Going Deeper with Images and Natural Language

Yufeng Ma

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Weiguo Fan, Chair
Edward A. Fox, Co-Chair
Bert Huang
Alan G. Wang
Zhongju Zhang

February 20, 2019
Blacksburg, Virginia 24061

Keywords: Image Captioning, Quasi-Supervised Learning, Image Aspect Mining,
GANs, Deep Learning

Going Deeper with Images and Natural Language

Yufeng Ma

ACADEMIC ABSTRACT

One aim in the area of artificial intelligence (AI) is to develop a smart agent with high intelligence that is able to perceive and understand the complex visual environment around us. More ambitiously, it should be able to interact with us about its surroundings in natural languages. Thanks to the progress made in big data collection, computing resources, and deep learning models, we've seen huge breakthroughs towards this goal over the last few years. The development has been extremely rapid in the computer vision community in which machines can now categorize images into multiple classes, and detect various objects within an image, with an ability that is competitive with or even surpasses that of humans. Meanwhile, we also have witnessed similar strides in natural language processing (NLP). It is quite often for us to see that now computers are able to almost perfectly do text classification, machine translation, etc. However, despite much inspiring progress, most of the achievements made are still within one domain, not handling inter-domain or multi-modal situations. The interaction between the visual and textual areas is still quite limited, although there has been progress in image captioning, visual question answering, etc.

In this dissertation we design models and algorithms that enable us to build in-depth connections between images and natural languages, which help us to better understand their inner structures. In particular, first we augment the conventional image captioning model using adversarial loss, to boost its performance on various metrics (e.g., BLEU, CIDEr). Generative adversarial networks (GANs) are connected with discrete caption data through the Gumbel-Softmax trick. The results of our experiments with the MSCOCO caption dataset show that our model is capable of generating captions with improved quality and higher readability.

Second, we develop a model for measuring review congruence, which takes an image and review text as input and quantifies the relevance of each sentence to the image. The whole model is trained in a purely unsupervised way, much like GANs. Although there are no manual tags involved in training the model, we build pseudo labels with the help of attention mechanisms, which is a novel unsupervised training paradigm. The model is able

to accurately estimate the relevance of sentences to an image based on our experimental results on the Yelp Restaurant dataset.

Lastly, we go deeper with fine-grained image understanding by identifying aspect level ratings and by detecting corresponding aspects from the image within the review context, which is considered a novel AI task. This would help merchants provide better service and readers make choices more easily. Adaptive models are proposed to automatically adjust weights placed on images and texts for overall rating prediction. It shows significant improvement as compared to our baseline model based on the Mean Squared Error (MSE) metric in terms of both overall and aspect level rating.

We argue that these models, the techniques involved, and the interactions they enable would be key components in the move towards general artificial intelligence. Furthermore, it's also promising that the deeper connections between images and natural languages will enable various practical applications.

Going Deeper with Images and Natural Language

Yufeng Ma

GENERAL AUDIENCE ABSTRACT

One aim in the area of artificial intelligence (AI) is to develop a smart agent with high intelligence that is able to perceive and understand the complex visual environment around us. More ambitiously, it should be able to interact with us about its surroundings in natural languages. Thanks to the progress made in deep learning, we've seen huge breakthroughs towards this goal over the last few years. The developments have been extremely rapid in visual recognition, in which machines now can categorize images into multiple classes, and detect various objects within an image, with an ability that is competitive with or even surpasses that of humans. Meanwhile, we also have witnessed similar strides in natural language processing (NLP). It is quite often for us to see that now computers are able to almost perfectly do text classification, machine translation, etc. However, despite much inspiring progress, most of the achievements made are still within one domain, not handling inter-domain situations. The interaction between the visual and textual areas is still quite limited, although there has been progress in image captioning, visual question answering, etc.

In this dissertation, we design models and algorithms that enable us to build in-depth connections between images and natural languages, which help us to better understand their inner structures. In particular, first we study how to make machines generate image descriptions that are indistinguishable from ones expressed by humans, which as a result also achieved better quantitative evaluation performance. Second, we devise a novel algorithm for measuring review congruence, which takes an image and review text as input and quantifies the relevance of each sentence to the image. The whole model is trained without any supervised ground truth labels. Finally, we propose a brand new AI task called Image Aspect Mining, to detect visual aspects in images and identify aspect level rating within the review context.

On the theoretical side, this research contributes to multiple research areas in Computer Vision (CV), Natural Language Processing (NLP), interactions between CV&NLP, and Deep Learning. Regarding impact, these techniques will benefit related users such as the visually impaired, customers reading reviews, merchants, and AI researchers in general.

Acknowledgement

There are varieties of people I must thank for playing a significant role in more than six years of my experience as a Ph.D. student.

First, I really would like to deliver my heartfelt gratitude to my advisor Dr. Weiguo (Patrick) Fan who, through countless emails, WeChat messages and meetings, molded and chiseled me from a student who knows nothing to an independent researcher. Dr. Fan is like a generative model of unique perspectives and insightful ideas, and can also be treated as a good discriminator, both of which direct me to explore with my own interests and capabilities (identify a prior input distribution) and then train me to produce competitive research results. Same sincere appreciation shall also be given to my co-advisor Dr. Edward A. Fox, whose passion, foresight, ambition and zeal for perfection have become my constant motivation to keep improving myself. It is their unfailing guidance, caring, encouragement, patience, and kindness, that supported me to successfully complete my Ph.D. program. Every moment we spent together on research discussion, paper and dissertation writing, and project collaboration, is highly cherished.

Besides my advisors, I am also very grateful to other professors in my committee: Dr. Bert Huang, Dr. Alan G. Wang, and Dr. Zhongju (John) Zhang. Thanks for generously serving as my committee members. I have benefited a lot from their diverse expertise, which helps deepen my understanding in research. Special thanks go to Dr. Stefan Lee who opened my door to (theoretical) machine learning and Dr. Dhruv Batra who guided me to explore and step further in deep learning. It's also fortunate to collaborate with Dr. Zheng (Phil) Xiang, Dr. Wenqi Shen, Dr. Florian Zach, and many others, who gave me insightful suggestions during research project collaborations.

Many thanks to my fellow colleagues who intersect with my day-to-day life and who

have made it an unforgettable experience for me at Virginia Tech. From the Digital Library Research Laboratory (DLRL) and Center for Business Intelligence and Analytics (CBIA): Xuan Zhang, Liuqing Li, Ziqian Song, Prashant Chandrasekar, Sunshin Lee, Mohamed Magdy, Zhilei Qiao, Qianzhou Du, Long Xia, Yu Wang, Xinyue Wang, Siyu Mi, Hongpeng Wang, etc.

I've been very fortunate to have the opportunity to be a research intern mentored by Dr. Rao Shen and Dr. Kapil Thadani at Yahoo! Research (Oath), who helped shape my thinking and research philosophy for model design and inspire me to think bigger and aim higher.

A big thank you to my 'brother' and roommate Yumin Dai who always has ten times more confidence in me than I do. It is his unending encouragement and great help that drove me to survive the hardest period during my Ph.D. life. And also a big thank you to my parents, who are always having faith in me and supporting me. It is in large part my determination to make them proud of me that motivates me to do better.

Thanks go to the Department of Computer Science, Advanced Research Computing (ARC), Pamplin College of Business, DLRL, Deloitte, and Mayfair Group for the funding and facilities which supported me through my Ph.D. research.

Last and foremost, it is so blessed for me to know God and become a Christian when I am about to finish my Ph.D. program. Thanks God!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the area of Artificial Intelligence (AI), humans have dreamed that one day in the future machines will be able to perceive and respond to the surrounding complex visual environment and interact with us in natural languages.

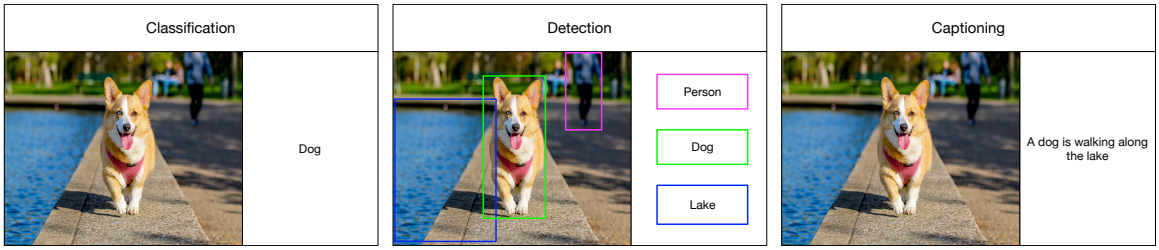


Figure 1.1: Visual recognition granularities. **Left**: an image is assigned a label. **Center**: localize and identify objects in the image. **Right**: describe an image with a sentence.

Humans would find it fairly straightforward to achieve varieties of tasks that incorporate complex image processing, object detection & recognition, and semantic understanding; ones that deal with interactions in natural languages; and tasks that build strong connections between both image and natural language modalities. For instance, just a glimpse at a picture is sufficient for humans to recognize all of the objects and describe the visual surroundings with a large number of details. With the example shown in Figure 1.1 indicating different levels of visual recognition, we can take a quick look at the image, and

then without any doubt point out the "walking dog", the "unclear pedestrian", the "people sitting on bench", or simply describe the entire image as "a dog is walking along the lake".

More capably, in the social media or online review domains, humans are adept at connecting semantics in text automatically with concepts identified in images. For example, in Figure 1.2, humans are inherently able to correlate relevant descriptions in a set of sentences with respect to a particular image. This situation is ubiquitous in reviews or posts that include both images and text. It is also not that difficult for us to identify aspect level sentiment in addition to the overall rating for a review. However, with the explosion of big data, we are unable to literally look through all these. Can we endow our computers with such capabilities so that machines can help us understand the context so that we can take one step closer to general AI?

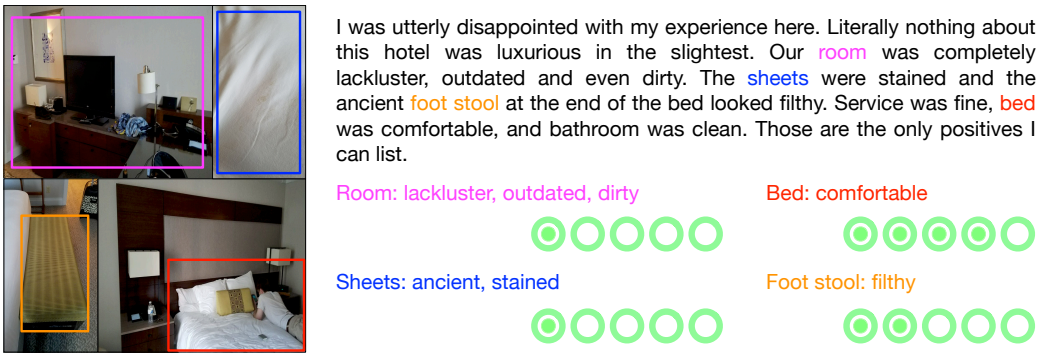


Figure 1.2: Deeper connection between text and image in review

Challenges. Since the above mentioned human ability feels so natural and effortless, it can be easy for us to forget how difficult these tasks are for a machine. In computers, images are usually characterized using big arrays of real numbers showing brightness at all positions. With such an input, a computer must learn to turn the patterns found in these numbers into high level, interpretable concepts like a "dog". Furthermore, a different breed of dog viewed with different brightness, in a different posture, or under a different camera angle, should still be depicted as "a dog is walking along the lake", but the statistics of real numbers in the array could be disparate. Conversely, various objects (coats, carpets, etc.) or other animals (monkeys, cats, bears, etc.) could also share some patterns, whose low-level statistics could be quite similar (e.g., fur-like high frequency patterns). The challenges from the natural language processing side are no less severe. In the computer, a sequence of integers, which are picked as indices of each word from a pre-defined vocabulary, may

be used to represent a natural language description such as "a dog is walking along the lake". Likewise, in order to point out and localize different parts of an image, a computer needs to pinpoint salient subsets of millions of pixels and simultaneously annotate them with integers, which should be a complex pattern recognition process. Clearly, the computational complexity is very high for the task of accurately predicting the relevance of each sentence of a review to an image, by correlating a sequence of integers with millions of pixel values. It would become even more difficult to extract human sentiment from image pixel values together with a text integer sequence, as it requires the identification of the semantic context first.

1.2 Problem Statement

Encouraging progress. Despite the difficulty of these tasks, dramatic advance has been witnessed in the area of visual recognition during the past decade. In particular, deep convolutional neural network (LeCun et al., 1998) based image classifiers have been current state of the art network architectures such as the VGG network (Simonyan and Zisserman, 2014), residual network (He et al., 2016), and densely connected network (Huang et al., 2017). They are capable of identifying thousands of visual categories, which is comparable to the ability of humans (Russakovsky et al., 2015). Similar rapid trends could also be discovered on related tasks such as object detection and semantic segmentation (Ren et al., 2015; Yu and Koltun, 2015). Meanwhile, on the natural language processing (NLP) side, excellent performance on machine translation (Wu et al., 2016a), abstractive summarization (See et al., 2017), etc., have been spotted with the help of diverse deep learning algorithms. All this progress has made many real-world applications achievable, including face recognition and detection, personal photo search based on text input, image style transfer, neural machine translation, text classification, etc. However, we can see that most of the achievements made are still within one domain, not handling inter-domain or multimodal situations. The interaction between the visual and textual areas is still quite limited, although there has been progress in image captioning (Lu et al., 2017; Xu et al., 2015), visual question answering (Antol et al., 2015; Srivastava and Salakhutdinov, 2012), etc.

Based on our motivation and the recent advances in the field of interactions between Computer Vision (CV) and NLP, we explore the following three problems, with related

goals:

1. Generating high quality free-form natural language descriptions for a given image. Specifically, captions generated by the machine should be indistinguishable from ones written by people.

2. Quantifying the congruence/relevance of each sentence to an image in the same review. Specifically, review sentences should be ranked correctly based on their relevance scores with respect to the given image, which are obtained through models trained without ground truth labels.

3. Recognizing topical aspects in a review represented in free-form natural language, matching these topical aspects with specific regions in an image, and outputting corresponding ratings. Identified aspects need to be detected in the image first. Then, together with the semantics inferred from the text, a fine-grained rating for each aspect should be produced.

1.3 Hypotheses

Our proposed approach would use datasets that have both images and texts. Their correct inner connections can be captured mathematically with complex neural networks.

Principal Hypothesis. Specifically, the output from a convolutional neural network (CNN) can more accurately deliver high-level abstractive concepts in images, while the recurrent neural network (RNN) or recent proposed Transformer (Vaswani et al., 2017) model can more accurately convey textual semantics, as compared to traditional human-crafted feature extraction methods. In particular, we make the following individual hypotheses, addressing our research problems.

1. Given training data available in our experiments, generative adversarial networks (GANs) are able to accurately approximate the underlying distribution of the data. Including a Gumbel-Softmax sampler in an image captioning model enables training of that model under the GANs framework by converting a discrete to a continuous representation, which we predict will boost quantitative evaluation metrics.

2. Quantitative relevance between review image and sentences can be estimated through our proposed Quasi-Supervised Learning (QSL) network trained in a data-driven manner. More specifically, using a Quasi-Supervised Learning network improves the accuracy of the relevance scores of the review sentences matching a related image, beyond that of existing heuristic based methods.

3. When topical aspects (objects) detected in images also appear in the review text content, correspondence between image aspects and the ones identified in text can be predicted using our proposed Adaptive model. Based on this inferred correspondence strength, it can also estimate fine-grained information such as aspect level ratings; these improve upon baseline approaches.

1.4 Research Questions and Dissertation Organization

In this dissertation, we design and augment several models for connecting images with natural languages, which help us to answer *how and to what extent can our machines understand the connections between images and natural languages*. In particular, we design specific neural networks that take in and connect images with natural language. Afterwards, the corresponding parameters are then trained end-to-end on datasets of samples with both modalities.

In **Chapter** 2, required mathematical background for deep learning is provided, which includes supervised learning, unsupervised learning (with a focus on GANs), backpropagation, optimization, neural network architectures, and attention mechanisms. Besides, we also describe typical rules of thumb for choosing neural net architectures, which are commonly used to deal with image and text inputs. In particular, convolutional neural networks for spatial topological data, and recurrent neural networks for data with temporal structure, are described in detail.

In **Chapter** 3, in order to determine *how and by how much we can improve the quality of captions generated by machines*, we augment the conventional encoder-decoder model for image captioning with generative adversarial networks (GANs). That is, the captioning model is first treated as a generator in the GANs framework, then, based on the real captions written by humans, we are training a discriminator to distinguish between real and fake captions. Meanwhile, a caption generator is tuned to fool this discriminator and make the

discriminator treat its generated captions as real ones. We find that our augmented system is able to generate more coherent captions and present better performance as compared to the original captioning model.

In **Chapter** 4, we study the problem of matching an image with sentences, in which both the image and sentences come from the same review. We address the research question: *What is an effective way of learning congruence between an image-sentence pair if there is no supervised label?* In particular, we propose a brand new learning framework based on the attention mechanism and review structures, which helps to build pseudo-labels for training. The idea again originates from the GANs training framework. Our experiments show that it is able to correctly build the connections between image and sentences, and identify the relevant sentences within a review.

Finally, in **Chapter** 5, we introduce a new AI task called Image Aspect Mining. In particular, we address the research problem: *How can we connect aspects mined from text and ones detected in images, and do fine-grained sentiment/rating inference?* As compared to previous overall rating prediction for the whole review, it can be treated as fine-grained level image understanding. Such models should bear the ability to detect image aspects based on object detection frameworks and then infer sentiment together with the context from review texts and image details. We propose intuitive neural network architectures (Adaptive models) to tackle the problem. Experimental results demonstrate our models' capability to extract hidden aspects from text semantics and infer aspect level ratings from image context. Our method also presents better quantitative performance with both the overall rating and aspect level rating mean squared errors (MSEs) as compared to our baseline models.

Chapter 2

Deep Learning Background Review

In this chapter, some basic knowledge on traditional machine learning and deep learning (neural network based approach) is provided. If readers would like to have a more detailed understanding about the background, we would highly recommend the book written by Goodfellow et al. (2016), which covers all kinds of necessary information required for deep learning research. Afterwards, we also discuss how we leverage the techniques introduced here to build our models for deeper understanding of images and texts.

2.1 Supervised Learning

In supervised learning, a large portion of problems can be converted to problems involving learning a mathematical mapping f, which transforms any sample in an input space X to an output space Y. For example, in image classification, we might take in images which constitute the input space X and output a value in between $[0, 1]$ that indicates how likely a dog appears in the image. Most of the time it is difficult to manually define a function f that fulfills such requirements (e.g., we don't know the exact process of how humans recognize a dog, which makes it harder to mathematically formalize such a function.) Fortunately, examples $(x, y \in X \rightarrow Y)$ that satisfy our desired mapping f are relatively more easily to get, which is what the supervised learning paradigm relies on. Therefore as in our visual recognition example, images annotated by humans indicating whether there is a dog inside, can be collected for the purpose of learning the mapping in supervised learning.

The objective function. More specifically, suppose we have an underlying data generating distribution D, from which n training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ are assumed to be sampled and it's obvious that they are independent and identically distributed. Our goal of learning the mapping $f : X \to Y$ then becomes finding a function among candidates, which should be the most consistent with our training dataset. To be more accurate, some particular class of functions $\mathcal{F}$ first needs to be selected for the candidates. Then we may turn to a scalar-valued evaluation function $L(\hat{y}, y)$, which should quantify how the estimated output $\hat{y}_i = f(x_i)$ for a particular $f \in \mathcal{F}$ aligns with human annotated label y_i. Finally, we would like to find the optimal function $f^* \in \mathcal{F}$, which is obtained by the following optimization:

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(x,y)\sim D} L(f(x), y). \tag{2.1}$$

That is to say, the function f^* is found by minimizing the expected loss among all examples, i.e., in the hope that these samples can mimic the underlying assumed distribution D. After we've identified this optimal function, in most practical applications, it is common that the training dataset is discarded and we only use the learned f^* to predict $\hat{y}$ for each input element x.

Unfortunately, it is difficult for us to pick a known distribution D that exactly matches the training dataset, which makes it infeasible to analytically evaluate the expectation. However, thanks to the i.i.d. assumption, we may just average the loss over all training samples to approximate the expectation in Equation 2.1, i.e., we are assuming that the data generating process satisfies the empirical distribution:

$$f^* \approx \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i). \tag{2.2}$$

Therefore, we are optimizing the loss function only with accessible samples (x_i, y_i), for which we hope it can accurately approximate the actual desired objective.

Regularization. However, some challenges would appear if we optimize Equation 2.2 rather than Equation 2.1. For example, our mapping function f can be enforced to map each x_i to its desired value y_i in the training data, but behave arbitrarily elsewhere. As desired, this would map x_i to y_i perfectly under Equation 2.2, which would always achieve

minimum values for any reasonable loss function, but what about all the remaining points in D that do not show up in the dataset? It's sensible that we would expect a very high loss for such functions, namely the generalization ability is not that high. Besides, varieties of functions could also achieve the same loss value with Equation 2.2, but for data points outside the available training data, their loss values would differ accordingly, which could indicate distinct generalization capability. Then which function $f \in F$ that arrives at the same loss should we turn to if we can get access to the training data? We may possibly alleviate both of these concerns if our original optimization function is incorporated with a regularization term R as:

$$f^* \approx \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + R(f), \tag{2.3}$$

in which R stands for a function that outputs a scalar value for any f. It measures f's alignment with our preference for some specific functions. This addition can also be justified from the perspective of Bayesian statistics. Instead of maximum likelihood, maximum a posteriori (MAP) estimation enables us to add some priors on the parameters of f. Therefore with the regularization term, desired simple solutions that satisfy our requirement for f would be encouraged, which should also behave well on the training data.

Summary. In supervised learning problems, a training dataset of n examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ in which $(x_i, y_i) \in X \to Y$ is first provided to us and then we need to identify the following three entities to mathematically define the problem:

1. $\mathcal{F}$: function candidate space, in which each reasonable f would map every input element in X to Y.

2. $L(\hat{y}, y)$: a scalar valued evaluation function that quantifies how different the prediction $f(x)$ is from the label y annotated by humans.

3. $R(f)$: another scalar valued function used for regularization purposes that tweaks f in favor of our pre-defined preferences.

In deep learning, we will typically turn to a neural network defined by parameters θ for the function candidate space $\mathcal{F}$. As for loss function L, the cross entropy loss (equivalent to the use of KL divergence loss) is typically selected in classification tasks while Euclidean loss

is preferred for regression. Finally, it's common to choose the l_2 norm of the parameters θ as the regularization term, whose behavior favors parameters that satisfy a Gaussian distribution with $\mathbf{0}$ mean.

As soon as we've finalized these three decisions, an optimization problem with the mathematical form $\theta = \arg\min_\theta g(\theta)$ would appear in front of us, which represents the learning objective for a supervised learning task. It should be noted that $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f_\theta(x_i), y_i) + R(f_\theta)$, in which θ stands for a vector that includes all trainable parameters. Besides, only the function f represented by a neural network would contain these parameters θ, among which no parameter belongs to either the loss function L or the regularization term R.

We next turn to another learning paradigm, i.e., unsupervised learning, with a focus on generative adversarial networks.

2.2 Unsupervised Learning - GANs

Generative adversarial networks or GANs (Goodfellow et al., 2014) are recognized as the most popular generative modeling approach proposed recently. They are composed of both differentiable generator and discriminator networks.

Generative adversarial networks are built upon an adversary game play, in which a generator network aims to compete against a discriminator network. The purpose of a generator network is to generate data samples $x = g(z; \theta^{(g)})$, which resemble ones from a real dataset. At the same time, a discriminator network (the generator's adversary) is incorporated to determine whether or not the input sample to the discriminator is sampled from the real training dataset. A probability of $d(x; \theta^{(d)})$ will be produced by the discriminator for each input x, which indicates how likely the input x is a real training sample instead of a fake one fabricated by the generator model.

In order to formulate the learning process in GANs, it's straightforward to incorporate a zero-sum game with both the generator and discriminator. A function $v(\theta^{(d)}, \theta^{(g)})$ is defined to determine the payoff output by the discriminator. At the same time, the generator will receive a payoff of $-v(\theta^{(d)}, \theta^{(g)})$ accordingly. Then in the process of learning, each

network/player will try to optimize based on its own payoff, which would result in

$$g^* = \arg\min_g \max_d v(d, g). \tag{2.4}$$

The default and most popular choice for v is

$$v(\theta^{(d)}, \theta^{(g)}) = \mathbb{E}_{x \sim p_{\text{model}}} \log(1 - d(x)) + \mathbb{E}_{x \sim p_{\text{data}}} \log d(x). \tag{2.5}$$

which was proposed in Goodfellow's original paper (Goodfellow et al., 2014). The objective function above trains the discriminator to do binary classification, choosing between real or fake. Meanwhile, the generator can be tuned to fool the adversary discriminator and make it believe that its generated samples are real. Finally, at convergence, both samples from the real dataset and the generator would be treated the same by the discriminator as it's supposed to output $\frac{1}{2}$ everywhere. Afterwards, we may discard the trained discriminator and just rely on the generator to produce unlimited data. The whole framework is depicted in Figure 2.1.

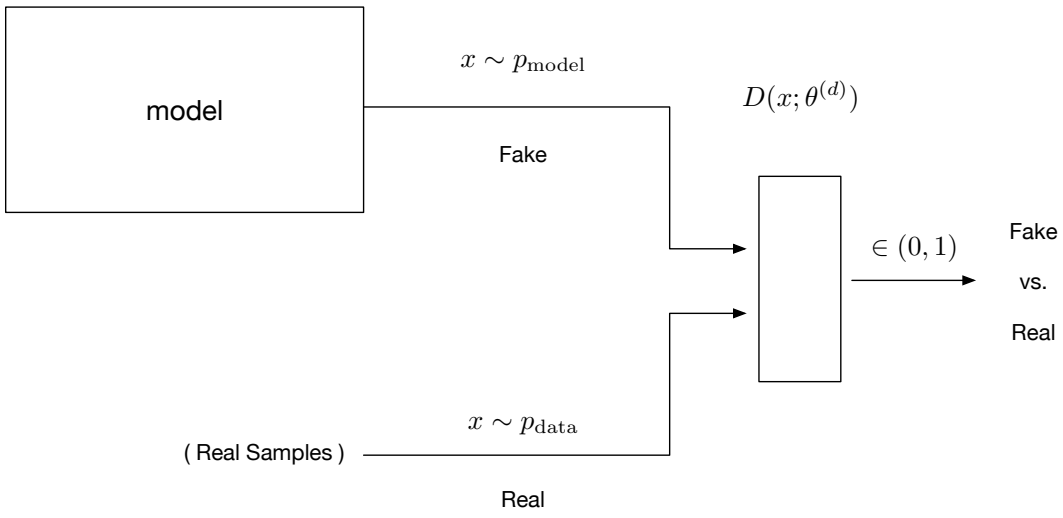


Figure 2.1: Generative adversarial networks framework

Both the previous supervised learning and the GANs framework here can be turned into some specific optimization problems. Following is a discussion of how to solve such optimization problems.

2.3 Optimization

As we've stated in the supervised learning Section 2.1, the task of learning a model can be reduced to an optimization problem, whose objective function is $\theta^* = \arg\min_\theta g(\theta)$. θ covers all the parameters and is treated as a column vector, while g stands for the average loss of training examples along with a regularization term R.

First order methods. Many of today's deep neural network models are restricted only to use differentiable functions. Based on the assumption of differentiation, the gradient $\nabla_\theta g$ can be computed by the use of the backpropagation process, the details of which are covered in the following Section 2.4. The gradient of a function at a specific point is a vector of partial derivatives, which has the same size as the original parameter vector. It tells us how quickly or slowly each dimension of θ would change with respect to the value of function g. The gradient is typically used as a search direction, which guides us on where we should move next. In particular, in order to get a lower value of g, we'll have to add a small amount of the negative gradient to the original parameter because what we want is to arrive at the minimum of g while the gradient indicates the direction and magnitude of increasing g. Inspired by this insight, the typical gradient descent (GD) algorithm coordinates with two processes: 1) forward with current input and apply backpropagation to compute the gradient; 2) update the parameters by moving along the negative direction of the gradient with a small step. Besides, the datasets we use in practice are often quite large with thousands up to millions of training examples, which makes it computationally infeasible to calculate the gradient efficiently. Instead, we would rather estimate it based on a small mini-batch of samples numbering around 32 or 64. In this way, many approximate updates could be executed rather than doing much fewer exact updates, which has been verified to be fairly effective both theoretically and empirically in real world applications (Smith et al., 2017). Algorithm 1 summarizes the mini-batch version of Stochastic Gradient Descent (SGD) that we've discussed above.

Advanced first order optimization methods. In practice, there are varieties of advanced approaches that converge much faster, which modify the update by taking advantage of the gradient direction. For example, the **Momentum** update just falls into this category, which accumulates updates from the consistent directions and neutralizes ones that are not. Let's use $\mathbf{g} = \nabla_\theta g(\theta)$ to denote the gradient vector. Then the update Δ_θ can

Algorithm 1 Mini-batch Stochastic Gradient Descent

Input: an initial point $\theta \in \mathbf{dom} g$.
Input: a small step size $\epsilon \in R^+$
repeat

1. Sample m examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ as a mini-batch from training data
2. Estimate the gradient $\nabla_\theta g(\theta) \approx \nabla_\theta [\frac{1}{m} \sum_{i=1}^{m} L(f_\theta(x_i), y_i) + R(f_\theta)]$ with backpropagation
3. Compute the update direction: $\Delta\theta := -\epsilon \nabla_\theta g(\theta)$
4. Perform a parameter update: $\theta := \theta + \Delta\theta$

until convergence

be calculated by first updating a temporary variable $v := \alpha v + \mathbf{g}$ that is initialized with zero, and flipping the direction of v as $\Delta\theta := -\epsilon v$. It should be noted that the variable v is an exponential decayed summation of previous gradients, which can help to cancel movements that are not directly pointing to the minimum. The Momentum update method is an example of manipulating the first moment of gradients, while there are also quite a few other methods that depend on the second moment. For instance, in the **Adagrad** update (Duchi et al., 2011), the summation of squared gradients is added to an intermediate variable as $\mathbf{r} := \mathbf{r} + \mathbf{g} \odot \mathbf{g}$, in which $\odot$ stands for element-wise multiplication. Afterwards, this variable $\mathbf{r}$ is used to change the magnitude of the gradient g as follows: $\nabla\theta := -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$, where a small epsilon δ is incorporated to prevent zero division. There is another method called **RMSProp** update (Tieleman and Hinton, 2012) that also relies just on the second moment, which calculates a running mean instead of pure addition. The running mean is calculated as $\mathbf{r} := \rho\mathbf{r} + (1-\rho)\mathbf{g} \odot \mathbf{g}$, where ρ is a hyper-parameter that balances how much weight we'd like to place on the history. Finally, there is the **Adam** update (Kingma and Ba, 2014) that combines RMSProp with Momentum, which incorporates both first and second moments tweaking. All these methods behave similarly as we update parameters: parameters that have traveled a longer distance would be assigned smaller moving steps and ones that move slowly can jump to a further point by taking larger steps.

2.4 Backpropagation

As we have seen above, we can apply stochastic gradient descent to minimize a function as long as we can get access to its gradient at each point. Accordingly, this corresponds to the process of finding mappings $f \in \mathcal{F}$ that are consistent with our training samples and also

match our prior preference characterized by regularization loss.

Now let's discuss how to perform the backpropagation process, which helps to calculate the gradients of scalar-valued functions with respect to training examples with high efficiency. The backpropagation algorithm applies the chain rule in calculus recursively. Recall that what we want is the gradients of function g, which is fed with the dataset of examples (x_i, y_i) and the parameters θ as input. To be more specific, it should be the gradient $\nabla_\theta g$ with respect to the parameters θ, which can be used to update the parameters. At the same time, if we want to visualize the examples based on their gradients, we may compute the gradients of each input x_i with the same procedure as well.

In short, the whole process of computing gradients can be reduced to successive products of local gradients, which corresponds to the Jacobian matrix between intermediate variables. In the scenario of a neural network function f, a forward pass is first performed based on a batch of training examples $\{(x_i, y_i)\}_1^m$ and parameters θ at the current point, during which all intermediate variables are evaluated and the final loss g is computed. Then during the backpropagation process, we start with the just evaluated final loss value and go backward through all the cached intermediate variables, which can be used to calculate the Jacobian matrix and then perform matrix multiplication. In many practical cases, there is no need to calculate the full Jacobian matrices as most activation functions are element-wise operations, which leads to diagonal matrix structure. This helps to chain the gradients with much higher efficiency.

2.5 Neural Networks

Previously, we've introduced that any differentiable function f can be used to transform inputs x to the desired outputs $\hat{y}$. Then the model might be optimized with the guidance from a loss function with the help of first order optimizers, such as the stochastic gradient descent algorithm. Now let's jump into the structures of the mapping f, which in our dissertation will typically be represented by a neural network.

2.5.1 Vanilla Neural Network

If we don't have any prior knowledge of the structure of input x, we may treat it as a column vector and construct neural nets in the form of matrix multiplication along with element-wise non-linear activations. For instance, a 2-layer neural network would be implemented as $f(x) = W_2\sigma(W_1x)$, where W_1, W_2 are parameter matrices while σ represents any element-wise non-linearity activation functions such as tanh or sigmoid. If we add one more hidden layer, a 3-layer network as shown in Figure 2.2 would have the form $f(x) = W_3\sigma(W_2\sigma(W_1x))$, etc. It should be noted that if we choose an identity function as the non-linearity, then the entire neural net would become a simple linear transformation. Typical non-linearity function choices include rectified linear unit (ReLU) $\max(0, x)$, tanh function $(e^x - e^{-x})/(e^x + e^{-x})$, and sigmoid function $1/(1 + e^{-x})$. Note that normally we would not apply any non-linearity function to the last layer of the neural net. It's also quite straightforward to notice that a simple linear mapping (a.k.a. a fully connected layer) is equivalent to a 1-layer neural network.

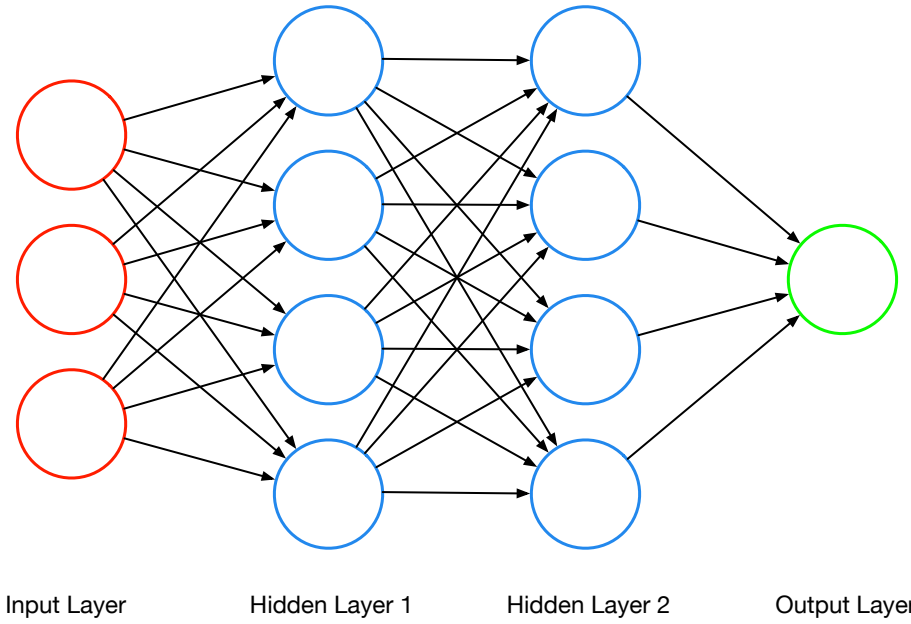


Figure 2.2: A neural network example composed of 3 layers showing the arrangement of neurons. Neurons in one layer are not connected with each other but are all connected to each neuron in the following layer. Such arrangement between neurons makes the evaluation of activation function quite efficient as we can apply simple matrix multiplications.

2.5.2 Convolutional Neural Network

If the data we process has a known grid-like topology such as images, videos, or sound spectrograms, which provide some prior knowledge about the dataset, we may design specific neural network architectures for them, i.e., the Convolutional Neural Networks (CNNs, or ConvNets) (LeCun et al., 1998). First, we should be aware that such input example x is represented in computers as a multi-dimensional array or a tensor. For example, a typical color image with the size of 256×256 would be a $256 \times 256 \times 3$ tensor as there are three color channels. We could represent a sound spectrogram as a $1,000 \times 128$ array, which indicates the amplitude of any one of 128 frequencies at different time steps $t = 1, \ldots, 1000$. What's more, we may segment some sentences into characters and represent them as an array of 100×26 if there are 100 characters in total and each is encoded as one-hot vectors. However, the dimensionality of these inputs would be extremely high (e.g., there will be approximately $200,000$ entries in the representation tensor), which will require fully connected layers with lots of connection. Therefore, neural networks that are sensitive to the inputs' spatial structure should be designed, which motivates the use of local connectivity and sharing parameters across different locations.

General Definition. Convolutional Neural Networks must have at least one Convolutional Layer, which is its key component that does the convolution operation by convolving an input tensor with some learnable filters. More generally, there are several properties that a convolutional layer must own for processing color images with multiple channels.

- It accepts a tensor of size $W_1 \times H_1 \times D_1$.
- There are 4 hyperparameters involved: the spatial size of a filter F, the number of filters K, the stride size we should skip when convolving S, and zero padding numbers around the input borders P.
- The output for a convolutional layer would be a tensor with size of $W_2 \times H_2 \times D_2$, where the output width $W_2 = (W_1 - F + 2P)/S + 1$, the height $H_2 = (H_1 - F + 2P)/S + 1$, and the depth $D_2 = K$.
- There are $F \times F \times D_1$ learnable parameters in each filter, which leaves us $(F \times F \times D_1) \times K$ weights and K biases to learn in total. It should also be noted that although the spatial extent for each filter is quite small $(F \times F)$, they will have to traverse along

the D_1 depth of the input tensor.

- For each output tensor, it is composed of D_2 slices of $W_2 \times H_2$ arrays, each of which results from convolution between the input tensor and one of the K filters.

Pooling layers. Besides the convolutional layers that help to reduce the number of learning parameters, pooling layers are usually incorporated to further decrease the representation size, which can control the overfitting problem as well. It is a fixed downsampling transformation, in which no learnable parameters are included. In particular, the pooling layers are applied independently upon each slice of the input tensor, which spatially downsample the activation map. The most common choice for a pooling layer is to do 2×2 max pooling with a stride step size of 2. As a result, the width and height of the input tensor is downscaled by 2 with the max pooling operation, for which the output size becomes $1/4$ of the original input size.

ConvNet Architectures. Finally, we can stack both CONV layers and pooling layers together to build our convolutional neural networks. Now let's turn to how we may organize these layers in more detail. A typical ConvNet takes the form of [INPUT, [CONV,CONV,POOL] $\times$3, FC, FC]. More specifically, mini-batches of images represented as tensors (e.g., 100 32×32 3-channel images result in a $[100 \times 32 \times 32 \times 3]$ tensor) make up the INPUT, CONV represents the typical convolution layer with some spatial size such as 3×3, POOL stands for a max pooling layer that downsamples the input by a factor of 2, and the FCs are linear mapping layers that are commonly used in a vanilla neural network. Normally, the last FC layer will map the previous input tensor to the dimensionality of classes, which can be used by a softmax classifier. A typical convolutional neural network for image classification is shown in Figure 2.3.

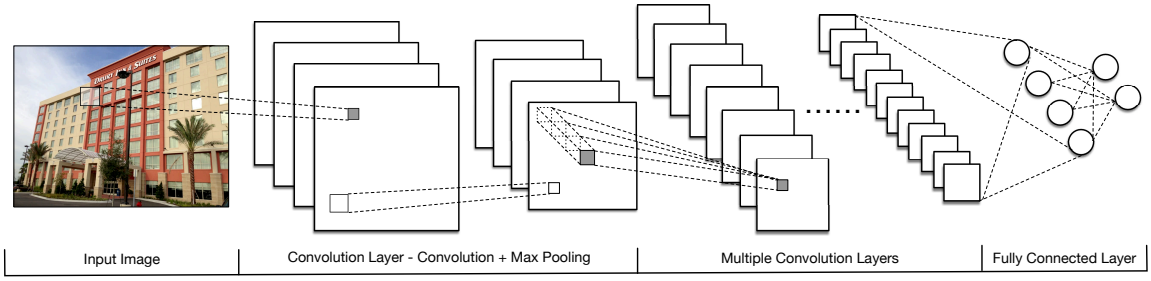


Figure 2.3: Typical convolutional neural network architecture for image classification

2.5.3 Recurrent Neural Network

Besides the spatial topology, there are many input or output spaces that show the structure of sequences, which are quite common in practical applications. For instance, we can model sentences as several sequences of words, in which each word is represented by a particular one-hot vector. Current deep learning models in NLP will typically start with one-hot vectors. It is a vector of all zeros except for the entry that stands for the index of the word in a pre-defined vocabulary. A sequence of vectors such as $x_1, \ldots, x_T$ can be fed into a recurrent neural network (RNN), which is adept at dealing with sequence data by using a recurrence formula. Its general form can be defined as $h_t = f_\theta(h_{t-1}, x_t)$, the details of which will be described below. It's worth noting that we will use the same parameters θ at every time step. It is this reusability that allows a RNN to process sequences of arbitrary length. As we go along different time-steps of a sequence, the hidden state vector h_t captures a summary of all vectors x it has witnessed before the time-step t and continues to update it with following input vectors. The starting hidden vector h_0 can be either initialized with $\vec{0}$ or treated as a parameter that should be learned. In different applications or models, the structures of the recurrence $(h_{t-1}, x_t) \to h_t$ vary a lot. Let's describe these in more detail next.

Vanilla Recurrent Neural Network (RNN) most often follows the recurrence form of

$$h_t = \tanh\left(W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right), \tag{2.6}$$

in which the current input x_t is concatenated with the previous hidden vector h_{t-1} and then they are linearly mapped to a new subspace with the learnable matrix W. Although Equation 2.6 is not the same as $h_t = \tanh(W_{\text{xh}} x_t + W_{\text{hh}} h_{t-1})$, we should be aware that they are actually equivalent considering that W_{xh} and W_{hh} are concatenated together and renamed as W. We've ignored the additional bias vector here for brevity. In terms of the activation function tanh here, some also have tried with either sigmoid or ReLU (Duch and Jankowski, 1999). Now let's look at the size of the parameter W, for which suppose our input x_t is of dimension D and there are H units in the hidden states h_t. Then the parameter W would be a $[H \times (D + H)]$ matrix. As we can see in the equation, the input elements x_t and the previous hidden states h_{t-1} are linearly transformed first, and then followed

by a non-linearity squashing function. It's desirable that the vanilla RNN's structure is quite simple, but unfortunately, various papers (Sutskever et al., 2011; Wu et al., 2016b) have shown that the naive addition is really weak for coupling the inputs together with the hidden states, i.e., the result hidden states won't be a good summary of all previous inputs. Besides, there appears to be unstable dynamics during backpropagation (Bengio et al., 1994) in models built with such vanilla RNNs. To be more specific, during the optimization, the gradients can either explode or vanish if the length of the input sequence is very large. We may heuristically clip the gradients at a specific maximum value (Pascanu et al., 2013) for gradient explosion, but the vanishing gradient problem still exists within such simple RNNs.

Long Short-Term Memory. To deal with the restrictions of vanilla RNN, the LSTM (Hochreiter and Schmidhuber, 1997) recurrence is specifically constructed. There are both multiplicative and additive interactions involved with the inputs x_t and the previous hidden states h_{t-1}, which allow them to behave more complicatedly but help the gradients to go backward more efficiently (Hochreiter and Schmidhuber, 1997). There is another memory vector c_t that is transmitted in addition to the usual hidden state h_t. The biggest contribution inside an LSTM would be the gating mechanism, which allows it to partially read from, write to, or reset the memory at each time step. Mathematically, one update of an LSTM cell can be formulated as:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \qquad \begin{aligned} c_t &= f \odot c_{t-1} + i \odot g \\ h_t &= o \odot \tanh(c_t) \end{aligned} . \tag{2.7}$$

There are two kinds of sigmoid activation functions here, namely sigm and tanh, which are element-wisely applied on top of their corresponding inputs. If the input x_t is of dimension D and the hidden state is a H-dimensional vector, then our parameter W would be a matrix of size $[4H \times (D + H)]$. We notice that there are three additional vectors $i, f, o \in R^H$ incorporated, which are the core building blocks that make LSTM a default choice for handling sequence data. They can be thought of as binary gates (either 0 or 1) that are able to control the intensity of the corresponding data flow. However, to keep the model still differentiable, sigmoid is selected as the activation function for these gates, which outputs a

value smoothly ranging in $(0, 1)$. Finally, the memory contents c_{t-1} are additively updated by the vector g, which relies on tanh as its activation and ranges between -1 and 1. It's critical to include this additive interaction, for which the gradients can be equally distributed during backpropagation. In this way, the gradients backpropagated through the memory cells won't be interrupted for a long period. In other words, the magnitude of the gradients can be kept during the backward pass, or just shrink by an active forget gate. This behavior is quite like the design of residual networks (He et al., 2016). A detailed diagram for LSTM data flow can be seen in Figure 2.4.

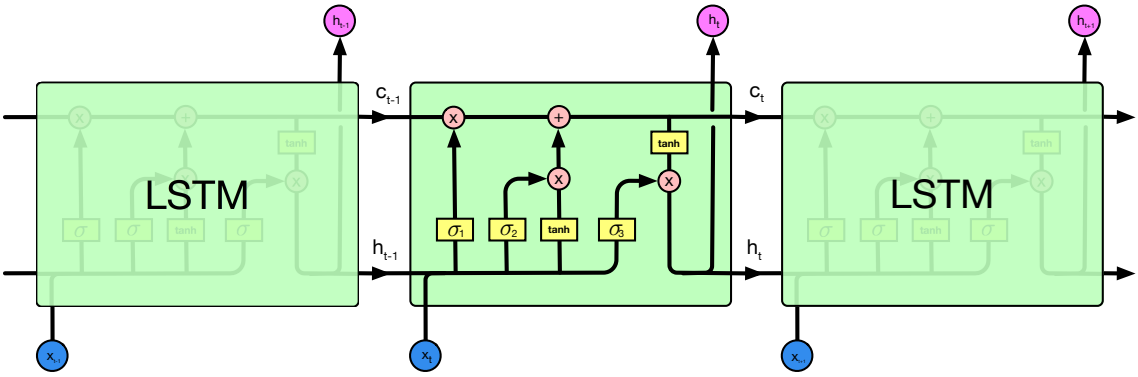


Figure 2.4: Data flow in Long and Short-Term Memory RNN

2.5.4 Transformer Self Encoder

Although Recurrent Neural Networks are good at dealing with sequence data of variable length, during the processing, they need to wait for the completion of previous input tokens before going forward, which may cause a time efficiency problem. In other words, the time complexity of a typical recurrent neural network would be $O(n)$, in which n stands for the total number of input tokens. With the introduction of the Transformer model (Vaswani et al., 2017) for machine translation, sequence transduction can be done via an attention mechanism together with some kind of positional encoding.

The input to a typical Transformer encoder is made up of queries Q, and key-value pairs K, V. Suppose we have q queries of dimension d_k, and n key-value pairs of dimension d_k

and d_v, then the scaled doc-product based attention is applied as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{2.8}$$

which gives us q retrieved embeddings represented by values vectors based on the similarity between queries and keys. The scale of $\sqrt{d_k}$ is to resolve the potential problem of small gradients when the dot product between Q and K has a large magnitude.

Instead of a single dot-product attention layer as implemented in Equation 2.8, it is beneficial to include multiple heads of such attention layers to push learning of various aspects. In terms of this, multi-head attention is incorporated, which will first map Q, K, V to some subspaces through linear transformations. Then we can do the attention transformation above and merge the learned embeddings together with another linear mapping. The mathematical formulation is described below as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.9}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2.10}$$

in which W_i^Q, W_i^K, and W_i^V do the linear mapping for subspace learning, while W^O maps the merged learned embedding for downstream task learning. Also it should be noted that h stands for the number of heads we desire, which is a hyper-parameter requiring tuning.

On top of the above multi-head attention, a feed-forward neural network is equipped, which is implemented as

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{2.11}$$

where W_1, W_2, b_1, and b_2 are parameters to be learned during training. Furthermore, each sub-module is followed by a residual connection (He et al., 2016) and layer normalization (Ba et al., 2016); this helps to stabilize the dynamics during optimization. The whole architecture of such a Transformer encoder is depicted in Figure 2.5.

Although we have successfully retrieved results based on the query process formulated in the above attention mechanism, by default we don't have such query and key-value pairs in a recurrent neural network learning problem. In terms of this, a self-encoder would make all the three entities the same, for which Q, K, and V all have the same value as all the

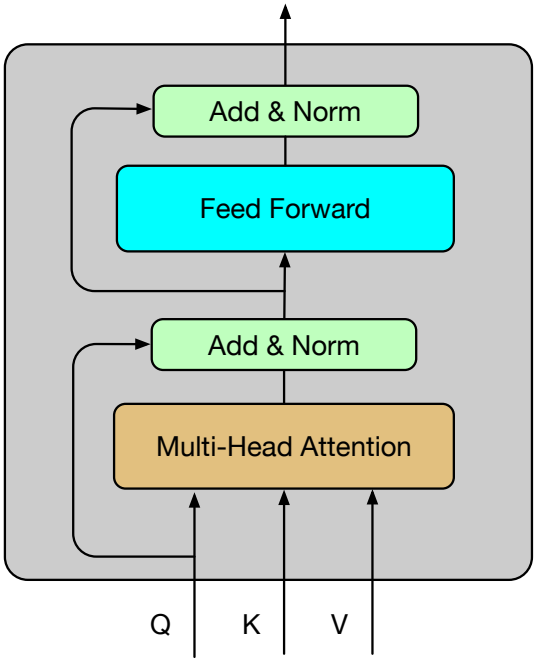


Figure 2.5: Transformer encoder architecture

input tokens. The intuition behind a self-encoder is to represent each input token using all the other tokens in the sequence, while a unidirectional recurrent neural network tries to encode the current input based on all previous tokens.

Besides, such transformer encoders should also incorporate position information since a RNN can inherently blend in this by processing input tokens one by one. However, for our transformer encoders, the input tokens can be treated as a bag of words; the order of words inside won't result in any difference. Therefore, in Vaswani et al. (2017), trigonometric functions like sine and cosine are used as:

$$PE(pos, 2i) = sin(pos/10000^{2i/d}) \tag{2.12}$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d}), \tag{2.13}$$

where *pos* stands for the position of an input token while i is the dimension in our final positional feature (a d-dimensional vector). It's intuitive to apply trigonometric functions here as the semantics of a word is determined by both previous and future words in the

word sequence, which behaves quite similarly to sine and cosine functions. For example, if we know the values of $sin(x), sin(y), cos(x)$, and $cos(y)$, then the values of these functions at a relative position like $x + y$ can be calculated as

$$sin(x + y) = sin(x)cos(y) + cos(x)sin(y) \tag{2.14}$$

$$cos(x + y) = cos(x)cos(y) - sin(x)sin(y). \tag{2.15}$$

This would mimic the relationship between the semantics of words in the same sequence, and we hope that trigonometric function can learn this with the following transformer encoder. To this end, the final input to the Transformer Self Encoder would be the summation of the positional encodings and corresponding word embeddings.

2.6 Attention Mechanism

Many of today's deep models that connect images and natural languages are under the encoder-decoder structure, which was originally proposed for machine translation (Sutskever et al., 2014). As in our scenario of image captioning, images will first go through deep convolutional neural networks, while recurrent neural networks are then applied on top in order to generate a text sequence. There is one potential issue that exists inside this encoder-decoder framework, where the encoder network will have to squeeze all required information inside a fixed-length feature vector. It places lots of computational burden upon the fixed-size context vector when the decoder is using it for output generation. However, with an attention mechanism, during each step of the output generation, it can help the decoder to have a fine-grained understanding of the source information and then attend to different parts accordingly.

Instead of encoding the input sequence/image regions into a single fixed context vector, we let the model adaptively generate the context vector based on different semantic context at different output time steps. To be more specific, we let the model learn which parts to attend to and how these parts should be merged based on the source information and what the decoder has provided. Consider the encoder-decoder diagram in Figure 2.6.

Here, the encoder RNN generates $h_1, h_2, h_3, \ldots, h_7$ from the inputs $x_1, x_2, x_3, \ldots, x_7$. Then we would have to find out the context vector c_i for each of the output time steps.

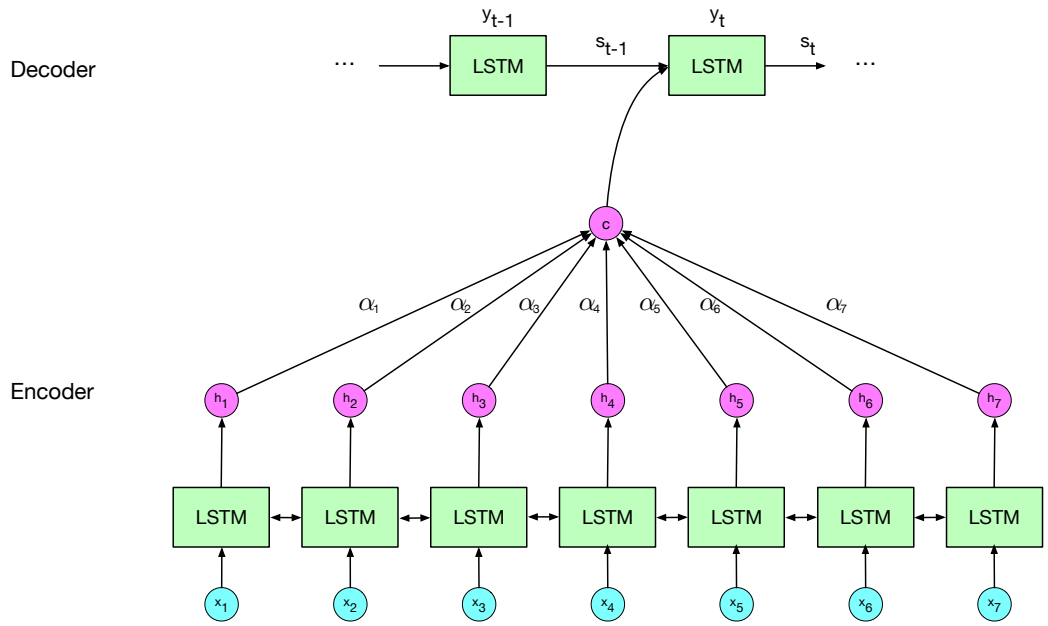


Figure 2.6: Attention mechanism in an encoder-decoder architecture

How can the context vector for each output time step be computed? We can define some alignment function as:

$$e_{t-1,j} = a(s_{t-1}, h_j), \tag{2.16}$$

in which a stands for a neural network that is trained simultaneously along with all the other sub-networks in the whole system. Typically, there are two ways of constructing this alignment function, i.e., additive and multiplicative, between s_{t-1} and h_j. This alignment model scores how well each encoded input h_j matches the current output of the decoder s_{t-1}. In order to output scalars for weighted summation, the alignment scores should be normalized according. Typically, a softmax function would be chosen as follows:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{7} \exp(e_{ik})}. \tag{2.17}$$

Correspondingly, we can get the context vector for current time step t by summing up the

encoder hidden states h_j with their normalized alignment weights α_{ij}.

$$c_i = \sum_{j=1}^{7} \alpha_{ij} h_j. \tag{2.18}$$

2.7 Typical Workflow

In summary, in order to apply neural networks for solving real-world problems, a typical workflow described below should be followed.

Data preparation. First, we need to collect a large number of data samples. For the purposes of this dissertation, we have assumed that in the scenario of a supervised learning task, each data sample consists of input feature x and ground truth label y. Then the dataset would be split into three folds, i.e., training, validation, and test. The common split ratio would be 60%, 20%, 20%, respectively. In particular, we use the training fold for parameter optimization based on backpropagation, hyper-parameter selection is done with the validation fold, and we will report our final results on the test fold.

Data pre-processing. It has been demonstrated that the convergence of neural networks benefits a lot from data preprocessing (LeCun et al., 1998). For images, they would commonly be preprocessed by data normalization, which does both mean subtraction and standard deviation division. Note that we will only standardize the data based on statistics collected from the training data. Both the validation and test fold would use the same mean and standard deviation as the training fold, which hopefully can simulate the process of deploying a trained system for real-world problem solving. For text sequences, preprocessing would typically require the operations of lowercasing, tokenization based on a specific tokenizer like PTBtokenizer (CoreNLP, 2016), vocabulary construction, etc.

Architecture design. We need to finalize the architecture of neural networks that might be explored with our dataset. As we have noted above, the structure of a neural network would guide how data should flow, which mathematically defines the function f. It is more of an art than a science. Here we may incorporate any prior knowledge, such as intuitions of how humans would solve a problem, into the architecture of a neural network. Normally, image pixel data would be processed with convolution operations while recurrent networks are good at input with sequence structure. When considering the network scale,

one rule of thumb that we should keep in mind is that the number of parameters should be roughly the same as the number of training examples. For instance, there are 1 million images in the ImageNet challenge (Russakovsky et al., 2015) dataset. Typical ConvNets that have successfully classified ImageNet images with high accuracy are often trained with around 10 million parameters. Regularization techniques such as data augmentation, $L2$ regularization, and dropout (Srivastava et al., 2014) are ubiquitously used in these networks to alleviate overfitting.

Optimization. The default optimizer to train a neural network would be to use Adam (Kingma and Ba, 2014). The learning rate is set approximately to $1e^{-3}$, while the hyper-parameters for its first and second moment are by default 0.9 and 0.99. Besides, learning rate decay has also been proved to be really effective for neural network training. Empirically, it is beneficial to start decaying only when we've reached the middle point of the training process. For sanity check, we may first try with only one training example to see if it is able to reach a training loss of zero. Afterwards, we may train the network with the full dataset.

Hyper-parameter selection. In addition to the learnable parameters that can be trained with the stochastic gradient descent method, hyper-parameters such as learning rate, depth of network (the number of layers), or hidden layer size (number of units), are infeasible to be tuned dynamically with the first order optimizers. Instead of selecting hyper-parameters with grid search, studies have proved that randomly sampling from some search ranges would achieve a better result (Bergstra and Bengio, 2012). Finally, the model that performs the best on the validation dataset might be chosen for deployment.

Evaluation. Whenever we've identified the best model, either with the lowest loss or achieving the best evaluation metric on the validation dataset, the final performance can be obtained by a single run of the model on the test dataset. Furthermore, model ensembles can further boost the performance, which requires training multiple models that are initialized with different parameters or trained with different hyper-parameters.

2.8 Extensions

On top of the above workflow with data-driven approaches, there are some restrictions when applying these components directly. Specifically, the GANs framework is employed to fine-tune the image captioning model in Chapter 3. However, GANs would require that both the generator and discriminator be continuous differentiable functions. Yet, our caption generator does not satisfy this requirement, as text data is discrete. To tackle this issue, we introduce another trick called Gumbel-Sotfmax (Jang et al., 2016) which is a continuous relaxation of our desired discrete categorical distribution. In this way, GANs can be plugged in to further improve image captioning. Besides, when quantifying the relevance of sentences to images, we are currently limited to datasets without such congruence labels. It is essential that we turn to unsupervised learning paradigms. Again inspired by GANs, we designed a new training framework specifically for our congruence measure, which through an attention mechanism, constructs pseudo-labels and thus makes it trainable as in supervised learning.

We organize the reminder of this dissertation around three research questions that we want to answer, each of which leverages the model design and training techniques we've introduced above to connect images with languages. Now let's discuss the details of each research topic one by one.

Chapter 3

Improved Image Captioning with Adversarial Loss

In this chapter, let's first look at the research question of *how and by how much we can improve the quality of image captions generated by machines*. Correspondingly, we make the hypothesis that including a Gumbel-Softmax sampler in an image captioning model enables training of that model under the GANs framework by converting a discrete to a continuous representation, which we hope will boost quantitative evaluation metrics.

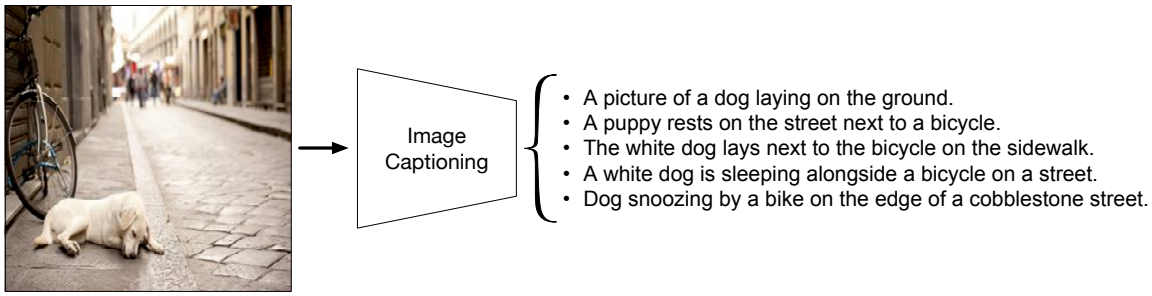


Figure 3.1: Image captioning example

3.1 Introduction

Image captioning requires a machine to generate complete and coherent sentences in natural language based on an input image. One such example is depicted above in Figure 3.1.

Not only do the objects in the image need to be captured, but the description also has to accurately express the relationship between these objects and what activities are involved (Vinyals et al., 2015). More formally, with an image I as input, a sequence of words $\{w_1, w_2, \dots\}$ will be produced, which describes the image adequately.

There are a variety of applications that captioning systems can help with, such as retrieving and tagging images, assisting the visually impaired by reading out generated captions, or making it easy for users to organize and navigate through unstructured visual data. With the help of deep learning models, it's inspiring that neural caption models, which use recurrent neural networks to decode images represented by the state–of-the-art convolutional neural networks and generate sentence descriptions, have performed quite promisingly on various evaluation metrics such as METEOR (Denkowski and Lavie, 2014) and CIDEr (Vedantam et al., 2015). Visual attentions (Karpathy and Fei-Fei, 2015; Xu et al., 2015; Yang et al., 2016) have also been explored to produce a spatial map of the image regions per the prediction of current word, which provides a fine-grained perception of the input image. Nevertheless, machine-generated/produced captions can still be easily distinguished from ones that are written by humans. It's particularly obvious if multiple captions for the same image are placed together and observed concurrently.

We train a new image captioning neural network that performs better than previous state of the art models on standard evaluation metrics. Besides, it can also generate captions less distinguishable from human ones, which in spirit matches the definition of the Turing Test. Inspired by the motivation behind GANs, we propose to incorporate adversarial loss instead of handcrafted ones to train the image captioning model, in the hope that it can generate more realistic captions. For this, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are built, in which our caption model is treated as a generator. As we know, GANs have been applied successfully to generate continuous data such as images (Denton et al., 2015; Radford et al., 2015), although in some cases (Luc et al., 2016) that may not work well. In contrast to images, captions are not continuous, but are discrete points distributed in some space. The operation of prediction with either argmax or beam search for caption generation makes the backpropagation infeasible in GANs. In terms of this, the Gumbel-Softmax sampler (Jang et al., 2016) trick is applied in our model, which makes end-to-end training possible with discrete categorical data and thus overcomes this obstacle.

3.2 Related Works

Image Description. Previously, captioning models are mostly implemented through two steps using template-based approaches. First, visual objects in the images are recognized, which includes attributes, objects, and activities, through classification, detection, or recognition. Then the description sentence is generated by incorporating either an n-gram model (Kulkarni et al., 2013), a template model (Farhadi et al., 2010), or statistical models designed for machine translation (Rohrbach et al., 2013). Farhadi et al. (2010) have tried to convert inferred triplets of visual elements to text sequence using templates. A Conditional Random Field (CRF) (Sutton et al., 2012) is adopted by Kulkarni et al. (2013) to jointly learn with detected objects, classified attributes, and inferred prepositions, which are finally used to fill the template slots. What's more, Kuznetsova et al. (2012) and Mitchell et al. (2012) proposed to apply more powerful language models with a syntactic tree to form the final sentence, with descriptive words coming from some attribute detectors.

Thanks to the development of deep neural network based sequence-to-sequence models originally proposed for machine translation (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014), now it's possible to end-to-end train caption models, in which a convolutional neural network is first used to extract visual features while a recurrent neural network is then applied to generate the final word sequence (Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015). This is inspired by that we may translate images to text as compared to the original machine translation task. With an input image and the previous word, Kiros et al. (2014) first proposed to apply a multi-layer perceptron together with a log-bilinear model to get the next word. Later, Mao et al. (2014) and Chen and Zitnick (2014) replaced the feed forward network with a recurrent neural network. Vinyals et al. (2015) then turned to an LSTM instead of a vanilla recurrent neural network for word sequence decoding. It should be noted that all these approaches are based on representations extracted from the last fully connected layer of a convolutional neural network. Then Karpathy and Fei-Fei (2015) started to adopt the convolution features from a CNN and later attention mechanisms (Xu et al., 2015) were incorporated to learn an alignment between current word prediction and potential image region focus. Lu et al. (2017) have also tried to reason about when we should rely on the image to predict the word and when we may turn to a language model for simple word prediction. Besides these basic encoder-decoder based neural image captioning models, reinforcement learning techniques have also been explored

to boost the corresponding evaluation metrics. Rennie et al. (2017) applied policy gradient with baseline to improve the quantitative performance of image captions. Reward of an inferred caption (model exploitation) is used as baseline to bias the reward of a sampled caption (model exploration), which is named as Self Critical Sequence Training (SCST) and obtained significant quantitative improvement. We've also adopted these techniques in our basic image caption generator.

Although coherent sentences can be generated by modern captioning models, if we look carefully, we may discover that they tend to be quite generic, some of which may even be replicated from the training dataset (Devlin et al., 2015). What's more, different people may describe the same image with different reasonable sentences. However, sentences generated with beam search (Rennie et al., 2017) by current neural caption models are often quite similar. On the other hand, the most common approach of training a recurrent language model currently is to feed in both all the previous ground truth words and the input image, and then predict the next word w_t. But during the inference time, our prediction is based on the input image and purely all previously predicted words. As a result, at test time if some words are incorrectly predicted, it can influence future word prediction, which can stack errors and cause big mistakes. Only training with the ground truth words is also called teacher forcing, which makes the model suffer from the exposure bias (Ranzato et al., 2015) problem. Whenever there is an error in prediction, the model won't have the capability to recover and predict correct words in the future. To avoid this, the scheduled sampling technique (Bengio et al., 2015) has been invented, which trains the caption model with actual words at the very beginning, but then gradually feeds in predicted words for future caption prediction during training. Furthermore, some also propose to alleviate the exposure bias issue by applying reinforcement learning approaches. Evaluation metrics such as BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), or CIDEr (Vedantam et al., 2015) are directly used as reinforcement reward, based on which models should be tuned to achieve a higher score (Liu et al., 2016; Rennie et al., 2017).

In this work, we introduce the generative adversarial networks, in which our caption model plays the role of a generator. Correspondingly, a discriminator is designed and explicitly trained to distinguish generated captions from human ones. The caption generator is then tuned to fool this discriminator based on the adversarial loss. Consequently, our caption model is able to generate captions that are not only quite similar to human ones,

but also achieve better performance on various metrics.

Generative Adversarial Networks. Generative models learned in the Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) framework do not need to define an explicit loss. Instead, an evaluation loss function is defined for an auxiliary discriminator, which does binary classification based on whether the input is real or fake, where the fake ones come from our desired trained generator. GANs have been proved to work amazingly well to generate real images (Denton et al., 2015; Radford et al., 2015). However, all these works share one common attribute, i.e., the data they generate is supposed to be continuous. In contrast, our captions, which are sequences of words, do not match this requirement. It's challenging to apply GANs for discrete sequence generation since we are unable to backpropagate gradients through a sampling (either argmax or beam search) operation.

Some have already tried to generate discrete distributions with GANs. In Luc et al. (2016), discrete semantic labels for each pixel are generated with support from GANs. Generative adversarial networks have also been used to generate text sequences unconditionally by training with the REINFORCE trick (Yu et al., 2017).

Most similar to our work are the ones that rely on the reinforcement rule (Williams, 1992) to deal with discrete sample backpropagation, such as in dialogue generation (Li et al., 2017) and image caption generation (Dai et al., 2017). In contrast, we plug a Gumbel Softmax sampler (Jang et al., 2016) into the GANs framework, which makes it fully trainable with gradient based optimizers. In short, it is a continuous approximation of a discrete categorical distribution. Additionally, both the adversarial and word level cross entropy losses are used in (Li et al., 2017) during each iteration of GAN training. We however first pre-train the generator with both teacher forcing and reinforcement learning, and then train the generator purely based on adversarial loss, with the hope that adversarial loss can further improve the quality of generated captions if feedback regarding what real captions should look like can be backpropagated. Besides, in order to stabilize the training dynamics of GANs framework, the recent proposed spectral normalization (Miyato et al., 2018) technique is also introduced in our work.

3.3 Model

3.3.1 Hierarchical Caption Generator

Our hierarchical caption model consists of two components, one of which is used to give guidance as to which regions we should focus on in an image during the prediction of one word. This is implemented as usual by an attention layer between image regions and one RNN hidden state. Based on such focus on image, then the other component shall provide more detailed instructions on word pickup in response to whether it's visually related.

Suppose the input image can be represented by a set of feature vectors V, we propose to borrow the idea of 'soft' top-down attention layer (Anderson et al., 2018) to weight each feature in V, which is guided by the current hidden state that has encoded all previous predicted words. Such design has been broadly used in other image captioning models (Lu et al., 2017; Rennie et al., 2017; Xu et al., 2015).

As stated above, at a high level, there are two LSTM (Hochreiter and Schmidhuber, 1997) layers in our captioning model, which follow the standard implementation in (Donahue et al., 2015). In the following sections, the data flow of the LSTM at one time step will be represented as follows:

$$h_t = \mathrm{LSTM}(x_t, h_{t-1}), \tag{3.1}$$

where the LSTM input vector is represented by x_t and h_t stands for the output hidden state vector. Here the memory cells in the standard LSTM have been neglected for notation brevity. Now let's specifically describe the content of input x_t and hidden state h_t for each of our LSTM layers. The overall basic caption generator is illustrated on the left side of Figure 3.2.

Attention LSTM. For the caption generator, the first LSTM is characterized as an attention layer for visual inspection, which looks at different regions of an image, and the second LSTM is just used as a language model for word prediction. We use superscripts in the equation to indicate the index of each layer. Now let's look at what makes up the input x_t^1 at each time step to this attention LSTM layer. It is composed of the previous hidden state from the top language model LSTM as h_{t-1}^2, the average of image features V represented as $\bar{v} = \frac{1}{k}\sum_{i=1}^{k} v_i$ and also the word embedding of the predicted word at the previous time

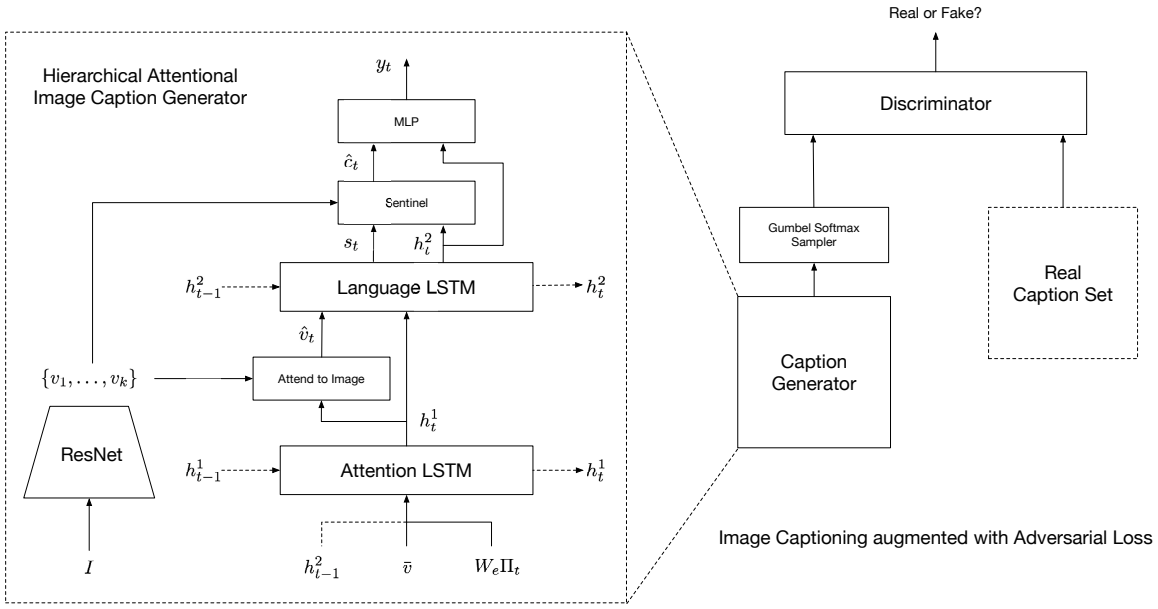


Figure 3.2: Image captioning augmented by GANs

step. More mathematically, it can defined as

$$x_t^1 = [h_{t-1}^2, \bar{v}, W_e \Pi_t], \tag{3.2}$$

where $W_e \in \mathbb{R}^{E \times \Sigma}$ stands for a large word embedding matrix which owns Σ words, and Π_t corresponds to the one-hot vector of the input word at current time t. As we can see, such formulation of input x_t^1 provides all we can get for the current time step to the attention LSTM layer, which includes the language LSTM state, the overall image content, and captions generated so far. It should also be noted that we don't pre-train word embedding W_e and it is learned along with the whole system.

Using the hidden state h_t^1 from the first attention LSTM as our current context, we calculate a normalized attention score $\alpha_{i,t}$ for each image features in V as below

$$a_{i,t} = w_a^\top \tanh(W_{va} v_i + W_{ha} h_t^1) \tag{3.3}$$

$$\alpha_t = \text{softmax}(a_t) \tag{3.4}$$

where $W_{va} \in \mathbb{R}^{H \times V}$, $W_{ha} \in \mathbb{R}^{H \times M}$, and $w_a \in \mathbb{R}^H$ are parameters to be tuned during training. Then the attended image embedding, which is part of the input to the language

LSTM, can be computed as a weighted summation of V:

$$\hat{v}_t = \sum_{i=1}^{K} \alpha_{i,t} v_i. \tag{3.5}$$

Language LSTM. The language LSTM input is quite simple as compared to the input of attention LSTM above. It's just a simple concatenation of hidden state vector h_t^1 from attention LSTM and attended image feature $\hat{v}_t$:

$$x_t^2 = [\hat{v}_t, h_t^1]. \tag{3.6}$$

Sentinel. Besides these two recurrent neural networks for sequence modeling, we also incorporate the sentinel mechanism (Lu et al., 2017) on top of the language LSTM for final word prediction. The main intuition behind the sentinel component is that although image captioning can benefit a lot from spatial attention based decoders, it's still difficult to discern whether we should just depend on the language model for word prediction at some time steps. The sentinel here will try to capture such information. Therefore, the above language LSTM is expanded accordingly as in (Lu et al., 2017) to generate the "sentinel" vector s_t as

$$g_t = \sigma(W_x x_t^2 + W_h h_{t-1}^2) \tag{3.7}$$

$$s_t = g_t \odot \tanh(m_t) \tag{3.8}$$

where W_x and W_h are learnable weight matrices, x_t^2 stands for the language LSTM input at the current time step, and g_t is designed as a gate that is element-wise multiplied by the language LSTM's memory cell m_t.

Based on this visual sentinel, the final context vector $\hat{c}_t$ for word prediction is generated by an adaptive attention model as follows

$$\hat{c}_t = \beta_t s_t + (1 - \beta_t) c_t, \tag{3.9}$$

in which c_t is calculated in a way similar to Equation 3.5 to model spatial attention again. To compute the new sentinel gate β_t, the above spatial attention component is slightly modified by inserting the sentinel s_t with all v_i. Detailed information can be found in (Lu

et al., 2017). Then we can calculate the probability of each word at time step t as:

$$p_t = \text{softmax}(W_p(\hat{c}_t + h_t^2)) \tag{3.10}$$

where W_p is the weight parameters to be learned. The probability of the complete output sequence should be a product of conditional probabilities implemented by the chain rule in statistics:

$$p(y_{1:T}) = \prod_{t=1}^{T} p(y_t|y_{1:t-1}). \tag{3.11}$$

3.3.2 Augmented with Generative Adversarial Networks

To augment the basic captioning model with the GANs framework, we need to deal with the discrete sample generation problem as this does not allow for backpropagation between the discriminator and the generator in GANs (since text data is discontinuous). Here the Gumbel-Softmax approximation trick, which is proposed in Jang et al. (2016), is applied in the hope that it can continuously relax discrete one-hot vectors. In this way, backpropagation can be performed with data sampled from a categorical distribution. Also, it's worth noting that we can simply plug in such a layer as a differentiable node, and there is no need to do extra steps for gradient estimation.

There are mainly two steps in the Gumbel-Softmax approximation. First, our original categorical distribution is re-parameterized by the Gumbel-Max trick, which adds Gumbel noise and does an arg max operation. Suppose we have a categorical distribution with C classes, which is parameterized as $\Theta = \theta_0, \ldots, \theta_{C-1}$, then the random variable r sampled from this distribution can be represented as:

$$r = \text{one\_hot}\left[\arg\max_i (g_i + \log\theta_i)\right] \tag{3.12}$$

where g_i's are i.i.d. random Gumbel noise. Next we can continuously relax our discrete sample r by replacing the arg max in Equation 3.12 with the softmax operation.

$$r' = \text{softmax}\left[\frac{g_i + \log\theta_i}{\tau}\right], \tag{3.13}$$

in which τ stands for a temperature hyper-parameter controlling how this Gumbel-Softmax distribution would behave like an one-hot distribution. To be more specific, if τ is quite near 0, samples from it would behave like one-hot representations. On the other hand, if a high value is picked for τ, the sampling tends to be more like the actual categorical distribution. The behavior of sampling from a Gumbel-Softmax distribution influenced by the temperature value τ is depicted in Figure 3.3.

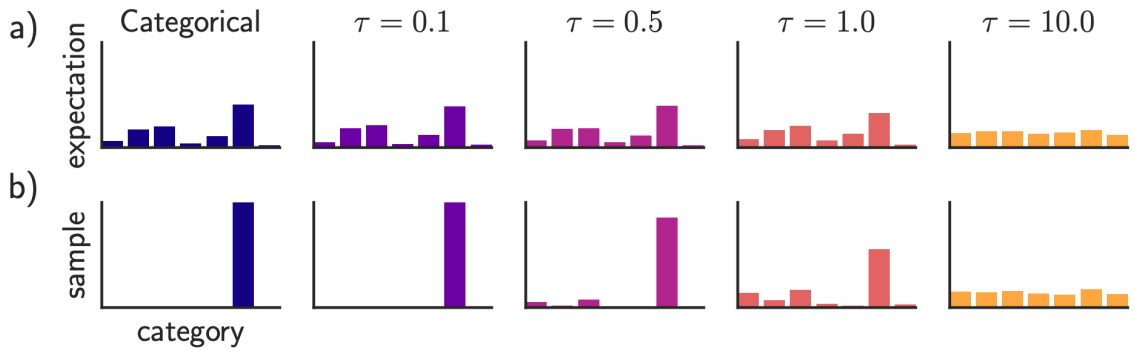


Figure 3.3: Interpolation between discrete one-hot distribution and continuous categorical densities by the Gumbel-Softmax distribution (Jang et al., 2016)

During our adversarial training, the output from the generator is armed with the straight-through variant of Gumbel-Softmax (Jang et al., 2016) to do word sampling. In this trick, the discrete sample r goes normally during the forward pass, while in the backward pass, the soft approximation r' is incorporated for backpropagation. More specifically, it can be fulfilled through the stop gradient (Abadi et al., 2016) or detach (Ketkar, 2017) function in different deep learning frameworks as shown below

$$\Pi_t = (r - r').\mathbf{detach} + r', \tag{3.14}$$

in which during the forward pass, one-hot representation is attained, while during the backward pass, gradients on r' are backpropagated. After we have obtained the index of a word (one-hot vector), we may retrieve the corresponding word embedding vector through matrix multiplication:

$$w_t = W_e \Pi_t, \tag{3.15}$$

which is fully differentiable. It should be noted that one-hot representation and matrix

multiplication are essential here as compared to the typical index-based word embedding retrieval, which is not trainable.

3.4 Optimization

Suppose θ includes all the parameters in our captioning model, and each training sample is also provided with a ground truth sequence $y^*_{1:T}$, our first target is to minimize word level cross entropy loss defined as:

$$L_{XE}(\theta) = -\sum_{t=1}^{T} \log(p_\theta(y^*_t | y^*_{1:t-1})), \tag{3.16}$$

which is also called teacher-forcing learning.

Then on the next stage after the above model has converged, a reinforcement learning based approach is employed to further boost performance. Here the negative expected reward is assumed to be minimized:

$$L_R(\theta) = -\mathbb{E}_{y_{1:T} \sim p_\theta}[r(y_{1:T})], \tag{3.17}$$

in which r stands for the reward function such as CIDEr, which calculates the score for each training example. Here we follow the REINFORCE (Williams, 1992) algorithm as used by the Self-critical Sequence Training (SCST) (Rennie et al., 2017). They estimate the gradient of this loss function as:

$$\nabla_\theta L_R(\theta) \approx -(r(y^s_{1:T}) - r(\hat{y}_{1:T}))\nabla_\theta \log p_\theta(y^s_{1:T}) \tag{3.18}$$

where $y^s_{1:T}$ stands for a caption sampled from our currently training network and $r(\hat{y}_{1:T})$ is the reward calculated with a sequence decoded by greedy search or beam search, which defines the baseline reward. This gradient $\nabla_\theta L_R(\theta)$ helps to encourage caption samples that have a higher score than ones inferred from the current model and suppress ones that are not.

Finally, we turn to the GANs framework to further fine-tune and improve our caption generator. Instead of using the original adversarial loss proposed in Goodfellow et al. (2014),

we applied the Wasserstein GAN introduced in Arjovsky et al. (2017). This presents much better training dynamics with fewer oscillations. Suppose the real caption's distribution is p_r. The adversarial loss is then defined as:

$$L_D(\theta^d) = -\mathbb{E}_{x\sim p_r}[D(x)] + \mathbb{E}_{x\sim p_\theta}[D(x)] \tag{3.19}$$

for the discriminator, in which $D(\cdot)$ is designed to be K- Lipschitz function following the WGAN requirement and its parameter is θ^d. For the generator, we want to minimize

$$L_G(\theta) = -\mathbb{E}_{x\sim p_\theta}[D(x)]. \tag{3.20}$$

We've also tried the spectral normalization (Miyato et al., 2018) technique in training the caption generator, in the hope that it can make the training more stable. Each weight of linear mappings in both our attention and language LSTMs (except ones implemented inside standard LSTM) is normalized by its spectral radius.

3.5 Experiments

All of our experiments are conducted with the MSCOCO dataset (Chen et al., 2015). The ubiquitously used 'Karpathy' split (Karpathy and Fei-Fei, 2015) is used for hyperparameter selection with validation dataset and offline test. There are 113,287 training images, each of which is paired with five caption annotations, and both the validation and testing splits have 5,000 images. You may find a summarized version in Table 3.1 and some examples in Figure 3.4.

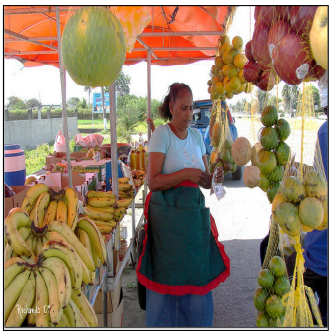
Table 3.1: MSCOCO dataset 'Karpathy' split statistics

Split	# of images	# of annotations
Training	113,287	566,435
Validation	5,000	25,000
Testing	5,000	25,000

In terms of text pre-processing, standard practice is followed, which includes sentence lowercase conversion, word tokenization based on white space, and that we only keep words

- A passenger bus near a bus stop, people next to it.
- The city bus is traveling down the road.
- A double-decker transit bus in the United Kingdom.
- A red double-decker bus is driving on a city street.
- A double decker bus rides down the street.

- Several plants are in the right side of a sink.
- Plants fill a sink near a kitchen counter.
- A kitchen with a counter some plants near a sink.
- A large houseplant resting in the kitchen sink.
- A kitchen sink with a potted plant sitting next to it.



- The woman has many bananas and other fruit at her stand.
- A lady making change at a roadside fruit stand.
- A woman in an apron is under an umbrella with a table of melons and banana.
- A woman standing in a market filled with market next to rep bananas.
- The woman is selling lots of fruit at the stand today.

Figure 3.4: MSCOCO image captioning examples, used with permission of MSCOCO

that appear more than five times. Finally, this results in a word vocabulary of $10,132$ words. Caption qualities are quantitatively evaluated by several standard evaluation metrics, i.e., ROUGE-L (Lin, 2004), BLEU (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), and METEOR (Denkowski and Lavie, 2014). We describe their underlying ideas in Table 3.2 and you may find more details in Kilickaya et al. (2016).

In the hierarchical captioning model that combines Lu et al. (2017) and Anderson et al. (2018), which we name as **Mixture**, the ResNet152 (He et al., 2016) backend is used for

Table 3.2: A summary of the evaluation metrics used for image captioning

Metric	Intuition	Originally proposed to evaluate
ROUGE	n-gram based recall	Document summarization
BLEU	n-gram based precision	Machine translation
CIDEr	n-gram similarity weighted by tf-idf	Image description generation
METEOR	n-gram matching w/ synonym replacing	Machine translation

encoding the input image. The hidden size M of each LSTM layer is set to $1,000$, there are 512 hidden units in the additive attention layer, and we set word embedding size E to $1,000$, which are the same as in Rennie et al. (2017). A beam size of 5 is chosen during both reinforcement learning optimization and final decoding. Basic captioning model training follows Anderson et al. (2018) and then an additional 20 epochs are trained under the GANs framework.

Table 3.3: Performance of image captioning models on MSCOCO Karpathy test split. Each metric has an upper-bound of 1 except for CIDEr, which can be up to 10.

Training Method	Model	BLEU-1	BLEU-4	ROUGE-L	METEOR	CIDEr	p-value
REINFORCE CIDEr	SCST	-	0.333	0.553	0.263	1.114	-
	Bottom-up	0.766	0.340	0.549	0.265	1.111	-
	Mixture	0.767	0.342	0.550	0.266	1.117	-
GANs	Mixture	**0.770**	**0.345**	**0.554**	**0.269**	**1.121**	0.0387
GANs w/ SN	Mixture	**0.782**	**0.350**	**0.561**	**0.272**	**1.137**	0.0324

In Table 3.3, the results of our mixture captioning model and ones trained with GAN augmentation are all reported in comparison to some near state-of-the-art baseline approaches on the test portion of the 'Karpathy' splits, i.e., SCST (Rennie et al., 2017) and Bottom-up (Anderson et al., 2018). SCST tried to apply the REINFORCE algorithm with image caption training, in which test time reward (such as CIDEr) is used as a baseline with policy gradient (Sutton et al., 2000). It has also been adopted in our two-stage training. On the other hand, Bottom-up takes the input of fine-grained image features from object detection like Faster R-CNN (Ren et al., 2015) instead of the whole image, which is proved to

be effective for providing useful image details and showed better performance as compared to previous works. In our caption generator, we chose to combine the spotlight component of various models into one model, which is further improved by our tweaked GANs training. As seen in Table 3.3, besides the improvement from mixture of previous basic captioning components, we can also see that after being augmented with the GANs framework training, various evaluation metrics can be further improved. As GANs are notorious for unstable dynamics during training, we've also incorporated the recent proposed spectral normalization trick during GANs optimization. It's expected that further performance boost can be obtained.

In order to demonstrate the effectiveness of GANs training on top of image captioning, we did a significance test based on the CIDEr metric with our Mixture model. Correspondingly, Mixture, Mixture with GANs, and Mixture with GANs & Spectral Normalization models are trained 10 times with different initializations. Afterwards, their individual performance on the CIDEr metric is calculated and the p-values are shown in Table 3.3 as compared to our basic Mixture model. Due to the inaccessibility of other baseline models, we do not run such analysis with our GANs training, which will be done in the near future to prove our method's generalization ability.

Figure 3.5: Qualitative captioning examples with the first attention layer visualized

Besides, we also provide qualitative caption examples in Figure 3.5, in which the scores

of the first attention layer are visualized on top of the corresponding image regions based on their position in the convolution layer. Like in the left side, when predicting the words "man", "woman", and "umbrella", the corresponding regions in the image are highlighted with a higher attention score as compared to the other part. Similar trends can be found in the right side image as for the words "dog" and "rug". For captioning examples with sentinel and corresponding spatial attention scores, samples are presented in Figure 3.6. We can find the sentinel score of turning to language model for guidance in brackets. As a

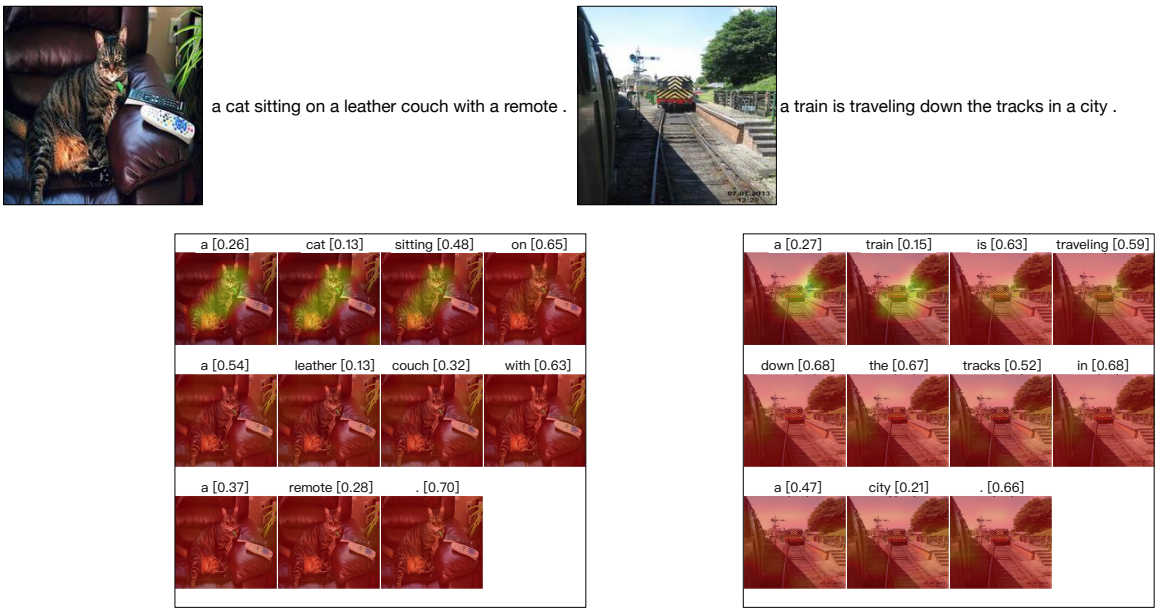
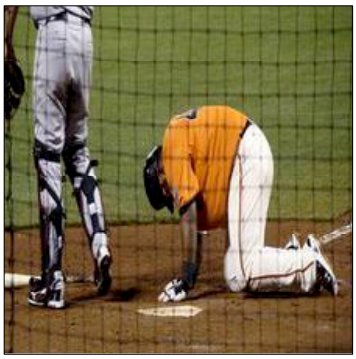


Figure 3.6: Qualitative captioning examples with sentinel (in brackets) and corresponding attention visualized

sentinel score indicates how likely we should turn to our language model for prediction, low sentinel would imply that we may need to look at the image and that the current word is visually related. It's obvious that the visual attention here is not that strong as in the first attention layer. For example, when we are predicting the word "cat", the corresponding sentiment score is 0.12, which is quite low. This indicates that our model is able to infer whether the word it is predicting is visually related or not. On the other hand, if the next word to be predicted is not based on the image such as some stop words, the corresponding sentinel score would be relatively high, which can be found out in the prediction of some words like "with" or the period mark at the end. It's obvious that such behavior is generic as we can find the same pattern in the right side image for both visual words "train" and

"city" as well as non-visual ones like "the", "in", etc. Finally, we present some caption examples that show improved quality with our GANs fine-tuning in Figure 3.7. It's clear that when GANs are incorporated, descriptive details and some verb words are added to make it a more coherent caption. Like in the left side image, the word "street" is replaced with the actual fact word "ramp", while in the right side image, the color word "orange" is inserted and the player's activity is characterized as "kneeling". Note that when GANs are trained with spectral normalization, caption sentences generated by the new converged model appears to be more grammatically complete and realistic, which further vindicates the effectiveness of our method. As we can see, more descriptive words like "brown" and "an" are captured and the period mark is identified to make the right side caption a more complete readable sentence.

Mixture: a man on a skateboard riding down the street .
GAN: a man on a skateboard is riding down the ramp .
GAN w/ SN: a man on a skateboard is riding down the brown ramp .



Mixture: a baseball player is swinging a bat at a ball
GAN: a baseball player in orange shirt is kneeling on the field
GAN w/ SN: a baseball player in an orange shirt is kneeling on the field .

Figure 3.7: Caption examples generated by our basic mixture model, GANs model, and GANs model w/ spectral normalization

3.6 Conclusion and Future Work

We introduce an adversarial caption generator which augments a conventional neural caption model with the GANs framework during fine-tuning. We achieved this by utilizing a discriminator network designed to push generated captions indistinguishable from human written captions. The Gumbel-Softmax trick is introduced to handle the discrete representation problem during caption generation. Qualitative results show that our adversarial

model produces captions with higher quality than selected baseline models. Besides, the adversarial model presented improved performance for common evaluation metrics. Therefore, we argue that the GANs framework should be used in training image captioning models. We will focus on incorporating more state of the art techniques for image captioning models in the near future. What's more, significance tests on other baseline caption models would be done to see the generalization ability our GANs fine-tuning technique. To be more specific, other baseline models besides SCST and Bottom-up augmented with GANs training will be tested on these evaluation metrics, in the hope that there will be consistent and significant improvement.

Chapter 4

Congruence Measure between Image and Sentences

In this chapter, we turn to the research question of *what is an effective way for learning the congruence of a sentence-image pair if there is no human annotated label?* In that respect, we make the following two hypotheses:

- Quantitative relevance between review image and sentences can be accurately estimated through our proposed Quasi-Supervised Learning (QSL) network trained in a data-driven manner based on review structures.
- Our QSL network based on the data priors (i.e., sentence-image structures in between reviews) is able to produce relevance scores of the sentences with respect to the image, which align better than ones from existing heuristic based methods.

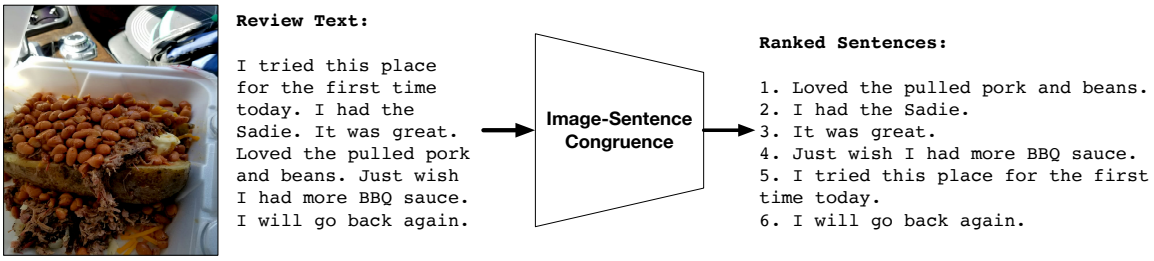


Figure 4.1: Image-sentence congruence measure example

In terms of matching review sentences to an image, suppose we have an image with corresponding review text context as shown above in Figure 4.1. The congruence model should be able to identify the quantitative relevance of each sentence to a given image. Finally, it will produce a ranked list of these sentences based on their relevance scores. We develop neural network architectures specifically for matching sentences to an image in the same review. It should be noted that this is an unsupervised learning or weakly supervised learning problem as there is no provided explicit label that indicates the relevance of each sentence-image pair. In other words, we have to figure out or infer such correspondence implicitly.

This problem is approached similarly in a data-driven manner. To be more specific, models that are made up of several neural networks are developed, which embed one image and all review sentences, and then calculate a relative scalar-valued score for each sentence-image pair. This quantity can be used to indicate how the image and each sentence are matched with each other. Afterwards, the model is trained end-to-end upon a dataset of reviews with both images and texts so that the highest score is assigned by the model to an image-sentence pair that matches perfectly, and a lower score otherwise. This is achieved by constructing relevant and non-relevant training pairs with the help of an attention mechanism, guided by how humans would perceive information by first looking at images and then reading text in a review. Our experimental results show our proposed Quasi-Supervised Learning network's effectiveness in learning the relative correspondence between image and sentences. Besides, a multi-task version of QSL is explored to alleviate the overfitting problem, which attains consistent improvement as compared to the vanilla QSL model.

4.1 Introduction

Matching images with sentences in terms of semantic relevance, i.e., the ability to connect images with natural language and infer their correspondence, is one of the central functions an intelligent machine should have in the sense of Turing test. It can not only help to differentiate between text content and image description, but also assist in the direct application of image retrieval (Liu et al., 2007). What's more, sentences that are highly related to an image can also be treated as ground truth captions for images (Lu et al., 2017; Xu

et al., 2015), which provides a way to reduce the cost of crowd sourcing for image captioning model training. As we go forward with multi-modal deep learning (Antol et al., 2015; Xu et al., 2015), many supervised learning models (Karpathy, 2016; Lin et al., 2014; Socher et al., 2014, 2011) have been proposed, which learn the embeddings jointly in order to rank highly relevant image-sentence pairs, but others lower. However, most of these approaches strongly depend on the quality and quantity of human annotated labels showing whether a sentence-image pair is relevant or not. They are super data intensive, and it takes a huge amount of time to do such manual labeling. Besides, the quality may vary among different kinds of people. In terms of this, we design a training framework that manipulates the inner structures of online reviews to pair review images with sentences.

Concretely, we develop a novel model that applies to the Yelp Restaurant dataset, which consists of reviews with both images and text descriptions. As with matching sentences with an image, we are required to pick relevant sentences in the review that best describe an attached image. But instead of relying on massive human annotations, we explored the relationships of sentences and images in reviews to construct pseudo-labels. To be more specific, it's most probable that text descriptions perceived after we have looked at some images in the same review should be more relevant as compared to images that we haven't seen before, the intuition of which is depicted in Figure 4.2. Based on this intuition, image

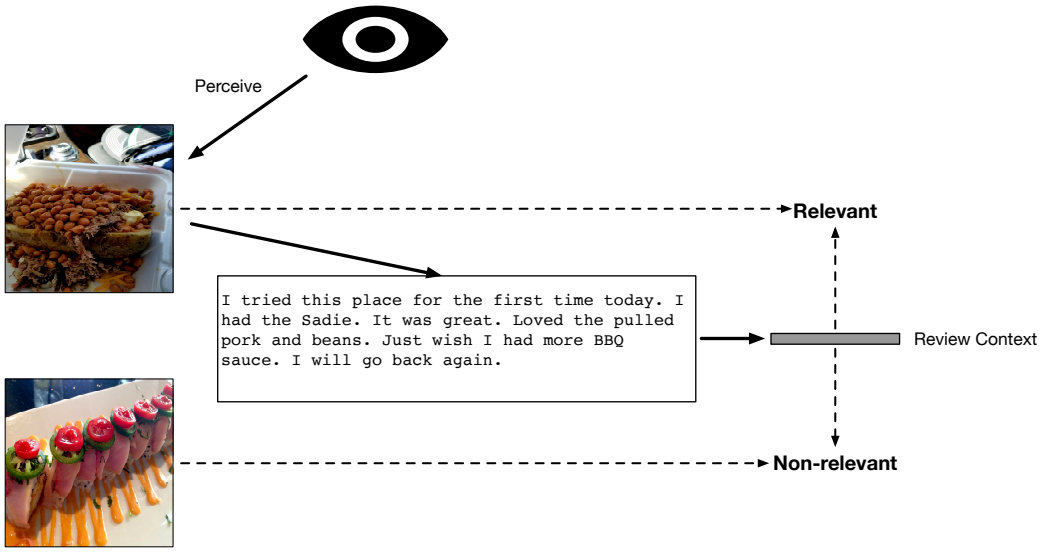


Figure 4.2: Intuition behind Quasi-Supervised Learning based on review structure

guided relevance attention to review sentences is incorporated to help select desired relevant

sentences, which provide us with relevant training pairs. On the other hand, the same text descriptions, coupled with images we haven't seen before, yield non-relevant (or less relevant) training pairs. With these 'ground truth' labels, a supervised learning model based on the ranking loss can be designed and trained. Since there is no actual human annotation, but it is trained in a supervised way, we name it as a Quasi-Supervised Learning (QSL) network.

Furthermore, we augment our vanilla QSL network with other downstream tasks to push the underlying learned features to be as generalizable as possible in terms of overfitting. Typical review writers will also provide overall ratings for the products or services they consume. Such rating information along with review texts can be used for sentiment analysis. With this in mind, we build another binary classifier on top of the learned sentence embeddings from QSL (review text encoder), which turns into a multi-task learning framework. Based on our experimental results, it can further improve the performance of our Quasi-Supervised Learning network on multiple information retrieval evaluation metrics.

Throughout this chapter, we try to demonstrate that

- Our proposed Quasi-Supervised Learning networks are able to identify congruences between review sentences and images, although no human annotated labels are provided during training. Besides, QSL can also accurately estimate the quantitative relevance between image and sentences based on our evaluation with a manually annotated dataset.
- Multi-task Learning models can often achieve much better performance as compared to single task driven approaches, which severely suffer from the ubiquitous overfitting problem in machine learning. With the auxiliary classification task augmented upon our QSL network, our model surpasses both the baseline and vanilla QSL network with large margins.

In short, we made the following three contributions. First, we propose and formalize the review image-sentence matching problem as a quasi-supervised learning task. This new task will not only find the matching sentences for the images in the online reviews, but also generate large-scale training corpora for the image caption task in e-Commerce

settings if highly related sentences can be identified. Second, two novel neural network based architectures are proposed to tackle the problem, in which no human annotated labels are required during training but we are still learning in a supervised manner with the incorporation of attention mechanisms and prior knowledge based on review structure. Besides, auxiliary learning tasks like sentiment analysis is introduced to make it a mult-task learning problem, which also alleviates the overfitting problem. Finally, we conducted large scale experimental evaluations using a real world Yelp Restaurant dataset and showed the efficacy of our proposed approaches.

4.2 Related Works

Generative Adversarial Networks. The approaches for matching a review image and sentences that we describe in this dissertation were originally inspired by the popular Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). With a trainable classifier or discriminator and a desired generator competing with one other, the classifier treats generated data as fake and samples from the actual dataset as real. On the other hand, the generator will be tuned and try to fool the classifier into predicting that its generated samples are real. Although our model does not exactly fit into the GANs definition, as we are not actually generating samples that are indistinguishable from real ones, the neural network structures are exactly the same. We built hierarchical encoders for sentence and review encoding, which can be treated as the 'generator'. Then we also have an image encoder in our model that can function as the real information receiver, which provides knowledge from "looking" at images and can be treated as "real data". Besides, we have a critic to evaluate and compare sentence-image pairs' relevance. This behaves exactly the same as what a classifier/discriminator is doing in GANs. Similarly, this critic is to be discarded when the model finally converges.

Image Sentence Matching. There are varieties of related works, which try to match description sentences to images by training with a margin ranking function (Karpathy, 2016; Lin et al., 2014; Socher et al., 2014, 2011). Socher et al. (2014, 2011) proposed Recursive Neural Networks based on the constituency parse tree or dependency parse tree for finding and describing images with sentences. They are trying to assign potential sentences to a given image. Lin et al. (2014) developed semantic graphs based on a complex text query for

video retrieval. The most similar to ours is Karpathy (2016). Their model jointly learns to match sentences with images and vice versa, while our model also maps the embeddings of image and text into the same subspace. All of these approaches applied the margin based ranking function to train the model, which is also used in our loss function. But it's worth noting that their models are all based on a human annotated dataset like Flickr8K (Hodosh et al., 2013), Flickr30K (Young et al., 2014), etc. It takes a huge amount of time to manually tag such sentence-image pairs. Furthermore, such labeling is not realistic for many real-world applications as the quality can't be ensured. There is a large disparity between what technology can supply and what users are actual asking for (Armitage and Enser, 1997). Our work differs from theirs in that we are working directly upon real-world sentence-image pairs that exist in online social media platforms like reviews, tweets, or posts. Intricate relationships between sentences and images inside one review and between reviews are comprehensively explored. To the best of our knowledge, no one has tried to address such sentence-image relevance measures before.

Multimodal Embeddings. Thanks to applications like image captioning (Xu et al., 2015), visual question answering (Antol et al., 2015), and visual dialog (Das et al., 2017a), multimodal embedding learning has attracted much attention in recent years. Such models typically project data from multiple sources such as images and texts, or different languages, into the same feature space. Srivastava and Salakhutdinov (2012) developed multimodal Deep Boltzmann Machines for image and text feature embedding learning, in the hope that they can be useful for downstream image-text comparison tasks. Socher and Fei-Fei (2010) introduced kernelized canonical correlation analysis to do fine-grained image segmentation and sentence pairing. Similar to their work, we try with various state of the art neural network architectures in deep learning to represent both images and review sentences (or word sequences) with high capacity. Then through some element-wise multiplication (Hudson and Manning, 2018) or additive mapping (Bahdanau et al., 2014), features from different sources can interact with each other, which results in some embeddings that are not only helpful for relevance evaluation but also generic, applicable as well to other tasks like binary classification.

4.3 Quasi-Supervised Learning

4.3.1 Model Architecture

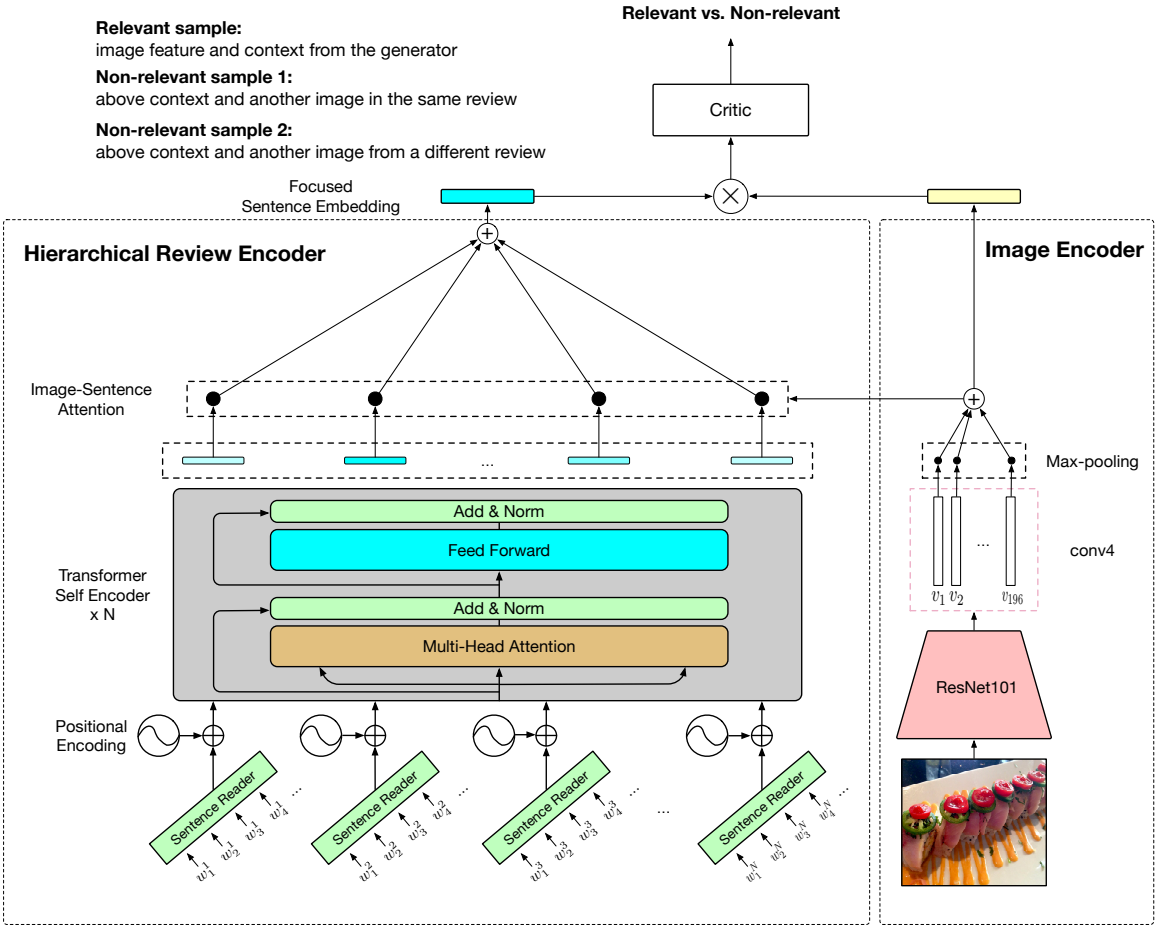


Figure 4.3: Quasi-Supervised Learning network architecture

In this section, we introduce the basic Quasi-Supervised Learning (QSL) network for review sentence and image matching, named as Vanilla QSL network. Given a review with both word sequences and images, our QSL model for matching a review sentence to an image, basically consists of three neural networks as shown in Figure 4.3, i.e., Hierarchical Review Encoder for sentence and review encoding, Convolutional Neural Network (CNN) for image feature embedding, and Multi-Layer Perceptron (MLP) critic to differentiate between relevant and non-relevant training pairs. Besides, we also augment our Vanilla QSL model

with an auxiliary learning task (binary sentiment classification) to alleviate the overfitting problem. Correspondingly, the multi-task learning version of QSL network is designed as shown in Figure 4.4. Next, we go over the details of each architecture one by one.

Vanilla QSL Network

Hierarchical Review Encoder. Assume we have N sentences in one review. A sentence reader represented by a unidirectional Recurrent Neural Network (RNN) or a pre-defined feature fusion function will first be used to get a fixed-length feature vector for each sentence. As we don't have any priors on the topics of each sentence, we would want this encoder to generalize well over all sentences. So each Sentence Reader (the bottom green box) in Figure 4.3 will share the same parameters if a trainable unidirectional RNN is picked here. Then, based on the following review encoding, we build the dependencies between these sentence vectors, which we expect can capture semantics of the whole review in terms of the connections between these vectors. Another bidirectional RNN like LSTM (Hochreiter and Schmidhuber, 1997) can be used for this purpose. This RNN would then try to model the connections and similarities between these sentence topics.

Table 4.1: Parameter size comparison of different sequence modeling layers

Layer Type	Parameter Size	Maximum Dependency Path
GRU	$3H \times (D + H)$	$O(n)$
LSTM	$4H \times (D + H)$	$O(n)$
Self-Attention (Transformer)	$4D^2 + O(D)$	$O(1)$

However, due to the large sentence lengths and the number of sentences in a review, it's easy for such hierarchical RNN models to suffer from memory explosion especially during backpropagation. In terms of this, we've also tried other gated RNN variants like GRU (Chung et al., 2014), and attention based encoder like the Transformer Self Encoder (Vaswani et al., 2017), for the sake of reducing the number of parameters to be tuned and the maximum lengths of dependency paths. The parameter size of each model is compared and listed in Tabel 4.1, for which we assume that our input has n tokens and each has D units, and the hidden state dimensionality is H. We've also compared the maximum dependency path between input tokens. The table shows, for each input token, how long it would take to get access to the other input token features. Since D is typically much

smaller than H, a Self-Attention layer would result in a much smaller number of parameters. Besides, its dependency path is $O(1)$ as compared to others' $O(n)$. Therefore, the Self-Attention layer is preferred for sequence modelling although we need to arm it with some additional positional encoding. To further reduce the parameter size, a pre-defined Sentence Reader function with no parameters is finally used as in Sukhbaatar et al. (2015) for sentence encoding. Suppose we have M words in each sentence, and each word is represented by some word embedding vector w. Sukhbaatar et al. (2015) proposed to use a pre-defined position encoding combined with word vectors to read and encode a sentence as

$$f_i = \sum_{j=1}^{M} l_j \circ w_j^i \tag{4.1}$$

$$l_{jd} = (1 - j/M) - (d/D)(1 - 2j/M), \tag{4.2}$$

in which l_j represents the position encoding at position j, w_j^i stands for the embedding of the j-th word of sentence i in a review, and the feature representation of sentence i is finally embedded in $f_i \in \mathbb{R}^D$. $\circ$ corresponds to the element-wise multiplication between two vectors.

Image Encoder. On the image side, it is almost a convention to use a pre-trained convolutional neural network (ResNet in our case) to encode an image. As is usually seen in practice, some intermediate convolution layer is used to represent the image, which not only keeps spatial information but also allows for fine-grained inference on details stored in these layers. During our implementation, the *conv*4 feature from residual network (He et al., 2016) is used, which produces a $14 \times 14 \times 1024$ feature map for each input image. Then we've added a 14×14 max pooling layer afterwards to produce a fixed length feature vector. Finally, a linear transformation layer is incorporated, which outputs an embedding that is compatible (having the same dimension) and can interact with the review encoding from the hierarchical review encoder. The whole pipeline for image embedding is depicted on the right side of Figure 4.3.

Image-Sentence Attention/Relevance. In order to estimate the relevance between review sentences and an image, an attention layer is used. Suppose the hidden states from the top review encoder are $H \in \mathbb{R}^{N \times d_{hid}}$ and the image feature after linear transformation is $I \in \mathbb{R}^{1 \times d_{hid}}$. We construct the attention layer through a MLP following the design in Ma

et al. (2017):

$$\alpha = \sigma\left(H \circ I \; w_r\right) \in \mathbb{R}^N,$$

in which w_r is a column vector of $\mathbb{R}^{d_{hid}}$ to be learned, and $\circ$ does the broadcasted element-wise multiplication between text and image features. σ represents the sigmoid activation function. The intuition behind the design of an element-wise multiplication is that if both entries on the same dimension are positive or negative, it will result in a positive value with this operation. This is desired as we want to measure the relevance or the alignment between sentence and image. For example, suppose that we are recognizing an image as a cat, and some dimensions of the feature vector represent how likely the "fur" feature appears in the image, if two such feature vectors have the same sign on these entries, it's most probably that either both of them do not have fur in them, or both own the fur. We want such feature to be captured between image and sentence embeddings so that better alignment can be learned.

As in the classical encoder-decoder architecture, we can then get the context vector for this review by a weighted summation of the hidden vectors h_i along with relevance scores α_i,

$$c = \frac{1}{N} \sum_{i=1}^{N} \alpha_i h_i. \tag{4.3}$$

Intuitively, this should give us a relevant semantic context of the whole review with respect to the given image. We want our model to select as many relevant sentences as possible by jointly training with the following critic network. More specifically, the sentences that are more relevant to the image content should have high attention scores, which leads to higher weights in the final review encoding vectors, and low scores for sentences that are not so relevant. Eventually, we can construct non-relevant training pairs by coupling such context vectors with other images in the same review or images from other reviews. Then we can train a discriminator/critic to distinguish these sample pairs, and backpropagate the gradient signal to the hierarchical encoder to further fine-tune our ultimate desired hierarchical encoder.

Transformer Self Encoder. As we have pointed out in the Section *Hierachical Review*

Encoder, hierarchical RNN encoders take plenty of computation resources during the optimization, especially during the backward pass. In order to further reduce the long term dependency path length (Gehring et al., 2017; Vaswani et al., 2017) between sentences and words, we replaced the top level RNN with the self attention encoder proposed in the Transformer model (Vaswani et al., 2017). In a Transformer encoder, a weighted combination of all the input tokens is used to encode each token out of the others, which produces the same set of hidden states as in a typical RNN. It starts with scaled dot-product attention between queries and keys. However, for self-attention encoders, queries, keys, and values are all the same, i.e., the input feature vectors. Besides, the authors also introduced the use of multi-head attentions, each head of which independently projects the input feature into some subspaces and then follows the multiplicative scaled dot-product attention as before. This, as they said, could push models to learn different aspects of attention features in different feature spaces. Another important feature in the Transformer self encoder is the positional encoding. Since self attention can't capture relative position relationships as in RNNs, it's essential to include such mechanisms to differentiate relative order between words. Otherwise, it would be similar to bag-of-words models and insensitive to word order. To model the relative position between sentences that appear in different locations, we follow their trigonometric sine and cosine functions for positional encoding

$$PE_{(pos,2i)} = sin(\text{pos}/10000^{2i/d_{\text{hid}}}) \tag{4.4}$$

$$PE_{(pos,2i+1)} = cos(\text{pos}/10000^{2i/d_{\text{hid}}}). \tag{4.5}$$

One intuition for selecting a trigonometric function is that it encodes relative position information naturally. We would highly recommend readers refer to their paper (Vaswani et al., 2017) for design details and supporting intuitions. We have also tried to give a short summary in Section 2.5.4.

Multi-task QSL Network

Furthermore, we augmented our vanilla QSL network with another module for other learning tasks, such as sentiment analysis. We chose sentiment analysis as the auxiliary task, mainly because this is the only human annotated label we have for a typical review. The detailed architecture for the Multi-task QSL network is depicted in Figure 4.4. The intuition is that

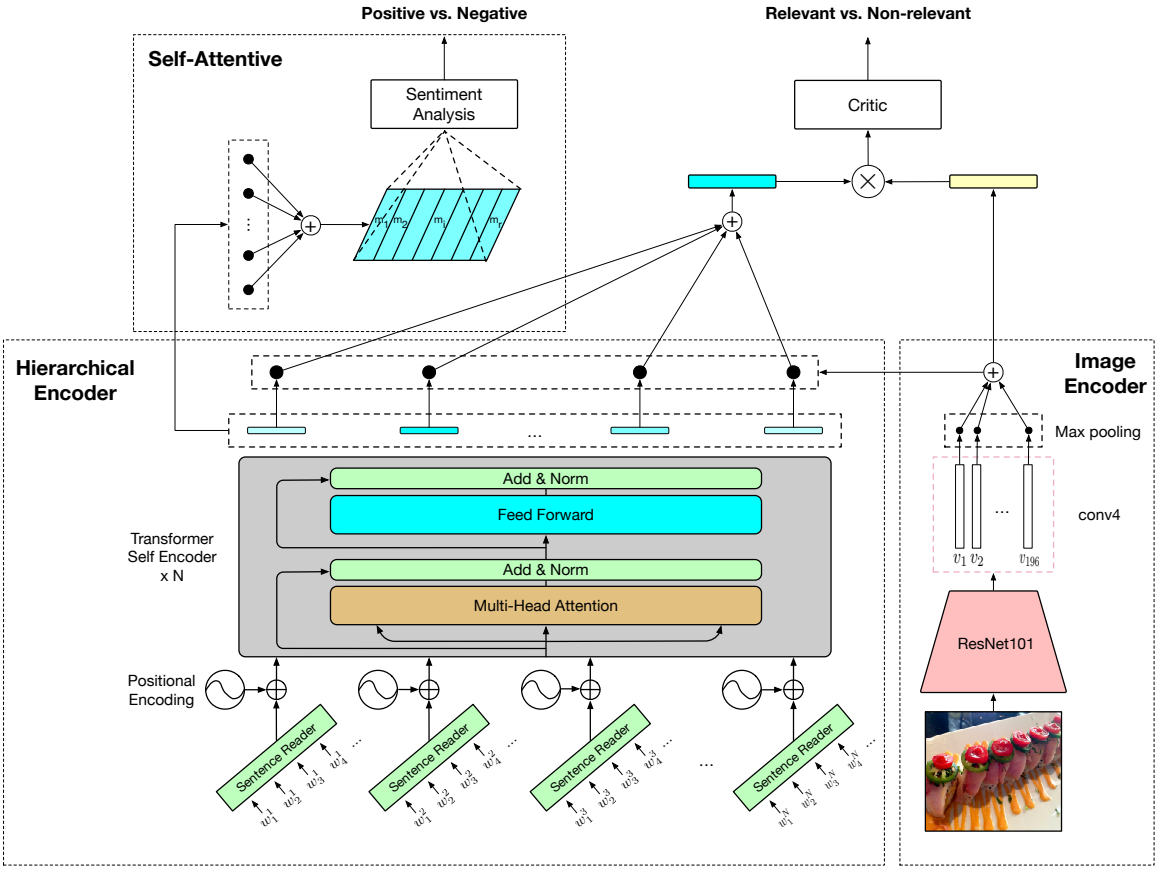


Figure 4.4: Multi-Task Quasi-Supervised Learning model architecture

when there is no image involved, readers may just depend on all the text words to identify the meaning and infer something. This naturally corresponds to the self-attentive model proposed in Lin et al. (2017). In this way, the hidden states from the top review encoder (the blue vectors from Transformer Self Encoder) would be more generalizable. Multi-task learning can also be thought of as a regularization technique, which can help reduce the overfitting problem in large deep neural network models.

4.3.2 Loss Function

For the vanilla Quasi-Supervised Learning network, suppose we have a review with some text T and images $I = \{I_t\}_{t=1}^{M}$. Then we can get the corresponding context vector $c_t = g(T, I_t)$, which is assumed to focus on sentences relevant to image I_t. Here g represents the

hierarchical encoder in the diagram when our discriminator is represented by some function f, which takes the CNN feature of input image I_t and a context vector c_t. Then we can define our objective function based on the max margin ranking loss as

$$L(\theta; T, I) = \frac{1}{M}\sum_{k=1}^{M}[\max(0, 1 - f(\text{CNN}(I_{t=k}), g(T, I_{t=k})) \tag{4.6}$$

$$+ f(\text{CNN}(I_{t\neq k}), g(T, I_{t=k})))], \tag{4.7}$$

in which CNN represents some pre-trained convolutional neural networks. In our case, it's *conv*4 from ResNet101 that is pre-trained on ImageNet. Thus, we want to find the parameters θ^* that minimize this function.

Next we turn to the multi-task learning scenario, where we will use the hidden states to do sentiment classification. Unlike the normal attention layer with outside query signal guidance, the self-attentive model (Lin et al., 2017) has no signal besides its own hidden states, which means it only contains key-value pairs. Here an attention layer can be represented as

$$a = \text{softmax}(w_{s2}\text{tanh}(W_{s1}H^\top)), \tag{4.8}$$

where w_{s2} and W_{s1} are learnable parameter matrices, and $H = (h_1, h_2, \ldots, h_N)$ representing the hidden states from the Transformer self encoder. The attention score should focus on some specific sentences as in our hierarchical encoder settings, but be driven by the sentiment classification goal. Finally, Lin et al. (2017) go one step further to incorporate multiple parallel attentions in order to make different attention layers focus on different parts or aspects. Correspondingly, a column vector w_{s2} is extended to a matrix W_{s2} as follows:

$$A = \text{softmax}(W_{s2}\text{tanh}(W_{s1}H^\top)). \tag{4.9}$$

The result review embedding can be obtained by matrix multiplication as:

$$M = AH, \tag{4.10}$$

which is a weighted summation of hidden states based on different aspect ('hops') of attentions. We can then feed this M into a MLP for sentiment classification.

The embedding matrix M can suffer from redundancy problems if the attention mechanism always provides similar summation weights for all the parallel attention layers. Thus we need a penalization term to encourage the diversity of the summation weight vectors across different hops of attention. As used in Lin et al. (2017), the same Frobenious penalization term is used

$$P = ||A^\top A - I||_F. \tag{4.11}$$

Here $||\cdot||_F$ stands for Frobenious norm. This encourages entries on the diagonal to be as close to 1 as possible and the rest to be near 0. Therefore, in different hops, only one word or sentence is focused and different hops would pay attention to different sentences. Therefore, our classification loss turns into:

$$L_{\text{sent}} = L_{XE} + \lambda_1 P, \tag{4.12}$$

in which L_{XE} stands for the cross entropy loss for binary sentiment classification and λ_1 is a hyper-parameter for multi-head attention strength.

Besides the QSL and sentiment classification loss, we also want the relevance scores between sentences and images to be as sparse as possible. This is based on the assumption that only one or two review sentences would be talking about the input image. Other sentences would focus on other images in the same review or not talk about any image content. This should result in sparse attention scores, in which most of the attention strength is close to 0. Inspired by the sparse encoding setting in Ma et al. (2017), we borrow the sparse attention penalty term here:

$$L_{\text{sparse}} = \sum_{t=1}^{N} \text{KL}(p||\hat{p}_t), \hat{p}_t = \frac{1}{m}\sum_{j=1}^{m} \alpha_{jt}, \tag{4.13}$$

in which p is a pre-defined hyper-parameter that represents the expected sparsity of attention scores (mean value of a Bernoulli distribution), and m is the batch size during the training process. Here we are using the KL divergence loss to quantify the disagreement between attention distribution and a pre-defined Bernoulli distribution with a mean of p.

Finally, together with our QSL cost, our loss becomes

$$L = L(\theta; T, I) + \lambda_2(L_{XE} + \lambda_1 P) + \lambda_3 L_{\text{sparse}}, \tag{4.14}$$

in which λ_1, λ_2, and λ_3 are the hyper-parameters that should be empirically selected to balance between different loss terms.

4.4 Experiments and Results

4.4.1 Dataset

Our experiments are based on a Yelp Restaurant review dataset that has both images and texts. It covers 18 major cities across the United States, like San Francisco, Chicago, Las

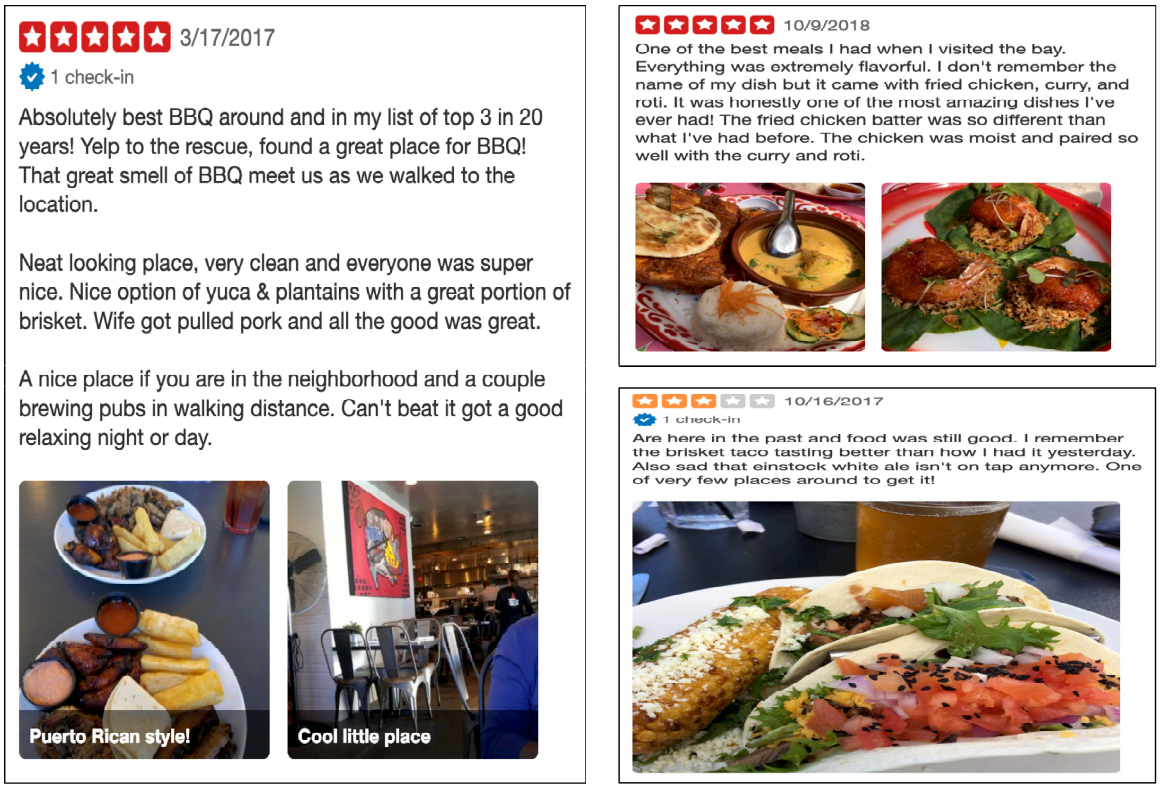


Figure 4.5: Review examples from Yelp Restaurant dataset, used with permission of Yelp

Vegas, etc. Some examples are picked and can be seen in Figure 4.5. That we are choosing

a restaurant dataset for evaluation is mainly because such reviews will talk more about the food and the environment of the restaurant, which is also probably depicted in the image. The dataset we use currently contains 234,820 reviews, each with up to 3 images We filter out reviews that either are empty or do not have images. The number of reviews in each city can be found in Table 4.2. They are merged and split into 'train', 'valid', and 'test'

Table 4.2: Number of reviews and images in each city from the Yelp Restaurant dataset

City and State	# of reviews	# of images
Aiken, SC	4,045	10,135
Americus, GA	5,764	13,297
Baltimore, MD	8,276	17,828
Bradenton, FL	10,334	21,489
Champaign, IL	13,043	38,245
Chattanooga, TN	9,127	16,368
Chicago, IL	18,227	26,459
Dallas, TX	17,298	21,894
Fort Worth, TX	15,682	30,225
Indianapolis, IN	82,45	24,707
Las Vegas, NV	18,757	58,359
Manhattan, NY	25,085	36,205
Memphis, TN	11,358	17,048
Myrtle Beach, SC	12,338	21,459
Orlando, FL	21,254	29,257
Pueblo, CO	8,940	20,422
San Francisco, CA	17,178	34,564
San Jose, CA	9,868	15,348

according to the ratios of 3:1:1, which results in training samples of 140,892 reviews, and validation as well as test with 46,964 reviews, respectively. Correspondingly, the training dataset has a total of 271,844 images, while in the validation we have 90,414 images and the test dataset has 91,051 images. Besides, in order to quantitatively evaluate our model's performance, we've asked humans to manually label each review sentence for a given image in the review. At the time of writing this dissertation, we have 6,701 reviews labeled for evaluation.

Preprocessing. The text word sequence is first segmented into sentences based on the NLTK (Bird and Loper, 2004) sentence tokenizer. Then each sentence is tokenized into multiple words with the Stanford NLP (Manning et al., 2014) PTBtokenizer. For images,

the only preprocessing step is to resize them to 256×256, which is required by a typical pre-trained CNN for ImageNet (Deng et al., 2009) classification.

4.4.2 Implementation Details

Below are the hyper-parameters that we use for training setup. Most of the building blocks just follow the default design of the referenced Transformer paper (Vaswani et al., 2017) and others (Lin et al., 2017). In terms of sensitivity analysis on these hyper-parameters, we will do more experiments with random search (Bergstra and Bengio, 2012) or grid search in the future to select the best model.

- Total training epochs: 50;
- Penalty strength: $\lambda_1 = \lambda_2 = \lambda_3 = 1$;
- Sparse attention Bernoulli distribution mean: $p = 0.05$;
- Word embedding: 200-d; GloVe (Pennington et al., 2014) vectors pretrained on Wikipedia and Gigaword are first fine-tuned on our corpus, and then trained along with the downstream task;
- Optimizer: Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.99$;
- Batch size = 40, which fits on 2 Tesla V100;
- Learning rate: 0.001, exponentially decayed by 10 with every 10 epochs starting from Epoch 10;
- CNN: ResNet101, start fine-tuning block [5-9] after epoch 20;
- Transformer $\times 2$: $n\_head = 8$, $d\_k = 128$, $d\_v = 128$, $d\_inner = 2048$;
- Self-Attentive binary classifier: attention hidden units $da = 300$, number of aspects ('hops') $r = 35$;
- Positional encoding: max_sent_num = 116, max_sent_len=598; it's computed based on training data statistics, and the valid and test split just follow this setup by discarding sentences or words that exceed the limit.

- Early stopping based on validation performance, for which it stops training if no further improvement is witnessed during the last 5 epochs.

Based on the above hyper-parameters and corresponding building blocks for our QSL networks, we showed the parameter size of each component in our model in Table 4.3. As we can see, the parameters of the image encoder (ResNet101) occupy the majority of the total parameters, in which most of them are frozen and only ones in higher convolutional and fully connected layers are learnable. Besides, we also listed the running time of training, validation, and testing in Table 4.4, in which the model has converged on Epoch 35 based on our early stopping criterion.

Table 4.3: Parameter size of different components in QSL networks

	Transformer	ResNet101	Critic	Self-Attentive	Total Params
	5M	44M	200	0.16M	
Vanilla QSL	✓	✓	✓		49M
Multi-task QSL	✓	✓	✓	✓	50M

Table 4.4: Running time of training, validation, and testing on Yelp Restaurant dataset

	Running Time (mins)	Avg secs/batch
Training	6,123.2 (4.25 days)	2.51
Validation	50.3	1.12
Testing	48.5	1.31

During the training process, the model is considered to have converged if its evaluation performance (selected information retrieval metrics) on our human annotated dataset hasn't improved in the last several epochs (i.e., 5 epochs in our case). Although we did early stopping based on the human annotated dataset, our whole training process does not take in any such ground truth information. As indicated by our loss function in Section 4.3.2, relevant training pairs of image and review text encoding should achieve a critic score that is at least some margin (e.g., 1 in our implementation) higher than the ones that are non-relevant. In the future, we may differentiate between training pairs in the same review and ones that are not, i.e., setting larger margins for sentences and images that do not come from the same review. Together with the constraints applied on attention weights and multi-task learning for generalizable features, no human annotated labels are involved

except the review overall rating. All of the training process is based on the inner structures between images and text that are inside or outside reviews. To be more specific, text descriptions perceived after we have looked at images in the review should be more relevant to images that we haven't seen before, which can be formalized as a ranking problem of image-sentence pairs.

4.4.3 Baseline and Metrics

Table 4.5: Image-sentence congruence measure baseline models

Methods	Descriptions
Randomized	A randomly shuffled sentence list is returned.
Caption	Word match between image caption and sentences is used for ranking.
Caption & Word2vec	Cosine similarity between average word embeddings of image captions and review sentences.

In terms of baseline models for comparison, to the best of our knowledge, we haven't seen any unsupervised or weakly supervised learning models with review sentence and image matching that have been proposed before. Here first we choose the randomized results as a benchmark to show the super-effectiveness of our QSL networks. We want to see what exact minimum performance we can achieve if no prior knowledge is applied. For each image and corresponding review text, we'll first shuffle the sentences, which then yields a list that represents the ranked order of review sentences in terms of their relevance to the image. Besides, we also applied our pre-trained **Mixture w/ GANs** model in Chapter 3 to represent the image context, and then rank review sentences based on the number of exact match words between them. However, exact word match may sound too simple as a baseline, for which we also applied the embedding similarity comparison method based on the average of all word embeddings in a sentence. All of our baseline models and their corresponding intuitions behind are summarized in Table 4.5.

For quantitative evaluation, we used the classical Information Retrieval (IR) metrics Precision@n (P@n) (Zhu, 2004) and Mean Reciprocal Rank (MRR) (Craswell, 2009) to assess our models. In particular, precision at up to 3 are evaluated during our experiments. The evaluation is performed upon the $3,351$ labeled reviews, as we have halved the human

Table 4.6: Model performance comparison: ST short for single task, MT for multi-task learning

Methods	MRR	P@1	P@2	P@3
Randomized	0.2256	0.1185	0.1398	0.1534
Caption	0.2878	0.2113	0.3254	0.3887
Caption & word2vec	0.3282	0.2412	0.3711	0.4232
QSL-ST	0.4873	0.3377	0.4712	0.6608
QSL-MT	**0.4996**	**0.3514**	**0.5013**	**0.6861**

annotated reviews (6701 in total) either for validation or testing.

4.4.4 Results

Finally, we've experimented with both the vanilla and multi-task versions of our QSL models, named as QSL-ST and QSL-MT, respectively. Detailed model performance is presented in Table 4.6. According to the table, our single task QSL model has already beaten all baseline models dramatically, which indicates the high effectiveness of Quasi-Supervised Learning network based on how humans would perceive information and the inner review structure of text and images. It's obvious that we achieved more than 10% absolute performance improvement across all the evaluation metrics, and on some of them even around 20% boost is obtained. What's more, when it's augmented with a sentiment classification task (multi-task learning), quantitative performance can be further improved. We achieved a more than 120% improvement on all the quantitative metrics we use, as compared to our baseline models. This evidently shows the usefulness of our QSL network and also the importance of multi-task learning for generalizable feature learning.

Besides, we've also shown some qualitative examples in Figure 4.6 from our multi-task learning model (QSL-MT). Both the model predicted and human annotated relevant sentences are highlighted in the figure. As for our prediction, a darker color would indicate higher relevance, while in human annotated cases only relevant sentences are highlighted (with the darkest color). If we look carefully at all the listed examples, we can see that at least every relevant sentence is found out and selected by our QSL-MT model. Furthermore, some colored sentences (with lighter background colors) that are not picked during human labeling are actually somewhat related to the input query image on the left. For example,

in the first sample, our model also indicates some relevance for the sentence "Worth the drive all the way from Mckinney", which is not picked by our human annotator. As inferred from the whole review, it's clear that this sentence is indirectly saying that the food shown in the picture is quite good. A similar pattern can also be discovered in the second sample; for that our model identifies as semantically relevant the sentence "This is way better than One pot (best of south bay), trust me". This implicitly implies that our model is able to recognize the semantics in the text besides the matching between image objects and text words in the review.

In order to see how our QSL model would rank review sentences based on the input image as compared to the selected baseline models, we also show the corresponding qualitative examples in Figure 4.7. All the ground truth relevant review sentences are highlighted with orange background color. As we can see from the figure, out of the four examples, our Caption baseline model is able to only find one relevant sentence (the 3rd example) based on exact word match. When semantic word embeddings are incorporated, slight improvement is obtained like in the first and last example as the ranks of relevant sentences are increased. Finally, if we compare these ranked lists with the one produced by our QSL-MT model, we can see that our designed model is capable of moving the ranking of all relevant sentences to the top.

On the other hand, we should bear in mind that our models are all trained upon samples that have both images and text descriptions, in which we assume that most images in each review are related to the corresponding text context in some sense, with only few non-relevant images treated as noise or low quality training samples. In other words, the text description has either talked about objects in the image or is semantically related to the image content. The ratio of non-relevant image-review pair should be quite low, otherwise it may affect the efficacy of our QSL networks. It should also be noted that we only have applied our QSL networks on top of the Yelp Restaurant dataset. More testing should be performed such as with the TripAdvisor and Yelp hotel reviews, etc., to demonstrate our QSL's ability of generalization on domains other than restaurant.

Also, we did another set of experiments in terms of sentiment analysis. Transformer plus Self-Attentive encoding for binary classification (positive or negative) was tried to see how it will perform without the features from images. The classification results on precision, recall, F-1,and accuracy are shown in Table 4.7. When the multi-task QSL network is involved,

Table 4.7: Sentiment analysis performance for models w/o and w/ QSL

Methods	Precision	Recall	F-1	Accuracy
Transformer+Self-Attentive	0.8277	0.8129	0.8202	0.8232
Multi-task QSL	0.8432	0.8387	0.8409	0.8415

sentiment classification performance is further boosted with around 2% improvement as compared to the basic classifier without images' intervention, which is consistent with that sentiment classification helps the QSL network. This also addresses the general research question of whether incorporating knowledge from another domain would help with feature learning. In that respect, our Quasi-Supervised Learning framework can be a test case for verification.

4.5 Conclusion and Future Work

In this chapter, an original unsupervised or weakly supervised learning paradigm is proposed, i.e., Quasi-Supervised Learning networks, that can accurately match review sentences to images based on how humans perceive both images and text sequences. Our model consists of three neural networks that take in review sentence-image pairs and work jointly. It is end-to-end trainable on review data that has both images and texts. Our experimental results demonstrate its effectiveness on learning quantitative relevance between review sentences and images. Future work will focus on doing experiments with more review datasets from other domains, in which each sample has both images and text descriptions, and also on trying with recent state of the art sentence-image matching models to explore the connections between these two modalities. Besides, exploration with hyper-parameters based on either random search or grid search should be done in order to demonstrate our results' reliability. In addition, other information retrieval evaluation metrics such as Discounted Cumulative Gain (DCG) would also be computed.

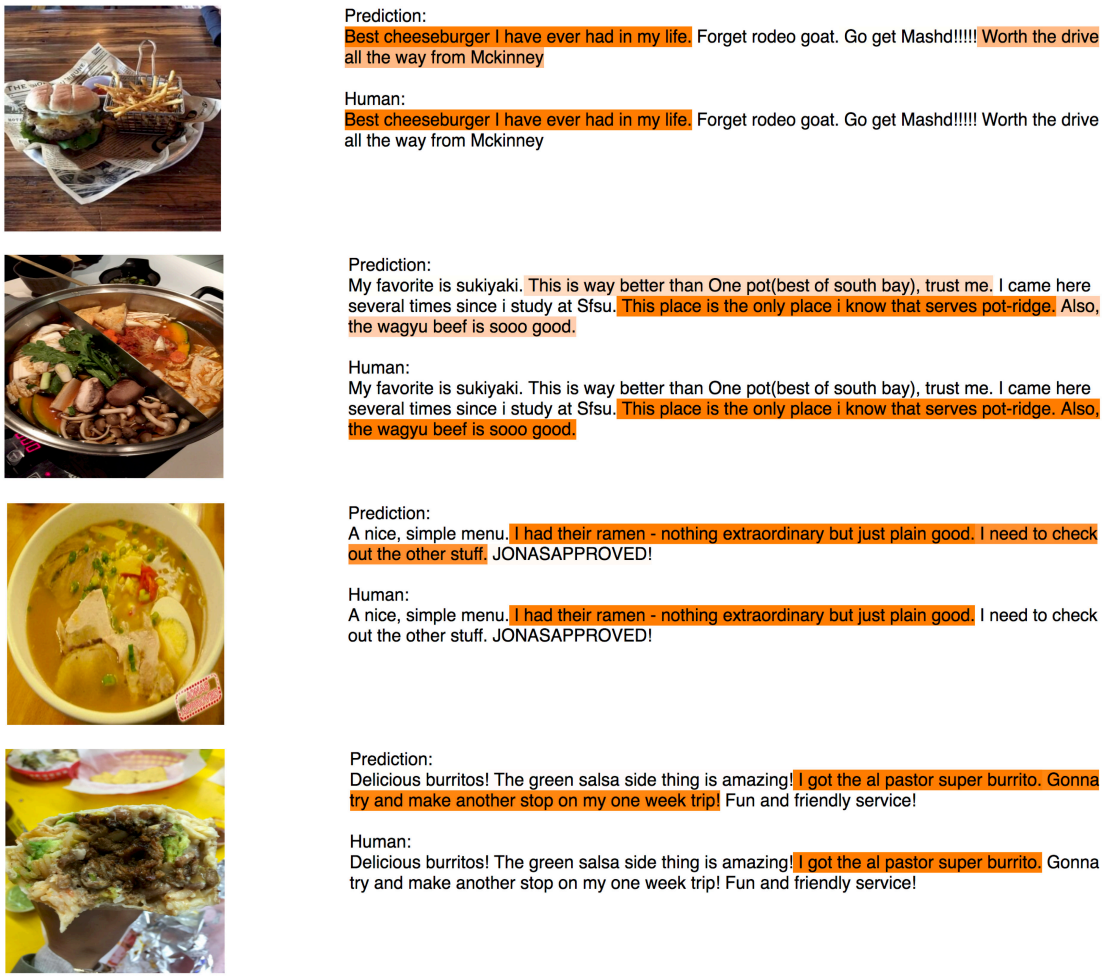
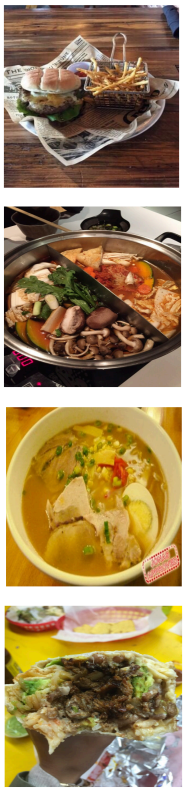


Figure 4.6: Qualitative sentence examples showing the effectiveness of our multi-task QSL network (darker color indicates higher relevance)



Caption	Caption & Word2vec	QSL-MT
1. Worth the drive all the way from Mckinney. 2. Best cheeseburger I have ever had in my life. 3. Go get Mashd!!!!! 4. Forget rodeo goat.	1. Best cheeseburger I have ever had in my life. 2. Go get Mashd!!!!! 3. Worth the drive all the way from Mckinney. 4. Forget rodeo goat.	1. Best cheeseburger I have ever had in my life. 2. Worth the drive all the way from Mckinney. 3. Go get Mashd!!!!! 4. Forget rodeo goat.
1. This is way better than One pot(best of south bay), trust me. 2. Also the wagyu beef is sooo good. 3. I came here several times since i stay at Sfsu. 4. This place is the only place I know that served pot-ridge. 5. My favorite is sukiyaki.	1. This is way better than One pot(best of south bay), trust me. 2. This place is the only place I know that served pot-ridge. 3. My favorite is sukiyaki. 4. Also the wagyu beef is sooo good. 5. I came here several times since i stay at Sfsu.	1. This place is the only place I know that served pot-ridge. 2. Also the wagyu beef is sooo good. 3. This is way better than One pot(best of south bay), trust me. 4. My favorite is sukiyaki. 5. I came here several times since i stay at Sfsu.
1. I had their ramen - nothing extraordinary but just plain good. 2. A nice, simple menu. 3. I need to check out the other stuff. 4. JONASAPPROVED.	1. I had their ramen - nothing extraordinary but just plain good. 2. I need to check out the other stuff. 3. A nice, simple menu. 4. JONASAPPROVED.	1. I had their ramen - nothing extraordinary but just plain good. 2. I need to check out the other stuff. 3. A nice, simple menu. 4. JONASAPPROVED.
1. The green salsa side thing is amazing! 2. Delicious burritos! 3. I got the al pastor super burrito. 4. Fun and friendly service. 5. Gonna try and make another stop on my one week trip!	1. The green salsa side thing is amazing! 2. I got the al pastor super burrito. 3. Delicious burritos! 4. Fun and friendly service. 5. Gonna try and make another stop on my one week trip!	1. I got the al pastor super burrito. 2. Gonna try and make another stop on my one week trip! 3. Delicious burritos! 4. The green salsa side thing is amazing! 5. Fun and friendly service.

Figure 4.7: Qualitative sentence examples from both baseline and our QSL-MT models (relevant ground truth sentences are highlighted in orange)

Chapter 5

Image Aspect Mining

In this chapter, we try to address our last research question of *how we can align topical aspects mined from text and ones detected in images, and then do fine-grained sentiment inference.* To this end, we made the hypothesis that when topical aspects (objects) detected in images also appear in the review text content, correspondence between image aspects and the ones identified in text can be built by our proposed Adaptive model. Based on this inferred correspondence strength, it can also estimate fine-grained information such as aspect level ratings.

First, we propose a new AI task named Image Aspect Mining. Specifically, given a review with images, the task for the machine is to *recognize product/service related topical aspects appearing both in the review represented in free-form natural language and in images, and then output corresponding fine-grained ratings.* This should be a more challenging task than simply answering what's inside an image and where it is. Our task requires more fine-grained image understanding, like how good or bad is the region (object) in the image w.r.t. the corresponding aspects identified in the text.

Again, we tackle this problem in a data-driven manner. Inspired by the observation that objects (aspects) detected in the image may not exactly appear in the review text, we introduce an Adaptive mechanism to let our models learn and automatically adjust how much weight we should place on image and text aspects, and then do overall rating inference. We trained such models upon datasets of reviews with both images and text, in the hope that the Adaptive layer can output a high weight if some aspects appear in both

image and text, and a low weight otherwise.

5.1 Introduction

We are witnessing rapid progress in both visual recognition (He et al., 2016; Huang et al., 2017; Krizhevsky et al., 2012) and natural language understanding (Devlin et al., 2018; Kim, 2014; LeCun et al., 2015; Vaswani et al., 2017; Winograd, 1972) with the emergence of deep learning. Sentiment analysis (Pang et al., 2008) of either images or texts is an interesting research problem that attracts lots of attention. However, most of the researchers (Kouloumpis et al., 2011; You et al., 2015) have only been focusing on inferring the sentiment of the whole image or text, which highly restricts the understanding of the context. For example, You et al. (2015) tried to figure out the overall sentiment of an image with deep convolutional neural networks based on large-scale Flickr images labeled by some baseline models, and fine-tune it with a small number of manually annotated samples. In terms of sentiment granularity, Wang et al. (2010) have tried to do aspect mining for online hotel reviews, which successfully mines topical aspect level ratings instead of the overall rating based on pure text context. But to the best of our knowledge, no one has done such fine-grained level aspect mining on images, assisted with the semantic context provided by texts.

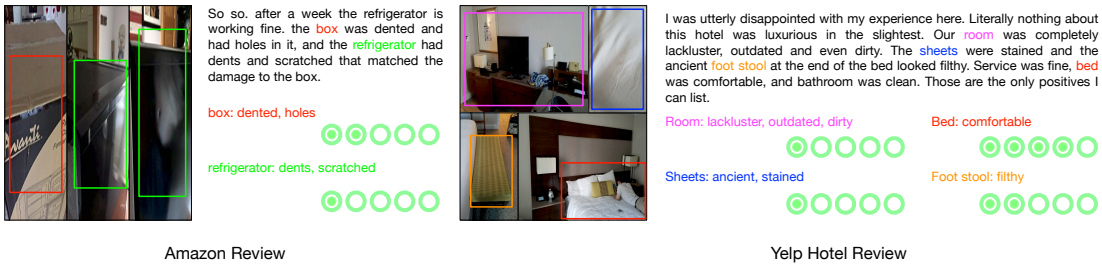


Figure 5.1: Image Aspect Mining example

With this in mind, we introduce a novel AI task – **Image Aspect Mining** – and propose to develop attention based adaptive deep learning models, and an evaluation protocol, for this task. More formally, given an image I with corresponding review text sequence $\{w_1, w_2, \dots\}$, the task for the machine is to *recognize product/service related topical aspects* A_i (word entity in text or object in an image) *appearing both in review text represented*

in free-form natural language and in images, and then output corresponding fine-grained sentiment/rating r_i.

Consider the review examples shown in Figure 5.1. With images and text as input, this task first requires the machine to selectively identify aspect attributes appearing in both review text and images (*i.e.*, room, bed, sheets, and foot stool). This calls for the successful discovery of potential aspects in text besides previous mentioned ones appearing in both sides (like hotel, service, and bathroom), detection of possible objects (TV set, pillow, table lamp, and so forth) in the input image and co-reference resolution (*e.g.*, shoe stool corresponds to the orange bounding box region in the image). Next, based on the text context description and fine details of the detected objects in the image, it should be able to determine each aspect's quality. Note that the description 'The sheets were stained' gives negative assessment for the aspect. Correspondingly on the image side, the details of 'sheets' should be perceived in terms of how clean they are. This also relates to the ***congruence*** requirement between review text and image content that we've tackled in Chapter 4, which can help identify potential problematical reviews. In the end, the machine needs to finalize a rating decision on each aspect. Such difficulties make our proposed task a highly challenging but also interesting one.

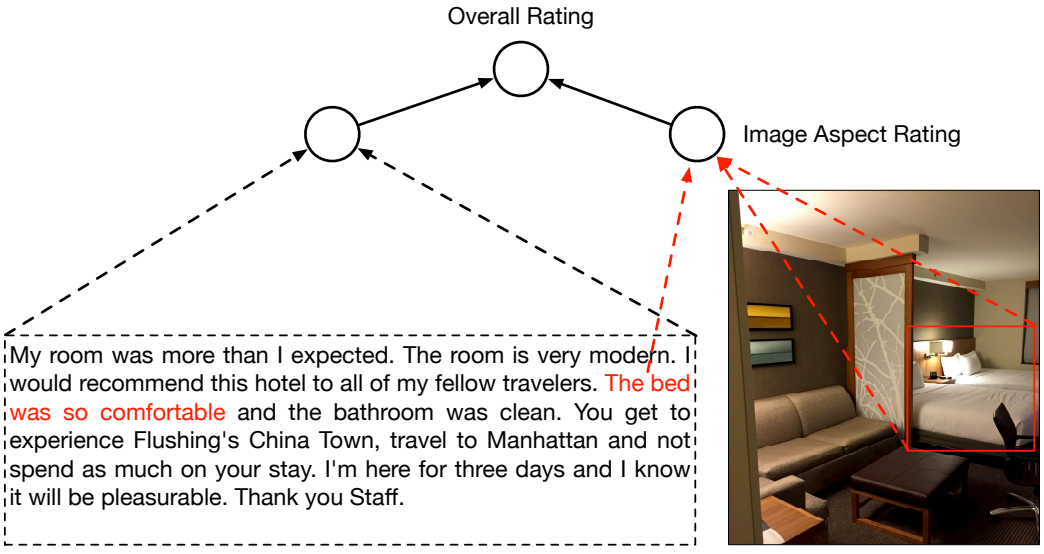


Figure 5.2: Intuitions behind our Adaptive Image Aspect Mining model

In this chapter, we show that based on the aspect level congruence/alignment learned between image objects and text entities in the review (whether an aspect appears in both

image and text), our proposed Adaptive model is able to perform better than baseline models. You may find the hierarchical inference intuitions for how we design the Adaptive model in Figure 5.2. Suppose we have identified the aspect 'bed' in the image as the red bounding box, and we want to infer how good or bad it is with the attached review text on the left. We would assume that we can figure out the rating of this image aspect only with the information in this bounding box and the corresponding descriptions highlighted in red. However, what we can only get access to is the overall rating for a whole review. This requires us to build a connection between the image aspect rating and the overall rating. In that respect, we make the further assumption that the remaining information necessary for overall rating prediction should only come from the text. Based on this intuition, we designed our Adaptive models with hierarchies on top of different neural networks.

5.2 Related Works

Multi-discipline Tasks. Image aspect mining is treated as a new AI task. In that respect, it is related to the multi-discipline tasks which involve knowledge of disparate domains like computer vision and natural language processing. Image captioning (Farhadi et al., 2010; Ferraro et al., 2015; Kulkarni et al., 2013; Lu et al., 2017; Mitchell et al., 2012) and video captioning (Guadarrama et al., 2013; Rohrbach et al., 2013) both require the machine to generate subtitles of sentences for the input image or video. Typically, a convolutional neural network is used for image or video encoding and a recurrent neural network is applied to decode the subtitle sequence. Like in our proposed Adaptive model, both CNN and RNN are deployed for image and text feature extraction, respectively. It's worth noting that our Adaptive model is inspired by the image captioning design of Lu et al. (2017), in which whether we should look at an image for caption word prediction is incorporated. Visual question answering (Antol et al., 2015; Fukui et al., 2016) and visual dialog (Das et al., 2017a,b) go one step further. Computers are asked to answer questions or chat with humans in natural language with a focus on an image. This puts forward to a deeper understanding of an image with respect to not only the objects inside, but also their relationships. All of them have input from both images and text sequences, which should share similar neural network architectures. However, in such tasks, images are mostly perceived as a whole, although some have tried to incorporate soft attention layers (Bahdanau et al., 2014; Olah and Carter, 2016) to do fine-grained feature embeddings. Since in our scenario, we have

to extract aspects (objects) from images and then align them with text sequences, more advanced layers bearing prior knowledge like object detection are required.

Fine-grained Visual Recognition. Image aspect mining can be seen as a task of fine-grained image perception. As indicated in our task description, we need to localize/identify the actual aspects (objects) from an entire image, which relates to some object detection frameworks. Ever since AlexNet (Krizhevsky et al., 2012) won the ImageNet (Deng et al., 2009) competition in 2012, convolution neural networks have been adopted everywhere in the computer vision community. Faster R-CNN (Ren et al., 2015) and YOLO (Redmon and Farhadi, 2017) are two key pipelines for image object detection. Both of them take convolution features for each region and try to accurately classify/localize simultaneously. Faster R-CNN is based on the original R-CNN (Girshick et al., 2014, 2016) framework, which first relies on some separate bounding box proposal approaches such as selective search (Uijlings et al., 2013), objectness (Alexe et al., 2012), constrained parametric min-cuts (CPMC) (Carreira and Sminchisescu, 2012), etc., to localize the potential object. Then a convolutional neural network is used to determine its object category. As compared to the R-CNN pipeline, YOLO would combine these two stages together and jointly learn both the location and class of the objects, which achieves higher computation efficiency during inference. In our work, we take the Faster R-CNN as the backbone network for our image aspect extraction, as it achieves higher accuracy, which is what we care about in terms of aspect recognition.

Text Encoding. Typical text sequence encoding would turn to gated recurrent neural networks like LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014). It's unfortunate that they are still not able to deal with long sequences when the dependency path between input features becomes longer, although the vanishing gradient problem (Hochreiter, 1998) can be alleviated by some degree. Recent progress in machine translation with the Transformer (Vaswani et al., 2017) model has dramatically reduced the dependency path between input tokens to $O(1)$, which is really surprising and inspiring. It builds the hidden state (as compared to the output of a RNN) of each input entity as a weighted combination of all the input features with the help of the attention mechanism. But it should be pointed out that such model by default won't incorporate position information of the sequence. An additional positional encoding module should be equipped to make it deal with sequence data. We describe the details of a Transformer self encoder and summarize

the intuition behind it, in Section 2.5.4. It would be helpful to take a look before jumping into the details of our Adaptive model design. Based on this, a bidirectional Transformer model (BERT) (Devlin et al., 2018) with masked language model pre-training has been proposed and used in backbone networks for various NLP tasks, among which it sets new records for all of them. The masked language model would first replace a specific word in the sequence with a special '[Mask]' token, and then, based on the embedding of this position from the Transformer encoder, we want to predict or recover the underlying exact word, which is quite similar to the continuous bag of words (CBOW) word2vec (Mikolov et al., 2013) training. With its inspiring results, it naturally motivates us to use a pre-trained language model with BERT architecture as our text feature encoder, in order to see how generalizable features would help.

5.3 Adaptive Models

In this section, we introduce the Adaptive neural network architectures we designed specifically for image aspect mining. Given an image and the corresponding review word sequence as semantic context, our network consists of three modules, i.e., text sequence encoding, image aspect extraction, and how these two interact with each other. Its direct purpose is to predict the review overall rating, but we can somehow implicitly figure out the aspect level rating with our Adaptive model. Next, these individual components are described, along with the intuitions behind them.

5.3.1 Text Semantics Encoding

Assume we are given one image and the attached review text sequence of T words as raw input. These words w_t will need to be merged together in some way, which can reflect the semantic context of the image. We turn to Recurrent Neural Networks as they inherently process sequence tokens one by one with variable length. In order to take account of future words in text embedding, it's straightforward to run a bidirectional RNN as follows.

$$\overrightarrow{h_t} = \text{RNN}(w_t, \overrightarrow{h_{t-1}}) \tag{5.1}$$

$$\overleftarrow{h_t} = \text{RNN}(w_t, \overleftarrow{h_{t-1}}) \tag{5.2}$$

Here $\overrightarrow{h_t}, \overleftarrow{h_t}$ are the hidden states from the forward and backward RNN that encode how each word at time step t should correlate with either previous or future words at other time steps. We can concatenate them together as $h_t = [\overrightarrow{h_t} : \overleftarrow{h_t}]$. Considering the long dependency and vanishing gradient problem with vanilla RNN (Goodfellow et al., 2016), we choose a gated RNN variant such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014). Since both of them behave similarly in terms of final evaluation, we finally build upon GRU as it contains fewer parameters. A comparison of parameter size between them can found in Table 4.1.

5.3.2 Image Aspect Detection

When talking about image feature extraction, Convolutional Neural Networks have taken the first place ever since deep learning models have appeared. However, in image aspect mining, what we care about is each specific object (aspect) and its detailed appearance along with the object, rather than perception of the whole image. Instead of the features for the entire input, we may need the embeddings for some specific regions, which is naturally an object detection task. In our network design, we use the Faster R-CNN (especially its Region Proposal Network) as our backbone network for image aspect feature detection.

Faster R-CNN. There are mainly two modules in the Faster R-CNN model (Ren et al., 2015). The first is designed for proposing regions, which is fulfilled by a fully convolutional neural network, instead of using traditional offline methods like selective search (Uijlings et al., 2013). It is plugged in along with the following object classifier and region regressor, both of which are fully trainable. The second module comes from the original Fast R-CNN (Girshick, 2015) detector with ROI (Region of Interest) pooling layers. The Region Proposal Network is trained upon whether a proposed region is background or not and also along with a bounding box regressor. The R-CNN detector is trained to both classify the object proposals and fine-tune the bounding box boundaries. During our experiments, we use the Faster R-CNN (Yang et al., 2017) with ResNet101 (He et al., 2016) as backend that has already been pre-trained on the MS-COCO (Chen et al., 2015) dataset. It contains objects of 90 categories, which are grouped into 5 aspects that we will describe at length in the next section. In terms of the region proposal features, we only take the regions of interest from the Region Proposal Network. At the same time, these regions should turn out to be valid objects through the following R-CNN detector. Let's assume we have an

image I, so such regions of interest can be represented by

$$\text{roi} = \text{RPN\_base}(I), \tag{5.3}$$

in which RPN_base stands for the Region Proposal base network. Then we may randomly select one roi_i from all potential regions of interest (roi) that have passed the final detector as valid regions, and use it to query among text features for further interaction.

5.3.3 Text/Image Feature Interaction

In order to mine the image aspect level rating, we make the assumption that this fine-grained rating is a hidden variable, which is part of the overall rating presented to readers. If the author of a review is shown both the text and the detected object, he/she will make the final rating based on both. Inspired by the intuition described at the end of Section 5.1, we proposed an **Adaptive** model for image aspect mining, which will automatically adjust the weight placed on either image aspect or text based rating.

First, we assume that there are two kinds of features embedded inside an aspect, i.e., object entirety and details. Humans are able to identify an object purely based on the overall outline or the sketches of an object, but we aren't able to tell whether it's good or bad until the detail appearances are elaborated. Thus, the ROI pooling feature might be mapped to two subspaces as

$$\text{f}_o = W_o\ \text{roi}_i + b_o \tag{5.4}$$

$$\text{f}_d = W_d\ \text{roi}_i + b_d, \tag{5.5}$$

in which f_o and f_d are the features encoding aspect entirety and details respectively, while W_d, W_o, b_d, b_o are learnable parameters to be trained.

Now based on the object entirety, we may turn to the text features to see whether such aspects would appear in the review context. This acts as the key component in our Adaptive model since the performance of all following modules depend on how well it behaves. We would use f_o as query to search among all the hidden states through an attention layer, and

get the final context vector as:

$$z_t^o = w_o^\top \tanh(W_f^o \, \mathrm{f}_o + W_h^o h_t) \tag{5.6}$$

$$\alpha_t^o = \mathrm{softmax}(z_t^o) \tag{5.7}$$

$$c_o = \sum_{t=1}^{T} \alpha_t^o h_t, \tag{5.8}$$

in which W_f^o, W_h^o, and w_o are parameters to be learned. Then we can feed these features into a multi-layer perceptron (a.k.a. MLP or 1-layer feed forward net) to generate the probability or strength of whether an image aspect would appear in the text semantics.

$$a_p = c_o \odot \mathrm{f}_o \tag{5.9}$$

$$p_i = \sigma(W_p a_p + b_p) \tag{5.10}$$

Here W_p and b_p are learnable parameters and $\odot$ stands for element-wise multiplication. $p_i \in (0, 1)$ should give us the preference an author placed on this image aspect when he/she is rating the product/service. To be more accurate, it reflects how likely the detected object would appear or be described in the text sequence.

Next let's estimate the corresponding image aspect rating. It's intuitive that we would rely on the detail feature f_d to predict it, but we should also consider the text semantics. We would first feed f_d and h_t together into another attention layer as in Eqs. 5.6–5.8 to get the context vector c_d as below.

$$z_t^d = w_o^\top \tanh(W_f^d \, \mathrm{f}_d + W_h^d h_t) \tag{5.11}$$

$$\alpha_t^d = \mathrm{softmax}(z_t^d) \tag{5.12}$$

$$c_d = \sum_{t=1}^{T} \alpha_t^d h_t, \tag{5.13}$$

Afterwards, we concatenate c_d and f_d together, and through another MLP, it is able to output the image aspect rating r_i.

$$a_i = [c_d : f_d] \tag{5.14}$$

$$r_i = \mathrm{Hardtanh}(W_i a_i + b_i) \tag{5.15}$$

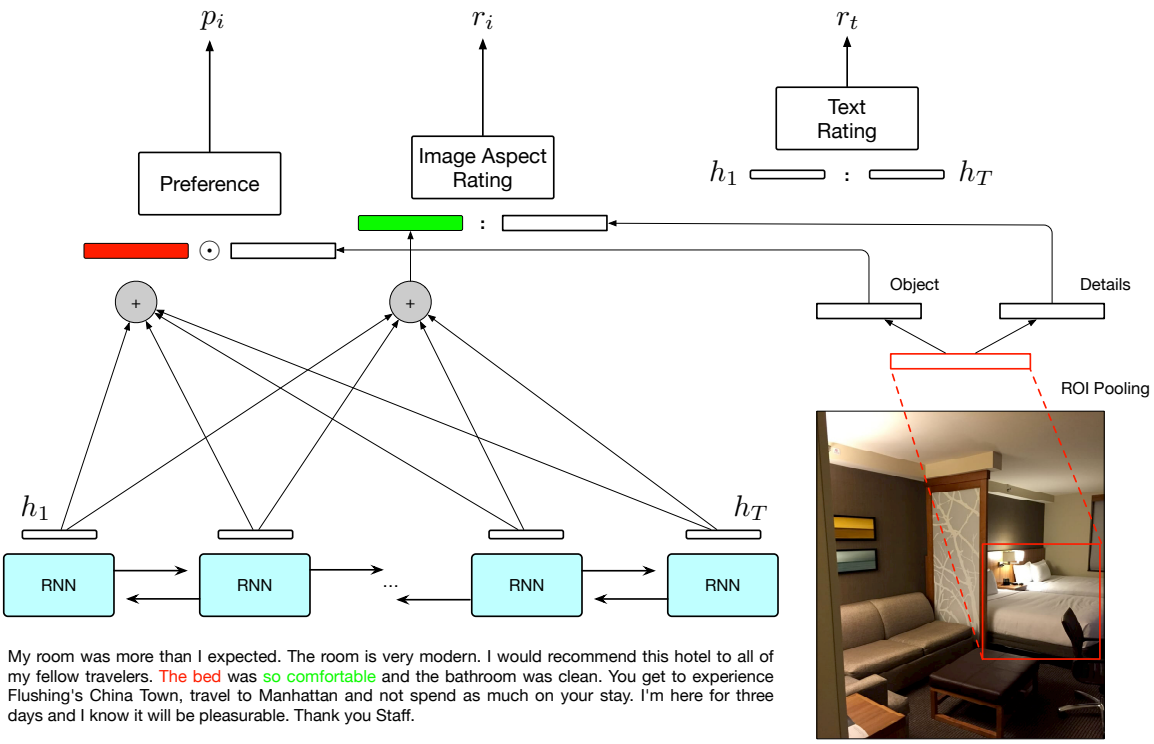


Figure 5.3: Adaptive model architecture for Image Aspect Mining

Here W_i and b_i are parameters and Hardtanh would clip its input in between 0 and 1, defined as:

$$\text{Hardtanh}(x) = \begin{cases} 0 & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x > 1 \\ x & \text{otherwise} \end{cases} . \tag{5.16}$$

We choose Hardtanh as the activation function because we've preprocessed our overall ratings and shrink them all into $[0, 1]$.

On the text side, we combine h_1 and h_T together as the input to a MLP, which would predict the overall rating based purely on text semantics.

$$t_i = [h_1 : h_T] \tag{5.17}$$

$$r_t = \text{Hardtanh}(W_t t_i + b_t) \tag{5.18}$$

Here W_t and b_t are parameters to be learned throughout the whole neural network.

Finally, we can combine all these signals together to predict the overall rating at hand:

$$r = p_i r_i + (1 - p_i)\ r_t. \tag{5.19}$$

The overall neural network architecture is shown in Figure 5.3. As the value of p_i is dynamically changing based on the alignment between the detected objects and the semantic context in the text, we name it as Adaptive model. We hope that our network would be able to learn such correspondence with review text and image training samples.

5.3.4 BERT as Language Model Pre-training

Transfer learning is ubiquitous within the deep learning community, as different tasks can probably share some low level features. For example, our image perception model here is directly borrowed from the pre-trained Faster R-CNN although its parameters are all fixed (i.e., without fine-tuning). We don't fine-tune the model here because it requires the following detector to both classify and outline the bounding box boundary. But it would be better to fine-tune the Region Proposal Network along with our Adaptive model if ground truth labels for our potential objects in the training images are available.

In the NLP field, it's common to fine-tune based on a pre-trained language model (Howard and Ruder, 2018) besides using word embeddings such as GloVe (Pennington et al., 2014) or Word2vec (Mikolov et al., 2013) that have previously been trained on a large corpus. It would be even better to fine-tune on our desired dataset first before training directly with the downstream tasks, if a commonly used word embedding is borrowed. Besides these techniques, what other embeddings or pre-trained models can we fine-tune? Motivated by the superior performance of BERT (Devlin et al., 2018) on various tasks, we also pre-train the "Masked" Language Model (MLM) with BERT architecture and then fine-tune it with our Adaptive model.

The key idea behind BERT is just a bidirectional version of the self-attention module proposed in Vaswani et al. (2017) that is trained with MLM. During the self-attention encoding, it can only get access to either all the previous words or the future words, which are implemented by two Masked Self-Attention layers. As in Vaswani et al. (2017), an

attention layer can be formalized with key-value pairs K and V, input query Q:

$$\text{context} = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{5.20}$$

Here $Q, K, V \in \mathbb{R}^{m \times d_k}$. They can be our m input word embeddings with dimension d_k. In order to prevent the encoder from getting access to future or previous words in the two unidirectional language model, maskings would be required as:

$$M_{ij}^{fw} = \begin{cases} 0, & i < j \\ -\infty, & \text{otherwise} \end{cases}, \quad M_{ij}^{bw} = \begin{cases} 0, & i > j \\ -\infty, & \text{otherwise} \end{cases} \tag{5.21}$$

in which $M^{fw}, M^{bw} \in \mathbb{R}^{m \times m}$ are the mask matrices that should be added in the attention layer before being normalized with a softmax function. For example, our forward context vector can be calculated as

$$\text{context}^{fw} = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M^{fw}\right)V, \tag{5.22}$$

which contains m hidden states as in a typical RNN. Then these two context vectors context^{fw} and context^{bw} would be concatenated together to predict the center word at the current time step, which is unseen and marked as "[Mask]" in the input layer.

After pre-training with the Masked Language Model, it can be plugged into our image aspect mining Adaptive model for further fine-tuning, namely the GRU for text encoding is replaced with this BERT architecture. You can find the details of our Adaptive model supported by the BERT backbone in Figure 5.4.

5.4 Experiments and Results

5.4.1 Evaluation Protocol

One major challenge for our image aspect mining task is the lack of a quantitative evaluation metric for the aspect rating extracted from images. Here we define the Mean Squared Error (MSE) upon this signal to assess aspect rating prediction. Formally, suppose s_{di}^* is the ground-truth rating for aspect A_i in a review d. Δ_{aspect}^2 directly computes the average of

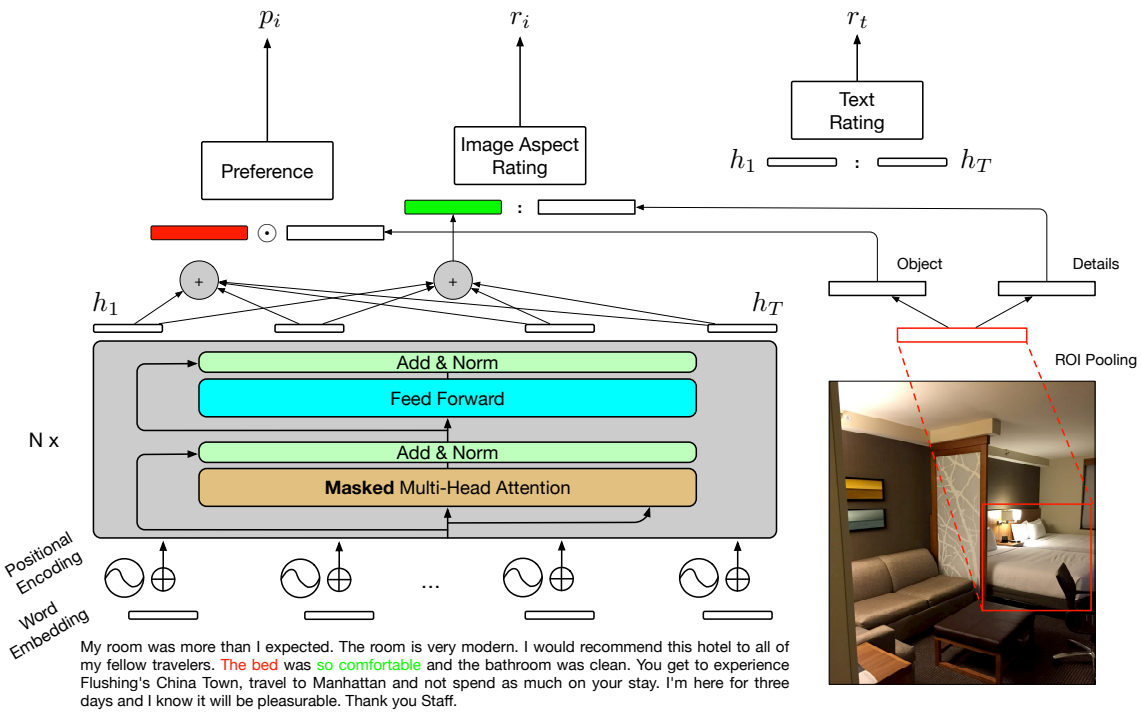


Figure 5.4: Adaptive model architecture for Image Aspect Mining with BERT pre-training

the absolute difference between the predicted aspect rating s_{di} and s^*_{di} among all the testing samples, and is defined as:

$$\Delta^2_{aspect} = \sum_{d=1}^{|D|} \sum_{i=1}^{k} (s_{di} - s^*_{di})^2 / (k \times |D|). \tag{5.23}$$

It's worth noting that there might be cases in which the model can't detect the ground-truth aspects. For these aspects, we would just set $(s_{di} - s^*_{di})$ as the maximum interval in our rating system (i.e., 1). Under such evaluation design, models with low Δ^2_{aspect} would be preferred. Besides, we could also use the MSE on the overall rating as an implicit indicator, which is also the value of our loss function during training. If a low training loss is obtained, we may infer that it benefits from a good estimate of the aspect level rating.

5.4.2 Dataset

Our experiments are done with both the TripAdvisor Hotel and Yelp Restaurant review datasets, in which each review is provided with both images and text descriptions. They include 18 major cities across the United States, among which we only select the reviews that include images and the text part of the ones that are not empty. After this filtering, we have 133,473 reviews in total for the TripAdvisor Hotel dataset, which are then divided into 'train', 'valid', and 'test'. For the hotel dataset, the number of reviews and corresponding images in each city is listed in Table 5.1. On the other hand, there are 234,820 reviews in

Table 5.1: Number of reviews and images in each city from the TripAdvisor Hotel dataset

City and State	# of reviews	# of images
Aiken, SC	2,045	5,176
Americus, GA	3,772	6,482
Baltimore, MD	6,276	9,985
Bradenton, FL	5,484	15,403
Champaign, IL	6,157	19,268
Chattanooga, TN	5,132	8,964
Chicago, IL	9,259	15,459
Dallas, TX	8,188	16,894
Fort Worth, TX	7,682	17,258
Indianapolis, IN	4,384	14,707
Las Vegas, NV	10,775	29,851
Manhattan, NY	14,085	28,205
Memphis, TN	7,854	10,946
Myrtle Beach, SC	7,775	18,496
Orlando, FL	12,278	25,357
Pueblo, CO	7,969	13,479
San Francisco, CA	9,382	24,767
San Jose, CA	4,976	10,545

total for the Yelp Restaurant dataset after initial screening. We've experimented with this dataset before in Chapter 4, in which Table 4.2 shows the number of reviews for each city. The same splitting ratio is applied here and the numbers of review samples and images with each split are described in Table 5.2 and Table 5.3.

In terms of the ground truth for aspect level ratings, the TripAdvisor hotel dataset provides 5 pre-defined categories, i.e., 'value', 'location', 'rooms', 'cleanliness', and 'service',

Table 5.2: TripAdvisor Hotel Image Aspect Mining dataset statistics

Split	# of Reviews	# of Images
train	80,083	174,653
valid	26,694	58,439
test	26,696	58,150

Table 5.3: Yelp Restaurant Image Aspect Mining dataset statistics

Split	# of Reviews	# of Images
train	140,892	271,844
valid	46,964	90,414
test	46,964	91,051

Figure 5.5: A typical review example from TripAdvisor Hotel dataset, used with permission of TripAdvisor

which can be used for our quantitative aspect level evaluation. A typical review sample from the TripAdvisor dataset is shown in Figure 5.5, in which images are put together as a whole. But it's obvious that only the 'rooms' aspect is for the exact objects that might be detected by a typical object detection framework, which could include various sub-objects like bed, sheets, TV, etc. On the other hand, based on the pre-trained Faster R-CNN (Yang et al., 2017) model on the MSCOCO (Chen et al., 2015) dataset, there are 90 classes in total. However, they are not directly aligned with the ground truth labels in our TripAdvisor dataset. Therefore, we need to manually group these categories together, to correspond to the image aspects we want to mine. The details of such correspondence are summarized in Table 5.4. However, for the Yelp Restaurant dataset, unfortunately there are NO ground truth aspect level ratings as shown in Figure 5.6. What we can only get access

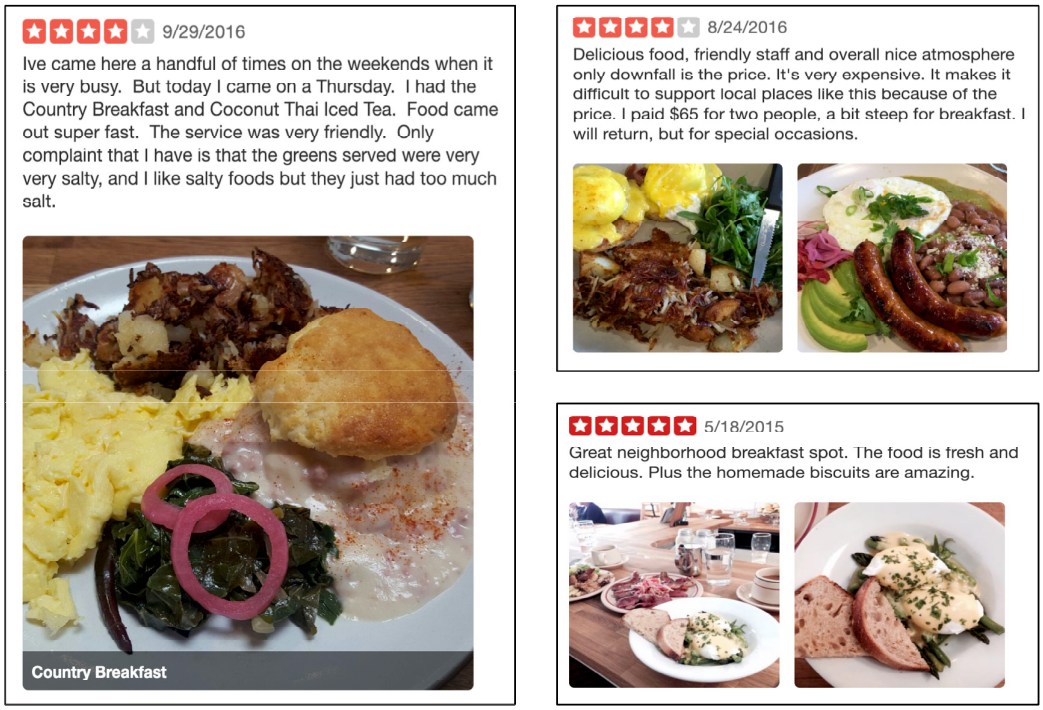


Figure 5.6: Typical review examples from Yelp Restaurant dataset, used with permission of Yelp

to is the overall rating information. In order to apply our Adaptive model on top of the Yelp Restaurant dataset and evaluate it, we also try to reply on implicit evaluation metrics such as the prediction for overall rating, for which we assume that better prediction of aspect rating can help the final overall rating prediction. In addition, similar correspondence between image aspects and object detection categories should also be defined in order for our Adaptive model to work on restaurant data, which is supposed to be largely different from the ones found in hotel images and is summarized in Table 5.5. It should be noted that its main aspects are 'food' and 'scenes' (or outdoor), and we incorporate ones originally appearing in the 'rooms' aspect into the 'others' aspect here. Since there is no ground truth aspect rating here, such grouping here is just for object detection and aspect definition. A better grouping might result in lower error performance, which clearly calls for more investigation in the future.

As can be seen from Table 5.4, based on MSCOCO categories, unfortunately there is only one image aspect in our defined groups that has corresponding ground truth in the TripAdvisor dataset, namely 'Rooms'. Currently, we calculate our proposed aspect rating

Table 5.4: Correspondence between MSCOCO categories and TripAdvisor Hotel image aspects

Our Image Aspects	MSCOCO Categories
Rooms	tv, bed, lamp, cup, chair, couch, etc.
Food	apple, sandwich, banana, donut, cake, etc.
Scenes	bench, building, ocean, etc.
Others	cellphone, stop sign, kite, etc.

Table 5.5: Correspondence between MSCOCO categories and Yelp Restaurant image aspects

Our Image Aspects	MSCOCO Categories
Food	apple, sandwich, banana, donut, cake, etc.
Scenes	bench, building, ocean, etc.
Others	tv, bed, lamp, stop sign, kite, etc.

metric based only on this 'Rooms' aspect. In the near future, we may rely on the congruence measure model in Chapter 4. The image-sentence matching model for this dataset will be trained first, and then the sentences with high relevance can be treated as descriptions for the image. If visual words, that appear in pre-defined MSCOCO categories, can be found in the sentence, we may treat the sentiment of the sentence scaled by the relevance score as our ground truth aspect rating. Besides, some detected objects might be related to different aspects per hotel aspect definition such as beds. Bed itself can be included in the 'Rooms' aspect, but the appearance of the bed could also be a good indicator of 'cleanliness'. This inspires us to explore with probabilistic graphical models (PGMs) for image aspect rating prediction, which will assign probability values to different object categories in the MSCOCO for each image aspect. That is planned to be done in our future work.

5.4.3 Implementation Details

Listed below are the hyper-parameters that we use for training setup. In terms of sensitivity analysis on these hyper-parameters, we will do more experiments with random search (Bergstra and Bengio, 2012) or grid search in the future to select the best model.

- Total training epochs: 50;

- Word embedding: 200-d, GloVe (Pennington et al., 2014) vector first fine-tuned on our corpus and then trained along with other networks;
- Optimizer: Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.99$;
- Batch size = 16, which fits on 2 Tesla V100;
- Learning rate: 0.001, exponentially decayed by 10 with every 10 epochs starting from Epoch 20;
- Gradient clipping (Pascanu et al., 2012): maximum gradient norm= 0.25 for weights inside GRU;
- Weight decay (or l_2 norm regularization strength): 5×10^{-4};
- GRU hidden states: 200-d;
- BERT: $n\_head = 8$, $d\_k = 128$, $d\_v = 128$, $d\_inner = 2048$;
- Positional encoding: max_sequence_len=927; it's computed based on training data statistics, and the valid and test split just follow this setup by chopping at the maximum length;
- Early stopping based on validation MSE loss, for which it stops training if no further improvement is witnessed during last 5 epochs.

5.4.4 Model Performance

To the best of our knowledge, image aspect mining can be treated as a brand new AI task and there are no previous approaches that have touched this. In that respect, it would be difficult to find and compare with suitable baseline models. Since our model is trained for the prediction of the review overall rating, one potential comparison could be based on the errors of this signal. In terms of this, our baseline model would be the overall rating prediction with pure text features, which uses concatenated first and last hidden states from a bidirectional GRU, and we name it as **GRU-text**. We use the encoded text information as input to directly estimate the actual overall rating as:

$$r = \text{Hardtanh}(W[h_1 : h_T] + b), \tag{5.24}$$

Table 5.6: Image Aspect Mining model performance comparison on TripAdvisor Hotel dataset

Model	Train MSE	Valid MSE	Test MSE	r^2	Δ^2_{aspect}
GRU-text	0.0927	0.1247	0.1252	89.73%	-
Adaptive	0.0198	0.0467	0.0473	96.12%	0.0265
Adaptive-BERT	0.0174	0.0448	0.0449	96.32%	0.0246

Table 5.7: Image Aspect Mining model performance comparison on Yelp Restaurant dataset

Model	Train MSE	Valid MSE	Test MSE	r^2
GRU-text	0.0872	0.1173	0.1177	90.21%
Adaptive	0.0174	0.0428	0.0439	96.34%
Adaptive-BERT	0.0158	0.0425	0.0421	96.49%

in which W and b are learned through the MSE loss for linear regression, and [:] stands for the operation of concatenation.

Then we experimented with our Adaptive model and also the one pre-trained with BERT Masked Language Modelling. The performance on two datasets for both the baseline and our adaptive models are shown in Table 5.6 and Table 5.7. We've listed the mean squared error for the sets train, valid, and test. It's obvious that when image aspects are involved, the errors for predicting overall ratings are dramatically reduced by more than 70% on the hotel dataset. More boost (around 80% improvement) is witnessed with the Yelp Restaurant dataset. These consistent improvements on both datasets show the generalization ability of our Adaptive model. Also, it implicitly shows our model's capability to infer aspect level rating, which evidently supports our assumption that aspect level rating could be a hidden variable imperative for overall rating prediction. Besides, our proposed quantitative metric Δ^2_{aspect} and the r^2 values are also calculated. Although we can only get access to the ground truth of the 'Rooms' aspect currently in the TripAdvisor hotel dataset (no ground truth for Yelp Restaurant dataset), our model is able to predict this fine-grained rating with high precision, as compared to the MSE errors we observed for overall rating. What's more, when the BERT MLM pre-training is incorporated, we can obtain further consistent improvements on both datasets. It not only reduces the MSE value on overall rating prediction, but also on our proposed Δ^2_{aspect} metric based on results from TripAdvisor

dataset. In addition, indicated by the r^2 value, our Adaptive model aligns much better than the baseline model based on text features. This demonstrates the validity of our Adaptive model design based on the overall rating prediction hierarchies. Finally, with the support from BERT MLM pre-training, the r^2 score can be improved by more than 6%, which further shows the effectiveness of our Adaptive model for fine-grained rating prediction and the necessity of model pre-training.

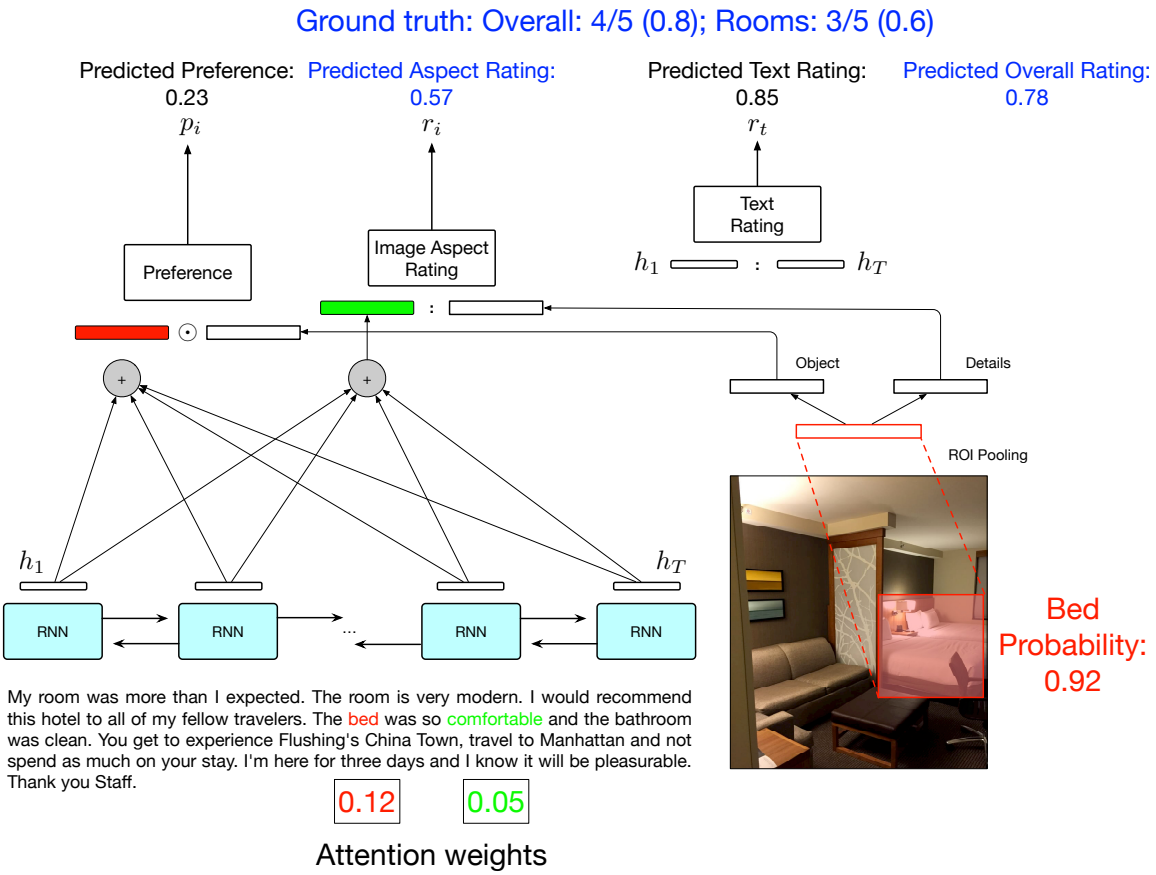


Figure 5.7: A cherry-picked example showing the process of our Adaptive model

We've also shown a cherry-picked qualitative example (based on the 'Rooms' aspect) in Figure 5.7, which describes how we first detect the image aspect and then align it with the words in the text sequence. The ground truth of both aspect level and overall ratings are shown in the figure. In this example, first the beds are detected by the pre-trained Faster R-CNN model with a probability of 0.92. Then the beds features (mapped to entirety and details subspaces) are used to query among the word entity sequence, which provides us

the attention weights for the words 'bed' and 'comfortable' as shown at the bottom of the figure. Afterwards, we can get the preference weight (0.23) and the rating (0.57) of the 'bed' object (i.e., the 'Rooms' aspect in our pre-defined category). Correspondingly, the predicted rating based on the text features is listed. In this example, it's obvious that our model is able to accurately predict both the overall and 'rooms' aspect rating, based on our provided ground truth information on the top of Figure 5.7.

5.5 Conclusion and Future Work

In this chapter, a novel AI task named Image Aspect Mining is proposed, which does fine-grained image understanding in terms of aspect level rating with review text as the semantic context. On top of that, we design the Adaptive model specifically for this task, and the corresponding evaluation protocol. The experimental results showed that our Adaptive model is able to effectively predict the aspect level rating. Also, it helps reduce the error for overall rating prediction, which implicitly confirms our assumption that image aspect level rating is part of the overall rating and their connection can be modeled by our Adaptive hierarchical models. Future work will focus on expanding our dataset for aspect level ground truth. This can help make our evaluations more comprehensive. We will also try to incorporate Bayesian inference between object detection categories and image aspects defined for a specific dataset, to allow more flexible and reasonable affiliation. At the same time, extensive experiments in terms of hyper-parameter exploration will be done to find the best model and achieve better performance.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation, we design and propose neural network architectures, techniques, and new research tasks that explore and advance the frontier of visual recognition and natural language processing by going deeper with modules from the field of deep learning. State of the art deep neural models are applied to build connections between images and texts in reviews. Detailed contributions in terms of our individual research question are listed below.

- Neural image captioning models are fine-tuned with generative adversarial networks to further improve both caption quality and performance on quantitative evaluation metrics. To the best of our knowledge, we are the first to incorporate a Gumbel-Softmax sampler into GANs training of an image captioning model, which successfully converts a discrete representation into a continuous one. Besides, significant improvements on common evaluation metrics are witnessed as compared to previous state of the art approaches.
- An original unsupervised or weakly supervised learning algorithm called Quasi-Supervised Learning (QSL) is proposed, specifically for matching review sentences to images based on their quantitative semantic relevance, which can be directly applied to review data with both images and texts. In terms of training, no human annotated labels for whether a sentence-image pair is relevant or not are required, for

which prior knowledge of how humans would perceive image and text information is incorporated. Inspiring results are obtained based on information retrieval metrics such as Precision@n, as compared to intuitive baseline models.

- A novel AI task - Image Aspect Mining - is introduced, which tries to recognize product/service related topical aspects appearing both in the review represented in free-form natural language and in images, and then output corresponding fine-grained ratings or sentiment scores. Accordingly, baseline approaches and evaluation protocols are built and explored to better connect objects in images with text semantics. In particular, our designed Adaptive model performs much better in terms of overall rating prediction, as compared to the baseline models. Besides, when BERT pre-training technique is deployed, consistent improvement is attained not only on overall rating but also on aspect level rating prediction, which suggests the use of better language understanding models in the future.

Overall, in terms of deep neural network model design, based on our experience, it would be better to first identify how we humans would solve a problem and then try to model this process with neural network 'lego' blocks. For example, in our research problem of matching review sentences to an image, we first find out that text descriptions perceived after we have looked at some images in the same review should be more relevant to these images, as compared to images that we haven't seen before. This motivates us to use CNN to 'look at' images and either RNN or Transformer self encoder to 'read' sentences. Afterwards, an attention layer is added to model their interaction, which mimics how our perceived image information would influence our perception of text semantics. All these modules are 'lego' blocks to be picked by us to build a powerful 'neural' processing system. Similar patterns are applied to the design of a sentinel component in image captioning and the hierarchical structure of the Adaptive model for image aspect mining.

6.2 Publications

A number of papers have been published during my Ph.D. program. They are shown in the list below.

[1]. Florian Zach, **Yufeng Ma**, and Edward Alan Fox, "A Preliminary Analysis of Images

in Online Hotel Reviews," in *e-Review of Tourism Research (eRTR)*, vol. 16, no. 2/3, pp. 156–164, 2019.

[2]. **Yufeng Ma**, Kapil Thadani, Rao Shen, Troy Chevalier and Ganesh Parameswaran, "Transformer Interaction and Beyond," in *Tech Pulse'18, Santa Clara, CA, Dec.* 2018.

[3]. **Yufeng Ma**, Zheng Xiang, Qianzhou Du, and Weiguo Fan, "Effects of user-provided photos on hotel review helpfulness: An analytical approach with deep leaning," *International Journal of Hospitality Management*, vol. 71, pp. 120–131, 2018.

[4]. **Yufeng Ma**, Tingting Jiang, Chandani Shrestha, Edward Alan Fox, Jian Wu, and C. Lee Giles, "Scenarios for advanced services in an ETD digital library," in *Proceedings of ETD2017, the 20th international symposium on electronic theses and dissertations, Washington, DC, August 7-9*, 2017.

[5]. Zheng Xiang, Qianzhou Du, **Yufeng Ma**, and Weiguo Fan, "Assessing reliability of social media data: Lessons from mining TripAdvisor hotel reviews," in *Information and Communication Technologies in Tourism 2017*, pp. 625–638, Springer, 2017 **Best Research Paper Award**.

[6]. Zheng Xiang, Qianzhou Du, **Yufeng Ma**, and Weiguo Fan, "A comparative analysis of major online review platforms: Implications for social media analytics in hospitality and tourism," *Tourism Management*, vol. 58, pp. 51–65, 2017.

[7]. **Yufeng Ma**, Long Xia, Wenqi Shen, Mi Zhou, and Weiguo Fan, "A surrogate-based generic classifier for Chinese TV series reviews," *Information Discovery and Delivery*, vol. 45, no. 2, pp. 66–74, 2017.

[8]. Long Xia, **Yufeng Ma**, and Weiguo Fan, "VTIR at the NTCIR-12 2016 lifelong semantic access task," *Proceedings of NTCIR-12, Tokyo, Japan*, 2016.

6.3 Future Work

We will continue the work on these three research projects, and report results, as follows.

- For each of the three projects, we plan to release the related datasets and code repositories upon the acceptance of corresponding papers.
- For the image captioning project, more recent state of the art image captioning models will be explored with our GANs training, along with significance tests. We hope that the same significant improvement can be obtained, which would demonstrate that our GANs training can be a general training technique for image captioning.
- For the review congruence measure project, additional quantitative evaluation metrics in information retrieval such as Discounted Cumulative Gain (DCG) will be calculated. Furthermore, more datasets from other than the restaurant domain are supposed to be explored to see our QSL model's generalizability. Also, it's essential to perform grid search or random search on hyper-parameters to analyze their sensitivity.
- In terms of the image aspect mining project, we plan to expand both TripAdvisor hotel and Yelp Restaurant datasets for aspect level ground truth labels, in order to do more comprehensive evaluation. We will also try to incorporate Bayesian inference between object detection categories and image aspects to allow for more flexible and reasonable affiliation/subordination connections.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *Operating Systems Design and Implementation*, volume 16, pages 265–283.

Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). VQA: Visual Question Answering. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2425–2433.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223.

Armitage, L. H. and Enser, P. G. (1997). Analysis of user need in image archives. *Journal of Information Science*, 23(4):287–299.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. *arXiv preprint arXiv:1607.06450*.

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

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Ttransactions on Neural Networks*, 5(2):157–166.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Bird, S. and Loper, E. (2004). NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Carreira, J. and Sminchisescu, C. (2012). CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328.

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

Chen, X. and Zitnick, C. L. (2014). Learning a recurrent visual representation for image caption generation. *arXiv preprint arXiv:1411.5654*.

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

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

CoreNLP, S. (2016). PTBTokenizer. *URL https://stanfordnlp.github.io/CoreNLP (Last accessed: 2019-03-05)*, page 3.

Craswell, N. (2009). Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.

Dai, B., Lin, D., Urtasun, R., and Fidler, S. (2017). Towards diverse and natural image descriptions via a conditional GAN. *arXiv preprint arXiv:1703.06029*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017a). Visual Dialog. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Das, A., Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017b). Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the 9th Workshop on Statistical Machine Translation*, pages 376–380.

Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494.

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

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., Zweig, G., and Mitchell, M. (2015). Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809*.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

Duch, W. and Jankowski, N. (1999). Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every picture tells a story: Generating sentences from images. In *Proceedings of IEEE European Conference on Computer Vision*, pages 15–29.

Ferraro, F., Mostafazadeh, N., Vanderwende, L., Devlin, J., Galley, M., Mitchell, M., et al. (2015). A survey of current datasets for vision and language research. *arXiv preprint arXiv:1506.06833*.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. (2016). Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*, volume 1. MIT Press Cambridge.

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

Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., and Saenko, K. (2013). YouTube2text: Recognizing and describing arbitrary

activities using semantic hierarchies and zero-shot recognition. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2712–2719.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

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

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

Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 328–339. Association for Computational Linguistics.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

Hudson, D. A. and Manning, C. D. (2018). Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*.

Karpathy, A. (2016). *Connecting Images and Natural Language*. PhD thesis, Stanford University.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Ketkar, N. (2017). Introduction to PyTorch. In *Deep Learning with Python*, pages 195–208. Springer.

Kilickaya, M., Erdem, A., Ikizler-Cinbis, N., and Erdem, E. (2016). Re-evaluating automatic metrics for image captioning. *arXiv preprint arXiv:1612.07600*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014). Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 595–603.

Kouloumpis, E., Wilson, T., and Moore, J. D. (2011). Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the International AAAI Conference on Web and Social Media*, pages 538–541.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2013). Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903.

Kuznetsova, P., Ordonez, V., Berg, A. C., Berg, T. L., and Choi, Y. (2012). Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 359–368. Association for Computational Linguistics.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521(7553):436–444.

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

Li, J., Monroe, W., Shi, T., Ritter, A., and Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, Barcelona, Spain.

Lin, D., Fidler, S., Kong, C., and Urtasun, R. (2014). Visual semantic search: Retrieving videos via complex textual queries. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2657–2664.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016). Optimization of image description metrics using policy gradient methods. *arXiv preprint arXiv:1612.00370*.

Liu, Y., Zhang, D., Lu, G., and Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 375–383.

Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

Ma, M., Zhao, K., Huang, L., Xiang, B., and Zhou, B. (2017). Jointly trained sequential labeling and classification by sparse attention neural networks. *arXiv preprint arXiv:1709.10191*.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Mitchell, M., Han, X., Dodge, J., Mensch, A., Goyal, A., Berg, A., Yamaguchi, K., Berg, T., Stratos, K., and Daumé III, H. (2012). Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 747–756. Association for Computational Linguistics.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Olah, C. and Carter, S. (2016). Attention and augmented recurrent neural networks. *Distill*, 1(9):e1.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

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

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Redmon, J. and Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

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

Rohrbach, M., Qiu, W., Titov, I., Thater, S., Pinkal, M., and Schiele, B. (2013). Translating video content to natural language descriptions. In *Proceedings of IEEE International Conference on Computer Vision*, pages 433–440.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Socher, R. and Fei-Fei, L. (2010). Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 966–973.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 207–218. Association for Computational Linguistics.

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 129–136.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Srivastava, N. and Salakhutdinov, R. R. (2012). Multimodal learning with deep Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2222–2230.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1017–1024.

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

Sutton, C., McCallum, A., et al. (2012). An introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

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

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 783–792.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

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

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., and Salakhutdinov, R. R. (2016b). On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057.

Yang, J., Lu, J., Batra, D., and Parikh, D. (2017). A faster PyTorch implementation of Faster R-CNN. *https://github.com/jwyang/faster-rcnn.pytorch*.

Yang, Z., Yuan, Y., Wu, Y., Cohen, W. W., and Salakhutdinov, R. R. (2016). Review networks for caption generation. In *Advances in Neural Information Processing Systems*, pages 2361–2369.

You, Q., Luo, J., Jin, H., and Yang, J. (2015). Robust image sentiment analysis using progressively trained and domain transferred deep networks. In *Proceeding of the 29th AAAI Conference on Artificial Intelligence*, pages 381–388.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 67–78. Association for Computational Linguistics.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceeding of the 31st AAAI Conference on Artificial Intelligence*, pages 2852–2858.

Zhu, M. (2004). Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2:30.