 109]. The word context vector u_w is randomly initialized and jointly learned during the training process.

Sentence Encoder Given the sentence vectors s_i, we can get a document vector in a similar way. We use a bidirectional GRU to encode the sentences:

$$\begin{aligned} \overrightarrow{h}_i =& \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L], \\ \overleftarrow{h}_i =& \overleftarrow{\mathrm{GRU}}(s_i), t \in [L, 1]. \end{aligned}$$

We concatenate $\overrightarrow{h}_i$ and $\overleftarrow{h}_j$ to get an annotation of sentence i, i.e., $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$. h_i summarizes the neighbor sentences around sentence i but still focus on sentence i.

Sentence Attention To reward sentences that are clues to correctly classify a document, we again use attention mechanism and introduce a sentence level context vector u_s and use the vector to measure the importance of the sentences. This yields

$$u_i = \tanh(W_s h_i + b_s), \tag{3.8}$$

$$\alpha_i = \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)}, \tag{3.9}$$

$$v = \sum_i \alpha_i h_i, \tag{3.10}$$

where v is the document vector that summarizes all the information of sentences in a document. Similarly, the sentence level context vector can be randomly initialized and jointly learned during the training process.

3.3.3 Document Classification

The document vector v is a high level representation of the document and can be used as features for document classification:

$$p = \text{softmax}(W_c v + b_c). \tag{3.11}$$

We use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_d \log p_{dj}, \tag{3.12}$$

where j is the label of document d.

3.4 Experiments

3.4.1 Data sets

We evaluate the effectiveness of our model on six large scale document classification data sets. These data sets can be categorized into two types of document classification tasks: sentiment estimation and topic classification. The statistics of the data sets are summarized in Table 4.1. We use 80% of the data for training, 10% for validation, and the remaining 10% for test, unless stated otherwise.

Yelp reviews are obtained from the Yelp Dataset Challenge in 2013, 2014 and 2015 [115]. There are five levels of ratings from 1 to 5 (higher is better).

IMDB reviews are obtained from [23]. The ratings range from 1 to 10.

Data set	classes	documents	average #s	max #s	average #w	max #w	vocabulary
Yelp 2013	5	335,018	8.9	151	151.6	1184	211,245
Yelp 2014	5	1,125,457	9.2	151	156.9	1199	476,191
Yelp 2015	5	1,569,264	9.0	151	151.9	1199	612,636
IMDB review	10	348,415	14.0	148	325.6	2802	115,831
Yahoo Answer	10	1,450,000	6.4	515	108.4	4002	1,554,607
Amazon review	5	3,650,000	4.9	99	91.9	596	1,919,336

Table 3.1: Data statistics: #s denotes the number of sentences (average and maximum per document), #w denotes the number of words (average and maximum per document).

Yahoo answers are obtained from [139]. This is a topic classification task with 10 classes: Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationships and Politics & Government. The document we use includes question titles, question contexts and best answers. There are 140,000 training samples and 5000 testing samples. The original data set does not provide validation samples. We randomly select 10% of the training samples as validation.

Amazon reviews are obtained from [139]. The ratings are from 1 to 5. 3,000,000 reviews are used for training and 650,000 reviews for testing. Similarly, we use 10% of the training samples as validation.

3.4.2 Baselines

We compare HAN with several baseline methods, including traditional approaches such as linear methods, SVMs and paragraph embeddings using neural networks, LSTMs, word-based CNN, character-based CNN, and Conv-GRNN, LSTM-GRNN. These baseline methods and results are reported in [115, 139].

Linear methods

Linear methods [139] use the constructed statistics as features. A linear classifier based on multinomial logistic regression is used to classify the documents using the features.

BOW and BOW+TFIDF The 50,000 most frequent words from the training set are selected and the count of each word is used features. Bow+TFIDF, as implied by the name, uses the TFIDF of counts as features.

n-grams and n-grams+TFIDF used the most frequent 500,000 n-grams (up to 5-grams).

Bag-of-means The average word2vec embedding [91] is used as feature set.

SVMs

SVMs-based methods are reported in [115], including **SVM+Unigrams, Bigrams, Text Features, AverageSG, SSWE**. In detail, **Unigrams** and **Bigrams** uses bag-of-unigrams and bag-of-bigrams as features respectively.

Text Features are constructed according to [62], including word and character n-grams, sentiment lexicon features etc.

AverageSG constructs 200-dimensional word vectors using word2vec and the average word embeddings of each document are used.

SSWE uses sentiment specific word embeddings according to [114].

Neural Network methods

The neural network based methods are reported in [115] and [139].

CNN-word Word based CNN models like that of [56] are used.

CNN-char Character level CNN models are reported in [139].

LSTM takes the whole document as a single sequence and the average of the hidden states of all words is used as feature for classification.

Conv-GRNN and LSTM-GRNN were proposed by [115]. They also explore the hierarchical structure: a CNN or LSTM provides a sentence vector, and then a gated recurrent neural network (GRNN) combines the sentence vectors from a document level vector representation for classification.

3.4.3 Model configuration and training

We split documents into sentences and tokenize each sentence using Stanford's CoreNLP [87]. We only retain words appearing more than 5 times in building the vocabulary and replace the words that appear 5 times with a special UNK token. We obtain the word embedding by training an unsupervised word2vec [91] model on the training and validation splits and then use the word embedding to initialize W_e.

The hyper parameters of the models are tuned on the validation set. In our experiments, we set the word embedding dimension to be 200 and the GRU dimension to be 50. In this case a combination of forward and backward GRU gives us 100 dimensions for word/sentence annotation. The word/sentence context vectors also have a dimension of 100, initialized at random.

For training, we use a mini-batch size of 64 and documents of similar length (in terms of the number of sentences in the documents) are organized to be a batch. We find that length-adjustment can accelerate training by three times. We use stochastic gradient descent to train all models with momentum of 0.9. We pick the best learning rate using grid search on the validation set.

3.4.4 Results and analysis

The experimental results on all data sets are shown in Table 3.2. We refer to our models as **HN-**$\{$**AVE, MAX, ATT**$\}$. Here HN stands for Hierarchical Network, AVE indicates averaging, MAX indicates max-pooling, and ATT indicates our proposed hierarchical attention model. Results show that HN-ATT gives the best performance across all data sets.

The improvement is regardless of data sizes. For smaller data sets such as Yelp 2013 and IMDB, our model outperforms the previous best baseline methods by 3.1% and 4.1% respectively. This finding is consistent across other larger data sets. Our model outperforms previous

	Methods	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo Answer	Amazon
Zhang et al., 2015	BoW	-	-	58.0	-	68.9	54.4
	BoW TFIDF	-	-	59.9	-	71.0	55.3
	ngrams	-	-	56.3	-	68.5	54.3
	ngrams TFIDF	-	-	54.8	-	68.5	52.4
	Bag-of-means	-	-	52.5	-	60.5	44.1
Tang et al., 2015	Majority	35.6	36.1	36.9	17.9	-	-
	SVM + Unigrams	58.9	60.0	61.1	39.9	-	-
	SVM + Bigrams	57.6	61.6	62.4	40.9	-	-
	SVM + TextFeatures	59.8	61.8	62.4	40.5	-	-
	SVM + AverageSG	54.3	55.7	56.8	31.9	-	-
	SVM + SSWE	53.5	54.3	55.4	26.2	-	-
Zhang et al., 2015	LSTM	-	-	58.2	-	70.8	59.4
	CNN-char	-	-	62.0	-	71.2	59.6
	CNN-word	-	-	60.5	-	71.2	57.6
Tang et al., 2015	Paragraph Vector	57.7	59.2	60.5	34.1	-	-
	CNN-word	59.7	61.0	61.5	37.6	-	-
	Conv-GRNN	63.7	65.5	66.0	42.5	-	-
	LSTM-GRNN	65.1	67.1	67.6	45.3	-	-
This work	HN-AVE	67.0	69.3	69.9	47.8	75.2	62.9
	HN-MAX	66.9	69.3	70.1	48.2	75.2	62.9
	HN-ATT	**68.2**	**70.5**	**71.0**	**49.4**	**75.8**	**63.6**

Table 3.2: Document Classification results, in percentage

best models by 3.2%, 3.4%, 4.6% and 6.0% on Yelp 2014, Yelp 2015, Yahoo Answers and Amazon Reviews. The improvement also occurs regardless of the type of task: sentiment classification, which includes Yelp 2013-2014, IMDB, Amazon Reviews and topic classification for Yahoo Answers.

From Table 3.2 we can see that neural network based methods that do *not* explore hierarchical document structure, such as LSTM, CNN-word, CNN-char have little advantage over traditional methods for large scale (in terms of document size) text classification. E.g. SVM+TextFeatures gives performance 59.8, 61.8, 62.4, 40.5 for Yelp 2013, 2014, 2015 and IMDB respectively, while CNN-word has accuracy 59.7, 61.0, 61.5, 37.6 respectively.

Exploring the hierarchical structure only, as in HN-AVE, HN-MAX, can significantly improve over LSTM, CNN-word and CNN-char. For example, our HN-AVE outperforms CNN-word by 7.3%, 8.8%, 8.5%, 10.2% than CNN-word on Yelp 2013, 2014, 2015 and IMDB respectively. Our model HN-ATT that further utilizes attention mechanism combined with hierarchical structure improves over previous models (LSTM-GRNN) by 3.1%, 3.4%, 3.5% and 4.1% respectively. More interestingly, in the experiments, HN-AVE is equivalent to using non-informative global word/sentence context vectors (e.g., if they are all-zero vectors, then the attention weights in Eq. 3.6 and 3.9 become uniform weights). Compared to HN-AVE, the HN-ATT model gives superior performance across the board. This clearly demonstrates the effectiveness of the proposed global word and sentence importance vectors for the HAN.

3.4.5 Context dependent attention weights

If words were inherently important or not important, models without attention mechanism might work well since the model could automatically assign low weights to irrelevant words and vice versa. However, the importance of words is highly context dependent. For example, the word `good` may appear in a review that has the lowest rating either because users are only happy with part of the product/service or because they use it in a negation, such as `not good`. To verify that our model can capture context dependent word importance, we plot the distribution of the attention weights of the words `good` and `bad` from the test split of Yelp 2013 data set as shown in Figure 3.3(a) and Figure 3.4(a). We can see that the distribution has a attention weight assigned to a word from 0 to 1. This indicates that our model captures diverse context and assign context-dependent weight to the words.

For further illustration, we plot the distribution when conditioned on the ratings of the review. Subfigures 3.3(b)-(f) in Figure 3.3 and Figure 3.4 correspond to the rating 1-5 respectively. In particular, Figure 3.3(b) shows that the weight of `good` concentrates on the low end in the reviews with rating 1. As the rating increases, so does the weight distribution. This means that the word `good` plays a more important role for reviews with higher ratings. We can observe the converse trend in Figure 3.4 for the word `bad`. This confirms that our model can capture the context-dependent word importance.

3.4.6 Visualization of attention

In order to validate that our model is able to select informative sentences and words in a document, we visualize the hierarchical attention layers in Figures 3.5 and 3.6 for several documents from the Yelp 2013 and Yahoo Answers data sets.

Every line is a sentence (sometimes sentences spill over several lines due to their length). Red denotes the sentence weight and blue denotes the word weight. Due to the hierarchical structure, we normalize the word weight by the sentence weight to make sure that only important words in important sentences are emphasized. For visualization purposes we display $\sqrt{p_s}p_w$. The $\sqrt{p_s}$ term displays the important words in unimportant sentences to ensure that they are not totally invisible.

Figure 3.5 shows that our model can select the words carrying strong sentiment like `delicious, amazing, terrible` and their corresponding sentences. Sentences containing many words like `cocktails, pasta, entree` are disregarded. Note that our model can not only select words carrying strong sentiment, it can also deal with complex across-sentence context. For example, there are sentences like `i don't even like scallops` in the first document of Fig. 3.5, if looking purely at the single sentence, we may think this is negative comment. However, our model looks at the context of this sentence and figures out this is a positive review and chooses to ignore this sentence.

Our hierarchical attention mechanism also works well for topic classification in the Yahoo Answer data set. For example, for the left document in Figure 3.6 with label 1, which de-

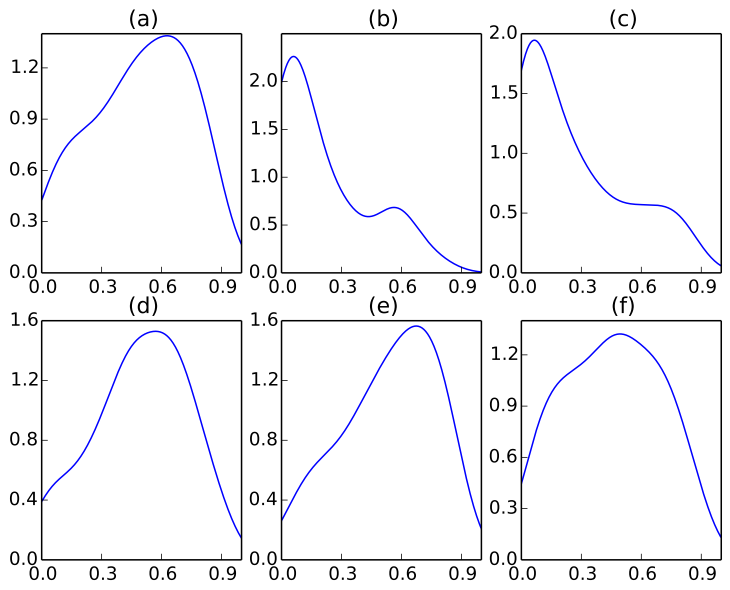


Figure 3.3: Attention weight distribution of `good`. (a) — aggregate distribution on the test split; (b)-(f) stratified for reviews with ratings 1-5 respectively. We can see that the weight distribution shifts to *higher* end as the rating goes higher.

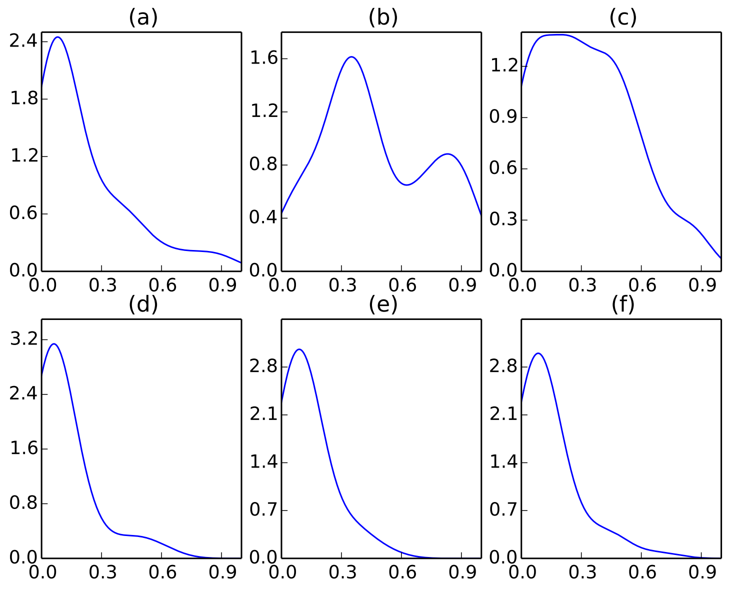
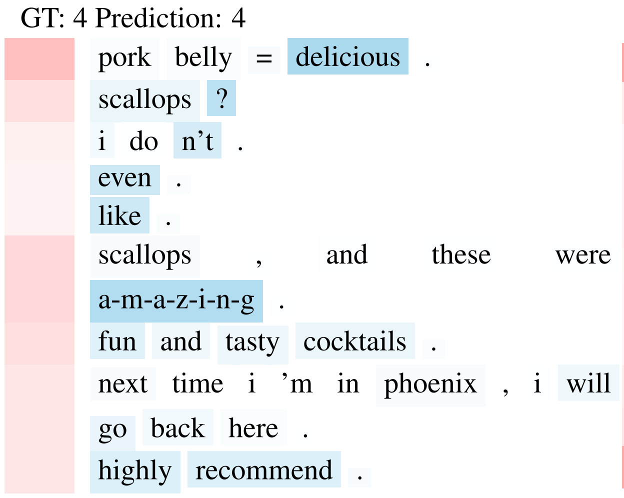
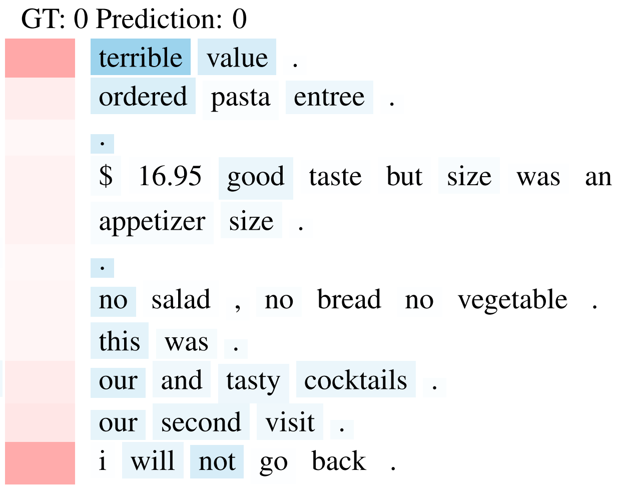


Figure 3.4: Attention weight distribution of the word `bad`. The setup is as above: (a) contains the aggregate distribution, while (b)-(f) contain stratifications to reviews with ratings 1-5 respectively. Contrary to before, the word `bad` is considered important for poor ratings and less so for good ones.

notes Science and Mathematics, our model accurately localizes the words `zebra, strips, camouflage, predator` and their corresponding sentences. For the right document with label 4, which denotes Computers and Internet, our model focuses on `web, searches,` `browsers` and their corresponding sentences. Note that this happens in a *multiclass* setting, that is, detection happens before the selection of the topic!

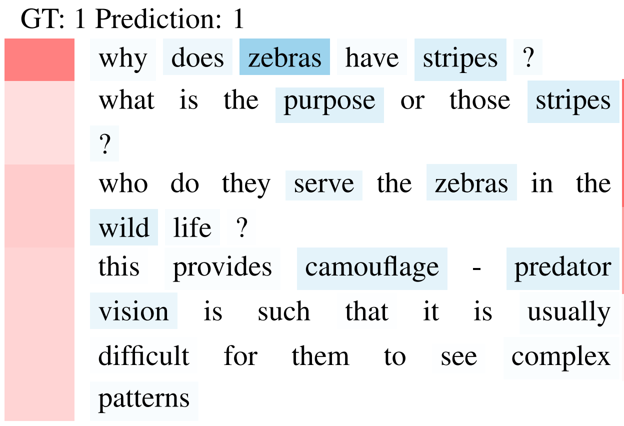

GT: 4 Prediction: 4

pork belly = delicious .
scallops ?
i do n't .
even .
like .
scallops , and these were
a-m-a-z-i-n-g .
fun and tasty cocktails .
next time i 'm in phoenix , i will
go back here .
highly recommend .

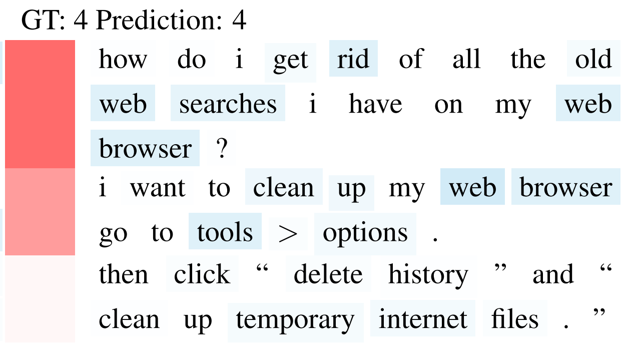

GT: 0 Prediction: 0

terrible value .
ordered pasta entree .
.
$ 16.95 good taste but size was an
appetizer size .
.
no salad , no bread no vegetable .
this was .
our and tasty cocktails .
our second visit .
i will not go back .

Figure 3.5: Documents from Yelp 2013. Label 4 means star 5, label 0 means star 1.


GT: 1 Prediction: 1

why does zebras have stripes ?
what is the purpose or those stripes
?
who do they serve the zebras in the
wild life ?
this provides camouflage - predator
vision is such that it is usually
difficult for them to see complex
patterns


GT: 4 Prediction: 4

how do i get rid of all the old
web searches i have on my web
browser ?
i want to clean up my web browser
go to tools > options .
then click " delete history " and "
clean up temporary internet files . "

Figure 3.6: Documents from Yahoo Answers. Label 1 denotes Science and Mathematics and label 4 denotes Computers and Internet.

3.5 Recent development

Our work is the first to apply attention mechanism on text classification problem. Since the publication of our work, it has attracted a lot of attention and the idea of the model has been adopted and applied in many other applications. For example, [116] applied our framework to aspect level sentiment classification. [93] generalized our model to the multilingual setting. [14] introduced a context vector for each user and then used user-product attention to predict the rating from each user. [96] applied the idea of our model to time series predictions. The authors from [137] adopted our idea to the task of referring expression comprehension.

Part II

Structural Bias for Unsupervised Learning

Chapter 4

Dilated CNN Decoders for VAEs

4.1 Overview

In this part, we are going to investigate generative models for text modeling, showing better model priors and learning objective makes generative models work better. Generative models play an important role in NLP, both in their use as language models and because of their ability to effectively learn from unlabeled data. By parameterzing generative models using neural nets, recent work has proposed model classes that are particularly expressive and can pontentially model a wide range of phenomena in language and other modalities. When designing generative models for NLP, there are some challenges due to its discrete and sequential structure. Hence the progress of the two classes of generative models—Variational Auto-Encoder (VAE) and Generative Adversarial Network (GAN)—on NLP is very limited. In this part of the thesis, we are going to investigate these problems and propose new models to overcome these difficulties.

In this chapter, we focus on a specific instance of this class: the variational autoencoder[1] (VAE) [59]. The generative story behind the VAE (to be described in detail in the next section) is simple: First, a continuous latent representation is sampled from a multivariate Gaussian. Then, an output is sampled from a distribution parameterized by a neural decoder, conditioned on the latent representation. The latent representation (treated as a latent variable during training) is intended to give the model more expressive capacity when compared with simpler neural generative models–for example, conditional language models. The choice of decoding architecture and final output distribution, which connect the latent representation to output, depends on the kind of data being modeled. The VAE owes its name to an accompanying variational technique [59] that has been successfully used to train such models on image data [35, 101, 128].

The application of VAEs to text data has been far less successful [11, 89]. The obvious choice for decoding architecture for a textual VAE is an LSTM, a typical workhorse in NLP. However, Bowman et al. [11] found that using an LSTM-VAE for text modeling yields higher perplexity on held-out data than using an LSTM language model. In particular, they observe that the LSTM decoder in VAE does not make effective use of the latent representation during training and, as a result, VAE collapses into a simple language model. Related work [70, 89, 92] has used simpler decoders that model text as a bag of words. Their results indicate better use of latent

[1]The name VAE is often used to refer to both a model class and an associated inference procedure.

representations, but their decoders cannot effectively model longer-range dependencies in text and thus underperform in terms of final perplexity.

Motivated by these observations, we hypothesize that the contextual capacity of the decoder plays an important role in whether VAEs effectively condition on the latent representation when trained on text data. We propose the use of a dilated CNN as a decoder in VAE, inspired by the recent success of using CNNs for audio, image and language modeling [53, 117, 118]. In contrast with prior work where extremely large CNNs are used, we exploit the dilated CNN for its flexibility in varying the amount of conditioning context. In the two extremes, depending on the choice of dilation, the CNN decoder can reproduce a simple MLP using a bags of words representation of text, or can reproduce the long-range dependence of recurrent architectures (like an LSTM) by conditioning on the entire history. Thus, by choosing a dilated CNN as the decoder, we are able to conduct experiments where we vary contextual capacity, finding a sweet spot where the decoder can accurately model text but does not yet overpower the latent representation.

We demonstrate that when this trade-off is correctly managed, textual VAEs can perform substantially better than simple LSTM language models, a finding consistent with recent image modeling experiments using variational lossy autoencoders [17]. We go on to show that VAEs with carefully selected CNN decoders can be quite effective for semi-supervised classification and unsupervised clustering, outperforming several strong baselines (from [21]) on both text categorization and sentiment analysis.

In this chapter, we empirically evaluate several dilation architectures with different capacities, finding that reduced contextual capacity leads to stronger reliance on latent representations. By picking a decoder with suitable contextual capacity, we find our VAE performs better than LSTM language models on two data sets. We also explore the use of dilated CNN VAEs for semi-supervised classification and find they perform better than strong baselines from [21]. Finally, we verify that the same framework can be used effectively for unsupervised clustering.

4.2 Related work

Variational inference via the re-parameterization trick was initially proposed by [59, 100] and since then, VAE has been widely adopted as generative model for images [35, 36, 45, 101, 128].

Our work is in line with previous works on combining variational inferences with text modeling [11, 43, 89, 103, 138]. [11] is the first work to combine VAE with language model and they use LSTM as the decoder and find some negative results. On the other hand, [89] models text as bag of words, though improvement has been found, the model can not be used to generate text. Our work fills the gaps between them. [103, 138] applies variational inference to dialogue modeling and machine translation and found some improvement in terms of generated text quality, but no language modeling results are reported. [9, 20, 27] embedded variational units in every step of a RNN, which is different from our model in using global latent variables to learn high level features.

Our use of CNN as decoder is inspired by recent success of PixelCNN model for images [118], WaveNet for audios [117], Video Pixel Network for video modeling [54] and ByteNet for machine translation [53]. But in contrast to those works showing using a very deep architecture leads to better performance, CNN as decoder is used in our model to control the contextual capacity, leading to better performance.

Our work is closed related the recently proposed variational lossy autoencoder [17] which is used to predict image pixels. They find that conditioning on a smaller window of a pixels leads to better results with VAE, which is similar to our finding. Much [17, 61, 99] has been done to come up more powerful prior/posterior distribution representations with techniques such as normalizing flows. We treat this as one of our future works. This work is largely orthogonal and could be potentially combined with a more effective choice of decoder to yield additional gains.

There is much previous work exploring unsupervised sentence encodings, for example skip-thought vectors [64], paragraph vectors [71], and sequence autoencoders [21]. [21] applies a pretrained model to semi-supervised classification and find significant gains, we use this as the baseline for our semi-supervised VAE.

4.3 Model

In this section, we begin by providing background on the use of variational autoencoders for language modeling. Then we introduce the dilated CNN architecture that we will use as a new decoder for VAE in experiments. Finally, we describe the generalization of VAE that we will use to conduct experiments on semi-supervised classification.

4.3.1 Background on Variational Autoencoders

Neural language models [90] typically generate each token x_t conditioned on the entire history of previously generated tokens:

$$p(\mathbf{x}) = \prod_t p(x_t|x_1, x_2, ..., x_{t-1}). \tag{4.1}$$

State-of-the-art language models often parametrize these conditional probabilities using RNNs, which compute an evolving hidden state over the text which is used to predict each x_t. This approach, though effective in modeling text, does not explicitly model variance in higher-level properties of entire utterances (e.g. topic or style) and thus can have difficulty with heterogeneous datasets.

Bowman et al. [11] propose a different approach to generative text modeling inspired by related work on vision [59]. Instead of directly modeling the joint probability $p(\mathbf{x})$ as in Equation 4.1, we specify a generative process for which $p(\mathbf{x})$ is a marginal distribution. Specifically, we first generate a continuous latent vector representation $\mathbf{z}$ from a multivariate Gaussian prior $p_\theta(\mathbf{z})$, and then generate the text sequence $\mathbf{x}$ from a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ parameterized using a neural net (often called the generation model or decoder). Because this model incorporates a latent variable that modulates the entire generation of each whole utterance, it may be better able to capture high-level sources of variation in the data. Specifically, in contrast with Equation 4.1, this generating distribution conditions on latent vector representation $\mathbf{z}$:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_t p_\theta(x_t|x_1, x_2, ..., x_{t-1}, \mathbf{z}). \tag{4.2}$$

To estimate model parameters θ we would ideally like to maximize the marginal probability $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z}$. However, computing this marginal is intractable for many decoder choices. Thus, the following variational lower bound is often used as an objective [59]:

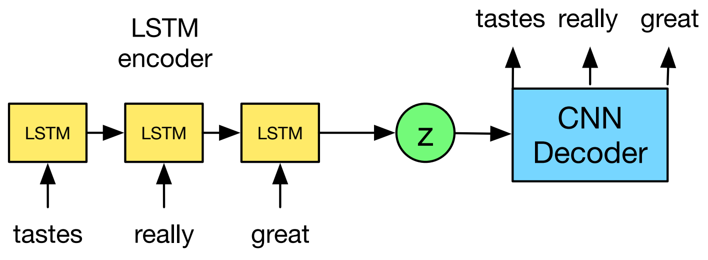
$$\begin{aligned} \log\ p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z} \\ &\geq\ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \end{aligned}$$

Here, $q_\phi(\mathbf{z}|\mathbf{x})$ is an approximation to the true posterior (often called the recognition model or encoder) and is parameterized by ϕ. Like the decoder, we have a choice of neural architecture to parameterize the encoder. However, unlike the decoder, the choice of encoder does not change the model class – it only changes the variational approximation used in training, which is a function of both the model parameters θ *a*nd the approximation parameters ϕ. Training seeks to optimize these parameters jointly using stochastic gradient ascent. A final wrinkle of the training procedure involves a stochastic approximation to the gradients of the variational objective (which is itself intractable). We omit details here, noting only that the final distribution of the posterior approximation $q_\phi(\mathbf{z}|\mathbf{x})$ is typically assumed to be Gaussian so that a re-parametrization trick can be used, and refer readers to [59].

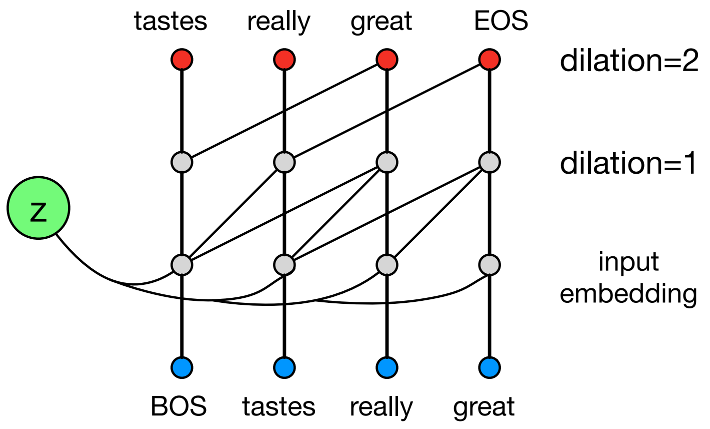
4.3.2 Training Collapse with Textual VAEs

Together, this combination of generative model and variational inference procedure are often referred to as a variational autoencoder (VAE). We can also view the VAE as a regularized version of the autoencoder. Note, however, that while VAEs are valid probabilistic models whose likelihood can be evaluated on held-out data, autoencoders are not valid models. If only the first term of the VAE variational bound $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is used as an objective, the variance of the posterior probability $q_\phi(\mathbf{z}|\mathbf{x})$ will become small and the training procedure reduces to an autoencoder. It is the KL-divergence term, $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$, that discourages the VAE memorizing each $\mathbf{x}$ as a single latent point.

While the KL term is critical for training VAEs, historically, instability on text has been evidenced by the KL term becoming vanishingly small during training, as observed by Bowman et al. [11]. When the training procedure collapses in this way, the result is an encoder that has duplicated the Gaussian prior (instead of a more interesting posterior), a decoder that completely ignores the latent variable $\mathbf{z}$, and a learned model that reduces to a simpler language model. We hypothesize that this collapse condition is related to the contextual capacity of the decoder architecture. The choice encoder and decoder depends on the type of data. For images, these are typically MLPs or CNNs. LSTMs have been used for text, but have resulted in training collapse as discussed above [11]. Here, we propose to use a dilated CNN as the decoder instead. In one extreme, when the effective contextual width of a CNN is very large, it resembles the behavior of LSTM. When the width is very small, it behaves like a bag-of-words model. The architectural flexibility of dilated CNNs allows us to change the contextual capacity and conduct experiments to validate our hypothesis: decoder contextual capacity and effective use of encoding information are directly related. We next describe the details of our decoder.



(a) VAE training graph using a dilated CNN decoder.



(b) Digram of dilated CNN decoder.

Figure 4.1: Our training and model architectures for textual VAE using a dilated CNN decoder.

4.3.3 Dilated Convolutional Decoders

The typical approach to using CNNs used for text generation [53] is similar to that used for images [40, 65], but with the convolution applied in one dimension. We take this approach here in defining our decoder.

One dimensional convolution: For a CNN to serve as a decoder for text, generation of x_t must only condition on past tokens $x_{<t}$. Applying the traditional convolution will break this assumption and use tokens $x_{\geq t}$ as inputs to predict x_t. In our decoder, we avoid this by simply shifting the input by several slots [118]. With a convolution with filter size of k and using n layers, our effective filter size (the number of past tokens to condition to in predicting x_t) would be $(k-1) \times n + 1$. Hence, the filter size would grow linearly with the depth of the network.

Dilation: Dilated convolution [135] was introduced to greatly increase the effective receptive field size without increasing the computational cost. With dilation d, the convolution is applied so that $d-1$ inputs are skipped each step. Causal convolution can be seen a special case with $d=1$. With dilation, the effective receptive size grows exponentially with network depth. In Figure 4.1b, we show dilation of sizes of 1 and 2 in the first and second layer, respectively. Suppose the dilation size in the i-th layer is d_i and we use the same filter size k in all layers, then the effective filter size is $(k-1)\sum_i d_i + 1$. The dilations are typically set to double every layer

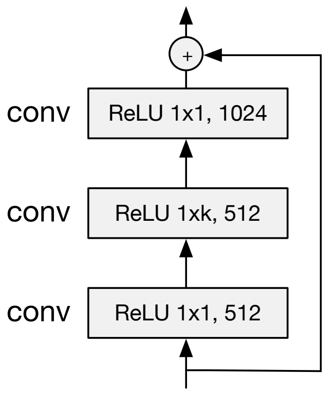


Figure 4.2: Residual connection for dilated CNN decoders.

$d_{i+1} = 2d_i$, so the effective receptive field size can grow exponentially. Hence, the contextual capacity of a CNN can be controlled across a greater range by manipulating the filter size, dilation size and network depth. We use this approach in experiments.

Residual connection: We use residual connection [40] in the decoder to speed up convergence and enable training of deeper models. We use a residual block (shown to the right) similar to that of [53]. We use three convolutional layers with filter size $1 \times 1, 1 \times k, 1 \times 1$, respectively, and ReLU activation between convolutional layers.

Overall architecture: Our VAE architecture is shown in Figure 4.1a. We use LSTM as the encoder to get the posterior probability $q(\mathbf{z}|\mathbf{x})$, which we assume to be diagonal Gaussian. We parametrize the mean μ and variance σ with LSTM output. We sample $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{x})$, the decoder is conditioned on the sample by concatenating $\mathbf{z}$ with every word embedding of the decoder input.

4.3.4 Semi-supervised VAE

In addition to conducting language modeling experiments, we will also conduct experiments on semi-supervised classification of text using our proposed decoder. In this section, we briefly review semi-supervised VAEs of [60] that incorporate discrete labels as additional variables. Given the labeled set $(x, y) \sim D_L$ and the unlabeled set $x \sim D_U$, [60] proposed a model whose latent representation contains continuous vector $\mathbf{z}$ and discrete label $\mathbf{y}$:

$$p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{y})p(\mathbf{z})p(\mathbf{x}|\mathbf{y}, \mathbf{z}). \tag{4.3}$$

The semi-supervised VAE fits a discriminative network $q(\mathbf{y}|\mathbf{x})$, an inference network $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and a generative network $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$ jointly as part of optimizing a variational lower bound similar that of basic VAE. For labeled data $(\mathbf{x}, \mathbf{y})$, this bound is:

$$\begin{aligned}\log p(\mathbf{x}, \mathbf{y}) \geq & \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y}, \mathbf{z})] \\ & - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z})) + \log p(\mathbf{y}) \\ = & L(\mathbf{x}, \mathbf{y}) + \log p(\mathbf{y}).\end{aligned}$$

For unlabeled data $\mathbf{x}$, the label is treated as a latent variable, yielding:

$$\begin{aligned}\log p(\mathbf{x}) \geq & U(\mathbf{x}) \\ = & \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \big[\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})] \\ & - \mathrm{KL}(q(\mathbf{z}|\mathbf{x},\mathbf{y})||p(\mathbf{z})) + \log p(\mathbf{y}) - \log q(\mathbf{y}|\mathbf{x}) \big] \\ = & \sum_y q(\mathbf{y}|\mathbf{x}) L(\mathbf{x},\mathbf{y}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y})).\end{aligned}$$

Combining the labeled and unlabeled data terms, we have the overall objective as:

$$\begin{aligned}J = & \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D_L}[L(\mathbf{x},\mathbf{y})] + \mathbb{E}_{\mathbf{x}\sim D_U}[U(\mathbf{x})] \\ & + \alpha\, \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D_L}[\log q(\mathbf{y}|\mathbf{x})],\end{aligned}$$

where α controls the trade off between generative and discriminative terms.

Gumbel-softmax: Jang et al. [50], Maddison et al. [83] propose a continuous approximation to sampling from a categorical distribution. Let u be a categorical distribution with probabilities $\pi_1, \pi_2, ..., \pi_c$. Samples from u can be approximated using:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{c} \exp((\log(\pi_j) + g_j)/\tau)}, \tag{4.4}$$

where g_i follows Gumbel(0, 1). The approximation is accurate when $\tau \to 0$ and smooth when $\tau > 0$. In experiments, we use Gumbel-Softmax to approximate the samples from $p(\mathbf{y}|\mathbf{x})$ to reduce the computational cost. As a result, we can directly back propagate the gradients of $U(\mathbf{x})$ to the discriminator network. We anneal τ so that sample variance is small when training starts and then gradually decrease τ.

Unsupervised clustering: In this section we adapt the same framework for unsupervised clustering. We directly minimize the objective $U(\mathbf{x})$, which is consisted of two parts: reconstruction loss and KL regularization on $q(\mathbf{y}|\mathbf{x})$. The first part encourages the model to assign $\mathbf{x}$ to label $\mathbf{y}$ such that the reconstruction loss is low. We find that the model can easily get stuck in two local optimum: the KL term is very small and $q(\mathbf{y}|\mathbf{x})$ is close to uniform distribution or the KL term is very large and all samples collapse to one class. In order to make the model more robust, we modify the KL term by:

$$\mathrm{KL}_{\mathbf{y}} = \max(\gamma, \mathrm{KL}(q(\mathbf{y}|\mathbf{x})|p(\mathbf{y})). \tag{4.5}$$

That is, we only minimize the KL term when it is large enough.

4.4 Experiments

4.4.1 Data sets

Since we would like to investigate VAEs for language modeling and semi-supervised classification, the data sets should be suitable for both purposes. We use two large scale document classification data sets: Yahoo Answer and Yelp15 review, representing topic classification and sentiment classification data sets respectively [115, 132, 139]. The original data sets contain millions of samples, of which we sample 100k as training and 10k as validation and test from the respective partitions. The detailed statistics of both data sets are in Table 4.1. Yahoo Answer contains 10 topics including Society & Culture, Science & Mathematics etc. Yelp15 contains 5 level of rating, with higher rating better.

4.4.2 Model configurations and Training details

We use an LSTM as an encoder for VAE and explore LSTMs and CNNs as decoders. For CNNs, we explore several different configurations. We set the convolution filter size to be 3 and gradually increase the depth and dilation from [1, 2, 4], [1, 2, 4, 8, 16] to [1, 2, 4, 8, 16, 1, 2, 4, 8, 16]. They represent small, medium and large model and we name them as SCNN, MCNN and LCNN. We also explore a very large model with dilations [1, 2, 4, 8, 16, 1, 2, 4, 8, 16, 1, 2, 4, 8, 16] and name it as VLCNN. The effective filter size are 15, 63, 125 and 187 respectively. We use the last hidden state of the encoder LSTM and feed it though an MLP to get the mean and variance of $q(\mathbf{z}|\mathbf{x})$, from which we sample $\mathbf{z}$ and then feed it through an MLP to get the starting state of decoder. For the LSTM decoder, we follow [11] to use it as the initial state of LSTM and feed it to every step of LSTM. For the CNN decoder, we concatenate it with the word embedding of every decoder input.

The architecture of the Semi-supervised VAE basically follows that of the VAE. We feed the last hidden state of the encoder LSTM through a two layer MLP then a softmax to get $q(\mathbf{y}|\mathbf{x})$. We use Gumbel-softmax to sample $\mathbf{y}$ from $q(\mathbf{y}|\mathbf{x})$. We then concatenate $\mathbf{y}$ with the last hidden state of encoder LSTM and feed them throught an MLP to get the mean and variance of $q(\mathbf{z}|\mathbf{y}, \mathbf{x})$. $\mathbf{y}$ and $\mathbf{z}$ together are used as the starting state of the decoder.

We use a vocabulary size of 20k for both data sets and set the word embedding dimension to be 512. The LSTM dimension is 1024. The number of channels for convolutions in CNN decoders is 512 internally and 1024 externally, as shown in Section 4.3.3. We select the dimension of $\mathbf{z}$ from [32, 64]. We find our model is not sensitive to this parameter.

We use Adam [58] to optimize all models and the learning rate is selected from [2e-3, 1e-3, 7.5e-4] and β_1 is selected from [0.5, 0.9]. Empirically, we find learning rate 1e-3 and $\beta_1 = 0.5$ to perform the best. We select drop out ratio of LSTMs (both encoder and decoder) from [0.3, 0.5]. Following [11], we also use drop word for the LSTM decoder, the drop word ratio is selected from [0, 0.3, 0.5, 0.7]. For the CNN decoder, we use a drop out ratio of 0.1 at each layer. We do not use drop word for CNN decoders. We use batch size of 32 and all model are trained for 40 epochs. We start to half the learning rate every 2 epochs after epoch 30. Following [11], we use KL cost annealing strategy. We set the initial weight of KL cost term to be 0.01 and increase it

Data	classes	documents	average #w	vocabulary
Yahoo	10	100k	78	200k
Yelp15	5	100k	96	90k

Table 4.1: Data statistics for experiments of VAEs

Model	Size	NLL (KL)	PPL
LSTM-LM	$< i$	334.9	66.2
LSTM-VAE**	$< i$	342.1 (0.0)	72.5
LSTM-VAE** + init	$< i$	339.2 (0.0)	69.9
SCNN-LM	15	345.3	75.5
SCNN-VAE	15	337.8 (13.3)	68.7
SCNN-VAE + init	15	335.9 (13.9)	67.0
MCNN-LM	63	338.3	69.1
MCNN-VAE	63	336.2 (11.8)	67.3
MCNN-VAE + init	63	334.6 (12.6)	66.0
LCNN-LM	125	335.4	66.6
LCNN-VAE	125	333.9 (6.7)	65.4
LCNN-VAE + init	125	**332.1 (10.0)**	**63.9**
VLCNN-LM	187	336.5	67.6
VLCNN-VAE	187	336.5 (0.7)	67.6
VLCNN-VAE + init	187	335.8 (3.8)	67.0

(a) Yahoo

Model	Size	NLL (KL)	PPL
LSTM-LM	$< i$	362.7	42.6
LSTM-VAE**	$< i$	372.2 (0.3)	47.0
LSTM-VAE** + init	$< i$	368.9 (4.7)	46.4
SCNN-LM	15	371.2	46.6
SCNN-VAE	15	365.6 (9.4)	43.9
SCNN-VAE + init	15	363.7 (10.3)	43.1
MCNN-LM	63	366.5	44.3
MCNN-VAE	63	363.0 (6.9)	42.8
MCNN-VAE + init	63	360.7 (9.1)	41.8
LCNN-LM	125	363.5	43.0
LCNN-VAE	125	361.9 (6.4)	42.3
LCNN-VAE + init	125	**359.1 (7.6)**	**41.1**
VLCNN-LM	187	364.8	43.7
VLCNN-VAE	187	364.3 (2.7)	43.4
VLCNN-VAE + init	187	364.7 (2.2)	43.5

(b) Yelp

Table 4.2: Language modeling results on the test set. ** is from [11]. We report negative log likelihood (NLL) and perplexity (PPL) on the test set. The KL component of NLL is given in parentheses. Size indicates the effective filter size. VAE + init indicates pretraining of only the encoder using an LSTM LM.

linearly until a given iteration T. We treat T as a hyper parameter and select it from [10k, 40k, 80k].

4.4.3 Language modeling results

The results for language modeling are shown in Table 4.2. We report the negative log likelihood (NLL) and perplexity (PPL) of the test set. For the NLL of VAEs, we decompose it into reconstruction loss and KL divergence and report the KL divergence in the parenthesis. To better visualize these results, we plot the results of Yahoo data set (Table 4.2a) in Figure 4.3.

We first look at the LM results for Yahoo data set. As we gradually increase the effective filter size of CNN from SCNN, MCNN to LCNN, the NLL decreases from 345.3, 338.3 to 335.4. The NLL of LCNN-LM is very close to the NLL of LSTM-LM 334.9. But VLCNN-LM is a little bit worse than LCNN-LM, this indicates a little bit of over-fitting.

We can see that LSTM-VAE is worse than LSTM-LM in terms of NLL and the KL term is nearly zero, which verifies the finding of [11]. When we use CNNs as the decoders for VAEs, we can see improvement over pure CNN LMs. For SCNN, MCNN and LCNN, the VAE results improve over LM results from 345.3 to 337.8, 338.3 to 336.2, and 335.4 to 333.9 respectively. The improvement is big for small models and gradually decreases as we increase the decoder model contextual capacity. When the model is as large as VLCNN, the improvement diminishes and the

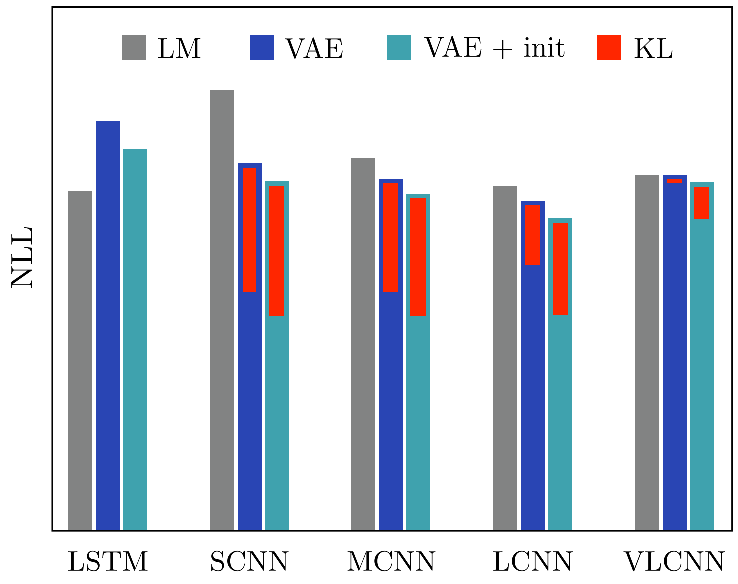


Figure 4.3: NLL decomposition of Table 4.2a. Each group consists of three bars, representing LM, VAE and VAE+init. For VAE, we decompose the loss into reconstruction loss and KL divergence, shown in blue and red respectively. We subtract all loss values with 300 for better visualization.

VAE result is almost the same with LM result. This is also reflected in the KL term, SCNN-VAE has the largest KL of 13.3 and VLCNN-VAE has the smallest KL of 0.7. When LCNN is used as the decoder, we obtain an optimal trade off between using contextual information and latent representation. LCNN-VAE achieves a NLL of 333.9, which improves over LSTM-LM with NLL of 334.9.

We find that if we initialize the parameters of *LSTM encoder* with parameters of LSTM language model, we can improve the VAE results further. This indicates better encoder model is also a key factor for VAEs to work well. Combined with encoder initialization, LCNN-VAE improves over LSTM-LM from 334.9 to 332.1 in NLL and from 66.2 to 63.9 in PPL. Similar results for the sentiment data set are shown in Table 4.2b. LCNN-VAE improves over LSTM-LM from 362.7 to 359.1 in NLL and from 42.6 to 41.1 in PPL.

Latent representation visualization: In order to visualize the latent representation, we set the dimension of $\mathbf{z}$ to be 2 and plot the mean of posterior probability $q(\mathbf{z}|\mathbf{x})$, as shown in Figure 4.4. We can see distinct different characteristics of topic and sentiment representation. In Figure 4.4a, we can see that documents of different topics fall into different clusters, while in Figure 4.4b, documents of different ratings form a continuum, they lie continuously on the x-axis as the review rating increases.

4.4.4 Semi-supervised VAE results

Motivated by the success of VAEs for language modeling, we continue to explore VAEs for semi-supervised learning. Following that of [60], we set the number of labeled samples to be

Model	ACCU	NLL (KL)
LSTM-VAE-Semi	51.9	345.5 (9.3)
SCNN-VAE-Semi	**65.5**	335.7 (10.4)
MCNN-VAE-Semi	64.6	332.8 (7.2)
LCNN-VAE-Semi	57.2	**331.3** (2.7)

Table 4.3: Semi-supervised VAE ablation results on Yahoo. We report both the NLL and classification accuracy of the test data. Accuracy is in percentage. Number of labeled samples is fixed to be 500.

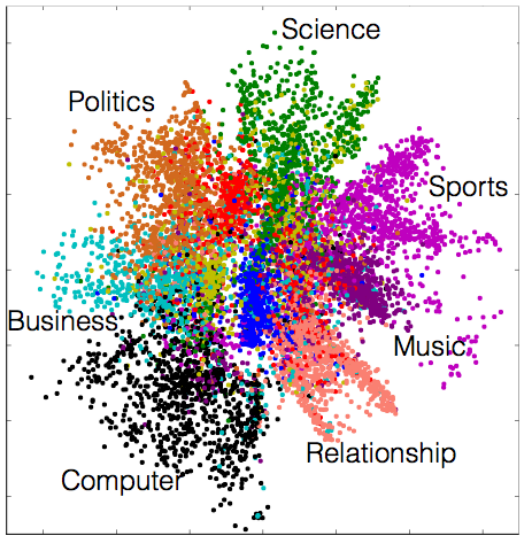
Model	100	500	1000	2000
LSTM	10.7	11.9	14.3	23.1
LA-LSTM [21]	20.8	42.2	50.4	54.7
LM-LSTM [21]	46.9	61.3	63.9	65.6
SCNN-VAE-Semi	55.4	65.6	66.0	65.8
SCNN-VAE-Semi+init	**63.8**	**65.4**	**66.6**	**67.4**

(a) Yahoo

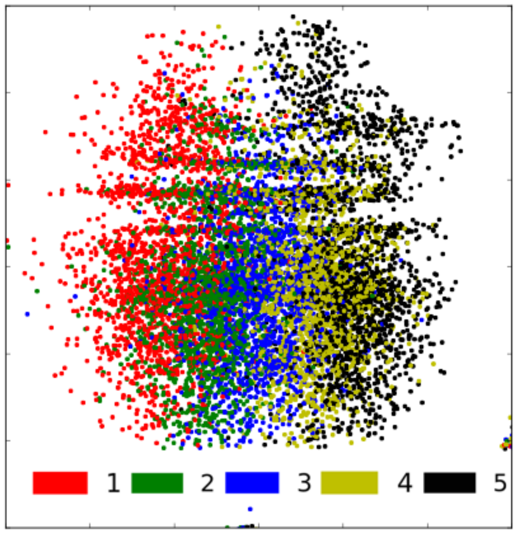
Model	100	500	1000	2000
LSTM	22.6	25.4	27.9	29.9
LA-LSTM [21]	35.2	46.4	49.8	52.2
LM-LSTM [21]	46.9	54.1	57.2	57.7
SCNN-VAE-Semi	51.4	53.5	55.3	57.4
SCNN-VAE-Semi+init	**52.6**	**57.3**	**58.9**	**59.8**

(b) Yelp

Table 4.4: Semi-supervised VAE results on the test set, in percentage. LA-LSTM and LM-LSTM come from [21], they denotes the LSTM is initialized with a sequence autoencoder and a language model.



(a) Yahoo



(b) Yelp

Figure 4.4: Visualizations of learned latent representations.

100, 500, 1000 and 2000 respectively.

Ablation Study: At first, we would like to explore the effect of different decoders for semi-supervised classification. We fix the number of labeled samples to be 500 and report both classification accuracy and NLL of the test set of Yahoo data set in Table. 4.5. We can see that SCNN-VAE-Semi has the best classification accuracy of 65.5. The accuracy decreases as we gradually increase the decoder contextual capacity. On the other hand, LCNN-VAE-Semi has the best NLL result. This classification accuracy and NLL trade off once again verifies our conjecture: with small contextual window size, the decoder is forced to use the encoder information, hence the latent representation is better learned.

Comparing the NLL results of Table 4.5 with that of Table 4.2a, we can see the NLL improves. The NLL of semi-supervised VAE improves over simple VAE from 337.8 to 335.7 for SCNN, from 336.2 to 332.8 for MCNN, and from 333.9 to 332.8 for LCNN. The improvement mainly comes from the KL divergence part, this indicates that better latent representations decrease the KL divergence, further improving the VAE results.

Comparison with related methods: We compare Semi-supervised VAE with the methods from [21], which represent the previous state-of-the-art for semi-supervised sequence learning. Dai and Le [21] pre-trains a classifier by initializing the parameters of a classifier with that of a language model or a sequence autoencoder. They find it improves the classification accuracy significantly. Since SCNN-VAE-Semi performs the best according to Table 4.5, we fix the decoder to be SCNN in this part. The detailed comparison is in Table 4.4. We can see that semi-supervised VAE performs better than LM-LSTM and LA-LSTM from [21]. We also initialize the encoder of the VAE with parameters from LM and find classification accuracy further improves. We also see the advantage of SCNN-VAE-Semi over LM-LSTM is greater when the number of labeled samples is smaller. The advantage decreases as we increase the number of labeled samples. When we set the number of labeled samples to be 25k, the SCNN-VAE-Semi achieves an accuracy of 70.4, which is similar to LM-LSTM with an accuracy of 70.5. Also, SCNN-VAE-Semi performs better on Yahoo data set than Yelp data set. For Yelp, SCNN-VAE-Semi is a little bit worse than LM-LSTM if the number of labeled samples is greater than 100, but becomes better when we initialize the encoder. Figure 4.4b explains this observation. It shows the documents are coupled together and are harder to classify. Also, the latent representation contains information other than sentiment, which may not be useful for classification.

4.4.5 Unsupervised clustering results

Model	ACCU
LSTM + GMM	25.8
SCNN-VAE + GMM	56.6
SCNN-VAE + init + GMM	57.0
SCNN-VAE-Unsup + init	**59.9**

Table 4.5: Unsupervised clustering results for Yahoo data set. We run each model 10 times and report the best results. LSTM+GMM means we extract the features from LSTM language model. SCNN-VAE + GMM means we use the mean of $q(\mathbf{z}|\mathbf{x})$ as the feature. SCNN-VAE + init + GMM means SCNN-VAE is trained with encoder initialization.

1 star	the food was good but the service was horrible . took forever to get our food . we had to ask twice for our check after we got our food . will not return .
2 star	the food was good , but the service was terrible . took forever to get someone to take our drink order . had to ask 3 times to get the check . food was ok , nothing to write about .
3 star	came here for the first time last night . food was good . service was a little slow . food was just ok .
4 star	food was good , service was a little slow , but the food was pretty good . i had the grilled chicken sandwich and it was really good . will definitely be back !
5 star	food was very good , service was fast and friendly . food was very good as well . will be back !

Table 4.6: Text generated by conditioning on sentiment label.

We also explored using the same framework for unsupervised clustering. We compare with the baselines that extract the feature with existing models and then run Gaussian Mixture Model (GMM) on these features. We find empirically that simply using the features does not perform well since the features are high dimensional. We run a PCA on these features, the dimension of PCA is selected from [8, 16, 32]. Since GMM can easily get stuck in poor local optimum, we run each model ten times and report the best result. We find directly optimizing $U(\mathbf{x})$ does not perform well for unsupervised clustering and we need to initialize the encoder with LSTM language model. The model only works well for Yahoo data set. This is potentially because Figure 4.4b shows that sentiment latent representations does not fall into clusters. γ in Equation 4.5 is a sensitive parameter, we select it from the range between 0.5 and 1.5 with an interval of 0.1. We use the following evaluation protocol [84]: after we finish training, for cluster i, we find out the validation sample $\mathbf{x}_n$ from cluster i that has the best $q(y_i|\mathbf{x})$ and assign the label of $\mathbf{x}_n$ to all samples in cluster i. We then compute the test accuracy based on this assignment. The detailed results are in Table 4.5. We can see SCNN-VAE-Unsup + init performs better than other baselines. LSTM+GMM performs very bad probably because the feature dimension is 1024 and is too high for GMM, even though we already used PCA to reduce the dimension.

Conditional text generation With the semi-supervised VAE, we are able to generate text conditional on the label. Due to space limitation, we only show one example of generated reviews conditioning on review rating in Table 4.6. For each group of generated text, we fix $\mathbf{z}$ and vary the label $\mathbf{y}$, while picking $\mathbf{x}$ via beam search with a beam size of 10.

4.5 Recent development

Following our work, there are many other works that investigated this posterior collapse problem. [57] pointed out the amortized variational inference (AVI) was the potential problem and proposed to use AVI to initialize the variational parameters and run stochastic variational to refine them. With the new approach they were able to train a model without posterior collapse. [24] added a skip connection between the latent variable and the final output to enforce that the latent variables were used. [126] used von Mises-Fisher (vMF) distribution to replace the traditional Gaussian distribution and found better performance. [39] found that the encoder network was not properly trained in variational inference procedure and proposed to update the encoder network multiple times before updating the decoder network.

Society	do you think there is a god ?
Science	how many orbitals are there in outer space ? how many orbitals are there in the solar system ?
Health	what is the difference between _UNK and _UNK
Education	what is the difference between a computer and a _UNK ?
Computers	how can i make flash mp3 files ? i want to know how to make a flash video so i can upload it to my mp3 player ?
Sports	who is the best soccer player in the world ?
Business	what is the best way to make money online ?
Music	who is the best artist of all time ?
Relationships	how do i know if a guy likes me ?
Politics	what do you think about Iran ?
Society	what is the meaning of life ?
Science	what is the difference between kinetic energy and heat ?
Health	what is the best way to get rid of migraine headaches ?
Education	what is the best way to study for a good future ?
Computers	what is the best way to install windows xp home edition ?
Sports	who do you think will win the super bowl this year ?
Business	i would like to know what is the best way to get a good paying job ?
Entertainment	what do you think is the best movie ever ?
Relationships	what is the best way to get over a broken heart ?
Politics	what do you think about the war in iraq ?
Society	what would you do if you had a million dollars ?
Mathematics	i need help with this math problem !
Health	what is the best way to lose weight ?
Education	what is the best college in the world ?
Computers	what is the best way to get a new computer ?
Sports	who should i start ?
Business	what is the best way to get a good paying job ?
Entertainment	who do you think is the hottest guy in the world ?
Relationships	what should i do ?
Politics	who do you think will be the next president of the united states ?
Society	do you believe in ghosts ?
Science	why is the sky blue ?
Health	what is the best way to get rid of a cold ?
Reference	what do you do when you are bored ?
Computers	why ca n't i watch videos on my computer ? when i try to watch videos on my computer , i ca n't get it to work on my computer . can anyone help ?
Sports	what do you think about the _UNK game ?
Business	what is the best way to get a job ?
Entertainment	what is your favorite tv show ?
Relationships	how do you know when a guy likes you ?
Politics	what do you think about this ?

Table 4.7: Text generated by conditioning on topic label.

1 star	the food is good , but the service is terrible . i have been here three times and each time the service has been horrible . the last time we were there , we had to wait a long time for our food to come out . when we finally got our food , the food was cold and the service was terrible . i will not be back .
2 star	this place used to be one of my favorite places to eat in the area .
3 star	i 've been here a few times , and the food has always been good .
4 star	this is one of my favorite places to eat in the phoenix area . the food is good , and the service is friendly .
5 star	my husband and i love this place . the food is great , the service is great , and the prices are reasonable .
1 star	this is the worst hotel i have ever been to . the room was dirty , the bathroom was dirty , and the room was filthy .
2 star	my husband and i decided to try this place because we had heard good things about it so we decided to give it a try . the service was good , but the food was mediocre at best .
3 star	we came here on a saturday night with a group of friends . we were seated right away and the service was great . the food was good , but not great . the service was good and the atmosphere was nice .
4 star	my husband and i came here for brunch on a saturday night . the place was packed so we were able to sit outside on the patio . we had a great view of the bellagio fountains and had a great view of the bellagio fountains . we sat at the bar and had a great view of the bellagio fountains .
5 star	my husband and i came here for the first time last night and had a great time ! the food was amazing , the service was great , and the atmosphere was perfect . we will be back !
1 star	this is the worst place i have ever been to . i will never go back .
2 star	i was very disappointed with the quality of the food and the service . i will not be returning .
3 star	this was my first time at this location and i have to say it was a good experience .
4 star	this is a great place to grab a bite to eat with friends or family .
5 star	i am so happy to have found a great place to get my nails done .
1 star	my wife and i have been going to this restaurant for years . the last few times i have been , the service has been terrible . the last time we were there , we had to wait a long time for our food to arrive . the food is good , but not worth the wait .
2 star	the food is good , but the service leaves something to be desired .
3 star	i have been here a few times . the food is consistently good , and the service is good .
4 star	my wife and i have been here a few times . the food is consistently good , and the service is friendly .
5 star	my husband and i have been coming here for years . the food is consistently good and the service is always great .

Table 4.8: Text generated by conditioning on sentiment label.

Chapter 5

Language Model Discriminators for GANs

5.1 Overview

In the previous chapter, we designed specific decoders for VAEs for text modeling according to the characteristics of language data to avoid the posterior collapse problem. In this chapter, we are going to investigate another type of generative model–Generative Adversarial Networks [33]–for text modeling and apply it on the task of unsupervised text style transfer. We first point out the problems of directly applying the binary classifiers as discriminators on text and then propose our solution of making the discriminator more structured. Before introducing the model in detail, we first review some background of unsupervised text style transfer task.

Unsupervised text style transfer requires learning disentangled representations of attributes (e.g., negative/positive sentiment, plaintext/ciphertext orthography) and underlying content. This is challenging because the two interact in subtle ways in natural language and it can even be hard to disentangle them with parallel data. The recent development of deep generative models like variational auto-encoders (VAEs) [59] and generative adversarial networks(GANs) [33] have made learning disentangled representations from non-parallel data possible. However, despite their rapid progress in computer vision—for example, generating photo-realistic images [97], learning interpretable representations [16], and translating images [143]—their progress on text has been more limited. For VAEs, the problem of training collapse can severely limit effectiveness [11, 133], and when applying adversarial training to natural language, the non-differentiability of discrete word tokens makes generator optimization difficult. Hence, most attempts use REINFORCE [111] to finetune trained models [74, 136] or uses professor forcing [68] to match hidden states of decoders.

Previous work on unsupervised text style transfer [44, 104] adopts an encoder-decoder architecture with style discriminators to learn disentangled representations. The encoder takes a sentence as an input and outputs a style-independent content representation. The style-dependent decoder takes the content representation and a style representation and generates the transferred sentence. [44] use a style classifier to directly enforce the desired style in the generated text. [104] leverage an adversarial training scheme where a binary CNN-based discriminator is used to evaluate whether a transferred sentence is real or fake, ensuring that transferred sentences match real sentences in terms of target style. However, in practice, the error signal from a binary classifier is sometimes insufficient to train the generator to produce fluent language, and optimization can be unstable as a result of the adversarial training step.

We propose to use an implicitly trained language model as a new type of discriminator, replacing the more conventional binary classifier. The language model calculates a sentence's likelihood, which decomposes into a product of token-level conditional probabilities. In our approach, rather than training a binary classifier to distinguish real and fake sentences, we train the language model to assign a high probability to real sentences and train the generator to produce sentences with high probability under the language model. Because the language model scores sentences directly using a product of locally normalized probabilities, it may offer more stable and more useful training signal to the generator. Further, by using a continuous approximation of discrete sampling under the generator, our model can be trained using back-propagation in an end-to-end fashion.

We find empirically that when using the language model as a structured discriminator, it is possible to eliminate adversarial training steps that use negative samples—a critical part of traditional adversarial training. Language models are *implicitly* trained to assign a low probability to negative samples because of its normalization constant. By eliminating the adversarial training step, we found the training becomes more stable in practice.

To demonstrate the effectiveness of our new approach, we conduct experiments on three tasks: word substitution decipherment, sentiment modification, and related language translation. We show that our approach, which uses only a language model as the discriminator, outperforms a broad set of state-of-the-art approaches on the three tasks.

5.2 Related Work

Non-parallel transfer in natural language: [32, 44, 95, 104] are most relevant to our work. [44] aim to generate sentences with controllable attributes by learning disentangled representations. [104] introduce adversarial training to unsupervised text style transfer. They apply discriminators both on the encoder representation and on the hidden states of the decoders to ensure that they have the same distribution. These are the two models that we mainly compare with. [95] use the back-translation technique in their model, which is complementary to our method and can be integrated into our model to further improve performance. [32] use GAN-based approach to decipher shift ciphers. [6, 69] propose unsupervised machine translation and use adversarial training to match the encoder representation of the sentences from different languages. They also use back-translation to refine their model in an iterative way.

GANs: GANs have been widely explored recently, especially in computer vision [16, 22, 49, 97, 102, 111, 143]. The progress of GANs on text is relatively limited due to the non-differentiable discrete tokens. Lots of papers [13, 74, 129, 136] use REINFORCE [111] to fine-tune a trained model to improve the quality of samples. There is also prior work that attempts to introduce more structured discriminators, for instance, the energy-based GAN (EBGAN) [140] and RankGAN [76]. Our language model can be seen as a special energy function, but it is more complicated than the auto-encoder used in [140] since it has a recurrent structure. [46] also proposes to use structured discriminators in generative models and establishes its the connection with posterior regularization.

Computer vision style transfer: Our work is also related to unsupervised style transfer in computer vision [31, 48]. [31] directly uses the covariance matrix of the CNN features and tries to align the covariance matrix to transfer the style. [48] proposes adaptive instance normalization for an arbitrary style of images. [143] uses a cycle-consistency loss to ensure the content of the images is preserved and can be translated back to original images.

Language model for reranking: Previously, language models are used to incorporate the knowledge of monolingual data mainly by reranking the sentences generated from a base model such as [12, 37, 38]. [15, 80] use a language model as training supervision for unsupervised OCR. Our model is more advanced in using language models as discriminators in distilling the knowledge of monolingual data to a base model in an end-to-end way.

5.3 Unsupervised Text Style Transfer

We start by reviewing the current approaches for unsupervised text style transfer [44, 104], and then go on to describe our approach in Section 5.4. Assume we have two text datasets $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$ and $\mathbf{Y} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n)}\}$ with two different styles $\mathbf{v}_x$ and $\mathbf{v}_y$, respectively. For example, $\mathbf{v}_x$ can be the positive sentiment style and $\mathbf{v}_y$ can be the negative sentiment style. The datasets are non-parallel such that the data does not contain pairs of $(\mathbf{x}^{(i)}, \mathbf{y}^{(j)})$ that describe the same content. The goal of style transfer is to transfer data $\mathbf{x}$ with style $\mathbf{v_x}$ to style $\mathbf{v_y}$ and vice versa, i.e., to estimate the conditional distribution $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y})$. Since text data is discrete, it is hard to learn the transfer function directly via back-propagation as in computer vision [143]. Instead, we assume the data is generated conditioned on two disentangled parts, the style $\mathbf{v}$ and the content $\mathbf{z}$[1] [44].

Consider the following generative process for each style: 1) the style representation $\mathbf{v}$ is sampled from a prior $p(\mathbf{v})$; 2) the content vector $\mathbf{z}$ is sampled from $p(\mathbf{z})$; 3) the sentence $\mathbf{x}$ is generated from the conditional distribution $p(\mathbf{x}|\mathbf{z}, \mathbf{v})$. This model suggests the following parametric form for style transfer where q represents a posterior:

$$p(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{z_x}} p(\mathbf{y}|\mathbf{z_x}, \mathbf{v_y}) q(\mathbf{z_x}|\mathbf{x}, \mathbf{v_x}) d\mathbf{z_x}.$$

The above equation suggests the use of an encoder-decoder framework for style transfer problems. We can first encode the sentence $\mathbf{x}$ to get its content vector $\mathbf{z_x}$, then we switch the style label from $\mathbf{v_x}$ to $\mathbf{v_y}$. Combining the content vector $\mathbf{z_x}$ and the style label $\mathbf{v_y}$, we can generate a new sentence $\tilde{\mathbf{x}}$ (the transferred sentences are denotes as $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$).

One unsupervised approach is to use the auto-encoder model. We first use an encoder model $\mathbf{E}$ to encode $\mathbf{x}$ and $\mathbf{y}$ to get the content vectors $\mathbf{z_x} = \mathbf{E}(\mathbf{x}, \mathbf{v_x})$ and $\mathbf{z_y} = \mathbf{E}(\mathbf{y}, \mathbf{v_y})$. Then we use a decoder $\mathbf{G}$ to generate sentences conditioned on $\mathbf{z}$ and $\mathbf{v}$. The $\mathbf{E}$ and $\mathbf{G}$ together form an auto-encoder and the reconstruction loss is:

$$\mathcal{L}_{\text{rec}}(\theta_{\mathbf{E}}, \theta_{\mathbf{G}}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{X}}[-\log p_{\mathbf{G}}(\mathbf{x}|\mathbf{z_x}, \mathbf{v_x})] + \mathbb{E}_{\mathbf{y}\sim\mathbf{Y}}[-\log p_{\mathbf{G}}(\mathbf{y}|\mathbf{z_y}, \mathbf{v_y})],$$

where $\mathbf{v_x}$ and $\mathbf{v_y}$ can be two learnable vectors to represent the label embedding. In order to make sure that the $\mathbf{z_x}$ and $\mathbf{z_y}$ capture the content and we can deliver accurate transfer between the

[1]We drop the subscript in notations wherever the meaning is clear.

style by switching the labels, we need to guarantee that $\mathbf{z_x}$ and $\mathbf{z_y}$ follow the same distribution. We can assume $p(\mathbf{z})$ follows a prior distribution and add a KL-divergence regularization on $\mathbf{z_x}$, $\mathbf{z_y}$. The model then becomes a VAE. However, previous works [11, 133] found that there is a training collapse problem with the VAE for text modeling and the posterior distribution of $\mathbf{z}$ fails to capture the content of a sentence.

To better capture the desired styles in the generated sentences, [44] additionally impose a style classifier on the generated samples, and the decoder $\mathbf{G}$ is trained to generate sentences that maximize the accuracy of the style classifier. Such additional supervision with a *discriminative* model is also adopted in [104], though in that work a binary real/fake classifier is instead used within a conventional adversarial scheme.

Adversarial Training [104] use adversarial training to align the $\mathbf{z}$ distributions. Not only do we want to align the distribution of $\mathbf{z_x}$ and $\mathbf{z_y}$, but also we hope that the transferred sentence $\tilde{\mathbf{x}}$ from $\mathbf{x}$ to resemble $\mathbf{y}$ and vice versa. Several adversarial discriminators are introduced to align these distributions. Each of the discriminators is a binary classifier distinguishing between real and fake. Specifically, the discriminator $D_\mathbf{z}$ aims to distinguish between $\mathbf{z_x}$ and $\mathbf{z_y}$:

$$\mathcal{L}^{\mathbf{z}}_{\text{adv}}(\theta_\mathbf{E}, \theta_{\mathbf{D_z}}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{X}}[-\log D_\mathbf{z}(\mathbf{z_x})] + \mathbb{E}_{\mathbf{y}\sim\mathbf{Y}}[-\log(1 - D_\mathbf{z}(\mathbf{z_y}))].$$

Similarly, $D_\mathbf{x}$ distinguish between $\mathbf{x}$ and $\tilde{\mathbf{y}}$, yielding an objective $\mathcal{L}^{\mathbf{x}}_{\text{adv}}$ as above; and $D_\mathbf{y}$ distinguish between $\mathbf{y}$ and $\tilde{\mathbf{x}}$, yielding $\mathcal{L}^{\mathbf{y}}_{\text{adv}}$. Since the samples of $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are discrete and it is hard to train the generator in an end-to-end way, professor forcing [68] is used to match the distributions of the hidden states of decoders. The overall training objective is a min-max game played among the encoder $\mathbf{E}$/decoder $\mathbf{G}$ and the discriminators $D_\mathbf{z}, D_\mathbf{x}, D_\mathbf{y}$ [33]:

$$\min_{E,G} \max_{D_\mathbf{z}, D_\mathbf{x}, D_\mathbf{y}} \mathcal{L}_{\text{rec}} - \lambda(\mathcal{L}^{\mathbf{z}}_{\text{adv}} + \mathcal{L}^{\mathbf{x}}_{\text{adv}} + \mathcal{L}^{\mathbf{y}}_{\text{adv}})$$

The model is trained in an alternating manner. In the first step, the loss of the discriminators are minimize to distinguish between the $\mathbf{z_x}, \mathbf{x}, \mathbf{y}$ and $\mathbf{z_y}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}$, respectively; and in the second step the encoder and decoder are trained to minimize the reconstruction loss while maximizing loss of the discriminators.

5.4 Language Models as Discriminators

In most past work, a classifier is used as the discriminator to distinguish whether a sentence is real or fake. We propose instead to use locally-normalized language models as discriminators. We argue that using an explicit language model with token-level locally normalized probabilities offers a more direct training signal to the generator. If a transfered sentence does not match the target style, it will have high perplexity when evaluated by a language model that was trained on target domain data. Not only does it provide an overall evaluation score for the whole sentence, but a language model can also assign a probability to each token, thus providing more information on which word is to blame if the overall perplexity is very high.

The overall model architecture is shown in Figure 5.1. Suppose $\tilde{\mathbf{x}}$ is the output sentence from applying style transfer to input sentence $\mathbf{x}$, i.e., $\tilde{\mathbf{x}}$ is sampled from $p_G(\tilde{\mathbf{x}}|\mathbf{z_x}, \mathbf{v_y})$ (and similary for

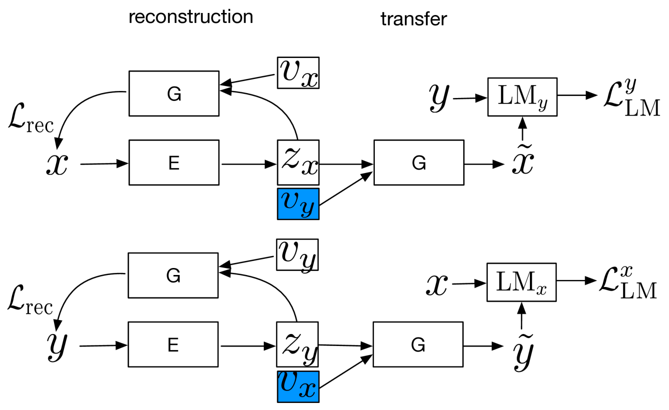


Figure 5.1: The overall model architecture consists of two parts: reconstruction and transfer. For transfer, we switch the style label and sample an output sentence from the generator that is evaluated by a language model.

$\tilde{\mathbf{y}}$ and $\mathbf{y}$). Let $p_{\text{LM}}(\mathbf{x})$ be the probability of a sentence $\mathbf{x}$ evaluate against a language model, then the discriminator loss becomes:

$$\mathcal{L}^{\mathbf{x}}_{\text{LM}}(\theta_{\mathbf{E}}, \theta_{\mathbf{G}}, \theta_{\text{LM}_\mathbf{x}}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{X}}[-\log p_{\text{LM}_\mathbf{x}}(\mathbf{x}))] + \gamma\mathbb{E}_{\mathbf{y}\sim\mathbf{Y},\tilde{\mathbf{y}}\sim p_G(\tilde{\mathbf{y}}|\mathbf{z}_\mathbf{y},\mathbf{v}_\mathbf{x})}[\log p_{\text{LM}_\mathbf{x}}(\tilde{\mathbf{y}})], \tag{5.1}$$

$$\mathcal{L}^{\mathbf{y}}_{\text{LM}}(\theta_{\mathbf{E}}, \theta_{\mathbf{G}}, \theta_{\text{LM}_\mathbf{y}}) = \mathbb{E}_{\mathbf{y}\sim\mathbf{Y}}[-\log p_{\text{LM}_\mathbf{y}}(\mathbf{y}))] + \gamma\mathbb{E}_{\mathbf{x}\sim\mathbf{X},\tilde{\mathbf{x}}\sim p_G(\tilde{\mathbf{x}}|\mathbf{z}_\mathbf{x},\mathbf{v}_\mathbf{y})}[\log p_{\text{LM}_\mathbf{y}}(\tilde{\mathbf{x}})]. \tag{5.2}$$

Our overall objective becomes:

$$\min_{E,G} \max_{\text{LM}_\mathbf{x},\text{LM}_\mathbf{y}} \mathcal{L}_{\text{rec}} - \lambda(\mathcal{L}^{\mathbf{x}}_{\text{LM}} + \mathcal{L}^{\mathbf{y}}_{\text{LM}}) \tag{5.3}$$

Negative samples: Note that Equation 5.1 and 5.2 differs from traditional ways of training language models in that we have a term including the negative samples. We train the LM in an adversarial way by minimizing the loss of LM of real sentences and maximizing the loss of transferred sentences. However, since the LM is a structured discriminator, we would hope that a language model trained on the real sentences will automatically assign high perplexity to sentences not in the target domain, hence negative samples from the generator may not be necessary. To investigate the necessity of negative samples, we add a weight γ to the loss of negative samples. The weight γ adjusts the negative sample loss in training the language models. If $\gamma = 0$, we simply train the language model on real sentences and fix its parameters, avoiding potentially unstable adversarial training steps. We investigate the necessity of using negative samples in the experiment section.

Training consists of two steps alternatively. In the first step, we train the language models according to Equation 5.1 and 5.2. In the second step, we minimize the reconstruction loss as well as the perplexity of generated samples evaluated by the language model. Since $\tilde{\mathbf{x}}$ is discrete, one can use the REINFORCE [111] algorithm to train the generator:

$$\nabla_{\theta_G}\mathcal{L}^{\mathbf{y}}_{\text{LM}} = \mathbb{E}_{\mathbf{x}\sim\mathbf{X},\tilde{\mathbf{x}}\sim p_G(\tilde{\mathbf{x}}|\mathbf{z}_\mathbf{x},\mathbf{v}_\mathbf{y})}[\log p_{\text{LM}}(\tilde{\mathbf{x}})\nabla_{\theta_G}\log p_G(\tilde{\mathbf{x}}|\mathbf{z}_\mathbf{x},\mathbf{v}_\mathbf{y})]. \tag{5.4}$$

However, using a single sample to approximate the expected gradient leads to high variance in gradient estimates and thus unstable learning.

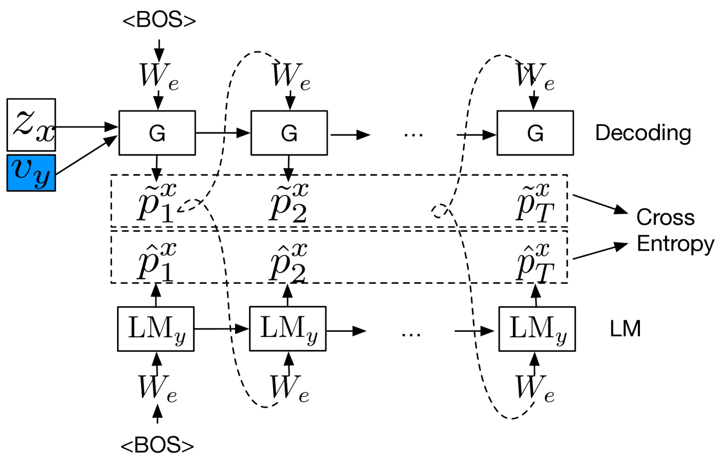


Figure 5.2: Continuous approximation of language model loss. The input is a sequence of probability distributions $\{\tilde{\mathbf{p}}^{\mathbf{x}}_t\}_{t=1}^T$ sampled from the generator. At each timestep, we compute a weighted embedding as input to the language model and get the sequence of output distributions from the LM as $\{\hat{\mathbf{p}}^{\mathbf{x}}_t\}_{t=1}^T$. The loss is the sum of cross entropies between each pair of $\tilde{\mathbf{p}}^{\mathbf{x}}_t$ and $\hat{\mathbf{p}}^{\mathbf{x}}_t$.

Continuous approximation: Instead, we propose to use a continuous approximation to the sampling process in training the generator, as demonstrated in Figure 5.2. Instead of feeding a single sampled word as input to the next timestep of the generator, we use a Gumbel-softmax [50] distribution as a continuous approximation to sample instead. Let u be a categorical distribution with probabilities $\pi_1, \pi_2, \ldots, \pi_c$. Samples from u can be approximated using:

$$p_i = \frac{\exp((\log \pi_i) + g_i)/\tau}{\sum_{j=1}^c \exp((\log \pi_j + g_j)/\tau)},$$

where the g_i's are independent samples from $\text{Gumbel}(0, 1)$.

Let the tokens of the transferred sentence be $\tilde{\mathbf{x}} = \{\tilde{x}_t\}_{t=1}^T$. Suppose the output of the logit at timestep t is $\mathbf{v}^{\mathbf{x}}_t$, then $\tilde{\mathbf{p}}^{\mathbf{x}}_t = \text{Gumbel-softmax}(\mathbf{v}^{\mathbf{x}}_t, \tau)$, where τ is the temperature. When $\tau \to 0$, $\tilde{\mathbf{p}}^{\mathbf{x}}_t$ becomes the one hot representation of token $\tilde{x}_t$. Using the continuous approximation, then the output of the decoder becomes a sequence of probability vectors $\tilde{\mathbf{p}}^{\mathbf{x}} = \{\tilde{\mathbf{p}}^{\mathbf{x}}_t\}_{t=1}^T$.

With the continuous approximation of $\tilde{\mathbf{x}}$, we can calculate the loss evaluated using a language model easily, as shown in Figure 5.2. For every step, we feed $\tilde{\mathbf{p}}^{\mathbf{x}}_t$ to the language model of $\mathbf{y}$ (denoted as $\text{LM}_{\mathbf{y}}$) using the weighted average of the embedding $W_e\tilde{\mathbf{p}}^{\mathbf{x}}_t$, then we get the output from the $\text{LM}_{\mathbf{y}}$ which is a probability distribution over the vocabulary of the next word $\hat{\mathbf{p}}^{\mathbf{x}}_{t+1}$. The loss of the current step is the cross entropy loss between $\tilde{\mathbf{p}}^{\mathbf{x}}_{t+1}$ and $\hat{\mathbf{p}}^{\mathbf{x}}_{t+1}$: $(\tilde{\mathbf{p}}^{\mathbf{x}}_{t+1})^\intercal \log \hat{\mathbf{p}}^{\mathbf{x}}_{t+1}$. Note that when the decoder output distribution $\tilde{\mathbf{p}}^{\mathbf{x}}_{t+1}$ aligns with the language model output distribution $\hat{\mathbf{p}}^{\mathbf{x}}_{t+1}$, the above loss achieves minimum. By summing the loss over all steps and taking the gradient, we can use standard back-propagation to train the generator:

$$\nabla_{\theta_G} \mathcal{L}^{\mathbf{y}}_{\text{LM}} \approx \mathbb{E}_{\mathbf{x}\sim\mathbf{X}, \tilde{\mathbf{p}}^{\mathbf{x}} \sim p_G(\tilde{\mathbf{x}}|\mathbf{z}_{\mathbf{x}}, \mathbf{v}_{\mathbf{y}})}[\nabla_{\theta_G} \sum_{t=1}^T (\tilde{\mathbf{p}}^{\mathbf{x}}_t)^\intercal \log \hat{\mathbf{p}}^{\mathbf{x}}_t]. \tag{5.5}$$

The above Equation is a continuous approximation of Equation 5.4 with Gumbel softmax distribution. In experiments, we use a single sample of $\tilde{\mathbf{p}}^{\mathbf{x}}$ to approximate the expectation.

Note that the use of the language model discriminator is a somewhat different in each of the two types of training update steps because of the continuous approximation. We use discrete samples from the generators as negative samples in training the language model discriminator step, while we use a continuous approximation in updating the generator step according to Equation 5.5.

Overcoming mode collapse: It is known that in adversarial training, the generator can suffer from mode collapse [5, 45] where the samples from the generator only cover part of the data distribution. In preliminary experimentation, we found that the language model prefers short sentences. To overcome this length bias, we use two tricks in our experiments: 1) we normalize the loss of Equation 5.5 by length and 2) we fix the length of $\tilde{\mathbf{x}}$ to be the same of $\mathbf{x}$. We find these two tricks stabilize the training and avoid generating collapsed overly short outputs.

5.5 Experiments

In order to verify the effectiveness of our model, we experiment on three tasks: word substitution decipherment, sentiment modification, and related language translation. We mainly compare with the most comparable approach of [104] that uses CNN classifiers as discriminators[2]. Note that [104] use three discriminators to align both $\mathbf{z}$ and decoder hidden states, while our model only uses a single language model as a discriminator directly on the output sentences $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$. Moreover, we also compare with a broader set of related work [28, 44, 75] for the tasks when appropriate. Our proposed model provides substantiate improvements in most of the cases. We implement our model with the Texar [47] toolbox based on Tensorflow [1].

Model Configurations: Similar model configuration to that of [104] is used for a fair comparison. We use one-layer GRU [19] as the encoder and decoder (generator). We set the word embedding size to be 100 and GRU hidden size to be 700. $\mathbf{v}$ is a vector of size 200. For the language model, we use the same architecture as the decoder. The parameters of the language model are not shared with parameters of other parts and are trained from scratch. We use a batch size of 128, which contains 64 samples from $\mathbf{X}$ and $\mathbf{Y}$ respectively. We use Adam [58] optimization algorithm to train both the language model and the auto-encoder and the learning rate is set to be the same. Hyper-parameters are selected based on the validation set. We use grid search to pick the best parameters. The learning rate is selected from $[1e-3, 5e-4, 2e-4, 1e-4]$ and λ, the weight of language model loss, is selected from $[1.0, 0.5, 0.1]$. Models are trained for a total of 20 epochs. We use an annealing strategy to set the temperature of τ of the Gumbel-softmax approximation. The initial value of τ is set to 1.0 and it decays by half every epoch until reaching the minimum value of 0.001.

Training Algorithms:

[2]We use the code from `https://github.com/shentianxiao/language-style-transfer`.

Algorithm 1 Unsupervised text style transfer.

```
Input: Data set of two different styles $\mathbf{X}, \mathbf{Y}$.
  Parameters: weight $\lambda$ and $\gamma$, temperature $\tau$.
  Initialized model parameters $\theta_{\mathbf{E}}, \theta_{\mathbf{G}}, \theta_{\text{LM}_\text{x}}, \theta_{\text{LM}_\text{y}}$.
  repeat
      Update $\theta_{\text{LM}_\text{x}}$ and $\theta_{\text{LM}_\text{y}}$ by minimizing $\mathcal{L}^{\text{x}}_{\text{LM}}(\theta_{\text{LM}_\text{x}})$ and $\mathcal{L}^{\text{y}}_{\text{LM}}(\theta_{\text{LM}_\text{y}})$ respectively.
      Update $\theta_{\mathbf{E}}, \theta_{\mathbf{G}}$ by minimizing: $\mathcal{L}_{\text{rec}} - \lambda(\mathcal{L}^{\text{x}}_{\text{LM}} + \mathcal{L}^{\text{y}}_{\text{LM}})$ using Equation 5.5.
  until convergence
Output: A text style transfer model with parameters $\theta_{\mathbf{E}}$, $\theta_{\mathbf{G}}$.
```

Model	20%	40%	60%	80%	100%
Copy	64.3	39.1	14.4	2.5	0
[104]*	86.6	77.1	70.1	61.2	**50.8**
`Our results:`					
LM	89.0	**80.0**	**74.1**	62.9	49.3
LM + adv	**89.1**	79.6	71.8	**63.8**	44.2

Table 5.1: Decipherment results measured in BLEU. Copy is directly measuring $\mathbf{y}$ against $\mathbf{x}$. LM + adv denotes we use negative samples to train the language model.*We run the code open-sourced by the authors to get the results.

5.5.1 Word substitution decipherment

As the first task, we consider the word substitution decipherment task previous explored in the NLP literature [25]. We can control the amount of change to the original sentences in word substitution decipherment so as to systematically investigate how well the language model performs in a task that requires various amount of changes. In word substitution cipher, every token in the vocabulary is mapped to a cipher token and the tokens in sentences are replaced with cipher tokens according to the cipher dictionary. The task of decipherment is to recover the original text without any knowledge of the dictionary.

Data: Following [104], we sample 200K sentences from the Yelp review dataset as plain text $\mathbf{X}$ and sample other 200K sentences and apply word substitution cipher on these sentences to get $\mathbf{Y}$. We use another 100k *parallel* sentences as the development and test set respectively. Sentences of length more than 15 are filtered out. We keep all words that appear more than 5 times in the training set and get a vocabulary size of about 10k. All words appearing less than 5 times are replaced with a ¡unk¿ token. We random sample words from the vocabulary and replace them with cipher tokens. The amount of ciphered words ranges from 20% to 100%. As we have ground truth plain text, we can directly measure the BLEU [3] score to evaluate the model.

Results: The results are shown in Table 5.1. We first investigate the effect of using negative samples in training the language model, as denotes by LM + adv in Table 5.1. We can see that using adversarial training sometimes improves the results. However, we found empirically that

[3]BLEU score is measured with `multi-bleu.perl`.

Model	Accu	BLEU	$\text{PPL}_{\mathbf{X}}$	$\text{PPL}_{\mathbf{Y}}$
[104]	79.5	12.4	50.4	52.7
[44]	87.7	**65.6**	115.6	239.8
`Our results:`				
LM	83.3	38.6	**30.3**	**42.1**
LM + Classifier	**91.2**	57.8	47.0	60.9

Table 5.2: Results for sentiment modification. $\mathbf{X}$ = negative, $\mathbf{Y}$ = positive. $\text{PPL}_{\mathbf{x}}$ denotes the perplexity of sentences transferred from positive sentences evaluated by a language model trained with negative sentences and vice versa.

using negative samples makes the training very unstable and the model diverges easily. This is the main reason why we did not get consistently better results by incorporating adversarial training.

Comparing with [104], we can see that the language model without adversarial training is already very effective and performs much better when the amount of change is less than 100%. This is intuitive because when the change is less than 100%, a language model can use context information to predict and correct enciphered tokens. It's surprising that even with 100% token change, our model is only 1.5 BLEU score worse than [104], when all tokens are replaced and no context information can be used by the language model. We guess our model can gradually decipher tokens from the beginning of a sentence and then use them as a bootstrap to decipher the whole sentence. We can also combine language models with the CNNs as discriminators. For example, for the 100% case, we get BLEU score of 52.1 when combing them. Given unstableness of adversarial training and effectiveness of language models, we set $\gamma = 0$ in Equation 5.1 and 5.2 in the rest of the experiments.

5.5.2 Sentiment Manipulation

We have demonstrated that the language model can successfully crack word substitution cipher. However, the change of substitution cipher is limited to a one-to-one mapping. As the second task, we would like to investigate whether a language model can distinguish sentences with positive and negative sentiments, thus help to transfer the sentiments of sentences while preserving the content. We compare to the model of [44] as an additional baseline, which uses a pre-trained classifier as guidance.

Data: We use the same data set as in [104]. The data set contains 250K negative sentences (denoted as $\mathbf{X}$) and 380K positive sentences (denoted as $\mathbf{Y}$), of which 70% are used for training, 10% are used for development and the remaining 20% are used as test set. The pre-processing steps are the same as the previous experiment. We also use similar experiment configurations.

Evaluation: Evaluating the quality of transferred sentences is a challenging problem as there are no ground truth sentences. We follow previous papers in using model-based evaluation. We measure whether transferred sentences have the correct sentiment according to a pre-trained sentiment classifier. We follow both [44] and [104] in using a CNN-based classifier. However, simply evaluating the sentiment of sentences is not enough since the model can output collapsed output such as a single word "good" for all negative transfer and "bad" for all positive transfer. We not only would like transferred sentences to preserve the content of original sentences, but also to be smooth in terms of language quality. For these two aspects, we propose to measure the BLEU score of transferred sentences against original sentences and measure the perplexity of transferred sentences to evaluate the fluency. A good model should perform well on all three metrics.

Results: We report the results in Table. 5.2. As a baseline, the original corpus has perplexity of 35.8 and 38.8 for the negative and positive sentences respectively. Comparing LM with [104], we can see that LM outperforms it in all three aspects: getting higher accuracy, preserving the content better while being more fluent. This demonstrates the effectiveness of using LM as the discriminator. [44] has the highest accuracy and BLEU score among the three models while the perplexity is very high. It is not surprising that the classifier will only modify the features of the sentences that are related to the sentiment and there is no mechanism to ensure that the modified sentence being fluent. Hence the corresponding perplexity is very high. We can manifest the best of both models by combing the loss of LM and the classifier in [44]: a classifier is good at modifying the sentiment and an LM can smooth the modification to get a fluent sentence. We find improvement of accuracy and perplexity as denoted by LM + classifier compared to classifier only [44].

Comparing with other models: Recently there are other models that are proposed specifically targeting the sentiment modification task such as [75]. Their method is feature based and consists of the following steps: (`Delete`) first, they use the statistics of word frequency to delete the attribute words such as "good, bad" from original sentences, (`Retrieve`) then they retrieve the most similar sentences from the other corpus based on nearest neighbor search, (`Generate`) the attribute words from retrieved sentences are combined with the content words of original sentences to generate transferred sentences. The authors provide 500 human annotated sentences as the ground truth of transferred sentences so we measure the BLEU score against those sentences. The results are shown in Table 5.3. We can see our model has similar accuracy compared with DeleteAndRetrieve, but has much better BLEU scores and slightly better perplexity.

We list some examples of transferred sentences in Table 5.5 in the appendix. We can see that [104] does not keep the content of the original sentences well and changes the meaning of the original sentences. [44] changes the sentiment but uses improper words, e.g. "maintenance is equally `hilarious`". Our LM can change the change the sentiment of sentences. But sometimes there is an over-smoothing problem, changing the less frequent words to more frequent words, e.g. changing "my goodness it was so gross" to "my `food` it was so good.". In general LM + classifier has the best results, it changes the sentiment, while keeps the content and the sentences are fluent.

5.5.3 Related language translation

In the final experiment, we consider a more challenging task: unsupervised related language translation [94]. Related language translation is easier than normal pair language translation since there is a close relationship between the two languages. Note here we don't compare with other sophisticated unsupervised neural machine translation systems such as [6, 69], whose models are much more complicated and use other techniques such as back-translation, but simply

Model	ACCU	BLEU	$\text{PPL}_{\mathbf{X}}$	$\text{PPL}_{\mathbf{Y}}$
[104]	76.2	6.8	49.4	45.6
[28]:				
StyleEmbedding	9.2	16.65	97.51	142.6
MultiDecoder	50.9	11.24	111.1	119.1
[75]:				
Delete	87.2	11.5	75.2	68.7
Template	86.7	18.0	192.5	148.4
Retrieval	**95.1**	1.3	**31.5**	**37.0**
DeleteAndRetrieval	90.9	12.6	104.6	43.8
Our results:				
LM	85.4	13.4	32.8	40.5
LM + Classifier	90.0	**22.3**	48.4	61.6

Table 5.3: Results for sentiment modification based on the 500 human annotated sentences as ground truth from [75].

compare the different type of discriminators in the context of a simple model.

Data: We choose Bosnian (bs) vs Serbian (sr) and simplified Chinese (zh-CN) vs traditional Chinese (zh-TW) pair as our experiment languages. Due to the lack of parallel data for these data, we build the data ourselves. For bs and sr pair, we use the monolingual data from Leipzig Corpora Collections[4]. We use the news data and sample about 200k sentences of length less than 20 for each language, of which 80% are used for training, 10% are used for validation and remaining 10% are used for test. For validation and test, we obtain the parallel corpus by using the Google Translation API. The vocabulary size is 25k for the sr vs bs language pair. For zh-CN and zh-TW pair, we use the monolingual data from the Chinese Gigaword corpus. We use the news headlines as our training data. 300k sentences are sampled for each language. The data is partitioned and parallel data is obtained in a similar way to that of sr vs bs pair. We directly use a character-based model and the total vocabulary size is about 5k. For evaluation, we directly measure the BLEU score using the references for both language pairs.

Note that the relationship between zh-CN and zh-TW is simple and mostly like a decipherment problem in which some simplified Chinese characters have the corresponding traditional character mapping. The relation between bs vs sr is more complicated.

Results: The results are shown in Table. 5.4. For sr–bos and bos–sr, since the vocabulary of two languages does not overlap at all, it is a very challenging task. We report the BLEU1 metric since the BLEU4 is close to 0. We can see that our language model discriminator still outperforms [104] slightly. The case for zh–tw and tw–zh is much easier. Simple copying already has a reasonable score of 32.3. Using our model, we can improve it to 81.6 for cn–tw and 85.5 for tw–cn, outperforming [104] by a large margin.

[4]`http://wortschatz.uni-leipzig.de/en`

Model	sr–bs	bs–sr	cn–tw	tw–cn
Copy	0	0	32.3	32.3
[104]	29.1	30.3	60.1	60.7
`Our results:`				
LM	**31.0**	**31.7**	**81.6**	**85.5**

Table 5.4: Related language translation results measured in BLEU. The results for sr vs bs in measured in BLEU1 while cn vs tw is measure in BLEU.

Model	Negative to Positive
Original	it was super dry and had a weird taste to the entire slice .
[104]	it was super friendly and had a nice touch to the same .
[44]	it was super well-made and had a weird taste to the entire slice .
LM	it was very good , had a good taste to the food service .
LM + classifier	it was super fresh and had a delicious taste to the entire slice .
Original	my goodness it was so gross .
[104]	my server it was so .
[44]	my goodness it was so refreshing .
LM	my food it was so good .
LM + classifier	my goodness it was so great .
Original	maintenance is equally incompetent .
[104]	everything is terrific professional .
[44]	maintenance is equally hilarious .
LM	maintenance is very great .
LM + classifier	maintenance is equally great .
Original	if i could give them a zero star review i would !
[104]	if i will give them a breakfast star here ever !
[44]	if i lite give them a sweetheart star review i would !
LM	if i could give them a _num_ star place i would .
LM + classifier	if i can give them a great star review i would !

Model	Positive to Negative
Original	did n't know this type cuisine could be this great !
[104]	did n't know this old food you make this same horrible !
[44]	did n't know this type cuisine could be this great !
LM	did n't know this type , could be this bad .
LM + classifier	did n't know this type cuisine could be this horrible .
Original	besides that , the wine selection they have is pretty awesome as well .
[104]	after that , the quality prices that does n't pretty much well as .
[44]	besides that , the wine selection they have is pretty borderline as atrocious .
LM	besides that , the food selection they have is pretty awful as well .
LM + classifier	besides that , the wine selection they have is pretty horrible as well .
Original	uncle george is very friendly to each guest .
[104]	if there is very rude to our cab .
[44]	uncle george is very lackluster to each guest .
LM	uncle george is very rude to each guest .
LM + classifier	uncle george is very rude to each guest .
Original	the food is fresh and the environment is good .
[104]	the food is bland and the food is the nightmare .
[44]	the food is atrocious and the environment is atrocious .
LM	the food is bad , the food is bad .
LM + classifier	the food is bland and the environment is bad .

Table 5.5: Sentiment transfer examples.

Chapter 6

Conclusions and future work

This thesis has investigated several models designed with new structural priors to address both supervised and unsupervised tasks. For supervised tasks, we demonstrated a model with iterative attention mechanism can effectively answer natural questions according to content of reference images. The attention maps learned in the model are not only interpretable, but also reflect the prior structures we injected into the model. We also introduced a new model to classify the documents that surpasses previous state-of-the-art. The new model has two main structural priors, the documents have a hierarchical structure and different words and sentences are differently informative. We showed the new structured model not only significantly improves accuracy, but also extracts the informative words and sentences out of documents.

For unsupervised learning, we designed specific modules for VAEs and GANs to deal with discrete and sequential inputs. We investigated the posterior collapse problems of VAEs for text modeling. By using a more structured decoder to control the contextual capacity, we are able to overcome the posterior collapse problem and make the model generate text according to desired attributes. The structured decoder can also be used in the semi-supervised learning setting and help supervised learning with large scale unsupervised data. Additionally, we introduced a structured discriminator—language model—for GAN-based unsupervised text style transfer. We find empirically a structured discriminator provides richer supervision signal to guide the generator in producing fluent and rich sentences.

There are several directions to explore in the future to further improve upon the work in this thesis. First of all, as we showed in Chapter 2, using more layers of attention (greater than two) was not helpful to improve the results. We can try to add residual connection [40] between the attention layers so that we can try deeper attention models. Other techniques such as layer normalization [7] can also be added to the current model. Combing our model with the recent self-attention mechanism [119] is another interesting direction to explore. Currently our model does not capture the relationship between the objects in the images, using self-attention from [119] may be helpful to improve the representation of images features, hence boosting the model performance.

The idea from Chapter 2 and Chapter 3 can be combined to improve our document classification model. Iterative attention mechanism can be used for word attention and sentence attention to find out the most relevant information in multiple steps. The RNN word encoders and sentence encoders can also be replaced by the Transformer encoder [119]. Our hierarchical attention network can extract the most important information out of documents. Some quantitative experiments can be done in the future to investigate the accuracy of those attention maps. If supported by quantitative evidence, our framework has potential application in the task of extractive summarization.

We can improve our work of Chapter 4 by using more powerful decoder architectures such as the Transformer decoder [119]. The Transformer decoder is similar to a CNN which is non-recurrent hence satisfies our need for contextual capacity control. Our new variational autoencoder model can be generalized to the setting with conditional inputs, which has very broad applications such as dialogues modeling. We can use our model to generate diverse responses to each utterance. It can also be applied in machine translation systems to generate diverse translation outputs for a given input sentence.

The model from Chapter 5 is a preliminary framework and can be made more powerful by adding multiple losses. One loss to add is the cycle-consistency constraint [143] to ensure that the content is preserved during the transfer. Our framework also has broad potential applications. One direction is to extend it to the semi-supervised setting and use it to help with many other sequence level tasks such as machine translation, summarization etc. With large amount of monolingual data, we can improve upon the state-of-the-art machine translation systems with our language model discriminators framework. It also can be applied to multi-modal settings with image, text or speech as inputs and outputs.

Bibliography

[1] 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. 5.5

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018. 2.5

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016. 2.5

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. *arXiv preprint arXiv:1505.00468*, 2015. 2.1, 2.2, 2.4.1, 2.4.2, 2.4.4, **??**

[5] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017. 5.4

[6] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017. 5.2, 5.5.3

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 6

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2.1, 3.1, 3.2, 3.3.1, 3.3.2

[9] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014. 4.2

[10] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014. 3.1

[11] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015. (document), 4.1, 4.2, 4.3.1, 4.3.2, 4.4.2, 4.2, 4.4.3, 5.1, 5.3

[12] Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007. 5.2

[13] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017. 5.2

[14] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659, 2016. 3.5

[15] Jianshu Chen, Po-Sen Huang, Xiaodong He, Jianfeng Gao, and Li Deng. Unsupervised learning of predictors from unpaired input-output samples. *arXiv preprint arXiv:1606.04646*, 2016. 5.2

[16] 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. 5.1, 5.2

[17] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016. 4.1, 4.2

[18] Xinlei Chen and C Lawrence Zitnick. Learning a recurrent visual representation for image caption generation. *arXiv preprint arXiv:1411.5654*, 2014. 2.2

[19] 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. 5.5

[20] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015. 4.2

[21] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015. (document), 4.1, 4.2, **??**, **??**, **??**, **??**, 4.4, 4.4.4

[22] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015. 5.2

[23] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM, 2014. 3.4.1

[24] Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018. 4.5

[25] Qing Dou and Kevin Knight. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275. Association for Computational Linguistics, 2012. 5.5.1

[26] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. From captions to visual concepts and back. *arXiv preprint arXiv:1411.4952*, 2014. 2.2, 2.3.2, 2.4.1

[27] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pages 2199–2207, 2016. 4.2

[28] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*, 2017. 5.5, **??**

[29] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. *arXiv preprint arXiv:1505.05612*, 2015. 2.1, 2.2, 2.3.1

[30] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2014. 3.1

[31] 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. 5.2

[32] Aidan N Gomez, Sicong Huang, Ivan Zhang, Bryan M Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*, 2018. 5.2

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. (document), 1.2, 5.1, 5.3

[34] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2.4.3

[35] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 4.1, 4.2

[36] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pages 3549–3557, 2016. 4.2

[37] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015. 5.2

[38] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing*

Systems, pages 820–828, 2016. 5.2

[39] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019. 4.5

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4.3.3, 6

[41] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*, 2015. 1.1, 3.2

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2.1, 3.1

[43] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 2017. 4.2

[44] 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. 5.1, 5.2, 5.3, 5.5, **??**, 5.5.2, **??**, **??**, **??**, **??**, **??**, **??**, **??**, **??**

[45] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017. 4.2, 5.4

[46] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Xiaodan Liang, Lianhui Qin, Haoye Dong, and Eric Xing. Deep generative models with learnable knowledge constraints. *arXiv preprint arXiv:1806.09764*, 2018. 5.2

[47] Zhiting Hu, Zichao Yang, Tiancheng Zhao, Haoran Shi, Junxian He, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Lianhui Qin, et al. Texar: A modularized, versatile, and extensible toolbox for text generation. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 13–22, 2018. 5.5

[48] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR, abs/1703.06868*, 2017. 5.2

[49] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017. 5.2

[50] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4.3.4, 5.4

[51] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998. 3.1

[52] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014. 3.1, 3.2

[53] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016. 4.1, 4.2, 4.3.3, 4.3.3

[54] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016. 4.2

[55] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014. 2.2

[56] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 2.3.2, 3.1, 3.2, 3.4.2

[57] Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018. 4.5

[58] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4.4.2, 5.5

[59] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. (document), 1.2, 4.1, 4.2, 4.3.1, 4.3.1, 5.1

[60] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 4.3.4, 4.4.4

[61] Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016. 4.2

[62] Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, pages 723–762, 2014. 3.4.2

[63] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 2.2

[64] 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. 4.2

[65] 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. 2.3.1, 4.3.3

[66] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015. 3.2, 3.3.2

[67] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 3.2

[68] 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. 5.1, 5.3

[69] Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017. 5.2, 5.5.3

[70] Hugo Larochelle and Stanislas Lauly. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716, 2012. 4.1

[71] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014. 4.2

[72] 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. 2.1, 3.2

[73] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015. 3.2

[74] Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017. 5.1, 5.2

[75] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*, 2018. (document), 5.5, 5.5.2, **??**, 5.3

[76] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165, 2017. 5.2

[77] Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907, 2015. 3.2

[78] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014. 2.4.1

[79] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015. 3.3.2

[80] Yu Liu, Jianshu Chen, and Li Deng. Unsupervised sequence classification using sequential output statistics. In *Advances in Neural Information Processing Systems*, pages 3550–3559, 2017. 5.2

[81] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016. 2.5

[82] Lin Ma, Zhengdong Lu, and Hang Li. Learning to answer questions from image using convolutional neural network. *arXiv preprint arXiv:1506.00333*, 2015. 2.2, 2.3.1, 2.4.1, 2.4.2, 2.4.4, **??**, **??**, **??**

[83] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A con-

tinuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016. 4.3.4

[84] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 4.4.5

[85] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690, 2014. 2.1, 2.2, 2.4.1, 2.4.2, 2.4.4, **??**, **??**, **??**, **??**

[86] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. *arXiv preprint arXiv:1505.01121*, 2015. 2.1, 2.2, 2.4.1, 2.4.2, 2.4.4, **??**, **??**

[87] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. 3.4.3

[88] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014. 2.2

[89] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *Proc. ICML*, 2016. 4.1, 4.2

[90] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010. 1.1, 4.3.1

[91] 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, 2013. 3.4.2, 3.4.3

[92] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014. 4.1

[93] Nikolaos Pappas and Andrei Popescu-Belis. Multilingual hierarchical attention networks for document classification. *arXiv preprint arXiv:1707.00896*, 2017. 3.5

[94] Nima Pourdamghani and Kevin Knight. Deciphering related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2513–2518, 2017. 5.5.3

[95] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*, 2018. 5.2

[96] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017. 3.5

[97] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 5.1, 5.2

[98] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. *arXiv preprint arXiv:1505.02074*, 2015. 2.1, 2.2, 2.3.1, 2.4.1, 2.4.2, 2.4.4, **??**, **??**, **??**

[99] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015. 4.2

[100] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 4.2

[101] Tim Salimans, Diederik P Kingma, Max Welling, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, volume 37, pages 1218–1226, 2015. 4.1, 4.2

[102] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. 5.2

[103] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*, 2016. 4.2

[104] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844, 2017. 5.1, 5.2, 5.3, 5.3, 5.5, **??**, 5.5.1, **??**, 5.5.1, 5.5.2, **??**, 5.5.3, **??**, **??**, **??**, **??**, **??**, **??**, **??**, **??**, **??**

[105] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014. 2.3.2, 3.1

[106] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2.3.1

[107] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*, 2013. 3.2

[108] 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. 2.4.3

[109] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015. 3.2, 3.3.2

[110] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 1.1, 2.3.2

[111] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in*

neural information processing systems, pages 1057–1063, 2000. 5.1, 5.2, 5.4

[112] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 2.3.1

[113] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL*, 2015. 3.2

[114] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565, 2014. 3.4.2

[115] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015. 3.1, 3.2, 3.4.1, 3.4.2, 3.4.2, 3.4.2, 4.4.1

[116] Duyu Tang, Bing Qin, and Ting Liu. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*, 2016. 3.5

[117] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016. 4.1, 4.2

[118] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 4.1, 4.2, 4.3.3

[119] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 6

[120] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015. 1.1

[121] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014. 3.2

[122] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014. 2.2

[123] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012. 3.1

[124] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994. 2.4.2

[125] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for

visual and textual question answering. In *International conference on machine learning*, pages 2397–2406, 2016. 2.5

[126] Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*, 2018. 4.5

[127] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015. 2.1, 2.2, 3.1, 3.2

[128] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016. 4.1, 4.2

[129] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*, 2017. 5.2

[130] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*, 2015. 3.2

[131] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016. 1.2

[132] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489, 2016. 1.2, 4.4.1

[133] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*, 2017. 1.2, 5.1, 5.3

[134] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. *arXiv preprint arXiv:1805.11749*, 2018. 1.2

[135] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 4.3.3

[136] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017. 5.1, 5.2

[137] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018. 3.5

[138] Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*, 2016. 4.2

[139] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*, 2015. 3.1, 3.2, 3.4.1, 3.4.2, 3.4.2, 3.4.2, 4.4.1

[140] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 5.2

[141] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015. 3.2

[142] Chen Zhu, Yanpeng Zhao, Shuaiyi Huang, Kewei Tu, and Yi Ma. Structured attentions for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1291–1300, 2017. 2.5

[143] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017. 5.1, 5.2, 5.3, 6