

Syracuse University

**SURFACE**

---

Dissertations - ALL

SURFACE

---

December 2020

# Learning Semantic Information from Multimodal Data using Deep Neural Networks

Krittaphat Pugdeethosapol  
*Syracuse University*

Follow this and additional works at: <https://surface.syr.edu/etd>



Part of the [Engineering Commons](#)

---

## Recommended Citation

Pugdeethosapol, Krittaphat, "Learning Semantic Information from Multimodal Data using Deep Neural Networks" (2020). *Dissertations - ALL*. 1259.  
<https://surface.syr.edu/etd/1259>

This Dissertation is brought to you for free and open access by the SURFACE at SURFACE. It has been accepted for inclusion in Dissertations - ALL by an authorized administrator of SURFACE. For more information, please contact [surface@syr.edu](mailto:surface@syr.edu).

# Abstract

During the last decades, most collective information has been digitized to form an immense database distributed across the Internet. This can also be referred to as Big data, a collection of data that is vast in volume and still growing with time. Nowadays, we can say that Big data is everywhere. We might not even realize how much it affects our daily life as it is applied in many ways, ranging from online shopping, music streaming, TV streaming, travel and transportation, energy, fighting crime, to health care. Many organizations and companies have been collecting and analyzing large volumes of data to solve domain-specific problems or making business decisions. One of the powerful tools that can be used to extract value from Big data is Deep learning, a type of machine learning algorithm inspired by the structure and function of the human brain called artificial neural networks that learn from large amounts of data. Deep learning has been widely used and applied in many research fields such as natural language processing, IoT applications, and computer vision. In this thesis, we introduce three Deep Neural Networks that used to learn semantic information from different types of data and a design guideline to accelerate Neural Network Layer on a general propose computing platform.

First, we focus on the text type data. We proposed a new feature extraction technique to preprocess the dataset and optimize the original Restricted Boltzmann Machine (RBM) model to generate the more meaningful topic that better represents the given document. Our proposed method can improve the generated topic accuracy by up to 12.99% on Open Movie, Reuters, and 20NewsGroup datasets.

Moving from text to image type data and with additional click locations, we proposed a human in a loop automatic image labeling framework focusing on aerial



images with fewer features for detection. The proposed model consists of two main parts, a prediction model and an adjustment model. The user first provides click locations to the prediction model to generate a bounding box of a specific object. The bounding box is then fine-tuned by the adjustment model for more accurate size and location. A feedback and retrain mechanism is implemented that allows the users to manually adjust the generated bounding box and provide feedback to incrementally train the adjustment network during runtime. This unique online learning feature enables the user to generalize the existing model to target classes not initially presented in the training set, and gradually improves the specificity of the model to those new targets during online learning.

Combining text and image type data, we proposed a Multi-region Attention-assisted Grounding network (MAGNet) framework that utilizes spatial attention networks for image-level visual-textual fusion preserving local (word) and global (phrase) information to refine region proposals with an in-network Region Proposal Network (RPN) and detect single or multiple regions for a phrase query. Our framework is independent of external proposal generation systems and without additional information, it can develop an understanding of the query phrase in relation to the image to achieve respectable results in Flickr30k entities and 12% improvement over the state-of-the-art in ReferIt game. Additionally, our model is capable of grounding multiple regions for a query phrase, which is more suitable for real-life applications.

Although Deep neural networks (DNNs) have become a powerful tool, it is highly expensive in both computational time and storage cost. To optimize and improve the performance of the network while maintaining the accuracy, the block-circulant matrix-based (BCM) algorithm has been introduced. It has been proven to be highly effective when implemented using customized hardware, such as FPGAs. However, its performance suffers on general purpose computing platforms. In certain cases, using the BCM does not improve the total computation time of the networks at all. With

this problem, we proposed a parallel implementation of the BCM layer, and guidelines that generally lead to better implementation practice is provided. The guidelines run across popular implementation language and packages including Python, numpy, intel-numpy, tensorflow, and nGraph.

# LEARNING SEMANTIC INFORMATION FROM MULTIMODAL DATA USING DEEP NEURAL NETWORKS

By

Krittaphat Pugdeethosapol

B.S., King Mongkut's University of Technology Thonburi, 2014

M.S., King Mongkut's University of Technology Thonburi, 2016

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy in Electrical and Computer Engineering

Syracuse University

December 2020

Copyright © 2020 Krittaphat Pugdeethosapol  
All Rights Reserved

# Acknowledgements

First of all, I would like to express my sincere gratitude to my doctoral advisor, Dr. Qinru Qiu, for her endless encouragement, comprehensive guidance, and her support that help me in all the time of my study, research, and my life in general. I could not imagine having a better doctoral advisor.

Second, I would like to thank the rest of my research committee members, Dr. Amit K. Sanyal, Dr. Senem Velipasalar, Dr. Fanxin Kong, Dr. C.Y. Roger Chen and Dr. Garrett Ethan Katz for their encouragement and insightful comments.

Third, I would like thank my fellow lab mates at Syracuse University, Dr. Khadeer Ahmed, Dr. Qiuwen Chen, Dr. Zhe Li, Amar Shrestha, Yilan Li, Ziyi Zhao, Haowen Fang, Chen Luo, Zhao Jin, Mingyang Li, Zaidao Mei and Daniel Patrick Rider. I am very grateful for your help and teamwork.

Finally, I would like to thank my family, Pakinee Ongkanupap, Chansri Pugdeethosapol, Vichai Pugdeethosapol, Pramot Ongkanupap, and Phanthira Pugdeethosapol for their loves. And to my wife Ruedeemart Jessadapattharakul for her full support and encouragement during the hard time.

# Table of Contents

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Outline and Main Contributions . . . . .	1
<b>2 Learning Topics using Semantic Locality</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Contributions . . . . .	7
2.3 Related Work . . . . .	7
2.4 Approach . . . . .	9
2.4.1 Feature Generation: Semantic Word Pair Extraction . . . . .	10
2.4.2 Feature Filtering: Two steps TF-IDF Processing . . . . .	11
2.4.3 Feature Coalescence: K-means Clustering . . . . .	12
2.5 Evaluation . . . . .	13
2.5.1 Experiments Setup . . . . .	13
2.5.2 Metric . . . . .	14
2.5.3 Results . . . . .	15
2.6 Conclusion . . . . .	21

<b>3</b>	<b>Automatic Image Labeling with Click Supervision on Aerial Images</b>	<b>22</b>
3.1	Introduction . . . . .	22
3.2	Contributions . . . . .	23
3.3	Related Works . . . . .	24
3.4	Hybrid Human-Machine Labeling Framework . . . . .	25
3.4.1	Prediction Model . . . . .	26
3.4.2	Adjustment Model . . . . .	30
3.5	Experiments . . . . .	32
3.5.1	Experimental Setup . . . . .	32
3.5.2	Evaluation Metric . . . . .	34
3.5.3	Prediction Model Results . . . . .	34
3.5.4	Adjustment Model Performance . . . . .	38
3.6	Conclusion . . . . .	40
3.7	Acknowledgment and Disclaimer . . . . .	40
<b>4</b>	<b>MAGNet: Multi-Region Attention-Assisted Grounding of Natural Language Queries at Phrase Level</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Contributions . . . . .	43
4.3	Related works . . . . .	44
4.4	Methodology . . . . .	47
4.4.1	Visual features . . . . .	48
4.4.2	Encoder-Decoder Language Model . . . . .	48
4.4.3	Attention model . . . . .	49
4.4.4	Attention-based Region Proposal Network . . . . .	51
4.4.5	Attention-based Region CNN . . . . .	52
4.5	Experiments . . . . .	53
4.5.1	Datasets . . . . .	53

4.5.2	Evaluation metrics . . . . .	54
4.5.3	Training details . . . . .	54
4.5.4	Quantitative Analysis . . . . .	55
4.5.5	Qualitative Analysis . . . . .	57
4.6	Ablation study . . . . .	60
4.7	Conclusion . . . . .	61
4.8	Supplementary Materials . . . . .	62
<b>5</b>	<b>Accelerating Block-circulant Matrix-based Neural Network Layer on a General Purpose Computing Platform: A Design Guideline</b>	<b>66</b>
5.1	Introduction . . . . .	66
5.2	Contributions . . . . .	67
5.3	Related Works . . . . .	68
5.4	Background of block-circulant weight matrix . . . . .	69
5.5	Acceleration for General Purpose Computing Platforms . . . . .	71
5.6	Design Space Exploration of BCM . . . . .	74
5.7	Impact of block size on performance . . . . .	78
5.8	Impact of number of CPU cores on performance . . . . .	82
5.9	Impact of batch size on performance . . . . .	85
5.10	Summary of Design Guidelines . . . . .	86
5.11	Conclusion . . . . .	87
<b>6</b>	<b>Conclusion</b>	<b>89</b>
6.1	Summary . . . . .	89
6.2	Future Research Directions . . . . .	90
6.2.1	Improvement on MAGNet framework . . . . .	90
6.2.2	Accelerating MAGNet framework by applying Structure Pruning to Deep Network . . . . .	91



# List of Figures

2.1	Model Structure . . . . .	9
2.2	Word Pair Extraction . . . . .	11
2.3	OMDb dataset mAP score evaluation . . . . .	18
2.4	Reuters dataset mAP score evaluation . . . . .	19
2.5	20NewsGroup dataset mAP score evaluation . . . . .	20
3.1	Example of aerial images. . . . .	24
3.2	The overall architecture of proposed work. . . . .	26
3.3	Generated user clicks location area. . . . .	34
3.4	An example of post-processing output at each step. . . . .	36
3.5	Examples when applying our prediction model and post-processing technique. . . . .	36
3.6	Examples when applying adjustment model. . . . .	36
3.7	Incremental learning results . . . . .	37
4.1	(a) Visual grounding approaches (b) Block diagram of our model. . .	46
4.2	(a) Encoder-Decoder Language Model. (b) Attention-based Region Proposal Network. (c) Attention-based Region CNN. . . . .	47
4.3	(a) Examples from our approach. (b) Predicted grounding for posi- tional cues. (c) Examples showing multiple region detection and dis- crepancies with ground truth. . . . .	58

4.4	Proposals generated by query “a red shirt” . . . . .	59
4.5	a man . . . . .	62
4.6	a thick-striped orange shirt . . . . .	63
4.7	a black pants . . . . .	63
4.8	his arms . . . . .	63
4.9	Examples of position clue . . . . .	63
4.10	Examples with different attributes . . . . .	64
4.11	Flickr30K examples . . . . .	64
4.12	Refclef examples . . . . .	64
4.13	Visual Genome examples . . . . .	65
5.1	An illustration of the partitioned weight and input matrices in FC layer	70
5.2	An illustration of the block-circulant matrix-based calculation [1]. . .	71
5.3	Total time used with different number of CPU cores. . . . .	72
5.4	An illustration of parallel block-circulant matrix-based algorithm. . .	73
5.5	Example use of Ray in Python implementation [2]. . . . .	74
5.6	Comparing the original and our ray-parallel implementation. . . . .	75
5.7	Total time used with different block size with single core in numpy package. . . . .	78
5.8	Total time used with different block size with single core in intel-numpy package. . . . .	79
5.9	Total time used with different block size with single core in tensorflow package. . . . .	79
5.10	Total time used with different block size with single core in tensor- flow+nGraph package. . . . .	79
5.11	Total time used with different block size with multiple cores in numpy package. . . . .	80

5.12	Total time used with different block size with multiple cores in intel-numpy package. . . . .	81
5.13	Total time used with different block size with multiple cores in tensorflow package. . . . .	81
5.14	Total time used with different block size with multiple cores in tensorflow+nGraph package. . . . .	81
5.15	Total time used with different number of CPU cores in numpy package.	83
5.16	Total time used with different number of CPU cores in intel-numpy package. . . . .	84
5.17	Total time used with different number of CPU cores in tensorflow package. . . . .	84
5.18	Total time used with different number of CPU cores in tensorflow+nGraph package. . . . .	85
5.19	Total computation time. . . . .	86
5.20	The time used per sample. . . . .	87

# List of Tables

2.1	LDA and RBM performance evaluation . . . . .	15
2.2	fixed total feature number word/word pair performance evaluation . .	16
2.3	Different Word Pair Generation Algorithms for OMDb . . . . .	19
3.1	Statistics of Neovision 2 Heli dataset. . . . .	33
3.2	Prediction model results.(a) YOLOv3 (b) Our model . . . . .	35
3.3	Adjustment model results. . . . .	36
3.4	Adjustment model results across classes. . . . .	38
4.1	Visual Grounding results . . . . .	55
4.2	Ablation Study . . . . .	61
4.3	Hit rates (N=200) of region proposal methods . . . . .	61
5.1	Size of inputs, blocks, weights before and after partitioning & FFT .	77

# Chapter 1

## Introduction

### 1.1 Outline and Main Contributions

In recent years, deep learning with the deep neural networks as the core has become the most powerful tools to solve many challenging problems and has been able to deliver impressive results. Who have thought that DeepMind's AlphaGo [3] could defeat some of the best Go players in the board game that has more possible moves than the number of atoms in the entire universe? Deep learning technique has excelled in many other tasks in our daily life situations such as identifying faces, reading handwritten digits and texts, recognizing/translating speeches, playing computer games, or even controlling self-driving cars. Many organizations and companies utilize deep learning to extract value from Big data to solve domain-specific problems or making business decisions. For example, the Amazon Go store uses deep learning to track and estimate the intention of every customer in the store. This is referred to as Just Walk Out technology, [4] which solves 6 core problems, sensor fusion, calibration, person detection, object detection, pose estimation, and activity analysis to provide the experience.

As in today's era, Big data is everywhere. We might not even realize how much it

affects our daily life. Ranging from online shopping, music streaming, TV streaming, travel and transportation, energy, fighting crime, to health care, many types of data has been collected and digitized to form an immense database distributed across the internet. With this data, there are several ways that deep learning models can be applied. For example, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) models can be used to process free text to perform sentiment analysis or text classification and tagging. They can also be used for automatic image caption generation where object is detected from the input image and sentences are generated based on the detected objects. When applied to healthcare applications, the aforementioned deep learning models can be used to help diagnose and treat patients by analyze blood samples, track glucose levels in diabetic patients or even perform tumors detection from medical images.

In this thesis, we present three Deep Neural Networks that are used to learn semantic information from different types of data ranging from text, image, user mouse activities and their combinations. A design guideline to accelerate Neural Network Layer on a general propose computing platform is also presented.

The remainder of the thesis is organized as follows. In chapter 2, deep learning model is applied to process text data. We present new feature extraction techniques to preprocess the dataset and optimize the original Restricted Boltzmann Machine (RBM) model to generate the more meaningful topic that better represents the given document. The method consists of three steps. First, it generates the word/word-pair from every single document. Then, it applies a two-way TF-IDF algorithm to word/word-pair for semantic filtering. Finally, it uses the K-means algorithm to merge the word pairs that have the similar semantic meaning. Then we replaced the original word only RBM model by introducing word pairs. The proposed new topic model is designed which combines two classes of text features as the model input. We demonstrate that a feature selection based on semantically related word pairs

provides richer information than the simple bag-of-words approach and the feature clustering effectively controls the model complexity. Compared to existing feature extraction and topic modeling approach, the proposed model improves the accuracy of the topic prediction by up to 12.99%.

In chapter 3, we present a human in a loop automatic image labeling framework focusing on aerial images without salient features for detection. The proposed model consists of two main parts, a prediction model and an adjustment model. The user first provides click location to the prediction model to generate a bounding box of a specific object. The bounding box is then fine-tuned by the adjustment model for more accurate size and location. A feedback and retrain mechanism is implemented that allows the user to manually adjust the generated bounding box and provide feedback to incrementally train the adjustment network during runtime. This unique online learning feature enables user to generalize existing model to target classes not initially presented in the training set, and gradually improves the specificity of the model to those new targets online. We further improve the feature pyramid in the YOLOv3 model to enhance its detection of small objects. Experimental results show that we can improve the IOU of the prediction by 35.6% in average. A further improvement of up to 45% can be reached after applying the adjustment network. The experimental results also show that we can utilize the feedback from users to incrementally train the model during runtime even with very small samples.

In chapter 4, we present a framework, the Multi-region Attention-assisted Grounding network (MAGNet), that utilizes spatial attention networks for image-level visual-textual fusion. It preserves local (word) and global (phrase) information to refine region proposals with an in-network Region Proposal Network (RPN) and detects single or multiple regions for a phrase query. Our contributions of this work are listed as the following:

- A model for image-level visual-textual fusion and natural language query through

the encoder-decoder language model with spatial attention.

- Spatial Attention representation for the global (i.e. phrase level) understanding alongside the local (i.e. word level) understanding of the query in relation to the input image.
- Attention-assisted proposal generation through in-network RPN trained with the aforementioned attention representation.
- Attention-assisted region detection through Region-CNN enabling single or multiple detections for the given query.

In chapter 5, we present a parallel design of the block-circulant based-matrix algorithm and demonstrated that this new design can achieve better performance than previous version of algorithm. We also provide guidelines on how to select block size, batch size, and number of cores in certain situations in order to achieve optimal performance in the least amount of time. The guidelines run across popular implementation language and packages including Python, numpy, intel-numpy, tensorflow, and nGraph

In Chapter 6, we conclude this thesis with a summarization of the results and discuss the future research directions.



# Chapter 2

## Learning Topics using Semantic Locality

### 2.1 Introduction

During the last decades, most collective information has been digitized to form an immense database distributed across the Internet. Among all, text-based knowledge is dominant because of its vast availability and numerous forms of existence. For example, news, articles, or even Twitter posts are various kinds of text documents. On one hand, it is difficult for human users to locate one's searching target in the sea of countless texts without a well-defined computational model to organize the information. On the other hand, in this big data era, the e-commerce industry takes huge advantages of machine learning techniques to discover customers' preference. For example, notifying a customer of the release of "Star Wars: The Last Jedi" if he/she has ever purchased the tickets for "Star Trek Beyond"; recommending a reader "A Brief History of Time" from Stephen Hawking in case there is a "Relativity: The Special and General Theory" from Albert Einstein in the shopping cart on Amazon. The content based recommendation is achieved by analyzing the theme of the items

extracted from its text description.

Topic modeling is a collection of algorithms that aim to discover and annotate large archives of documents with thematic information[5]. Usually, general topic modeling algorithms do not require any prior annotations or labeling of the document while the abstraction is the output of the algorithms. Topic modeling enables us to convert a collection of large documents into a set of topic vectors. Each entry in this concise representation is a probability of the latent topic distribution. By comparing the topic distributions, we can easily calculate the similarity between two different documents[6]. The availability of many manually categorized online documents, such as Internet Movie Database (IMDb) movie review [7], Wikipedia articles, makes the testing and validation of topic models possible.

Some topic modeling algorithms are highly frequently used in text-mining[8], preference recommendation[9] and computer vision[10]. Many of the traditional topic models focus on latent semantic analysis with unsupervised learning [5]. Latent Semantic Indexing (LSI) [11] applies Singular-Value Decomposition (SVD) [12] to transform the term-document matrix to a lower dimension where semantically similar terms are merged. It can be used to report the semantic distance between two documents, however, it does not explicitly provide the topic information. The Probabilistic Latent Semantic Analysis (PLSA)[13] model uses maximum likelihood estimation to extract latent topics and topic word distribution, while the Latent Dirichlet Allocation (LDA) [14] model performs iterative sampling and characterization to search for the same information. Restricted Boltzmann Machine (RBM) [15] is also a very popular model for the topic modeling. By training a two layer model, the RBM can learn to extract the latent topics in an unsupervised way. Moreover, a lot of Deep Neural Network (DNN) based topic modeling methods have been proposed in recent years[16, 17].

Almost all of the existing works are based on the bag-of-words model, where a document is considered as a collection of words. The semantic information of words

and interaction among objects are assumed to be unknown during the model construction. Such simple representation can be improved by recent research in natural language processing and word embedding. In this work, we will explore the existing knowledge and build a topic model using explicit semantic analysis.

This work studies effective data processing and feature extraction for topic modeling and information retrieval. We investigate how the available semantic knowledge, which can be obtained from language analysis, can assist in the topic modeling.

The rest of the chapter is structured as follows: In Section 2.2, we summarize our contributions of this work. In Section 2.3, we review the existing methods, from which we got the inspirations. This is followed in Section 2.4 by details about our topic models. Section 2.5 describes our experimental steps and analyzes the results. Finally, Section 2.6 concludes this work.

## 2.2 Contributions

Compare to existing published feature extraction and topic modeling approach [14, 18], the proposed word/word pair combined model can improve the *mAP* score up to 10.48% in OMDb dataset, up to 1.11% in Reuters dataset and up to 12.99% in the 20NewsGroup dataset. A new topic model is designed which combines two classes of text features as the model input. We demonstrate that a feature selection based on semantically related word pairs provides richer information than simple bag-of-words approach and the proposed semantic based feature clustering effectively controls the model complexity.

## 2.3 Related Work

Many topic models have been proposed in the past decades. This includes LDA, Latent Semantic Analysis(LSA), word2vec, and RBM, etc. In this section, we will

compare the pros and cons of these topic models for their performance in topic modeling.

LDA was one of the most widely used topic models. LDA introduces sparse Dirichlet prior distributions over document-topic and topic-word distributions, encoding the intuition that documents cover a small number of topics and that topics often use a small number of words [14]. LSA was another topic modeling technique which is frequently used in information retrieval. LSA learned latent topics by performing a matrix decomposition (SVD) on the term-document matrix [18]. In practice, training the LSA model is faster than training the LDA model, but the LDA model is more accurate than the LSA model.

Traditional topic models did not consider the semantic meaning of each word and cannot represent the relationship between different words. Word2vec can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary [19]. During the training, the model generated word-context pairs by applying a sliding window to scan through a text corpus. Then the word2vec model trained word embeddings using word-context pairs by using the continuous bag of words (CBOW) model and the skip-gram model [20]. The generated word vectors can be summed together to form a semantically meaningful combination of both words.

RBM was proposed to extract low-dimensional latent semantic representations from a large collection of documents [15]. The architecture of the RBM is an undirected bipartite graphic, in which word-count vectors are modeled as Softmax input units and the output units are binary units. The Contrastive Divergence learning was used to approximate the gradient. By running the Gibbs sampler, the RBM reconstructed the distribution of units [21]. A deeper structure of neural network, the Deep Belief Network (DBN), was developed based on stacked RBMs.

In this work, we adopt Restricted Boltzmann Machine (RBM) for topic modeling,

and investigate feature selection for this model. Another state-of-the-art model in topic modeling is the LDA model. As mentioned in Section 2.3, LDA is a statically model that is widely used for topic modeling. However, previous research [22] shows that the RBM based topic modeling gives 5.45%~19.94% higher accuracy than the LDA based model. In Section 2.5, we also compare the MAP score of these two when applied to three different datasets. Our results also show that the RBM model has better efficiency and accuracy than the LDA model. Hence, we focus our discussion only for the RBM based topic modeling.

## 2.4 Approach

Our feature selection contains three steps handled by three different modules: feature generation module, feature filtering module and feature coalescence module. The whole structure of our framework as shown in Figure 2.1. Each module will be elaborated in the next.

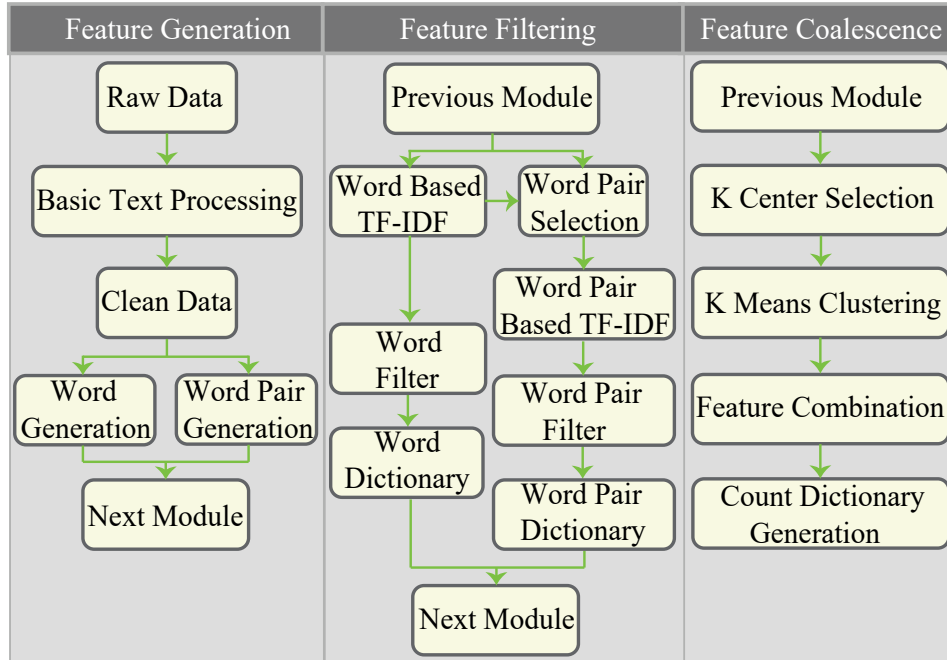


Figure 2.1: Model Structure

The proposed feature selection is based on our observation that word dependencies provide additional semantic information than that simple word counts. However, there are quadratically more depended word pairs relationships than words. To avoid the explosion of feature set, filtering and coalescing must be performed. Overall those three steps perform feature generation, screening and pooling.

### 2.4.1 Feature Generation: Semantic Word Pair Extraction

Current RBM model for topic modeling uses the bag-of-words approach. Each visible neuron represents the number of appearance of a dictionary word. We believe that the order of the words also exhibits rich information, which is not captured by the bag-of-words approach. Our hypothesis is that including word pairs (with specific dependencies) helps to improve topic modeling.

In this work, Stanford natural language parser [23, 24] is used to analyze sentences in both training and testing corpus, and extract word pairs that are semantically dependent. Universal dependency(UD) is used during the extraction. For example, given the sentence: *“Lenny and Amanda have an adopted son Max who turns out to be brilliant.”*, which is part of description of the movie “Mighty Aphrodite” from the OMDb dataset. Figure 2.2 shows all the depended word pairs extracted using the Standford parser. Their order is illustrated by the arrows connection between them, and their relationship is marked beside the arrows. As you can see that the depended words are not necessarily adjacent to each other, however they are semantically related.

Because each single word may have combinations with many other different words during the dependency extraction, the total number of the word pairs will be much larger than the number of word in the training dataset. If we use all depended word pairs extracted from the training corpus, it will significantly increase the size of our dictionary and reduce the performance. To retain enough information with manage-

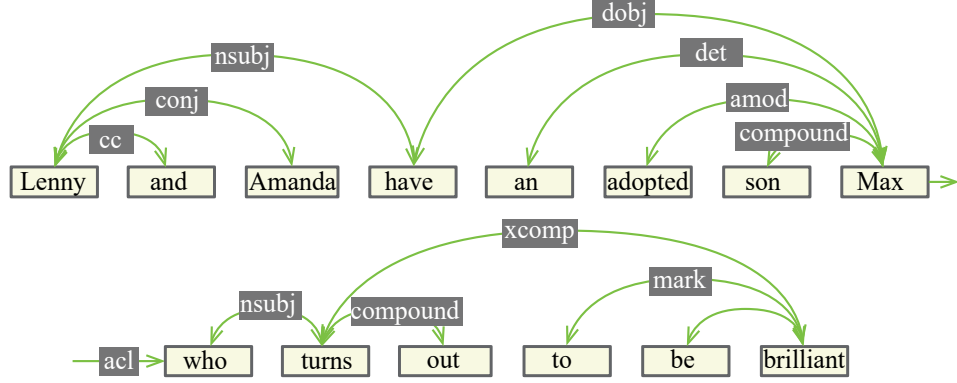


Figure 2.2: Word Pair Extraction

able complexity, we keep the 10,000 most frequent word pairs as the initial word pair dictionary. Input features of the topic model will be selected from this dictionary. Similarly, we use the 10,000 most frequent words to form a word dictionary. For both dictionary, stop words are removed.

#### 2.4.2 Feature Filtering: Two steps TF-IDF Processing

The word dictionary and word pair dictionary still contain a lot of high frequency words that are not very informational, such as "first", "name", etc. *Term frequency-inverse document frequency (TF-IDF)* is applied to screen out those unimportant words or word pairs and keep only important ones. The equation to calculate TF-IDF weight is as following:

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \quad (2.1)$$

$$IDF(t) = \log \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \quad (2.2)$$

$$TF - IDF(t) = TF(t) * IDF(t) \quad (2.3)$$

Equation 2.1 calculates the *Term Frequency (TF)*, which measures how frequently

a term occurs in a document. Equation 2.2 calculates the *Inverse document frequency* (*IDF*), which measures how important a term is. The TF-IDF weight is often used in information retrieval and text mining. It is a statically measure to evaluate how important a word is to a document in a collection of corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus [25, 26, 27, 28, 29].

As shown in Figure 2.1, Feature Filtering module, a two-step TF-IDF processing is adopted. First, the word-level TF-IDF is performed. The result of word level TF-IDF is used as a filter and a word pair is kept only if the TF-IDF scores of both words are higher than the threshold (0.01). After that, we treat each word pair as a single unit, and the TF-IDF algorithm is applied again to the word pairs and further filter out word pairs that are either too common or too rare. Finally, this module will generate the filtered word dictionary and the filtered word pair dictionary.

### 2.4.3 Feature Coalescence: K-means Clustering

Even with the TF-IDF processing, the size of the word pair dictionary is still prohibitively large. We further cluster semantically close word pairs to reduce the dictionary size. Each word is represented by their embedded vectors calculated using Google’s word2vec model. The semantic distance between two words is measured as the Euclidean distance of their embedding vectors. The words that are semantically close to each other are grouped into K clusters.

We use the index of each cluster to replace the words in the word pair. If the cluster ID of two word pairs are the same, then the two word pairs are semantically similar and be merged. In this we can reduce the number of word pairs by more than 63%. We also investigate how the number of the cluster centrum (i.e. the variable K) will affect the model accuracy. The detailed experimental results on three different datasets will be given in 2.5.



## 2.5 Evaluation

### 2.5.1 Experiments Setup

The proposed topic model will be tested in the context of content-based recommendation. Given a query document, the goal is to search the database and find other documents that fall into the category by analyzing their contents. In our experiment, we generate the topic distribution of each document by using RBM model. Then we retrieve the top N documents whose topic is the closet to the query document by calculating their Euclidean distance. The number of hidden units of the RBM is 500 which represents 500 topics. The number of visible units of the RBM equals to total number of different words and words pairs extracted as input features. The weights are updated using a learning rate of 0.01. During the training, momentum, epoch, and weight decay are set to be 0.9, 15, and 0.0002 respectively.

Our proposed method is evaluated on 3 datasets: OMDb, Reuters, and 20News-Group. All the datasets are divided into three subsets: training, validation, and testing. The split ratio is 70:10:20. For each dataset, a 5-fold cross-validation is applied.

- OMDb, the Open Movie Database, is a database of movie information. The OMDb dataset is collected using OMDb APIs [30]. The training dataset contains 6043 movie descriptions; the validation dataset contains 863 movie descriptions and the testing dataset contains 1727 movie descriptions. Based on the genre of the movie, we divided them into 20 categories and tagged them accordingly.
- The Reuters, is a dataset consists of documents appeared on the Reuters newswire in 1987 and were manually classified into 8 categories by personnel from Reuters Ltd. There are 7674 documents in total. The training dataset contains 5485

news, the validation dataset contains 768 news and the testing dataset contains 1535 news.

- The 20NewsGroup dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The training dataset contains 13174 news, the validation dataset contains 1882 news and the testing dataset contains 3765 news. Both Reuters and 20NewsGroup dataset are download from [31].

### 2.5.2 Metric

We use *mean Average Precision* ( $mAP$ ) score to evaluate our proposed method. It is a score to evaluate the information retrieval quality. A higher  $mAP$  score is better. Compare with the traditional  $F1$  score, this evaluation method considers the effect of orders in the information retrieval results. If the relational result is shown in the front position (i.e. ranks higher in the recommendation), the score will be close to 1; if the relational result is shown in the back position (i.e. ranks lower in the recommendation), the score will be close to 0.  $mAP1$ ,  $mAP3$ ,  $mAP5$ , and  $mAP10$  are used to evaluate the retrieval performance. For each document, we retrieve 1, 3, 5, and 10 documents whose topic vectors have the smallest Euclidean distance with that of the query document. The documents are considered as relevant if they share the same class label. Before we calculate the  $mAP$ , we need to calculate the *Average Precision* ( $AveP$ ) for each document first. The equation of  $AveP$  is described below,

$$AveP = \frac{\sum_{k=1}^n (P(k) \cdot rel(k))}{\text{number of relevant documents}}, \quad (2.4)$$

where  $rel(k)$  is an indicator function equaling 1 if the item at rank  $k$  is a relevant document, 0 otherwise [32]. Note that the average is over all relevant documents and the relevant documents not retrieved get a precision score of zero.

The equation of the *mean Average Precision* ( $mAP$ ) score is as following,

$$mAP = \frac{\sum_{q=1}^Q AveP(q)}{Q}, \quad (2.5)$$

where  $Q$  indicates the total number of queries.

### 2.5.3 Results

#### 2.5.3.1 LDA and RBM Performance Comparison

In the first experiment, we investigate the topic modeling performance between LDA and RBM. For the training of the LDA model, the training iteration is 15 and the number of generated topics is 500 which are as the same as the RBM model. As we can see from the Table 2.1. The RBM outperforms the LDA in all datasets. For example, using the  $mAP5$  evaluation, the RBM is 30.22% greater than the LDA in OMDb dataset, 18.18% greater in Reuters dataset and 25.25% greater in 20NewsGroup dataset. To have a fair comparison, the RBM model here is based on word only features. In the next we will show the including word pairs can further improve its  $mAP$  score.

Table 2.1: LDA and RBM performance evaluation

mAP	OMDb		Reuters		20NewsGroup	
	LDA	RBM	LDA	RBM	LDA	RBM
mAP 1	0.12166	<b>0.14772</b>	0.84919	<b>0.94407</b>	0.68669	<b>0.73959</b>
mAP 3	0.07473	<b>0.09381</b>	0.79976	<b>0.92604</b>	0.55410	<b>0.65530</b>
mAP 5	0.05723	<b>0.07453</b>	0.77500	<b>0.91589</b>	0.48796	<b>0.61115</b>
mAP 10	0.03914	<b>0.05273</b>	0.74315	<b>0.90050</b>	0.44719	<b>0.55338</b>

#### 2.5.3.2 Word/Word Pair Performance Comparison

In this experiment, we compare the performance of two RBM models. One of them only considers words as the input feature, while the other has combined words and

Table 2.2: fixed total feature number word/word pair performance evaluation

mAP	F = 10.5K		F = 11K		F = 11.5K		F = 12K		F = 12.5K		F = 15K	
	word	word pair	word	word pair	word	word pair	word	word pair	word	word pair	word	word pair
OMDB												
mAP 1	<b>0.14772</b>	0.14603	0.13281	<b>0.14673</b>	0.13817	<b>0.14789</b>	0.13860	<b>0.14754</b>	0.14019	<b>0.14870</b>	0.13686	<b>0.14708</b>
mAP 3	0.09381	<b>0.09465</b>	0.08606	<b>0.09327</b>	0.08933	<b>0.09507</b>	0.08703	<b>0.09517</b>	0.09054	<b>0.09657</b>	0.09009	<b>0.09537</b>
mAP 5	0.07453	<b>0.07457</b>	0.06835	<b>0.07380</b>	0.07089	<b>0.07508</b>	0.06925	<b>0.07485</b>	0.07117	<b>0.07635</b>	0.07175	<b>0.07511</b>
mAP 10	0.05273	<b>0.05387</b>	0.04862	<b>0.05340</b>	0.04976	<b>0.05389</b>	0.04900	<b>0.05322</b>	0.05019	<b>0.05501</b>	0.05083	<b>0.05388</b>
Reuters												
mAP 1	0.94195	<b>0.95127</b>	0.94277	<b>0.95023</b>	0.94407	<b>0.95179</b>	0.94244	<b>0.94997</b>	0.94277	<b>0.95270</b>	0.94163	<b>0.94984</b>
mAP 3	0.92399	<b>0.93113</b>	0.92448	<b>0.93117</b>	0.92604	<b>0.93276</b>	0.92403	<b>0.93144</b>	0.92249	<b>0.93251</b>	0.92326	<b>0.93353</b>
mAP 5	0.91367	<b>0.92123</b>	0.91366	<b>0.91939</b>	0.91589	<b>0.92221</b>	0.91367	<b>0.92051</b>	0.91310	<b>0.92063</b>	0.91284	<b>0.92219</b>
mAP 10	0.89813	<b>0.90425</b>	0.89849	<b>0.90296</b>	0.90050	<b>0.90534</b>	0.89832	<b>0.90556</b>	0.89770	<b>0.90365</b>	0.89698	<b>0.90499</b>
20NewsGroup												
mAP 1	0.73736	<b>0.77129</b>	0.73375	<b>0.76093</b>	0.68720	<b>0.75865</b>	0.73959	<b>0.75846</b>	0.72280	<b>0.76768</b>	0.72695	<b>0.75583</b>
mAP 3	0.65227	<b>0.68905</b>	0.64848	<b>0.68042</b>	0.60356	<b>0.67546</b>	0.65530	<b>0.67320</b>	0.63649	<b>0.68455</b>	0.63951	<b>0.66743</b>
mAP 5	0.60861	<b>0.64620</b>	0.60548	<b>0.63783</b>	0.56304	<b>0.63321</b>	0.61115	<b>0.62964</b>	0.59267	<b>0.64165</b>	0.59447	<b>0.62593</b>
mAP 10	0.55103	<b>0.58992</b>	0.54812	<b>0.58057</b>	0.51188	<b>0.57839</b>	0.55338	<b>0.57157</b>	0.53486	<b>0.58500</b>	0.53749	<b>0.56969</b>

word pairs as the input feature. The total feature size varies from 10500, 11000, 11500, 12000, 12500, 15000. For the word/word pair combined RBM model, the number of word feature is fixed to be 10000, and the number of word pair features is set to meet the requirement of total feature size.

Both models are first applied to the OMDb dataset, and the results are shown in Table 2.2, section 1, the word/word pair combined model almost always performs better than the word-only model. For the *mAP1*, the *mAP5* and the *mAP10*, the most significant improvement occurs when total feature size is set to = 11000. About 10.48%, 7.97%, and 9.83% improved were found compared to the word-only model. For the *mAP3*, the most significant improvement occurs when the total feature size is set to = 12000, and about 9.35% improvement is achieved by considering word pair.

The two models are further applied to the Reuters dataset, and the results are shown in Table 2.2, section 2. Again, the word/word pair combined model outperforms the word-only model almost all the time. For the *mAP1, 3, 5 and 10* up to 1.05%, 1.11%, 1.02% and 0.89% improvement are achieved.

The results for 20NewsGroup dataset are shown in Table 2.2, section 3. Similar to previous two datasets, all the results from word/word pair combined model are better than the word-only model. For the *mAP1, 3, 5 and 10*, the most significant improvement occurs when the total feature size is set to = 11500. Up to 10.40%,

11.91%, 12.46% and 12.99% improvements can be achieved.

### 2.5.3.3 Cluster Centrum Selection

In the third experiment, we focus on how the different  $K$  values affect the effectiveness of the generated word pairs in terms of their ability of topic modeling. The potential  $K$  values are 100, 300, 500, 800 and 1000. Then we compare the  $mAP$  between our model and the baseline model, which consists of word only input features.

The OMDb dataset results are shown in Figure 2.3. As we can observe, all the  $K$  values give us better performance than the baseline. The most significant improvement occurs when in  $K = 100$ . Regardless of the size of word pair features, in average we can achieve 2.41%, 2.15%, 1.46% and 4.46% improvements in  $mAP1, 3, 5, \text{ and } 10$  respectively.

The results of Reuters dataset are shown in Figure 2.4. When the  $K$  value is greater than 500, all  $mAP$  scores for word/word pair combination model are better than the baseline. Because the  $mAP$  score for Reuters dataset in original model is already very high (almost all of them are higher than 0.9), compared to OMDb, it is more difficult to further improve the  $mAP$  score of this dataset. For the  $mAP1$ , disregard the impact of input feature size, in average the most significant improvement happens when  $K = 500$ , which is 0.31%. For the  $mAP3$ , the  $mAP5$  and the  $mAP10$ , the most significant improvements happen when  $K = 800$ , which are 0.50%, 0.38% and 0.42% respectively.

The results for 20NewsGroup dataset results are shown in Figure 2.5. Similar to the Reuters dataset, when the  $K$  value is greater than 800, all  $mAP$  scores for word/word pair combination model are better than the baseline. For the  $mAP1, 3, 5, \text{ and } 10$ , in average the most significant improvements are 2.82%, 2.90%, 3.2% and 3.33% respectively, and they all happen when  $K = 1000$ .

In summary, a larger  $K$  value generally gives a better result, like the Reuters

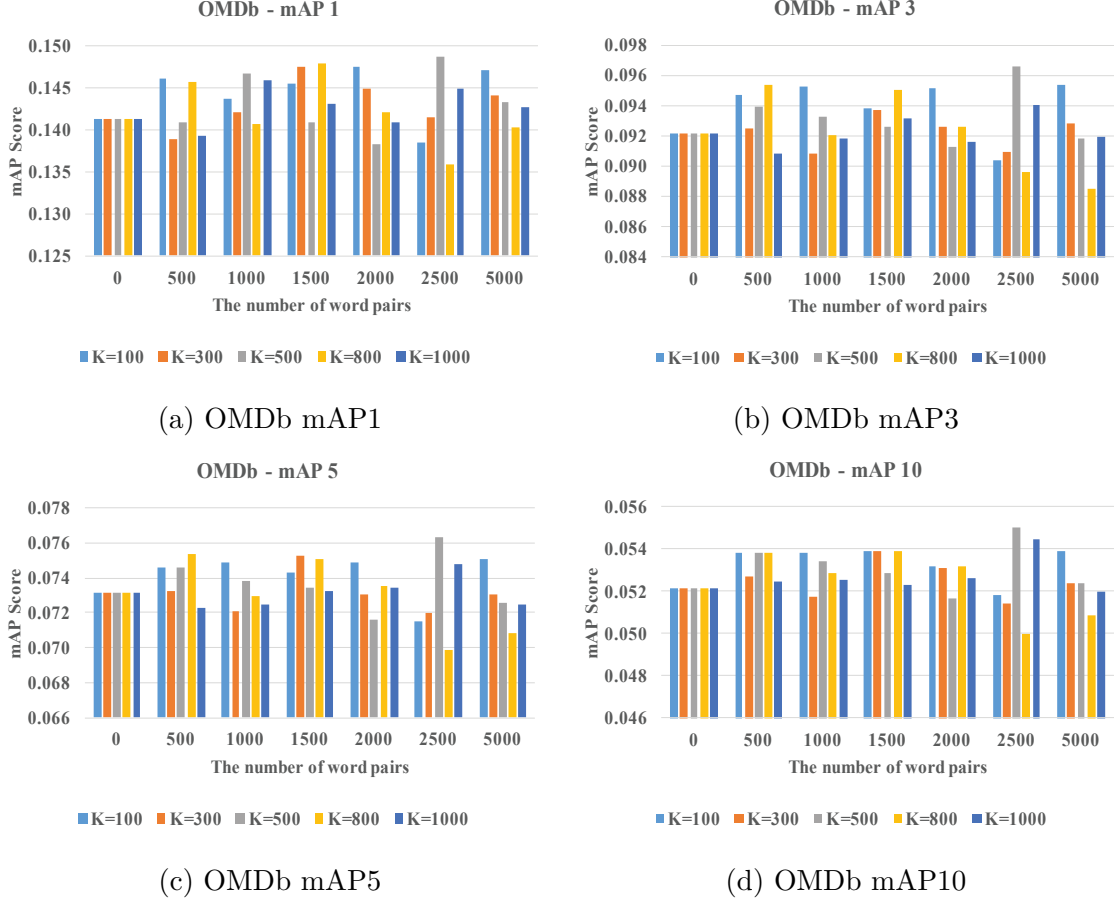


Figure 2.3: OMDb dataset mAP score evaluation

dataset and the 20NewsGroup dataset. However, for some documents sets, such as OMDb, where the vocabulary semantically has a wide distribution, keeping the number of clusters small will not lose too much information.

#### 2.5.3.4 Word Pair Generation Performance

In the last experiment, we compare different word pair generation algorithms with the baseline. Similar to previous experiments, the baseline is the word-only RBM model whose input consists of the 10000 most frequent words. The “semantic” word pair generation is the method we proposed in this work. The proposed technique is compared to a reference approach that applies the idea from the skip-gram [20] algorithm, and generates the word pairs from each word’s adjacent neighbor. We

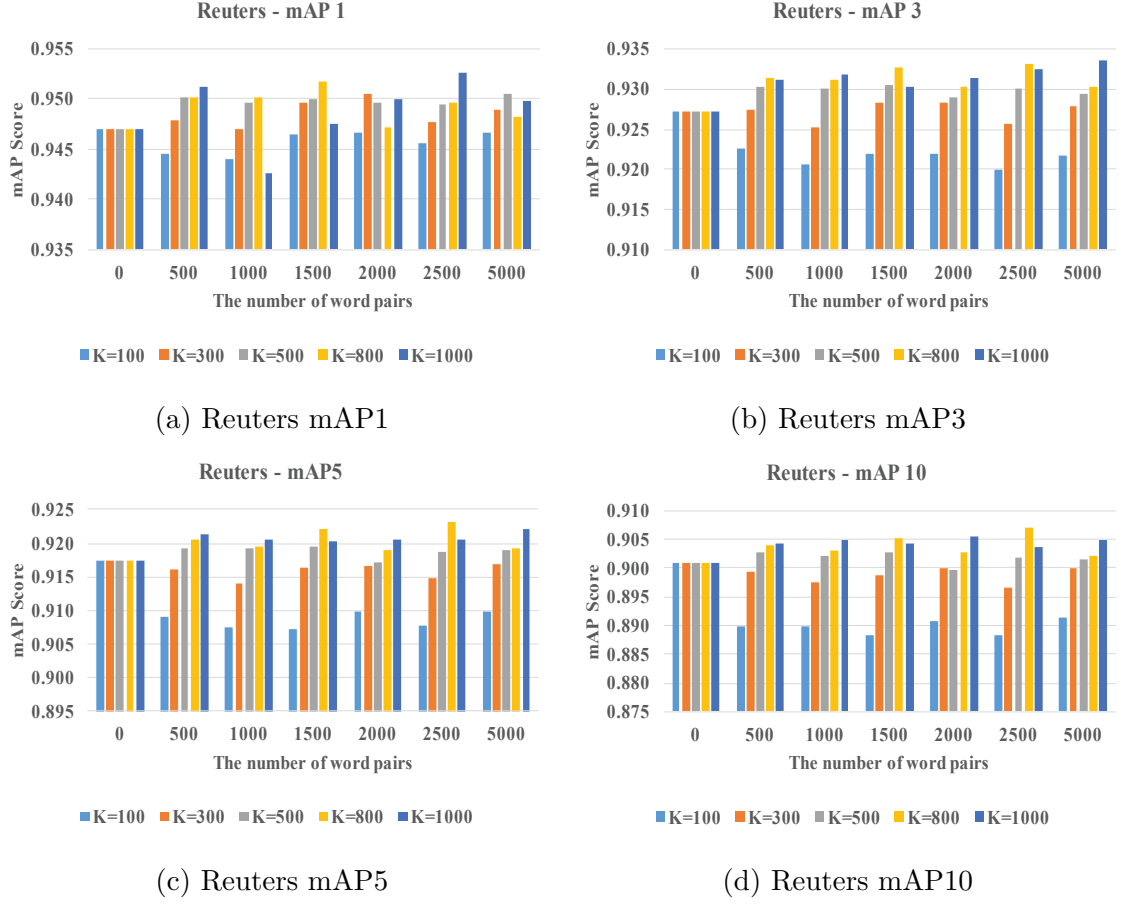


Figure 2.4: Reuters dataset mAP score evaluation

call it “N-gram” word pair generation. And the window size used in here is  $N = 2$ . For the “Non-K” word pair generation, we use the same algorithm as the “semantic” except that no K-means clustering is applied to the generated word pairs.

Table 2.3: Different Word Pair Generation Algorithms for OMDb

mAP	Baseline	Semantic	N-gram	Non-K
mAP 1	0.14134	<b>0.14870</b>	0.13202	0.14302
mAP 3	0.09212	<b>0.09657</b>	0.08801	0.09406
mAP 5	0.07312	<b>0.07635</b>	0.07111	0.07575
mAP 10	0.05113	<b>0.05501</b>	0.05132	0.05585

The first thing we observe from the Table 2.3 is that both “semantic” word pair generation and “Non-K” word pair generation give us better *mAP* score than the baseline; however, the *mAP* score of the “semantic” generation is slightly higher than

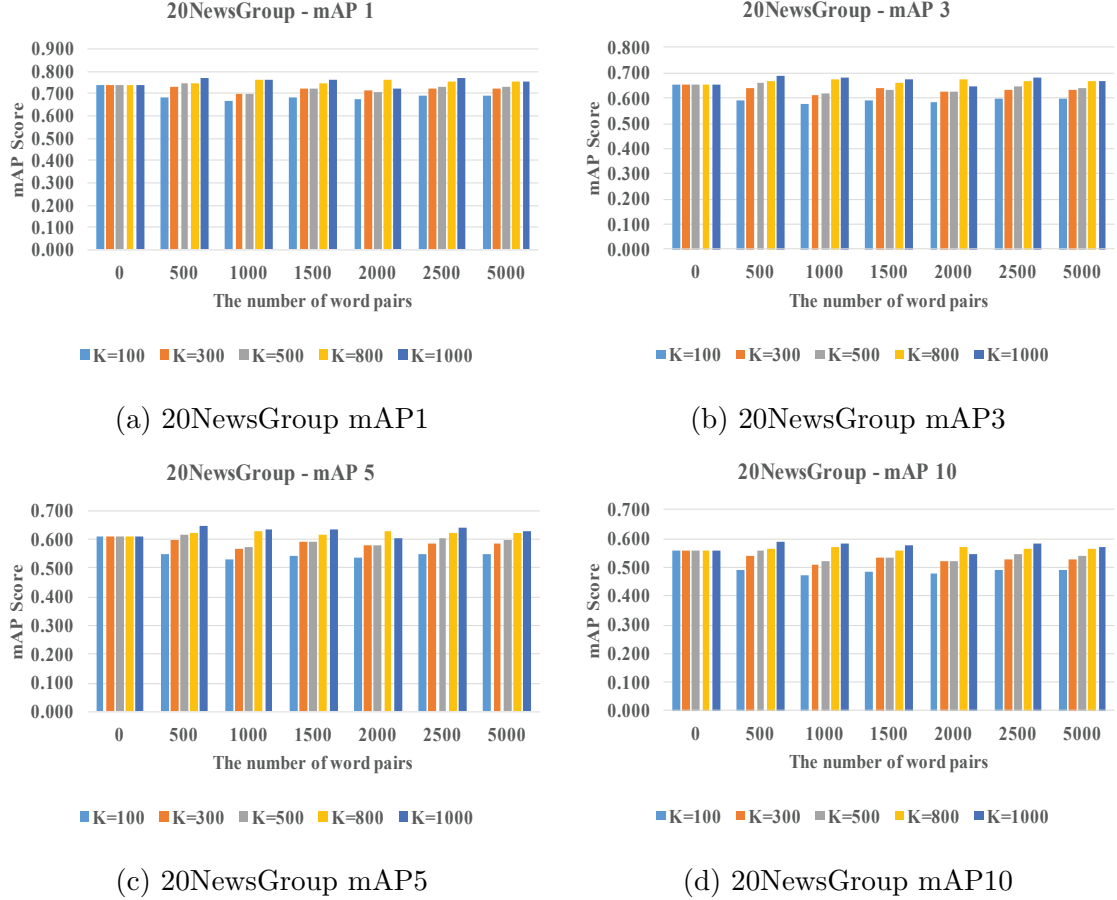


Figure 2.5: 20NewsGroup dataset mAP score evaluation

the “Non-K” generation. This is because, although both “Non-K” and “semantic” techniques extract word pairs using natural language processing, without the K-means clustering, semantically similar pairs will be considered separately. Hence there will be lots of redundancies in the input space. This will either increase the size of the input space, or, in order to control the input size, reduce the amount of information captured by the input set. The K-means clustering performs the function of compression and feature extraction.

The second thing that we observe is that, for the “N-gram” word pair generation, its *mAP* score is even lower than the baseline. Beside the OMDb dataset, other two datasets show the same pattern. This is because the “semantic” model extracts word pairs from natural language processing, therefore those word pairs have the



semantic meanings and grammatical dependencies. However, the “N-gram” word pair generation simply extracts words that are adjacent to each other. When introducing some meaningful word pairs, it also introduces more meaningless word pairs at the same time. These meaningless word pairs act as noises in the input. Hence, including word pairs without semantic importance does not help to improve the model accuracy.

## 2.6 Conclusion

In this chapter, we proposed a few techniques to preprocess the dataset and optimize the original RBM model. During the dataset preprocessing, first, we used a semantic dependency parser to extract the word pairs from each sentence in the text document. Then, by applying a two-way TF-IDF processing, we filtered the data in word level and word pair level. Finally, K-means clustering algorithm helped us merge the similar word pairs and remove the noise from the feature dictionary. We replaced the original word only RBM model by introducing word pairs. At the end, we showed that proper selection of K value and word pair generation techniques can significantly improve the topic prediction accuracy and the document retrieval performance. With our improvement, experimental results have verified that, compared to original word only RBM model, our proposed word/word pair combined model can improve the *mAP* score up to 10.48% in OMDb dataset, up to 1.11% in Reuters dataset and up to 12.99% in the 20NewsGroup dataset.

# Chapter 3

## Automatic Image Labeling with Click Supervision on Aerial Images

### 3.1 Introduction

Object detection has become one of the proliferating research fields in recent years. With the convolution based network structure and novel backpropagation technique [33], models can be trained to directly extract features from images or videos for object detection and classification. Many state-of-the-art models [34, 35, 36, 37, 38, 39] have demonstrated impressive results. Although they have different training techniques and network architectures, what is common is that they all require carefully labeled data for supervised training. Obtaining a set of high-quality labeled data is one of the major challenges for those who have to build and train their own model using supervised learning. Although there are many available public datasets, i.e. [40, 41, 42] training an application specific model requires additional domain specific data.

Creating labeled data for object detection is time consuming. According to [43], annotators take about 35 seconds to draw and annotate a bounding box in the ILSVRC dataset. Do this repeatedly for every object in the training image is a tedious

task for human annotator and impairs their productivity. On the other hand, human cognition is necessary for robust detection, especially in the cases where salient image feature is missing. Examples of such cases are given in Fig 3.1. Conventional CNN based object detection model such as YOLOv3 [44] generates bounding boxes only when it has a relatively high confidence that the region is not part of the background. Given the aerial image in Fig 3.1, the confidence level reduces. Human cognition is much more reliable in these cases.

In this work, we present a human-in-loop automatic image labeling model that can be used to improve the annotator’s productivity while maintaining the labeling accuracy. The proposed model mainly focuses on aerial images (top-down view) which has less salient features for detection and classification. It adopts a hybrid approach to automatically generate a bounding box around the object identified by a user click. The rest of the chapter is organized as the following. In section 3.2, we summarize our contributions of this work. In section 3.3, we review the existing methods that has been used to help training object class detectors. This is followed by Section 3.4 that provides details of our framework. Section 3.5 describes our experimental steps and analyzes the results. Finally, Section 3.6 concludes this work.

## 3.2 Contributions

We present a framework that fuse human object detection capability with the machine intelligence in feature extraction and semantic understanding to improve annotators productivity in labeling. We improve the feature pyramid in the YOLOv3 model [44] to enhance the detection of small objects. We also proposed an adjustment network that is attached to the original detection model to dynamically learn how to improve the bounding boxes and adapt to new target classes that are not in the training set. Compare to [44], we can improve IOU in prediction model by 35.6% in average.



Figure 3.1: Example of aerial images.

A further improvement of up to 45% can be reached after applying the adjustment network. The experimental results also show that we can utilize the feedback from users to incrementally train the model during runtime even with very small samples.

### 3.3 Related Works

Human in a loop click annotation method has been used to help training object class detectors in many researches [45, 46, 47, 48, 49].

Papadopoulos et al. [48] utilized crowdsourcing framework (Amazon Mechanical Turk) to collect click annotations. Annotators must pass a simple qualification test before they can proceed to the real annotation tasks which are divided into batches of 20 images. During the test, annotators will be asked to click as close as possible to the center of synthetic polygons. The purpose of the test is to ensure that annotators can locate the center of the object regardless of the shape. The click annotations are then used to incorporate into a reference Multiple Instance Learning framework designed for weakly supervised object detection [40] results in improvements of object class detectors. In 2011, Wah et al. [49] proposed a visual recognition system which combined a computer and a human into a single system to identify the class of the objects. The system asked the annotator to provide click location on the specific part of the image along with answering binary questions. The system will predict the most likely class based on the given information. The procedures will continue until the

correct class was presented.

Other than object class detection, user click information has also been used in semantic segmentation [50, 51, 52, 53]. For example, Bearman et al. [53] combining user click information with state-of-the-art Convolutional Neural Network (CNN) for semantic segmentation by adding in training loss function to help infer the boundary of the objects.

Either object class detection or semantic segmentation requires the CNN to incorporate with user click location to complete the tasks. There are many state-of-the-art methods for class object detections. They can be divided into two main categories, one-stage methods [36, 39], and two-stage method [35]. In one-stage method, the model processes and predicts object bounding boxes regularly across the image, while in the two-stage method, the model will generate location proposals and then classify each proposal into one of the object classes or the background. There are trade-offs between these two methods in terms of speed and accuracy, the one-stage method is faster while the two-stage method is more accurate.

In this work, we choose the one-stage method and incorporate it with user click location. More specifically we choose YOLOv3 [44] as our based model due to its speed and state-of-the-art accuracies.

### 3.4 Hybrid Human-Machine Labeling Framework

In this section, we introduce the overall architecture of the proposed framework. Its input is the image and the  $(x, y)$  coordinates of user click, which indicate the location of a target object. Its output is the bounding box of the target object. The framework can be divided into two main parts, prediction model and adjustment model. The prediction model generates the bounding box of the object selected by the user click, while the adjustment model refines the predicted bounding box to better fit the target

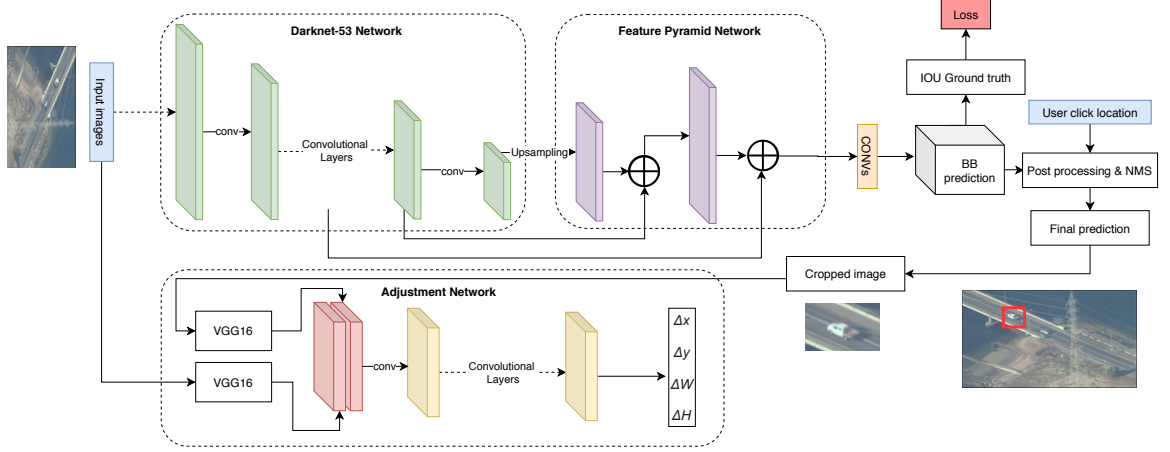


Figure 3.2: The overall architecture of proposed work. There are two main parts, prediction model on the top section and adjustment model on the bottom. There are two types of input to the model which include image and user click location. Image is used to feed into both prediction and adjustment model, while user click location is used for post processing to find the final bounding box prediction. The prediction model predicts bounding boxes based on user click location, while the adjustment model predicts how much the predicted box should adjust to fit more to the target object.

object. The overall architecture is shown in Fig 3.2.

### 3.4.1 Prediction Model

The prediction model can be further divided into three main parts, a backbone network in which Darknet-53 is used, a Feature Pyramid Network [39], and a post-processing module, which is only used during the inference. During the inference, features extracted from different layers in the backbone network will be combined in the feature pyramid network to get the bounding box predictions. The Post-processing module is used to filter and select the final predicted box based on the user click information.

#### 3.4.1.1 Backbone Network

Darknet-53 is used for the backbone network for its high speed and state-of-the-art performance. It is a hybrid approach that combines Darknet-19 [36], residual network,

and skip connection techniques. This results in 53 convolutional layers using only  $1 \times 1$  or  $3 \times 3$  kernels. The residual block replicates itself  $1\times, 2\times, 4\times$ , and  $8\times$  times at each part of the network. The output size of the residual block is always the same as the input because of padding.

#### **3.4.1.2 Feature Pyramid Network**

We apply the feature pyramid network on top of Darknet-53. The last three feature maps of residual block are used as the inputs of feature pyramid network. Each feature map is up-sampled by  $2\times$  to match the size of the output of earlier layers. We use fusion to combine the up-sampled high level features with the low level features coming from the darknet. With this technique, the combined features allow us to extract information at higher resolution, and help to locate and detect the object with different scales and different feature representations. While YOLOv3 predicts boxes at three different scales and using concatenation when merging. Our unique feature pyramid network can identify small objects in the image more efficiently.

#### **3.4.1.3 Post-processing**

Post-processing module is used during inference. The goal of this module is to find the best bounding box prediction based on user click information. In traditional object detection methods, a threshold of the detection confidence must be set. If the threshold is too high, the model may not predict any boxes, and if the threshold is too low, the model will generate multiple boxes at the same location which sometimes makes it difficult to select the best one. The aerial images that we target at have low resolution and most of the bounding boxes will be filtered out due to a low confidence. However, a user click will significantly boost the detection confidence of the bounding boxes located at the clicked area. Instead of increasing the confidence of those bounding boxes, we lower the confidence threshold to keep more detected

boxes, then use the post-processing algorithm to filter out the boxes not adjacent to the click location.

The input of the post-processing is all bounding boxes generated by the backbone and feature pyramid network, albeit with low confidence. Overlapped boxes have been filtered using Non-Maximum Suppression (NMS). With the assumption that there is exactly one object at the click location and the user already knew the class of the object; the post-processing module filters the input bounding boxes and selects the most fit one. The algorithm is shown below:

---

**Algorithm 1** Post-processing steps

---

**Input**

Bounding boxes generated from model:  $B = B_{(x,y,w,h,c,s)}^i$  where  $i = 0, 1, 2, \dots, n$

User click location and predefined class:  $P = (p_x, p_y, p_c)$

**Output**

Final predicted box at user click location:  $F = (f_x, f_y, f_w, f_h)$

**List L:** list of boxes that contain user click location

**List M:** list of boxes that contain user click location and match pre-define class

**foreach**  $b \in B$  **do**

    select boxes that contain user click location

$x_{min} = b_x - b_w/2$ ,  $x_{max} = b_x + b_w/2$

$y_{min} = b_y - b_h/2$ ,  $y_{max} = b_y + b_h/2$

**if**  $x_{min} \leq p_x \leq x_{max}$  **and**  $y_{min} \leq p_y \leq y_{max}$  **then**

$L.add(b)$

**foreach**  $l \in L$  **do**

    filter out boxes that don't match pre-define class

**if**  $l_c == p_c$  **then**

$M.add(l)$

**if**  $M == \emptyset$  **then**

$M = L$  (use boxes from other classes)

**for**  $m \in M$  **do**

    find boxes that have a minimum distance to a click location

    calculate Euclidean distance between center of a box and user click location

$d_m = \sqrt{(m_x - p_x)^2 + (m_y - p_y)^2}$

**sort**( $M$  by  $d_m$ ): sort List M by distance from smallest-largest

$M = top_{30}(M)$ : keep 30% smallest distance

$F = max(M)$ : select the box that has the highest score

**return**  $F$

---

First, we filter boxes that do not contain the user click point. Then we filter



boxes whose class information does not match the target class of the user click. In some cases, there will be no boxes left after the filtering. If this happens, we will alternatively select boxes belonging to other classes as an approximation since the model may miss classifying the objects that have similar features, e.g. bus as train, helicopter as an airplane, etc. The possible error led by such approximation can be mitigated by the adjustment network later. In the next step, we calculate the Euclidean distance between the center of boxes and user click location, sort and select the top 30% of remaining boxes that have the shortest distance. Finally, we select the box that has the highest detection score.

#### 3.4.1.4 Training and Inference

Following YOLOv3, we divide an image into  $S \times S$  grids, each grid predicts 9 anchor boxes with different aspect ratios and predefined width and height using k-mean clustering algorithm.

For each box, the model predicts 5 values:  $x$ ,  $y$ , width, height, and confidence score. As a result, the final prediction features are encoded into a  $S \times S \times 45$  tensor, where  $S = \text{image size}/16$  in our model.

We use a pre-trained Darknet-53 network on the MS-COCO dataset [42] as our backbone network. The prediction model is trained using three losses: location of boxes, width and height of boxes, and the confidence score of the box showing the probability of having an object. The loss equations are defined below:

$$L = \alpha * L_{xy} + \beta * L_{wh} + \gamma * L_{conf} \quad (3.1)$$

$$L_{xy} = \sum_{i=0}^{B^+} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \quad (3.2)$$

$$L_{wh} = \sum_{i=0}^{B^+} (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \quad (3.3)$$

$$L_{conf} = - \sum_{i=0}^{B^+} p_i \log(p_i) + \delta * - \sum_{i=0}^{B^-} p_i \log(p_i) \quad (3.4)$$

where  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$  are weighted parameters for each part of the loss. We use a square loss for both location and size of the box. It only considers the set of boxes, denoted as  $B^+$ , which are responsible for the ground truth (i.e. IOU between the predicted box and ground truth more than a threshold). For confidence loss, we use cross entropy loss where  $p_i$  is an object probability calculated using *sigmoid* activation function, It is calculated only for the set of boxes, denoted as  $B^-$ , which are not responsible for the ground truth.

Instead of predicting  $x$ ,  $y$ , *width*, and *height* directly, we predict  $x$  and  $y$  in relative to the location of the grid cell using *sigmoid* activation function and predict width and height using the anchor boxes [36].

During inference, click location can be asked in two ways. [48] ask annotator to click on the center of the object while in [53] ask to click anywhere on a target object. In this work, we choose to ask annotators to click anywhere on the target due to its feasibility in real applications. Some of the images may have small objects, which will be hard for annotators to click exactly at the center. Furthermore, concentrating on clicking at the center will slow down the annotation process and reduce the human productivity.

### 3.4.2 Adjustment Model

The performance of the aforementioned model is highly determined by the training set. If the annotated image deviates from the training image, accurate bounding boxes may not be generated. Given the initial bounding box found by the aforementioned

prediction model at the user click location, the goal of the adjustment model is to dynamically adapt to the testing data and adjust the size and shape of the box to be more precise over the target object. This is achieved by allowing the user to manually adjust the initially predicted bounding box to the correct size, and to collect the difference,  $(\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{W}, \text{ and } \Delta\hat{H})$ , between the predicted and corrected boxes. The difference will be referred to as the ground truth adjustment and will be used to train the adjustment model.

The adjustment model receives two inputs, the input image and the image of the target object cropped based on the predicted bounding box. Both images pass through VGG16 to extract features. The stacked features run through multiple convolution layers before finally used to predict the bounding box adjustment  $\Delta\hat{x}$ ,  $\Delta\hat{y}$ ,  $\Delta\hat{W}$ , and  $\Delta\hat{H}$ . The total loss to train the model is defined as follow:

$$L = L_{\Delta x} + L_{\Delta y} + L_{\Delta W} + L_{\Delta H} \quad (3.5)$$

where smoothed L1 loss is used for each loss:

$$L_{...} = \begin{cases} 0.5d^2 & |d| < 1 \\ |d| - 0.5 & otherwise \end{cases} \quad (3.6)$$

where  $d$  is the difference between the predicted adjustment  $(\Delta x, \Delta y, \Delta W, \text{ and } \Delta H)$  and ground truth adjustment  $(\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{W}, \text{ and } \Delta\hat{H})$ . We maintain an adjustment network for each object class. Compared to Darknet or VGG net, the adjustment model is a relatively small and low-cost and hence is more suitable for online training.

The significance of the adjustment model is beyond refining the size and the location of the predicted bounding boxes. It also enables transfer learning. As we will show in the experimental results, the adjustment model only needs a few samples

in order to adapt. By leveraging this property, we can apply the learned prediction model of one target class to predict the bounding boxes of objects of another target class that has not been included in the training set, as long as the two have similar features, and apply the adjustment model to adapt to the correct bounding boxes. For example, if we have a trained prediction model for trucks, but not for cars. We can still use the existing truck model to generate bounding boxes of cars at user click location. At first, the predicted box will not fit perfectly to the object because all observed features will be interpreted as trucks and be fit into a truck bounding box. Based on the user feedback, the adjustment model will gradually learn to adjust the bounding box specifically for the “cars”.

The adjustment network is especially useful when the labeled data of a specific target class is limited. Training the adjustment model (rather than the prediction model) for this class will be more effective as there will be less overfitting. Furthermore, it will be more difficult to locate an object than adjusting a box that has already existed.

## 3.5 Experiments

In this section, we describe the details of the experiments, including the dataset, the implementation details, and evaluation metrics. The experimental results will be presented for prediction models compared to a state-of-the-art method.

### 3.5.1 Experimental Setup

For prediction model, the pre-trained Darknet-53 on MS-COCO is used as the backbone network. We then fine-tuned the model on Neovision2 Heli dataset [54]. We chose this dataset to mainly focus on aerial images.

The dataset contains 32 video clips for training and 37 video clips for testing.

Table 3.1: Statistics of Neovision 2 Heli dataset.

Class	Number of boxes	
	Train	Test
All	8839	17626
Car	3791	8065
Boat	251	1150
Helicopter	23	39
Person	1196	3867
Container	654	2824
Cyclist	240	426
Plane	2576	792
Tractor	12	130
Truck	96	243
Bus	0	90

The videos were filmed by a helicopter over the Los Angeles area and have 10 classes: car, boat, helicopter, person, container, cyclist, plane, tractor, truck, and bus. The annotation of videos is divided into frames. We only choose the frames and boxes that are not part of the Don’t Care Regions (DCR), not ambiguous, and have confidence score equal to 1.0. With these criteria, there are no ground truth boxes left for “buses”, so we exclude this class during the training. The total number of boxes for each class in training and testing sets are shown in table 3.1.

We compare our model to a state-of-the-art YOLOv3 model [55]. Both models are trained on NVIDIA TitanX (Pascal) and Tensorflow r1.13. In the experiment, we set  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\delta$  to 5.0, 5.0, 1.0, and 0.5 respectively.

During inference, we lower the model threshold at predict phase to 0.01 to get additional bounding boxes before applying post-processing to select the final prediction. The locations of user click were generated using uniform distribution with a specific range of the lowest and highest value. We used scale of 0, 0.25, 0.5, 0.75, and 1.0. At scale 0, we assumed that a user click is located at the center of the ground truth box. At scale 0.25, 0.5, and 0.75, user click locations were randomly selected within a rectangular area whose size is 25%, 50% and 75% of the ground truth bounding box

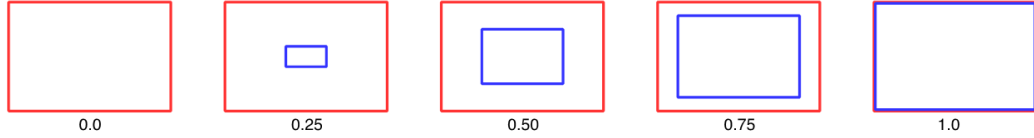


Figure 3.3: Generated user clicks location area.

located at its center. Obviously the larger the rectangular area is, the more possible the user click will deviate from the center. And at scale 1.0, user click locations can be anywhere within the ground truth box. Fig 3.3. shows the area where user click locations are generated at different range.

### 3.5.2 Evaluation Metric

To measure the accuracy of our predicted boxes, we use intersection of union (IOU) as the evaluation metric. If no predicted boxes were generated at user click locations, possibly due to very low detection confidence, the IOU is considered to be 0.

### 3.5.3 Prediction Model Results

We performed experiments on Neovision 2 Heli dataset, where we trained the model while considering all classes and each class individually. The threshold of 0.01 and IOU of 0.5 were used during the inference. We compared our model with a state-of-the-art YOLOv3 in which post-processing technique was applied for both approaches. The randomly generated user click location at scale of 0, 0.25, 0.5, 0.75, and 1.0 were used in the experiments. As mentioned above, “bus” class was excluded in this experiment since there is no labeled training data.

The results are shown in table 3.2. Each row represents an average IOU of corresponding class objects at different user clicks. and average percentage improvement. Compared to the YOLOv3, our models have better IOU in every class and user click scale. This is because our model has a better detection rate with the help of user

Table 3.2: Prediction model results.(a) YOLOv3 (b) Our model

Click location	0			0.25			0.50		
	(a)	(b)	% improvement	(a)	(b)	% improvement	(a)	(b)	% improvement
<b>Class</b>									
All	0.2446	0.3647	49.10	0.2399	0.3615	50.69	0.2028	0.3319	63.66
Car	0.2519	0.3095	22.87	0.2506	0.3071	22.55	0.2203	0.2718	23.38
Boat	0.1298	0.1349	3.93	0.1262	0.1614	27.89	0.1077	0.1208	12.16
Helicopter	0.2442	0.2790	14.25	0.2442	0.2790	14.25	0.2301	0.2790	21.25
Person	0.0804	0.1879	133.71	0.0728	0.1763	142.17	0.0653	0.1297	98.62
Container	0.1359	0.2579	89.77	0.1292	0.2500	93.50	0.1086	0.2167	99.54
Cyclist	0.4216	0.4884	15.84	0.4225	0.4764	12.76	0.3580	0.4638	29.55
Plane	0.3844	0.4172	8.53	0.3785	0.4141	9.41	0.3128	0.3447	10.20
Tractor	0.1720	0.1828	6.28	0.1576	0.1798	14.09	0.1405	0.1537	9.40
Truck	0.1785	0.2965	66.11	0.1708	0.2965	73.59	0.1468	0.2508	70.84
Click location	0.75			1.00			Average		
	(a)	(b)	% improvement	(a)	(b)	% improvement	(a)	(b)	% improvement
<b>Class</b>									
All	0.1320	0.2449	85.53	0.0846	0.1717	10.96	0.1808	0.2949	63.15
Car	0.1491	0.1806	21.13	0.0968	0.1193	23.24	0.1937	0.2377	22.67
Boat	0.0695	0.0861	23.88	0.0410	0.0628	53.17	0.0948	0.1132	19.36
Helicopter	0.1921	0.2194	14.21	0.1113	0.1830	64.42	0.2044	0.2497	21.28
Person	0.0471	0.0770	63.48	0.0334	0.0482	44.31	0.0468	0.1238	164.57
Container	0.0761	0.1439	89.09	0.0538	0.0899	67.10	0.1007	0.1917	90.31
Cyclist	0.2302	0.3295	43.14	0.1580	0.2031	28.54	0.3181	0.3922	23.32
Plane	0.2018	0.2279	12.93	0.1221	0.1559	27.68	0.2799	0.3120	11.45
Tractor	0.1246	0.1247	0.08	0.1081	0.1087	0.56	0.1406	0.1499	6.67
Truck	0.1003	0.1876	87.04	0.0750	0.1302	63.60	0.1343	0.2323	73.01
<b>Average</b>							0.1694	0.2297	35.60

click information. We can also observe that the IOU dropped when the range of the user click expanded. This mean that the user click location plays an important role in the final prediction. The closer the user clicks locate near the object center, the better the predicted box can be.

Additionally, we tested the model on aerial video which is not part of the Neovision2 Heli dataset. Fig 3.4. shows the result when applying post-processing technique to get the final prediction box. From left to right, the first image represents all possible bounding boxes when the threshold has been lowered. The second image shows all boxes that contain user click location while the third image shows the top 30% of all boxes that have a minimum distance from click location. Finally, the last image shows the final prediction box which has the highest score. Fig 3.5. shows additional results at the final prediction.



Figure 3.4: An example of post-processing output at each step.

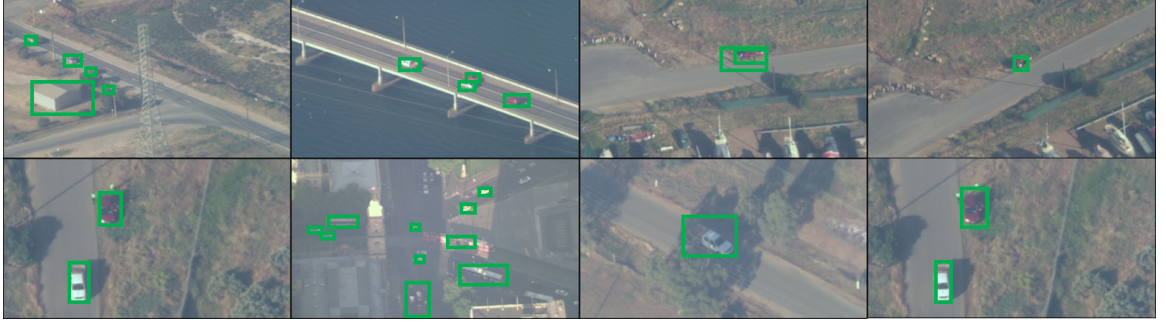


Figure 3.5: Examples when applying our prediction model and post-processing technique.

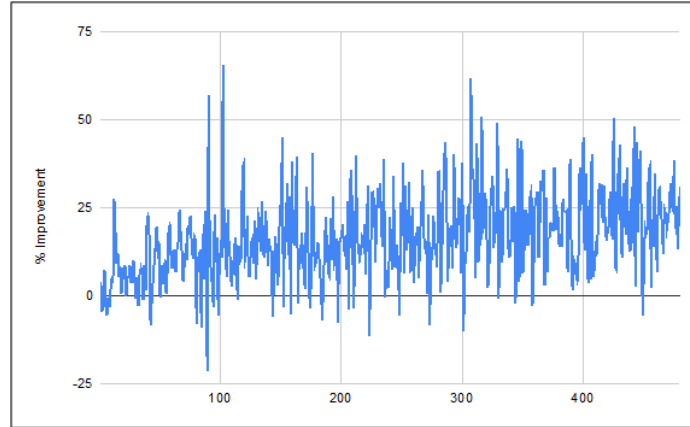


Figure 3.6: Examples when applying adjustment model. Left: before adjustment. Right: after adjustment. Red: ground truth box. Blue: predicted box.

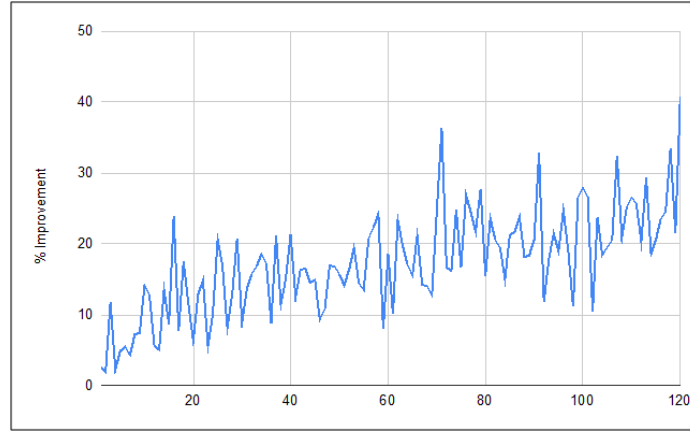
Table 3.3: Adjustment model results.

Training data	Average IOU		Percentage Improvement
	Before	After	
All	0.4416	0.5650	27.94
Car	0.4975	0.6095	22.51
Boat	0.4240	0.7199	69.78
Helicopter	0.3858	0.7419	92.30
Person	0.4649	0.6395	37.55
Container	0.4652	0.6410	37.79
Cyclist	0.5094	0.6382	25.28
Plane	0.4997	0.7252	45.12
Tractor	0.2246	0.3397	51.24
Truck	0.3665	0.5205	42.01
Average	0.42792	0.61404	45.15

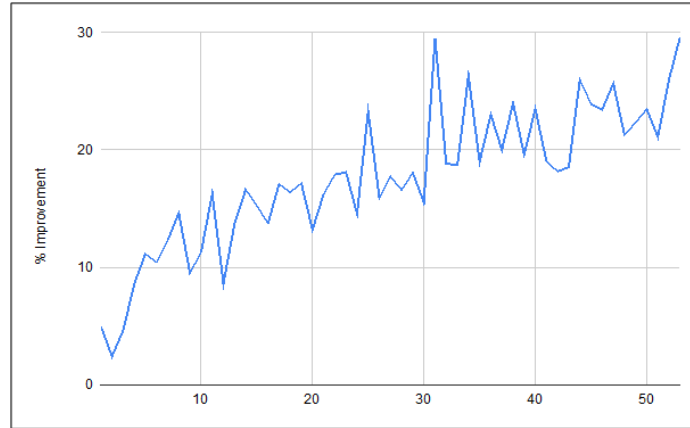




(a)  $N=10$



(b)  $N=40$



(c)  $N=90$

Figure 3.7: Incremental learning results. y-axis represent percent improvement, x-axis represent number of batch where the model has been retrained when new data is available. (a) size 10, (b) size 40, (c) size 90. The size indicates how many new feedback boxes will trigger the model to retrain itself.

Table 3.4: Adjustment model results across classes.

Training Data	Average IOU		Detection rate
	Before	After	
(1) Car	0.4975	0.6095	60.01%
(2) Vehicle except car	0.3337	0.6098	82.00%
(3) All except car	0.3432	0.4580	26.00%
(4) All	0.4290	0.5593	84.80%

### 3.5.4 Adjustment Model Performance

The second experiment evaluates the performance of the adjustment model. In this experiment, we assume the center click. We first apply the prediction model to obtain the initial bounding boxes. The results are then split into 70:30 ratio for training and testing. The adjustment model is trained while considering all classes and each class individually.

The results are shown in Table 3.3. Each row represents the average IOU of corresponding class before and after applying the adjustment model. The IOU is calculated only if predicted boxes were generated at user click locations. It shows that, with adjustment model, IOU can be improved up to 45% in average.

We also evaluated the transfer learning capability of the adjustment network by using the pre-trained prediction model from different classes that have similar features to predict a new class of objects that has not been trained. We trained four prediction models using different training sets that consists of only: (1) cars, (2) all vehicles except the cars, (3) all classes in Table 3.3 except the cars, (4) all classes in Table 3.3. Then we apply the prediction model to predict the bounding box of cars followed by the adjustment model that learns to fine tune the bounding boxes. Table 3.4. shows the prediction results of the four configurations. Each row represents the detection rate of the prediction model, the IOU before adjustment, and the IOU after adjustment.

As we can observe, when the model is trained with data belong to all other classes

except the car (i.e. 3<sup>rd</sup> row in Table 3.4), we got the lower detection rate and IOU before and after the adjustment. This happens because there were classes in the training data that is totally different from cars. Using a prediction model learned from those examples results in lower overall performance.

We can also see that we got promising results if the model is trained based on data from all other vehicle classes except the car (i.e. scenario 2 in Table 3.4). Even though the IOU without adjustment is lower than that of scenario 1 (i.e. prediction model trained using car images), it has a higher detection rate and the IOU can be improved by adjustment. This result confirms that we can adjust the model trained for other classes to predict the bounding box of new target class, if they share some similarities. Fig 3.6. shows some example of bounding boxes before and after apply adjustment model. The same results can also be observed for other classes.

Since the adjustment model is a relatively low-cost network, it can utilize the feedback from users to incrementally train the model during runtime. Fig 3.7. shows how adjustment model improves the IOU over the incremental learning procedure. In this experiment, the target class is the “car” class and the prediction model was trained using the image of cars. We use the ground truth bounding box of the testing samples as the user feedback. The adjustment network is incrementally re-trained every N samples, and N varies from 10 to 40 and 90. The plots show the improvement in IOU after applying the adjustment model. As we can see even with a small batch N=10, the adjustment network adapt to the testing samples after 20 batches and start to deliver around 10% improvements over the original prediction model. For all different N, after about 3,000 samples, the adjust network can about 25% of improvements in average

## 3.6 Conclusion

In this chapter, we proposed a human in a loop automatic image labeling model. The combined prediction, post-processing, and adjustment model can be used to streamline the analyst’s job in annotating images or videos. The model focuses on aerial images, which has less salient features for detection. We demonstrate promising results on Neovision 2 Heli dataset with a comparison to YOLOv3.

Also, by leveraging user click information and the adjustment model, we can improve the overall IOU and extend the framework during runtime to adapt to new classes whose labeled training data is not readily available.

## 3.7 Acknowledgment and Disclaimer

This work was received and approved for public release by AFRL on 04/30/2020, case number 88ABW-2020-1588. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

# Chapter 4

## MAGNet: Multi-Region Attention-Assisted Grounding of Natural Language Queries at Phrase Level

### 4.1 Introduction

Object detection has been the bread and butter of computer vision with the recent advances in deep learning leading to super-human performances in terms of accuracy and speed [56, 57]. A variation of the object detection task is visual grounding where the objective is to detect objects/regions of interests in the image referenced by a descriptive phrase instead of a pre-defined set of classes. The visual grounding task can have various specific objectives: (a) Phrase localization [58, 59]: The language query is a local phrase from a caption describing an image such that an image region linked to the phrase may or may not be independent of the broader context of the full caption. This makes the queries inherently ambiguous. (b) Referring expression

[60, 61, 62, 63, 64]: the query is an expression referring to a particular region of an image. It is less ambiguous. (c) Natural language object retrieval [65, 66]: a query is used to retrieve images from a set of images. (d) Visual question answering [67, 68, 69]: a query is in the form of a question and the image region is the associated answer.

In this work, we will mainly focus on the tasks (a) and (b). Recently, various approaches have been developed to solve the above-mentioned specific tasks. Most state-of-art visual grounding systems have a two-stage framework [59, 62, 63, 70, 71, 72, 73] which rely on an explicit pre-trained object detector to obtain proposed object bounding boxes and rank their ROI-pooled features based on the encoded feature obtained from the query. This essentially limits these systems to a fixed set of object classes that the detector was trained on. One-stage approaches [74, 75, 76, 77] adopt object detection frameworks to generate image features of all possible regions and fuse them with separately encoded features for the query (proposal-level visual-textual fusion) to rank them. Such proposal-level fusion doesn't build an understanding of the whole image in relation to the phrase query. Some datasets [58] also provide annotations in addition to the query such as class, attribute, etc. described by the query and thus are used in various works [63, 75, 78]. This makes them dependent on the information provided by the dataset and not purely based on the natural language query. To reduce the ambiguity in phrase localization, [24] also utilizes the full sentence to describe the image along with the query to develop relationships between multiple queries in the sentence.

Evaluation metrics used to measure the performance also adds bias to some existing works. The conventional *Recall@K* metric essentially expects the predicted region in an image to be ranked in the top K spots. Thus, most works are designed to predict one region per query even if the query might suggest multiple regions in the image irrespective of how the dataset has marked the ground truth.

In this work, to address the mentioned issues, we utilize an encoder-decoder language model with spatial attention for image-level visual-textual fusion of the input image and the natural language query which encodes both the local (word) and global (full query/phrase) understanding of the query in relation to the input image. We utilize this context generated from the attention distribution to train a Faster-RCNN framework [79] such that the proposal generation through in-network Region Proposal Network (RPN) is trained to understand the multi-modal relationship and is not limited to a fixed set of classes, and the Region-CNN network is trained to detect one or multiple regions that can relate to the given query. We depend only on the phrase query - ground truth pair information to make the model independent of the constraints of the datasets i.e. additional attributes, context etc. We call this framework Multi-region Attention-assisted Grounding network (MAGNet).

## 4.2 Contributions

The contributions of this work are listed as follows:

- Image-level visual-textual fusion of the input image and the natural language query through the encoder-decoder language model with spatial attention.
- Spatial Attention distribution representing global (phrase) understanding alongside the local (word) understanding of the query in relation to the input image.
- Attention-assisted proposal generation through in-network RPN trained on the context generated from attention.
- Attention-assisted region detection through Region-CNN trained on the context generated from attention enabling single or multiple detections for a single query.

We evaluate our approach on Flickr30k entities [58], ReferIt game [60] and Visual Genome [80] datasets. Compare to existing work in table 4.1, Our model achieves 12.30% better R@1 performance than the current state-of-the-art one-stage [77] and two-stage [81] approaches. This performance boost can be attributed to the following key points: (1). encoding the image and the query together ensures that the query is understood in relation to the given image. Thus, for independent queries, the generated context vector relates the query closely to the image. This is especially effective for queries with positional cues. Hence, our framework is better at handling the visual oriented language information. (2) the attention-assisted RPN produces better quality proposals than other pre-trained proposal generators. We also achieve state-of-the-art  $R@1$  performance on Visual Genome. However, there aren't many reported visual grounding results on the dataset.

### 4.3 Related works

As we intend to focus on the phrase localization and referring expression tasks in a supervised setting, we compare our work to related works specifically for those tasks. Fig. 4.1a shows the types of approaches.

**Two-stage approach.** The majority of the grounding systems follow a two-stage approach: proposal generation and ranking. Proposal generation is performed either through a pre-trained RPN [70, 81] or Faster-RCNN [63, 78], proposal generation algorithms such as Edgebox [72, 59], Multibox [73], Selective Search [82] or proposal candidates based on all the ground truths in the image [71, 78]. The proposals are then matched with an encoding of the query and then ranked using ranking algorithm or network based on their matching scores. The performance of these two-stage systems relies heavily on the proposal generation. And as the proposal generation mostly focuses on just objects when it's trained as object detectors, the regions unrelated to



objects are often missed. For example, generated proposals may contain “person”, “tree”, “car” etc. but it might not contain “sky to the left of the tree”, “a group of people”, etc.

**One-stage approach.** [74, 75, 76, 77] adopt object detection frameworks such as to SSD [83], YOLO [84], FPN [85] and Retina Net [86] to generate image features of all possible regions and fuse them with separately encoded features for the query to rank them. As an encoding does not capture the entire information, fusing them after encoding might lead to loss of relationships between the two modes (i.e. image and query).

**Additional information for reducing ambiguity.** Various works also utilize additional information other than the image and query-ground truth pair to reduce ambiguity and fine-tune the grounding predictions. Some datasets [58] provide annotations in addition to the query such as class, attribute, etc. described by the query. As some of the two-stage approaches [63, 75], one stage approaches [75, 78] also utilizes the attribute classes to refine the grounding. The help of these attribute is noticeable in the performance but are not available for most visual grounding datasets. In phrase localization task, the image caption is available. This helps reduce the ambiguity of just utilizing the query phrase. [81, 70] utilize the image caption to form relations between the query phrases to improve grounding performance. But, datasets for referring expressions do not have such captions relating query phrases in the image. And most of the works including one-stage approaches [74, 76, 77] also encode the spatial information as an 8-dimensional feature vector to bias predictions for queries based on their location.

**One query – one region approach.** The majority of the related works are designed to output only one region for a query. This bias is derived from the current formalization of the visual grounding problem and prevalent use of Recall@K metric to evaluate the performance. This metric essentially expects the predicted region in

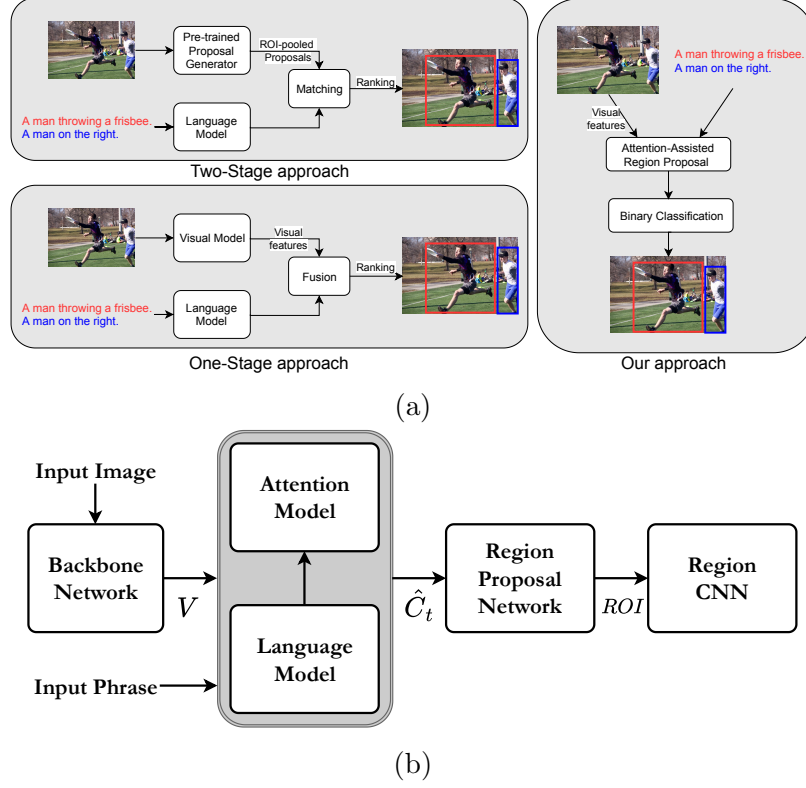


Figure 4.1: (a) Visual grounding approaches (b) Block diagram of our model.

an image to be ranked in the top  $K$  spots. Thus, most work either utilize matching network/algorithms to generate a matching score to produce ranking for proposed regions or simply utilize a softmax over the proposed regions. These systems are thus unable to localize a query to multiple regions in the image even though multiple objects matching the query exists.

**Our approach.** In this work, we intend to enable visual grounding for single or multiple regions in an image based on natural language query without the use of any additional information other than the image and query-ground truths pairs and any pre-trained proposal generation systems. The approach and evaluation of the approach are described in the following sections.

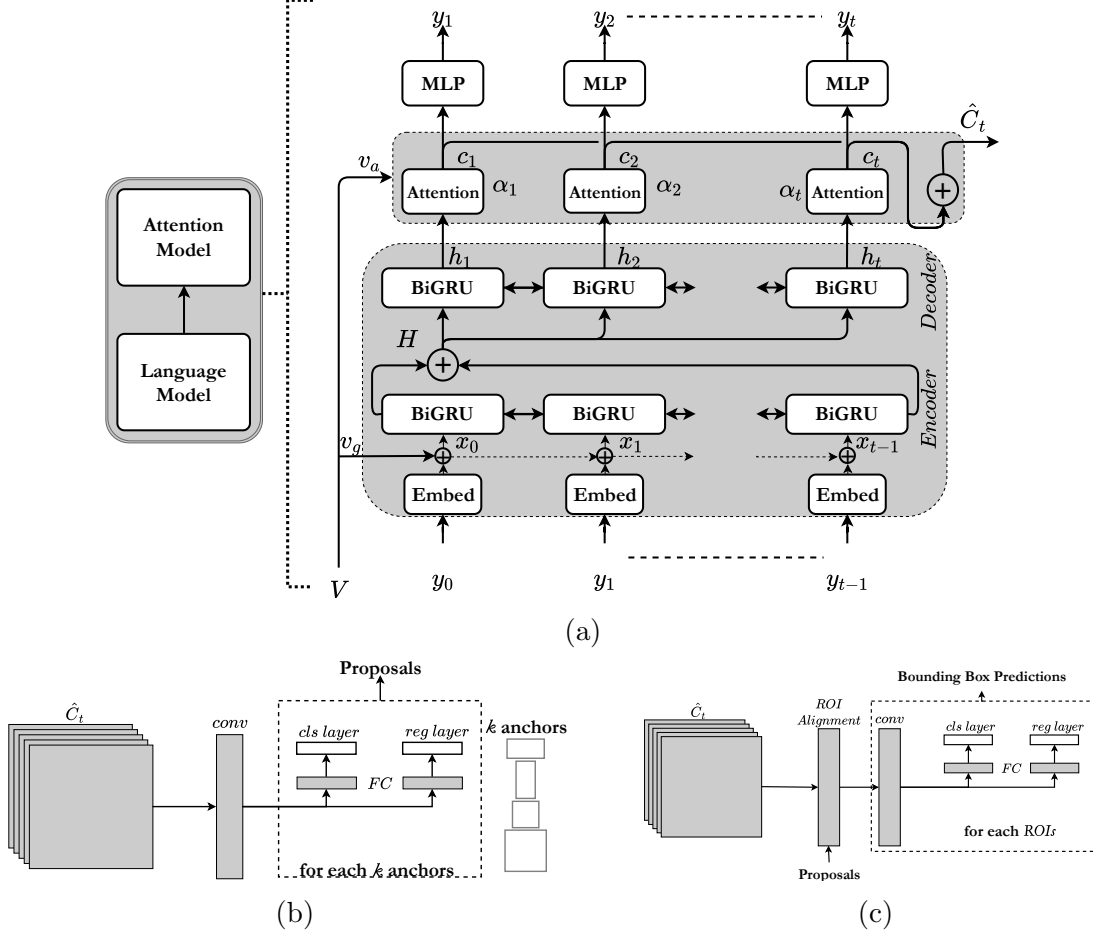


Figure 4.2: (a) Encoder-Decoder Language Model. (b) Attention-based Region Proposal Network. (c) Attention-based Region CNN.

## 4.4 Methodology

In this section, we describe the MAGNet framework. Our approach involves encoding the image and phrase using an Encoder-Decoder framework (Section 4.4.2), identifying regions of interest using a spatial attention model (Section 4.4.3) embracing both local and global information and integrating the attentions into a region proposal network (Section 4.4.4) and region-CNN (Section 4.4.5). In the following sections, we introduce our model. In Section 4.6 we perform an ablation study. The block diagram of the overall framework is shown in Fig. 4.1b.

#### 4.4.1 Visual features

For our model, we use ResNet-50 [87] as the backbone network to extract visual features of the input image. The input image is resized and padded with zeros to get a square image of size  $512 \times 512$ . The visual features are extracted from the  $C4$  layer with size  $32 \times 32 \times D_c$  such that the feature map is  $1/16$  of the input image and  $D_c$  is the number of feature maps. The choice of  $C5$  ( $16 \times 16$ ) and  $C4$  ( $32 \times 32$ ) makes minimal difference in the performance.  $D_c$  varies with the choice of the backbone and the input image size. So, to standardize the model, we add a  $1 \times 1$  convolutional layer with a ReLU and  $D_e = 512$  filters to produce the final visual features  $V$ . This visual feature is further encoded separately for the language model and the spatial attention model.

#### 4.4.2 Encoder-Decoder Language Model

To encode the image and the phrase together, we first adopt the encoder-decoder framework [88] and modify it to encode the image and corresponding phrase together. Fig. 4.2a shows the encoder-decoder model we utilize.

In this work, for our model, we adopt Gated Recurrent Units (GRU) instead of Long-Short Term Memory (LSTM) as it has demonstrated state-of-the-art performances with significantly lower number of parameters. Alongside, we adopt a bidirectional version of GRUs to encode the phrases from both front-to-back and back-to-front. The encoder encodes the combination of the image and phrase producing the global embedding  $H$  where it is computed over the entire time steps.

$$H = \text{Encoder}(x_t, eh_{t-1}, m_{t-1}, eh_{t+1}, m_{t+1}) \quad (4.1)$$

$m_t$  is the memory cell vector and  $eh_t$  is the hidden state of the encoder at the time before and after  $t$ .  $x_t$  is the input vector formed from the concatenation of the visual

features from the image and embedding of the words  $y_t$  in the phrase at each time  $t$  given as  $E_t$ . For our model, the visual features from the backbone network  $V$  are embedded using global average pooling (GAP) such that  $v_g = GAP(V)$ . The phrase is embedded using pre-trained 300-dimensional GLOVE embedding vectors [89]  $w_t$  trained on Wikipedia2014 and Gigaword. Such that  $x_t = [E_t, v_g]$

The encoder may consist of multiple recurrent layers, but for our model, we only use a single layer of BiGRU. The decoder also only consists of a single layer of BiGRU such that its hidden state is represented as:

$$h_t = Decoder(H, h_{t-1}, m_{t-1}, h_{t+1}, m_{t+1}) \quad (4.2)$$

Utilizing the context vector  $c_t$  generated from attention distribution to be detailed in Section 4.4.3 and hidden vector  $h_t$ , the probability distribution of  $y_t$  over the word vocabulary is generated as:

$$p(y_t|I) = f(h_t, c_t) \quad (4.3)$$

### 4.4.3 Attention model

In the attention-based frameworks such as [90] at time  $t$  based on the hidden state, the decoder would focus on the specific regions of the image with a distribution  $a_t$  and compute  $c_t$  using the spatial image features from visual backbone network. Such that the context vector  $c_t$  is defined as:

$$c_t = g(v_a, h_t) \quad (4.4)$$

where  $g$  is the attention function that will be given later by equation 4.8, and  $v_a = Conv_{1 \times 1}(V)$  with number of filters  $D_a$  to match dimensions with  $h_t$ .  $v_a \in \mathbb{R}^{D_a \times D_f}$  where  $D_f = w_f \times h_f$  is the number of pixels in a single visual feature map. For a

$512 \times 512$  input image,  $D_f$  is  $32 \times 32$  and each pixel in the feature map corresponds to a  $16 \times 16$  region in the input image.

$c_t$  in equation 4.4 captures the region of focus in the visual features pertaining to the current word  $t$  in the phrase. In localizing the phrase, it is important to preserve the global information of the whole phrase when focusing on a region.  $H$  encodes the entire phrase in the encoder layer such that it is a viable candidate to generate the context vector. The importance of having  $H$  will be shown in Section 4.6. With the global information  $H$ , the context vector  $c_t$  can be derived by the following:

$$c_t = g(v_a, h_t, H) \quad (4.5)$$

where  $H \in \mathbb{R}^{D_a \times 1}$  and  $h_t \in \mathbb{R}^{D_a \times 1}$ . Given  $v_a$ ,  $h_t$  and  $H$ , we apply a simple neural network and a softmax function to generate the attention distribution  $\alpha_t$  over the spatial image features at time  $t$ :

$$z_t = W_z^T \tanh(W_v v_a + (W_h h_t) \mathbb{1}^T + (W_H H) \mathbb{1}^T) \quad (4.6)$$

$$\alpha_t = \text{softmax}(z_t) \quad (4.7)$$

where  $W_v, W_h, W_H \in \mathbb{R}^{D_f \times D_a}$ , and  $W_z \in \mathbb{R}^{D_f \times 1}$  are weight coefficients learned from the training process, and  $\mathbb{1} \in \mathbb{R}^{D_f \times 1}$  such that  $\alpha_t \in \mathbb{R}^{1 \times D_f}$ . The context vector  $c_t$  at time  $t$  can now be obtained as:

$$c_t = v_a(W_H H) \alpha_t \quad (4.8)$$

Such that we model the probability distribution over  $y_t$  in equation 4.3 as

$$p(y_t|I) = f(h_t, c_t) = \text{softmax}(W_p(c_t + h_t \mathbb{1}^T)) \quad (4.9)$$

where  $W_p$  is learned weight matrix. The log probability distribution  $y_t$  is maximized with a cross-entropy loss. Applying this loss as an auxiliary loss enables training the attention vector without any grounding supervision. This infers the possibility for pre-training the attention model. More specifically, the attention model in Section 4.4.3 can be trained separately with just the auxiliary loss in equation 4.9 to predict the next words of the query, similar to seq-to-seq language models. In doing so, the attention model can focus its gaze to the relevant area of the image, i.e. learn the soft attention ( $\alpha_t$ ) and thus the context vector ( $c_t$ ) without the grounding supervision, i.e. the ground truth bounding boxes. This can provide a way to pre-train the attention model before adding it into the MAGNet framework where fine-tuning of the attention model can be done along with training the attention-based RPN and Region CNN.

#### 4.4.4 Attention-based Region Proposal Network

Instead of using a pre-trained RPN to generate proposals in conventional two-stage phrase localization works, we intend to train the RPN with assistance from the context ( $c_t$ ) derived from encoding the visual and phrase features together.

The context vector  $c_t$  from equation 4.8 represents the understanding of the word in a phrase at time  $t$  in terms of focus on the image. To utilize this context, we need to combine the context over the entire phrase. In our model, we simply average the context over the time dimension such that the resulting context  $\hat{C}_T$  still has the same dimensions as the original visual feature  $v_a$ .

$$\hat{C}_T = \frac{1}{T} \sum_{t=1}^T c_t \quad (4.10)$$

where  $T$  represents the number of words in the phrase.

We use this average context vector  $\hat{C}_T$  as the input of the RPN. Similar to Faster-

RCNN, our RPN as shown in Fig. 4.2b takes the average context vector as input and outputs a set of rectangular object proposals (*reg* layer), each with an objectness score (*cls* layer). Here, we define "objectness" not literally but based on how the phrases are grounded in the dataset. For example, the phrase "a red shirt" refers to an actual object whereas the phrase "a group of people" might not fit the literal definition of the word "object" but still is taken as such based on the dataset.

We also adopt the same multi-task loss as in [79] to train the *cls* layer (binary classification) and the *reg* layer (regression). In Section 4.6, we demonstrate the efficacy of utilizing the learned context for training the RPN instead of using a pre-trained RPN.

#### 4.4.5 Attention-based Region CNN

Now we utilize the proposals generated by RPN for region-based phrase detection CNN. For the detection network, we again adopt Faster-RCNN as shown in Fig. 4.2c. Again, the proposals are used to perform ROI alignment on the context vector  $\hat{C}_T$ . As we do not have classes as the detection network in Faster-RCNN, we define the task of the *cls* layer in the phrase detection network as detecting how much the proposal represents the given phrase. For this purpose, *cls* layer classifies each proposal as either not related or related to the given phrase using a softmax. This essentially means, instead of ranking these proposals, we detect how related these proposals are to the phrase such that we can detect multiple instances of the phrase in the image. The *reg* layer is now used to regress to the final bounding box for the phrase. After this, we perform a further step of non-maximum suppression to fine-tune the detections.



## 4.5 Experiments

In this section, we present experiments to evaluate our proposed model MAGNet on varieties of datasets with multiple evaluation metrics and compare our results to the state-of-the-art visual grounding methods [58, 59, 65, 70, 72, 74, 75, 77, 81, 82]. Results of ablation studies with different configurations will also be reported to further explain the design decisions of the proposed model.

### 4.5.1 Datasets

We evaluate MAGNet on 3 different datasets Flickr30K entities [58], ReferItGame [60], and Visual Genome [80]. Flickr30K Entities provides region phrase correspondence annotations to the original Flickr30K. The 31,783 images in Flickr30K have 427K referred entities. We follow the same training/test split used in the previous work [77] in our experiments. The queries in Flickr30K are region phrases extracted from a full sentence description of the image. The ground truth image object provided for each query is an object described in the image caption. The contextual information of the image caption imposes extra constraints in visual grounding, such that the dataset ignores other objects in the “background” that also match the query phrase. The MAGNet focuses on the referring expressions itself with no other context information, its training and testing are done solely based on the query phrases. As we will show in this section, it detects more matching objects for the given query. Some of them are not in the ground truth of Flickr30K. ReferItGame has 20,000 images from the SAIAPR-12 dataset [91] and contains 130,525 expressions, 96,654 distinct objects, and 19,894 photographs of natural scenes. The queries are expressions referring to one or more regions in the image. The Visual Genome dataset has a total of 108,077 images with 5.4 million region descriptions.

### 4.5.2 Evaluation metrics

We evaluate the models with two metrics: Recall@K and mean average precision (mAP). Recall@K (R@K) for  $K = 1, 5$  and  $10$  is defined as the proportion of all positive examples ranked above a given rank  $K$ . mAP metric is adapted from the PASCAL VOC challenge [92] used for object detection tasks. The mAP considers both precision and recall of a model and enables evaluation of the model when there are more than one region to be detected for a single query. and is defined as the mean precision at a set of eleven equally spaced recall levels  $[0, 0.1, \dots, 1]$ . Detailed description of the mAP metric can be found in [92]. For both metrics, the predicted bounding box is considered positive if it is classified as related to the given query phrase and the intersection over union (IOU) is 0.5 or more.

### 4.5.3 Training details

We reshape the input image to size  $512 \times 512$  while keeping the original aspect ratio and padding the smallest dimension with zero pixels. No other data augmentation is performed. Query phrases are prepended with a start token and appended with an end token and embedded with the GLOVE 300D embedding [89]. Shorter phrases are padded with pad tokens and are limited to 18 words.

We utilize ResNet-50 [93] trained on ImageNet [94] as the backbone network for visual features. The two layers of BiGRU contains 512 units each. For the RPN and Region-CNN, we use the same architecture and dimensions as the original Faster-RCNN. For the RPN, we use 9 anchors (3 aspect ratios and 3 sizes) for each feature in the context vector. All the modules in the model are trained together to allow the attention distribution to correlate better with the region proposals and the final region predictions. In the experiments, we found that training the RPN and Region-CNN separately hindered the performance of our approach.

Table 4.1: Visual Grounding results

	Methods	Flickr30k Entities			ReferItGame			Visual Genome		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
2-Stage	SCRC[65]	27.80	-	62.90	17.93	-	45.27	11.00	-	-
	DSPE[95]	43.89	64.46	69.66	-	-	-	-	-	-
	GroundedR[82]	47.81	-	-	26.93	-	-	-	-	-
	CCA[58]	50.89	71.09	75.73	-	-	-	-	-	-
	Similarity Net[59]	51.05	70.30	75.04	-	-	-	-	-	-
	MSRC[96]	57.53	-	-	32.31	-	-	-	-	-
	QRN[70]	60.21	-	-	43.57	-	-	-	-	-
	QRC[70]	65.14	-	-	44.07	-	-	-	-	-
	CITE[72]	61.89	-	-	34.13	-	-	24.43	-	-
	PIRC Net[81]	<b>72.83</b>	-	-	59.13	-	-	-	-	-
1-Stage	IGOP[74]	53.97	-	-	34.70	-	-	-	-	-
	SSG[75]	-	-	-	54.24	-	-	-	-	-
	ZSGNet[76]	63.39	-	-	59.63	-	-	-	-	-
	[77]	68.69	-	-	59.30	-	-	-	-	-
	<b>MAGNet(Ours) VGG16</b>	49.85	67.00	67.85	63.55	72.50	72.75	22.35	34.90	36.35
	<b>MAGNet(Ours) Resnet50</b>	60.20	78.85	79.90	<b>71.60</b>	81.00	81.20	<b>28.85</b>	48.50	50.70

#### 4.5.4 Quantitative Analysis

Table 4.1 compares our approach with prior works on Flickr30k entities, ReferIt, and Visual Genome datasets in terms of Recall@K metric where  $K = 1, 5$ , and 10. We separate the prior works into two-stage and one-stage approaches. and compare the results with our model described in Section 4.4 and 4.5.3. Results for the prior works are collected from their respective publications.

For the phrase localization task on Flickr30k entities, the phrase queries extracted from the image caption ignore the context in the original sentence and thus are highly ambiguous especially in terms of the positional cues of the region. Some examples are given in Figure 4.3c with the ground truths. As we explained in Section 4.5.1, MAGNet searches for the matching objects solely based on the query phrase without considering any additional contextual information. Additionally, we modelled MAGNet as a detection framework to detect single or multiple regions for a query instead of just specifically one region. Therefore, it is able to detect all matching objects in the image, and the one mentioned in the image caption may not necessarily have the highest score. That is why the R@5 and R@10 score of MAGNet is significantly better than its R@1 score. From this perspective, the Flickr30k entities is not the

ideal dataset to evaluate our approach, because only the objects within the context of the image captions are identified as the ground truth, while other objects are ignored even though they also match the query description.

The authors of [70] [81] utilize the entire caption to either build relationships between multiple queries in a single image or as the context to reduce the ambiguity in the query phrase. Therefore, they are able to locate the object in the context (i.e. image caption) that matches the phrase description. However, for many applications the caption of the image is usually not available. Furthermore, focusing only on the context given by the image caption prohibits the model to locate all possible matching objects to the query in the image. That is why the performance of these approaches degrade significantly when applied to the ReferItGame dataset.

In addition to focus only on a single region, [76] [77] explicitly code spatial features for each position of the spatial dimensions to add positional information thus reducing positional ambiguity. These works also encode the queries and images separately, thus enabling them to utilize powerful pre-trained language models like BERT [97]. However, separately trained language and image model also means that the model is less effective in extracting language features that have salient image information or vice versa. This probably is another reason that these works perform worse when applied to ReferItGame dataset, where many objects are located based on the position cue in the phrase description.

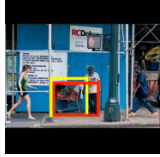

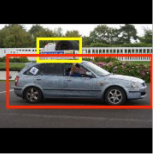
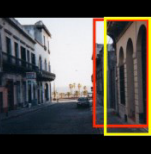
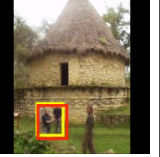
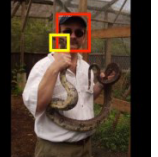
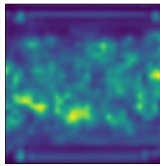
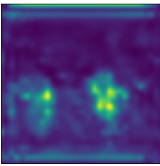
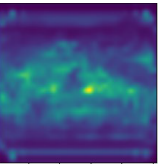
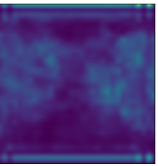
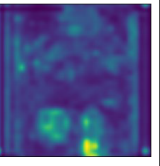
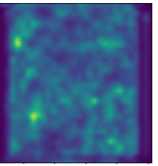
The advantages of our approach become apparent in the referring expression task in ReferIt and Visual Genome datasets as shown Table 4.1. In this task, the queries in the image are independent of each other. The queries are self-sufficient with specific positional cues and thus less ambiguous. Hence, the ground truth has better precision in this dataset. Our model achieves 12.30% better R@1 performance than the current state-of-the-art one-stage [77] and two-stage [81] approaches. This performance boost can be attributed to the following key points: (1). encoding the image and the query

together ensures that the query is understood in relation to the given image. Thus, for independent queries, the generated context vector relates the query closely to the image. This is especially effective for queries with positional cues as shown by predicted grounding in Fig. 4.3b. Hence, our framework is better at handling the visual oriented language information. (2) the attention-assisted RPN produces better quality proposals than other pre-trained proposal generators. Table 4.3 shows the hit rates for various region proposal methods for the number of proposals  $N = 200$ . The MAGNet (a) in the table is the original MAGNet model, however the input of its RPN (Figure 4.2b) is the visual feature  $V$  instead of the attention enhanced visual feature  $\hat{C}_t$ . Our attention-assisted RPN produces the highest quality proposals for ReferIt and Visual Genome.

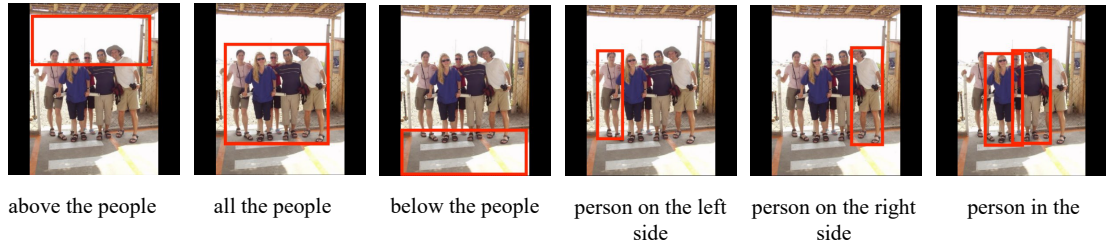
We also achieve state-of-the-art  $R@1$  performance on Visual Genome. However, there aren't many reported visual grounding results on the dataset.

#### 4.5.5 Qualitative Analysis

Fig. 4.3a shows some examples of visual grounding performance of our approach and the attention distribution for Flickr30k entities and ReferIt datasets. For each dataset, the leftmost and middle columns show the visual grounding and attention distribution when the query is grounded correctly, and the rightmost column when the query phrase is grounded incorrectly. The yellow bounding box represents the ground truth and the red bounding box represents the predicted bounding box. As can be seen from the examples, the attention is distributed as suggested by the query focusing on relevant parts as described by the words in the phrase. The incorrect region grounding occurs mainly in cases of high ambiguity in the query. For example, for Flickr30k entities in the rightmost column, the query is “the load” with an image of a car with baggage on its top. The image caption “the passenger is holding on to the load on top of the car” is also provided. The word “load” itself has various

Ex ps	Flickr30K			ReferItGame		
	Positive		Negative	Positive		Negative
	a shopping cart	a red shirt	the load	the beige building on the front right	people to the left	damn that snake head
Predictions						
Attention						

(a)



(b)



(c)

Figure 4.3: (a) Examples from our approach. (b) Predicted grounding for positional cues. (c) Examples showing multiple region detection and discrepancies with ground truth.

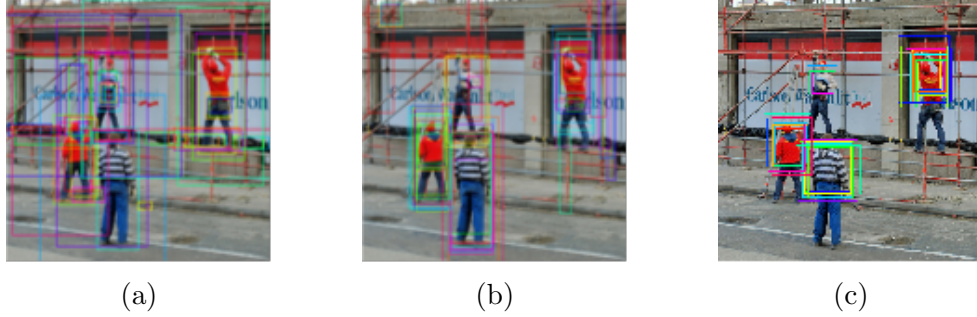


Figure 4.4: Proposals generated by query “a red shirt” by (a) RPN(COCO) (b) RPN without attention (c) attention-assisted RPN. The bounding boxes are colored only for distinguishing between the dense boxes

meanings. Without the context of the car in the query or the context of the full caption, this example becomes highly ambiguous.

Our approach also detects multiple regions for a query. This causes some discrepancies between our prediction and the ground truth, esp. in Flickr30k entities. We show some examples of discrepancies of grounding of queries in the two datasets, perceived correct grounding and our approach’s predicted grounding in Fig. 4.3c. For example, the query “two boys” is grounded showing only one of the boys in the dataset whereas our approach is able to predict regions for both the boys in the image which is perceived to be correct.

As mentioned in Section 4.5.4, the attention-assisted RPN produces better quality proposals than other pre-trained proposal generators. Fig. 4.4 shows an example of proposals generated by an (a) RPN trained on MSCOCO, (b) RPN without attention and (c) our attention-assisted RPN. As it can be seen, the RPN (a) produces lots of proposals unrelated and not useful to ground the given phrase, whereas our attention-assisted RPN produces very focused proposals based on the query.

## 4.6 Ablation study

We study the effect of some variations in our model to demonstrate the effectiveness of some design choices. Table 4.2 shows the R@1 and mAP for three variations from the final model.

Variation (a) studies the effect of not using a word embedding. Instead of using the GLOVE 300D embedding, we allow the model to learn the embedding during the training. This variation has minimal impact on ReferIt whereas a bigger impact on the Flickr30k entities. This is expected as the vocabulary size of ReferIt queries are smaller ( $\sim 1500$ ) than that of Flickr30k entities ( $\sim 4000$ ). And also learning the embedding just from the vocabulary doesn't allow the model to generalize as it does when using a word embedding trained on a large corpus.

Variation (b) studies the effect of not utilizing the encoding of the full query  $H$  in attention distribution. In this model, (4.6) (4.7) and (4.8) are reduced to the following:

$$z_t = w_z^T \tanh(W_v v_a + (W_h h_t) \mathbb{1}^T) \quad (4.11)$$

$$a_t = \text{softmax}(z_t) \quad (4.12)$$

$$c_t = \alpha_t v_a \quad (4.13)$$

This variation has a clear impact on the performance of our approach for all the datasets. Without the knowledge of the full query, the attention distribution only tends to represent the focus towards the latest word in the query, thus missing the context of the full query. For example, in a query "a red shirt", the attention distribution without  $H$  only focuses on shirts at the end of the query, whereas with  $H$ , the attention is now focusing on red shirt.

Variation (c) studies the effect of training the RPN without the use of the context



Table 4.2: Ablation Study

Method	Flickr30k R@1	Entities mAP	ReferIt R@1	Game mAP	Visual Genome R@1	Genome mAP
<b>MAGNet</b>	<b>60.20</b>	<b>0.4956</b>	<b>71.60</b>	<b>0.6052</b>	<b>28.85</b>	<b>0.1892</b>
MAGNet(a)	52.90	0.4293	69.95	0.6129	29.00	0.1823
MAGNet(b)	49.65	0.3672	68.00	0.5505	26.50	0.1316
MAGNet(c)	52.90	0.3730	68.95	0.5813	28.40	0.1144

MAGNet: Our model (a) without word embedding (b) without global H (c) without attention-assisted RPN

Table 4.3: Hit rates (N=200) of region proposal methods

Method	Flickr30k	ReferIt	Visual Genome
RPN(COCO)[79]	76.60	46.50	-
Edgebox[98]	83.69	68.26	-
Selective Search[99]	85.68	80.34	-
PGN (N=100)[70]	89.61	-	-
<b>MAGNet</b>	<b>89.78</b>	<b>92.68</b>	<b>68.59</b>
<b>MAGNet(a)</b>	78.22	83.98	50.90

MAGNet(a) MAGNet without attention-assisted RPN

vector  $\hat{C}_T$ . In this variation, we directly utilize  $v_a$  from equation (10) to train the RPN and utilize  $\hat{C}_T$  only to train the Region-CNN. This variation of training RPN is similar to the Proposal Generation Network (PGN) in [70] but with regular *cls* and *reg* RPN loss instead of the proposal generation loss dependent on the context of the full caption. This variation also creates a measurable impact on the performance of our approach. This can be understood as the proposals generated by this RPN are of lower quality than the attention-assisted RPN as shown in Table 4.3 and Fig. 4.4.

## 4.7 Conclusion

In this chapter, we utilize an encoder-decoder language model to fuse the input image and the natural language query and train an attention distribution over the input image which encodes both the local and global understanding of the query in relation

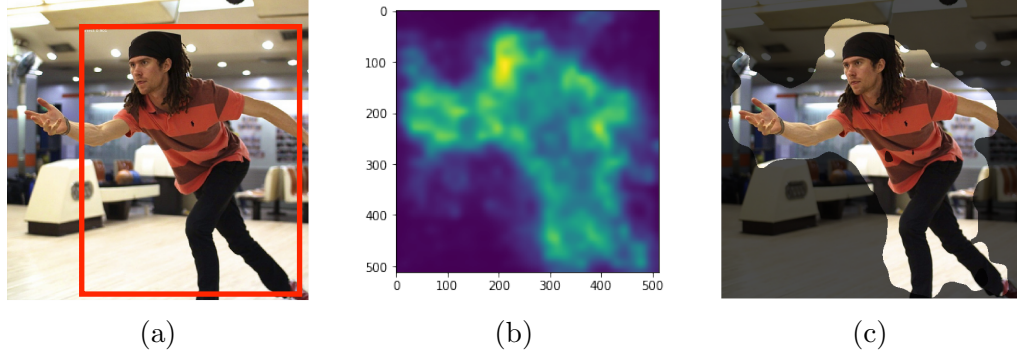


Figure 4.5: a man

to the input image. We utilize the generated context to train an attention-assisted region proposal network to generate proposals relevant to the query phrase and train an attention-assisted region CNN to classify these proposals in a Faster-RCNN framework. We call this framework the Multi-region Attention-assisted Grounding network (MAGNet). With this MAGNet framework, our model is independent of external proposal generation systems and without additional information it can develop understanding of the query phrase in relation to the image to achieve respectable results in Flickr30k entities and 12% improvement over the state-of-the-art in ReferIt game. Additionally, our model is capable of grounding multiple regions for a query phrase, which is more suitable for real-life applications. The use of attention distribution also makes the model more interpretable than other existing works.

## 4.8 Supplementary Materials

We show more examples of using our model. Fig. 4.5, 4.6, 4.7, and 4.8 show the predicted bounding box (a), attention (b), and masking (c) of a query expression. Fig. 4.9 shows predicted grounding for positional cues, while Fig. 4.10 shows examples of different attributes. Fig. 4.11, 4.12, and 4.13 show more examples in each data set where the red boxes are predicted boxes and yellow boxes are ground truth boxes.

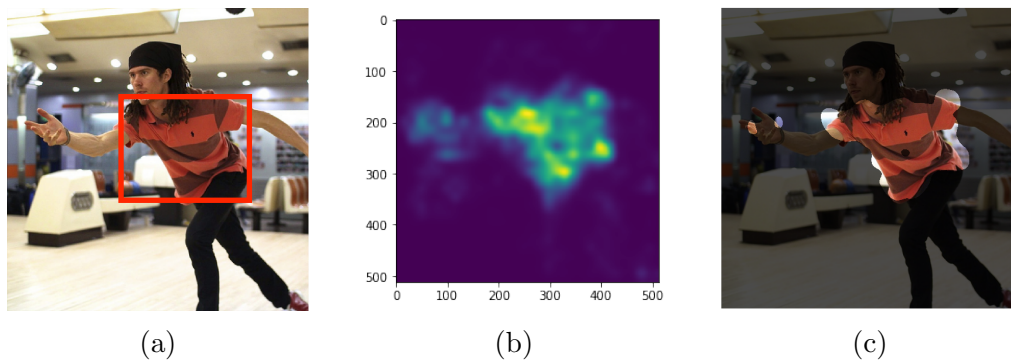


Figure 4.6: a thick-striped orange shirt

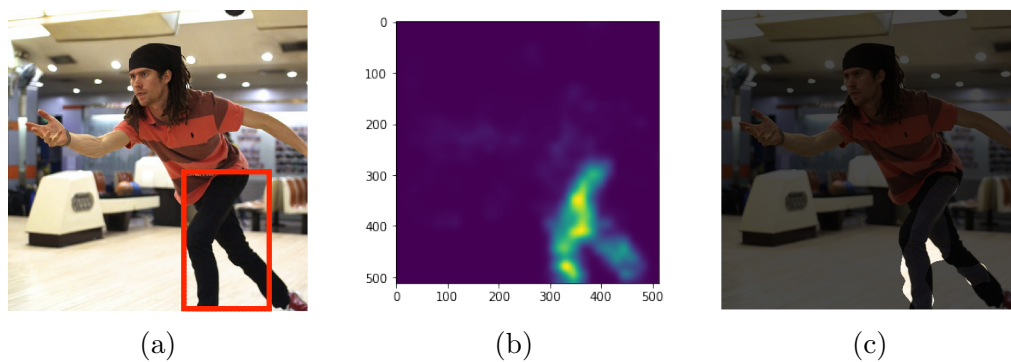


Figure 4.7: a black pants

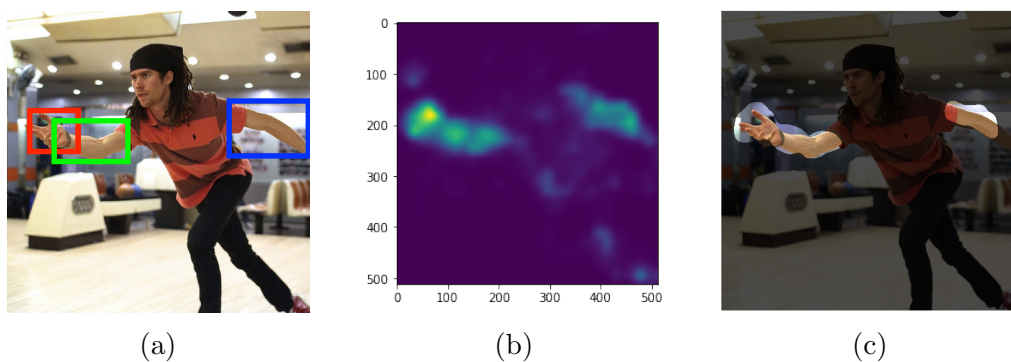


Figure 4.8: his arms



Figure 4.9: Examples of position clue

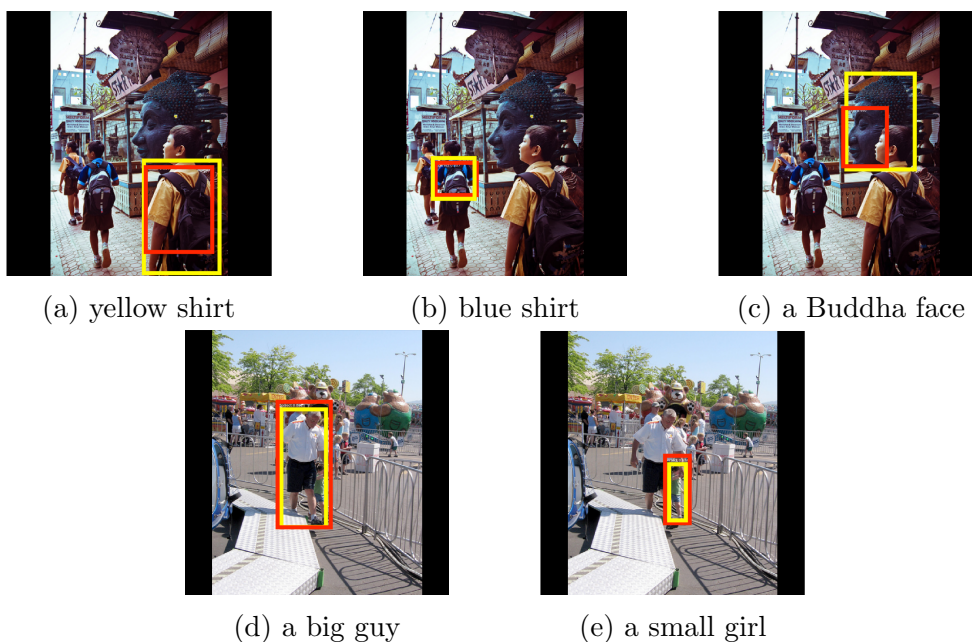


Figure 4.10: Examples with different attributes



Figure 4.11: Flickr30K examples

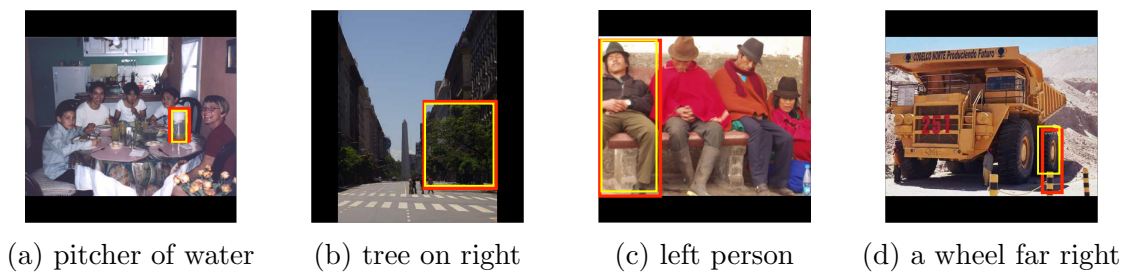
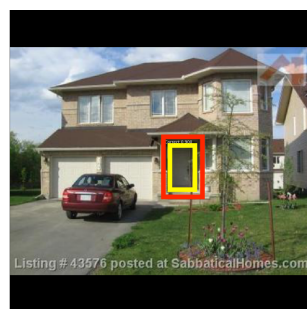


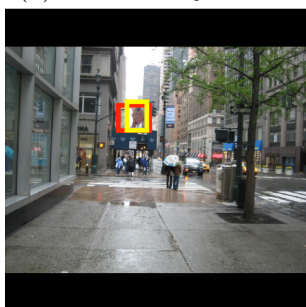
Figure 4.12: Refclef examples



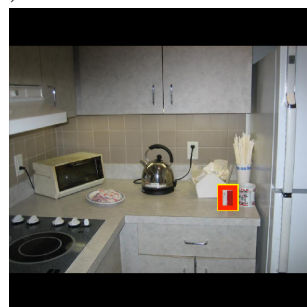
(a) bat held by batter



(b) white colored front door



(c) a flag hanging from a building



(d) the red coffee mug

Figure 4.13: Visual Genome examples

## Chapter 5

# Accelerating Block-circulant Matrix-based Neural Network Layer on a General Purpose Computing Platform: A Design Guideline

### 5.1 Introduction

In the past few years, driven by increasing amounts of data and processing speed, Deep Neural Network (DNN) has been able to deliver impressive results for many complex and challenging problems. Particularly large-scale DNNs have significantly enhanced object recognition accuracy and led a revolution in many real-world applications, such as automatic machine translation [100], self-driving systems [101], and drug discovery [102]. The resurgence of neural networks has attracted both academic and industry in evaluation, improvement and promotions.

The deep neural networks consist of multiple layers of various parameters and thousands of neurons. Recent research has proven that the depth of DNN structure is crucial to satisfactory that stands out accuracy [93]. As a result, large scale DNNs require computation and memory remarkably. Driven by this challenge, more and more techniques have been proposed to compress deep neural network size with a negligible accuracy loss. One strategy is the block-circulant matrix-based (BCM) algorithm [1], a principled approach utilizing Fast Fourier Transform (FFT) and block-circulant matrices to reduce both computational and memory complexity. Compared to other compression techniques such as weight pruning [103], BCM algorithm has three main advantages. First, it allows us to derive a tradeoff between accuracy and acceleration. Second, the BCM algorithm reduces storage complexity from  $O(n^2)$  to  $O\left(\frac{n^2}{k}\right)$  by compressing the weight matrix into  $k$  dense vector, whereas conventional weight pruning gives a sparse weight matrix that requires additional memory footprint for indexing. Lastly, BCM algorithm maintains the regular network structure and retains a rigorous mathematical foundation on a compression ratio and accuracy [1].

## 5.2 Contributions

In prior work, the BCM algorithm has only been evaluated for embedded platforms due to their portability, versatility, and energy efficiency [1]. We aim to solve two remaining questions. First, can the BCM algorithm be implemented on software-based platforms, especially in Python which is the most popular programming language used for deep learning. Second, how to configure the BCM algorithm to balance the tradeoff between accuracy and compression/acceleration. We proposed in this work to guide users to implement the BCM algorithm and achieve the best performance. To solve the two questions, we will evaluate the performance of the algorithm in

Python using numpy, intel-numpy, tensorflow, and nGraph packages. Additionally, we design the parallel BCM algorithm that effectively utilize multiple cores in the target systems.

## 5.3 Related Works

In the past decade, numerous techniques have been proposed to compress neural network. These include structured weight matrices [104, 105], parameter pruning [106, 103, 107] and quantization [108, 109]. Recently weight pruning methods have become more and more popular. Although weight pruning could achieve an amazing compression ratio, but the network structure and weight storage after pruning become irregular, hence indexing is required which weaken the performance improvement. Specially, when implemented in embedded system, it requires a customized hardware capable of loading sparse matrices and/or performing sparse matrix-vector operations [110]. Inherently, irregular memory access and extra storage footprint reduce the speed of the weight pruning.

Frequency domain operation was first proposed by LeCun to accelerate the computations of convolution layer by replacing the convolution operation using element wise multiplication in frequency domain [111]. No weight compression was considered in [111]. Circulant-weight matrix was first proposed in [104] in 2015, as a mean to reduce the storage complexity of fully connected neural networks. By compressing a weight matrix into a circulant weight matrix, it reduces the space complexity from  $O(d^2)$  to  $O(d)$ . As a property of circulant matrix, the matrix-vector multiplication (between the weights and the inputs) can be done as element wise vector-vector multiplication in frequency domain, and hence reduce the time complexity from  $O(d^2)$  to  $O(d)$ . Unlike conventional weight pruning, the circulant weight matrix has a dense structure and it could be used to optimize both speed and space. FFT is used to



transform weights and inputs to frequency domain.

For very large weight matrices, the circulant matrix approach provides very significant compressing ratio, but also will lead to considerable quality degradation of the neural network. Block-circulant weight matrix was first proposed by Ding et al. [1] as a way to balance the storage complexity and neural network quality during the compression. The authors also proposed the CirCNN to implement the BCM based Deep Neural Networks on hardware, such as ASIC and FPGA. With the customized pipeline structure, the FFT and element-wise operation achieve their best performance on the customized hardware implementation. However, the efficiency of the BCM on general purpose computing platforms, which is still normally used by machine learning community, has not been studied.

In this work, we consider all the potential overheads in software and propose a parallel design of the block-circulant matrix-based algorithm for general purpose computing platform. We evaluate its performance on popular deep learning frameworks/packages and provide guidelines that can generally lead to better implementations.

## 5.4 Background of block-circulant weight matrix

The block-circulant matrix-based algorithm can be applied to both Fully Connected (FC) and Convolutional (CONV) layers. Since the benefit of compression is more noticeable for FC layers, in this work we will focus our discussions on FC layers. The similar result can be extended to CONV layers as well.

In FC layers,  $W \in R^{m \times n}$ , a weight matrix  $W$  with the size of  $m \times n$  will be partitioned into 2D blocks of square submatrices where each submatrix is a circulant matrix. After partitioning, there will be  $p \times q$  blocks where  $p = m \div k$ , and  $q = n \div k$  while  $k$  represents the size of square submatrices or block size.  $X \in R^l$ ,

the input vector  $X$  with the size of  $l$  will also be partitioned into  $r$  blocks where  $r = l \div k$ . Fig 5.1. shows an example of partitioned matrices of input and weight where  $m = 6, n = 9, l = 9$ , and  $k = 3$ .

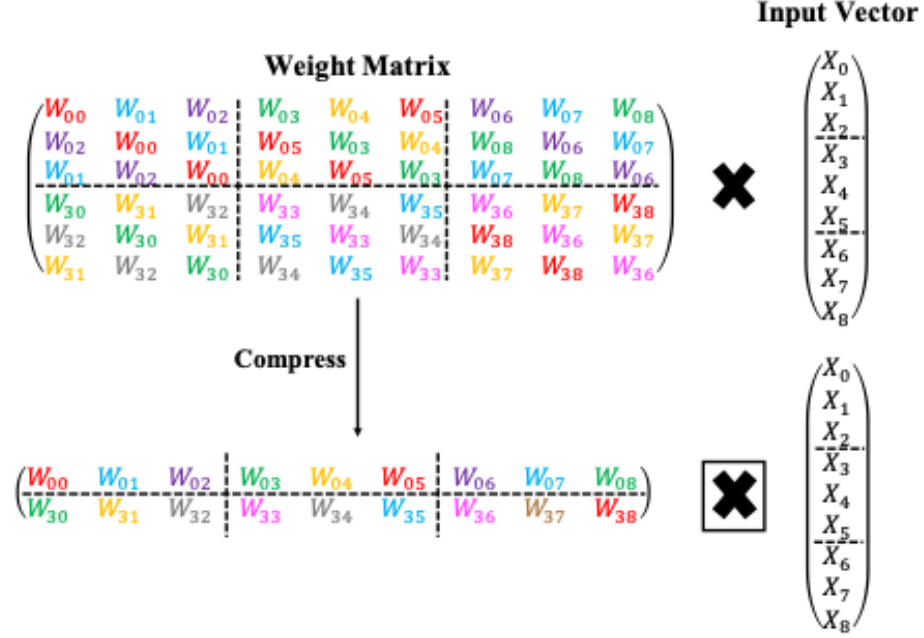


Figure 5.1: An illustration of the partitioned weight and input matrices in FC layer

Following the partitioning, the weight matrix becomes  $W = [W_{ij}], i \in \{1...p\}, j \in \{1...q\}$ , and the input matrix becomes  $X = [x_1^T, x_2^T, ..., x_q^T]^T$ . As a result, the output of each block can be calculated as:

$$a_i = \sum_{j=1}^q W_{ij} x_j^T \quad (5.1)$$

where  $a_i \in R^k$  is an output column vector. According to circulant convolution theorem [112], the compressed weight matrix  $W_{ij}$  is defined by the first row of a vector  $w_{ij}$  as shown in Fig 5.1. The output of each block can be calculated as:

$$a_i = IFFT(\sum_{j=1}^q FFT(w_{ij}) \circ FFT(x_j^T)) \quad (5.2)$$

where  $\circ$  denotes element-wise multiplication, and FFT denotes Fast Fourier trans-

form. An illustration of the BCM calculation is shown in Fig 5.2.

By using the BCM algorithm, we can reduce the storage complexity from  $O(mn)$  to  $O(pqk)$ . Since we only need to store  $FFT(w_{ij})$  for each submatrix, it is equivalent to  $O(n)$  for small  $p$  and  $q$  values. Additionally, the computational complexity of FC layer is reduced from  $O(n^2)$  to  $O(n \log n)$ , and from  $O(WHr^2CP)$  to  $O(WHQ \log Q)$  where  $Q = \max(r^2C, P)$  for CONV layer.

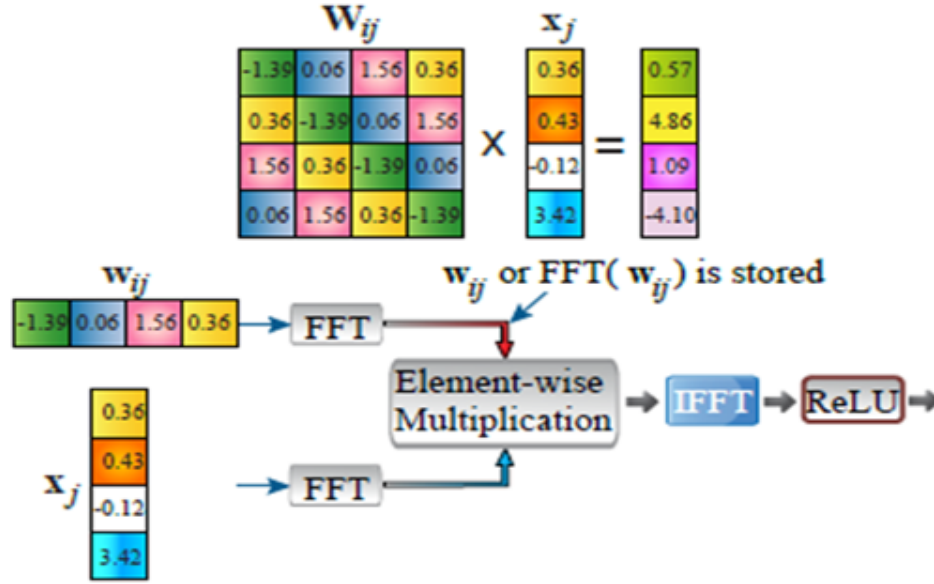


Figure 5.2: An illustration of the block-circulant matrix-based calculation [1].

## 5.5 Acceleration for General Purpose Computing Platforms

Despite its success in hardware implementation, the BCM based approach has not been widely adopted by machine learning community that works mainly on general purpose computing platforms. This is because, compared to matrix multiplication, which is highly optimized for multi-core systems in many programming paradigms, FFT, IFFT, and element-wise multiplication are not nearly optimized. This sig-

nificantly affect the performance of the BCM algorithm. In our investigations, we ran experiments comparing matrix multiplication with the BCM algorithm by using different numbers of CPU cores.

Our implementation is based on Python programming language with numpy, intel-numpy, tensorflow, and nGraph packages. We ran the experiments with one FC layer which contains 4,096 hidden neurons. A batch size of 1024, and a block size of 128. Fig 5.3. shows the results of matrix multiplication and BCM algorithm with different number of CPU cores.

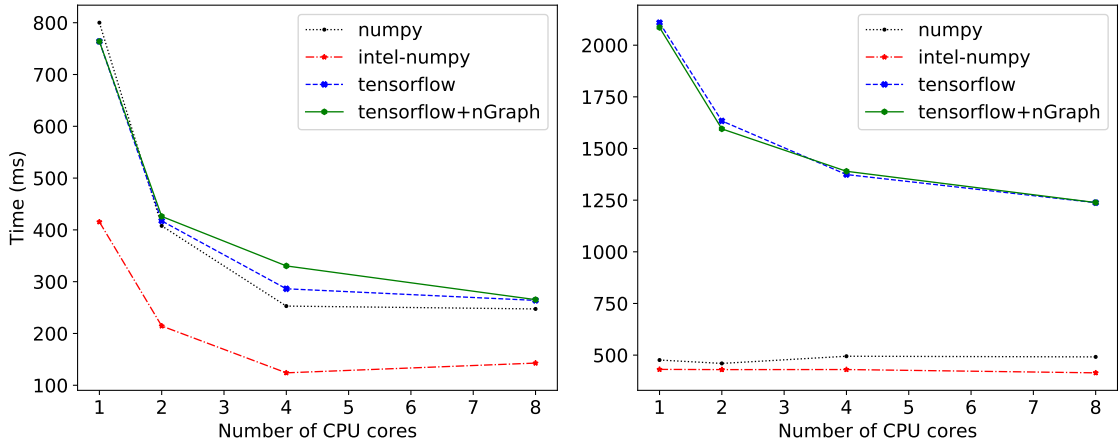


Figure 5.3: Total time used with different number of CPU cores. Left panel: Matrix multiplication results. Right panel: Block-circulant matrix-based algorithm results. The y-axis display time used in milliseconds, the x-axis shows the number of CPU cores which the maximum number is 8 since the machine has 4 physical cores and 8 threads, and each label represents different packages.

As shown in Fig 5.3, the time used in matrix multiplication decreases as the number of cores increases. This can be explained through utilization of multiple cores in each package by using either multiprocessing or multithreading. In contrast, the time used in the block-circulant matrix-based algorithm slightly decreases in tensorflow and tensorflow+nGraph while remains stable in numpy and intel-numpy.

Therefore, we design the parallel block-circulant matrix-based algorithm to accelerate the computations. The key idea is to separate each computation block and

run it on different processes as each block can be calculated independently. Fig 5.4. represents parallel block-circulant matrix-based algorithm. We partition the block-circulant matrix by row and run it on the different processes. Once the calculations are completed, we combine and convert the matrices to get the final output.

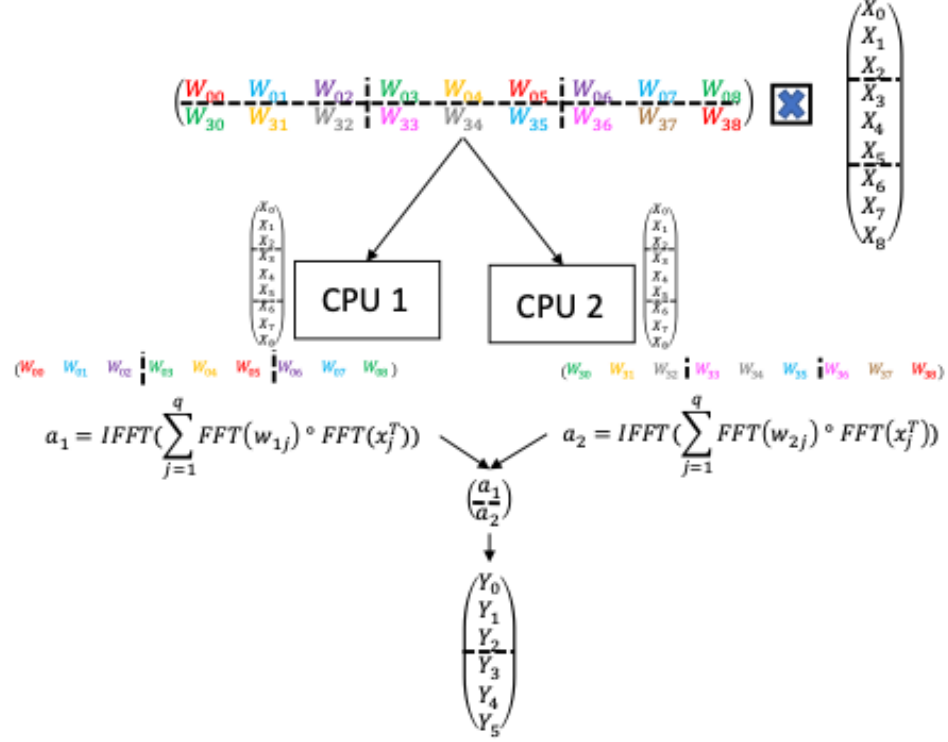


Figure 5.4: An illustration of parallel block-circulant matrix-based algorithm.

In case there are multiple inputs, we can use data parallelism to separate these inputs into processes such that each portion of data is assigned to different processes. The portion of data is defined as *input size / number of CPU cores*.

In terms of implementation, we initially use native multithreading and multiprocessing provided in Python. However, Python has Global Interpreter Lock (GIL) that only allows one thread to hold the control of its interpreter, creating the performance bottleneck in multithreading. In contrast to multithreading, multiprocessing uses sub processes to solve GIL which allows the program to optimize multiple cores in a given machine. Nevertheless, there are overhead in spawning processes and sending data.

To improve the performance, we use Ray [2]: a simple framework that has been proven to be faster than native multiprocessing and multithreading. Additionally, Ray can be easily integrated with Python. Fig 5.5. represents the basic sample code of how to use Ray to accelerate the parallel block-circulant matrix-based algorithm.

Basic Python	Distributed with Ray
<pre># Execute f serially.  def f():     time.sleep(1)     return 1  results = [f() for i in range(4)]</pre>	<pre># Execute f in parallel.  @ray.remote def f():     time.sleep(1)     return 1  ray.init() results = ray.get([f.remote() for i in range(4)])</pre>

Figure 5.5: Example use of Ray in Python implementation [2].

Using parallel design and Ray, we can achieve better performance than the original block-circulant matrix-based algorithm when increasing the number of CPU cores. Fig 5.6. shows the results of our parallel version versus the previous version. The new parallel version can achieve a stable speedup ratio up to 4 cores which is the number of physical cores.

## 5.6 Design Space Exploration of BCM

In this work, the block-circulant matrix-based algorithm has been applied to the model during inference phase. When it comes to computational complexity, the block-circulant matrix-based algorithm is faster than matrix multiplication. However, when it comes to implementation, we need to examine the overhead from *IFFT*, *FFT*, and matrix reshaping. Design parameters such as the batch size, block size, and number of CPU cores all will affect the calculation time. For some combinations, matrix multiplication may be faster than the BCM, while for others the BCM may be more

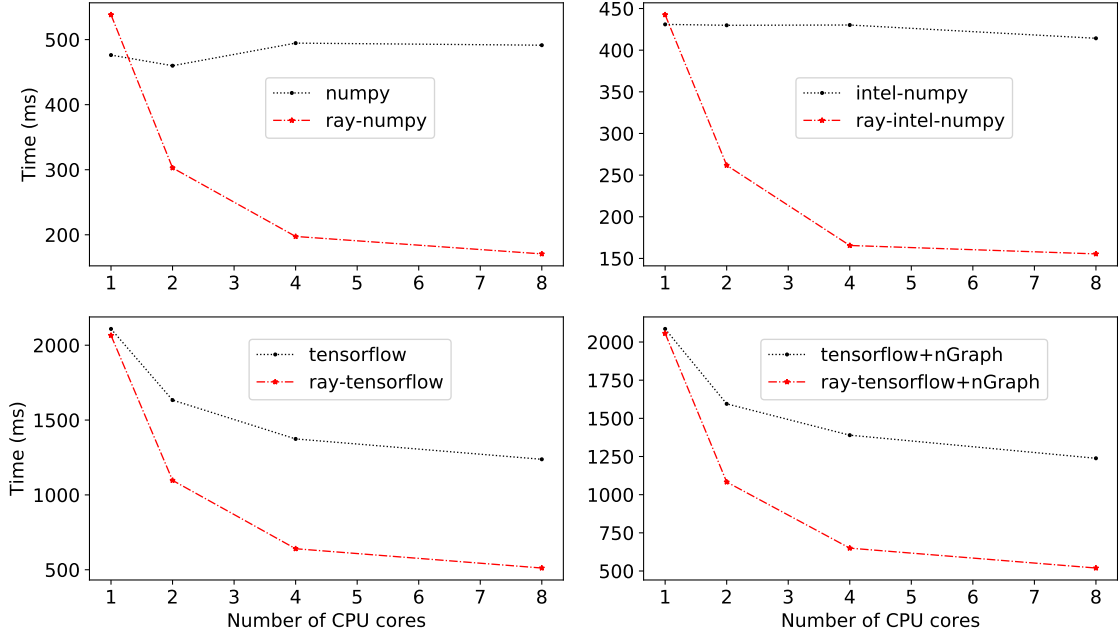


Figure 5.6: Comparing the original and our ray-parallel implementation. Top-left panel: numpy. Top-right panel: intel-numpy. Bottom-left panel: tensorflow. Bottom-right panel: tensorflow+nGraph. A batch size of 1024, and a block size of 128 are used in these experiments. The y-axis display time used in milliseconds, and the x-axis shows the number of CPU cores.

effective than matrix multiplication. While increasing the block size always reduces storage and computing complexity, it also lowers the model capacity of the neural network and hence may lead to larger prediction error. It has been shown in [1] that with a compression ratio of up to 30-50x, sometimes the loss may be negligible, and the compressed models may even outperform the baseline models. However, in some cases the loss is noticeable. In general, the loss is monotonically increasing with the compression ratio. By focusing on the speed during the inference phase, users must decide whether the accuracy loss is acceptable.

In order to choose the best model with the most efficient configuration and acceptable accuracy without exhaustively exploring the entire design space, the designer needs to know how the performance is affected by these design parameters including the batch size, block size, and number of CPU cores. In this work, we designed a set of

benchmark programs that characterize the performance of different configurations of BCM with a comparison to the matrix based implementation. Guidelines in choosing the configuration of the BCM was derived from the result. These guidelines will help designers to choose the configuration without having to attempting all combinations.

The study is performed on Intel(R) Xeon(R) W-2123 CPU @3.6GHz which has 4 physical cores and 8 threads. Matrix multiplication and the BCM algorithm are implemented using Python programming language with various packages. The following lists of possible choices of the hardware/software configurations and design parameters that were evaluated.

- Packages: numpy, intel-numpy, tensorflow, and tensorflow+nGraph.
- Number of CPU cores: 1, 2, 4, and 8
- Block size ( $M$ ): 128, 256, 512, 1024, and 2048
- Batch size ( $N$ ): 128, 256, 512, 1024, 2048, 4096, and 8192

The evaluation results reported in this work are platform specific, however, the benchmarks and methodologies can be applied to other platforms. The model that we considered is a fully connected layer with 4,096 hidden neurons. The weights and inputs that have the size of (4096, 4096), and (number of batches, 4096) respectively. Table 5.1 shows the size of inputs, blocks, and weights used in the experiments.

We assume that the block-circulant weight matrix has already been trained and the FFT of each block has been calculated. Please refer to [1] about how to train a block circulant weight matrix. For a fully connected layer whose input size and output size are  $X$  and  $Y$ , let  $M$  denote the block size, then there will be  $\lceil \frac{X}{M} \rceil \times \lceil \frac{Y}{M} \rceil$  circulant blocks. Each block, after FFT, will be represented as a vector of size  $\frac{M}{2} + 1$  since we only compute the real part of discrete Fourier Transform. Overall we will represent the weight as 4D tensor with size  $1 \times \lceil \frac{X}{M} \rceil \times \lceil \frac{Y}{M} \rceil \times (\frac{M}{2} + 1)$ . In Table



Table 5.1: Size of inputs, blocks, weights before and after partitioning & FFT

Input size	Weight size before partitioning & FFT (X,Y)	Block size (M)	Weight size after partitioning & FFT (1, $\lceil \frac{X}{M} \rceil$ , $\lceil \frac{Y}{M} \rceil$ , $\frac{M}{2} + 1$ )
(N, 4096)	(4096, 4096)	128	(1, 32, 32, 65)
		256	(1, 16, 16, 129)
		512	(1, 8, 8, 257)
		1024	(1, 4, 4, 513)
		2048	(1, 2, 2, 1025)

5.1, N represents the number of batches. The weight size after partitioning & FFT has 4 dimensions, including 1,  $\lceil \frac{X}{M} \rceil$ ,  $\lceil \frac{Y}{M} \rceil$ , and size after FFT where 1 represents additional dimension that help matching the number of batches in input size during element-wise multiplication.

In each experiment, we record the time used starting from the initial step until we receive the output from the algorithm. The algorithm consists of six steps as the following:

1. Reshaping input X into 4 dimensions  $(N, \lceil \frac{X}{M} \rceil, 1, M)$  to match the size of weight tensor.
2. Calculating the *FFT* of input X from step 1,  $FFT(X)$  where the size of this step becomes  $(N, \lceil \frac{X}{M} \rceil, 1, \frac{M}{2} + 1)$ .
3. Calculating element-wise multiplication  $FFT(W) \circ FFT(X)$ . The output size becomes  $(N, \lceil \frac{X}{M} \rceil, \lceil \frac{Y}{M} \rceil, \frac{M}{2} + 1)$ .
4. Summing the output from step 3 using the formula:  $\sum_{i=1}^{\lceil \frac{X}{M} \rceil} FFT(w_{ij}) \circ FFT(x_i)$ , where  $j$  is each block in  $\lceil \frac{Y}{M} \rceil$ . The output size becomes  $(N, \lceil \frac{Y}{M} \rceil, \frac{M}{2} + 1)$ .
5. Calculating *IFFT* along the third dimension of the tensor from step 4 and get output Y. The output size becomes  $(N, \lceil \frac{Y}{M} \rceil, M)$ .

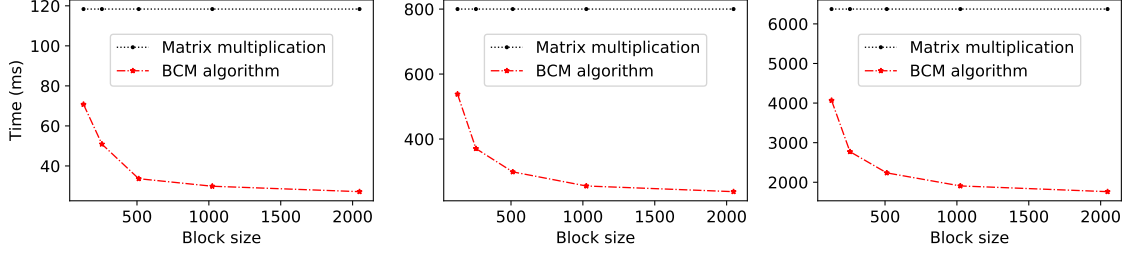


Figure 5.7: Total time used with different block size with single core in numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

6. Reshaping the output into  $(N, \text{output size})$  where the output size is 4,096.

The dimensions that have been set to 1 are used for broadcasting which are available in all packages that we use in our experiment, reducing and simplifying the codes.

## 5.7 Impact of block size on performance

Varying the block size will result in different compression ratios and accuracies. In this experiment, we set the number of blocks to 128, 256, 512, 1024, and 2048 to observe the performance in relative to increasing number of blocks in different configurations (i.e. number of CPU cores and batches).

**Single core.** Even though multiple cores/GPUs may be readily available, some specific embedded system/machine may only have single core. Therefore, running deep model on this system may require significant amount of time and memory. BCM based approach is especially effective for this type of resource constrained platforms. Applying the block-circulant matrix-based algorithm will reduce the amount of memory used and increase the speed. Fig 5.7, 5.8, 5.9, and 5.10. Show how inference time reduces as block size increases.

In all packages, increasing the block size reduces the time used. However, this

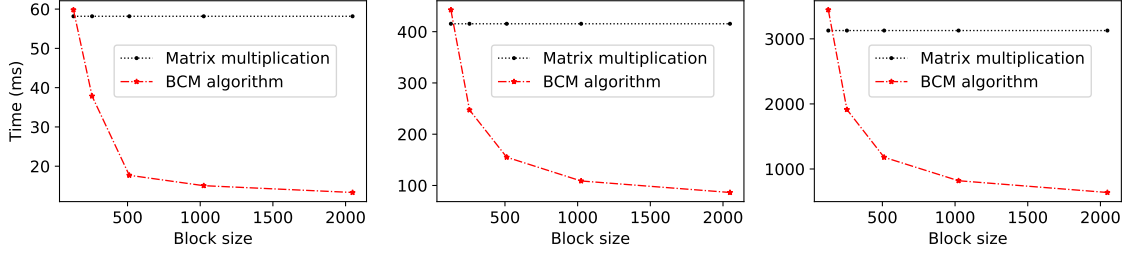


Figure 5.8: Total time used with different block size with single core in intel-numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

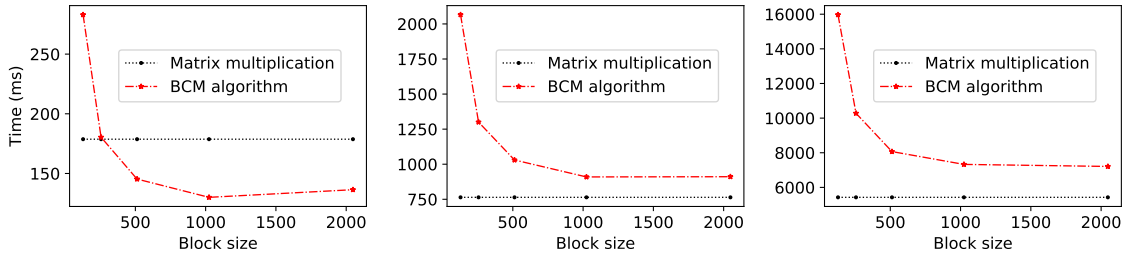


Figure 5.9: Total time used with different block size with single core in tensorflow package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

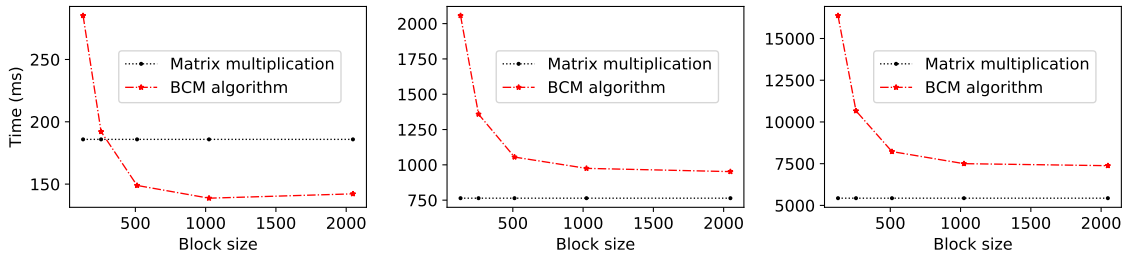


Figure 5.10: Total time used with different block size with single core in tensorflow+nGraph package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

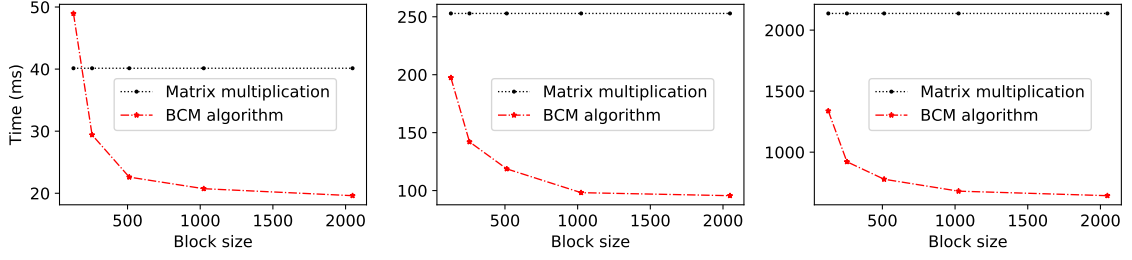


Figure 5.11: Total time used with different block size with multiple cores in numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

correlation is not linear, increasing the block size twice does not reduce the time by half. Once the block size reaches 1024, the speedup ratio remains stable.

In numpy, using the BCM algorithm is faster than matrix multiplication in all block size, small, medium, and large batch size. Meanwhile, the BCM algorithm is faster when the block size is larger than 128 in intel-numpy. The same results apply to all batch size with more difference observed in the large batch size.

In contrast to numpy and intel-numpy, tensorflow and tensorflow+nGraph display a slower speed in the BCM algorithm when compared to matrix multiplication. This outcome applies to small batch size when block size less than 256, and all block size for medium and large batch size. The results can be explained by the overhead time used to compute FFT, IFFT, and to create session to run the calculation.

**Multiple cores.** Exploiting the resources of multiple cores of the system/machine will help increase the overall performance. As mentioned earlier, we use Ray library to implement the parallel block-circulant matrix-based algorithm. In this experiment, we set the number of cores to 4 and ran the experiments using small, medium, and large batch size. Fig 5.11, 5.12, 5.13, and 5.14. represents how much time used when we increase block size in each package using multiple cores.

Due to different techniques used in multiple cores in each package, different results are expected. In numpy, the BCM algorithm appears to be faster than matrix

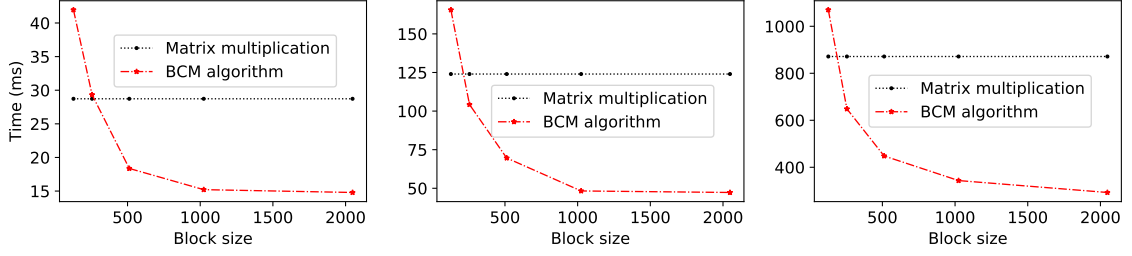


Figure 5.12: Total time used with different block size with multiple cores in intel-numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

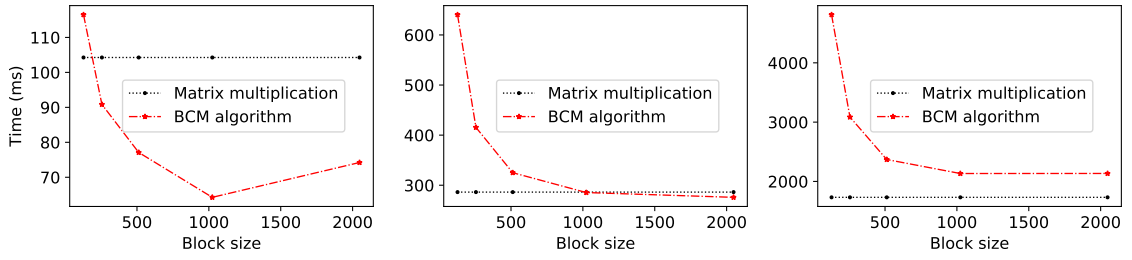


Figure 5.13: Total time used with different block size with multiple cores in tensorflow package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

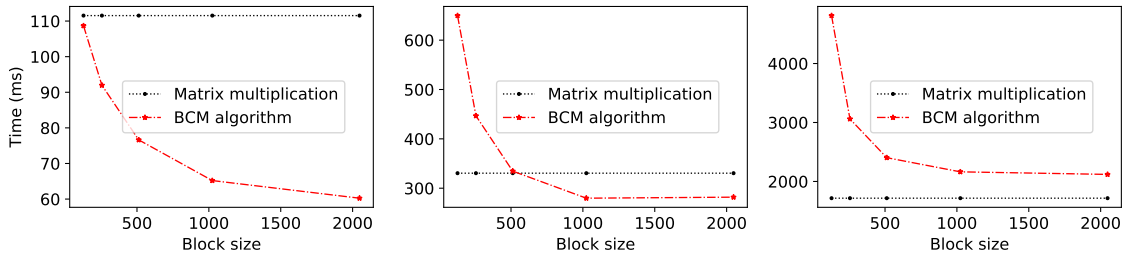


Figure 5.14: Total time used with different block size with multiple cores in tensorflow+nGraph package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, and the x-axis shows different block size.

multiplication once the block size is above 128 and small batch size is used, while it appears to be faster with all block sizes when using a medium and large batch size.

With intel-numpy, matrix multiplication is always faster than the BCM algorithm for block size below 128. The BCM outperforms matrix multiplication only for relatively larger blocks. This outcome can be explained by highly optimized matrix multiplication by Intel Math Kernel Library (MKL) packages which is designed specifically for intel CPU.

With tensorflow and tensorflow+nGraph, the break-even block size where BCM and matrix multiplication has similar performance moves left. We now need even larger blocks to outperform matrix multiplication. Similar to single core system, the BCM performance still prefers smaller batch size. The break-even block size gets bigger when the batch size increases.

## **5.8 Impact of number of CPU cores on performance**

The parallel block-circulant matrix-based algorithm utilizes multiple cores by spawning multiple processes. However, the amount of speed up is not linearly proportional to the hardware resources. Increasing the number of cores exceeds a certain point will cause the program to run slower because of the communication bottleneck. Each process uses more time in synchronization, such that the amount of increased communication time outweighs the amount of computing time saved by add more cores. With some configurations, applying matrix multiplication is more advantageous.

In this experiment, the number of cores is set to 1, 2, 4 and 8. We ran the experiments using a small, medium, and large batch size of 128, 1024, and 8192 in numpy, intel-numpy, tensorflow, and tensorflow+nGraph respectively. Fig 5.15, 5.16, 5.17, and 5.18. display how much performance gain when we increase the number of

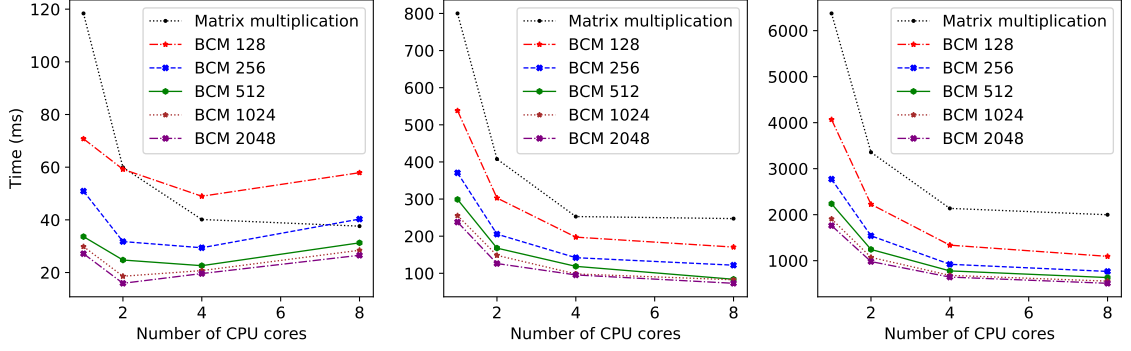


Figure 5.15: Total time used with different number of CPU cores in numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, the x-axis shows different number of CPU cores, and the legend represents matrix multiplication and different block size.

cores in each package using a different block size.

In general, using BCM algorithm can gain a stable speedup ratio up to 4 cores, while in some cases, it become slower when using 8 cores as a result of parallel slowdown, each process uses more time in communication and spawning process than the increased processing power that it achieves.

Meanwhile in numpy, BCM algorithm is faster than matrix multiplication, However, when a small batch size, block size of 128 & 4 cores, and block size of 256 & 8 cores are used, matrix multiplication is faster. In intel-numpy, regardless of the number of cores and batch size, the block-circulant is faster than matrix multiplication when the block size is larger than 128.

For tensorflow and tensorflow+nGraph, the BCM algorithm is slower than matrix multiplication when using larger batch size. Although at small batch size, it is faster when setting block size to be greater than 128.

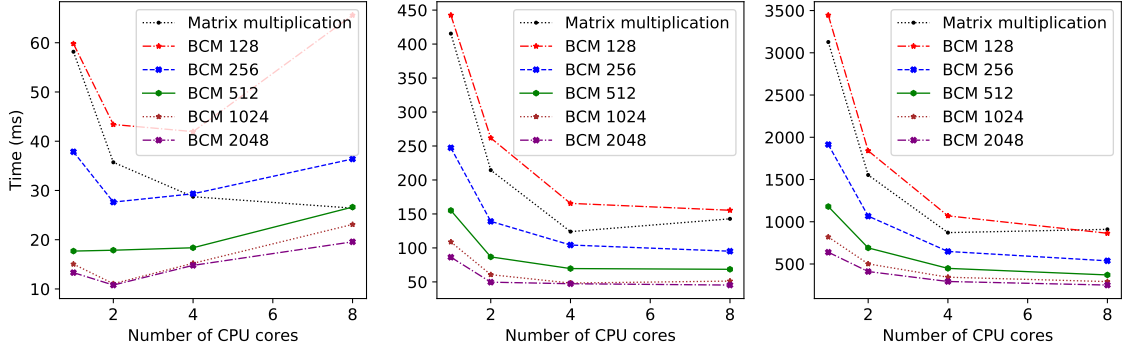


Figure 5.16: Total time used with different number of CPU cores in intel-numpy package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, the x-axis shows different number of CPU cores, and the legend represents matrix multiplication and different block size.

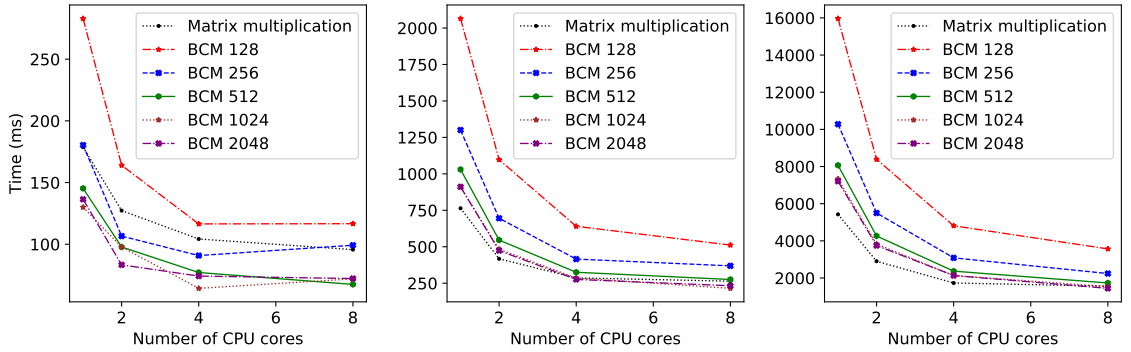


Figure 5.17: Total time used with different number of CPU cores in tensorflow package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, the x-axis shows different number of CPU cores, and the legend represents matrix multiplication and different block size.



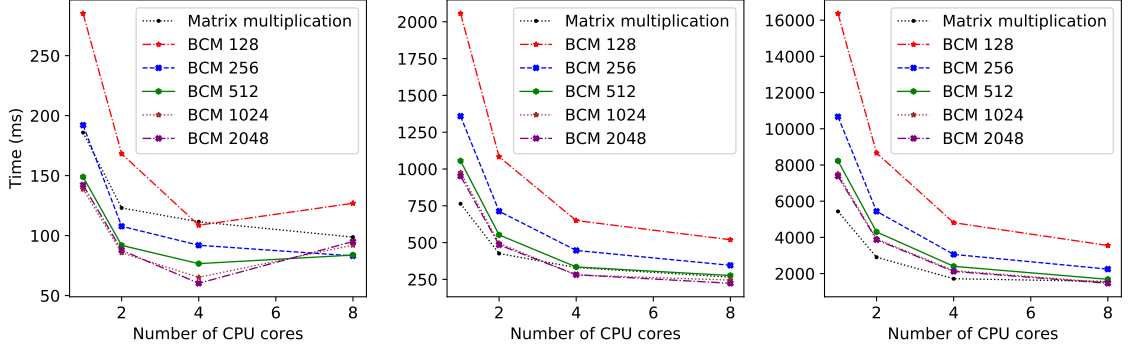


Figure 5.18: Total time used with different number of CPU cores in tensorflow+nGraph package. Left panel: Small batch size (128). Middle panel: Medium batch size (1024). Right panel: Large batch size (8192). The y-axis display time used in milliseconds, the x-axis shows different number of CPU cores, and the legend represents matrix multiplication and different block size.

## 5.9 Impact of batch size on performance

Since the batch size affects the computational speed, choosing appropriate batch size for particular system will significantly improve the performance. For instance, using larger batch size will improve the computation speed in multiple cores system/machine. However, there is a saturation point where increasing the batch size will no longer decrease the computational speed.

In contrast, using smaller or larger batch size with a single core does not significantly affect the computational speed since it has to go through data one at a time. Fig 5.19 displays total compute time when increase batch size, while Fig 5.20 shows compute time per sample. A block size of 128 & 256, and number of CPU cores of 1 & 4 are used in this experiment.

As shown in Fig 5.19, a positive near linear relationship can be observed between the batch size and the total computational time. However, the amount of time is different in each configuration which is explained through the slope differences.

The computational time used per sample is not decreasing as the batch size increase as shown in Fig 5.20. However, it still decreases a little at the beginning because

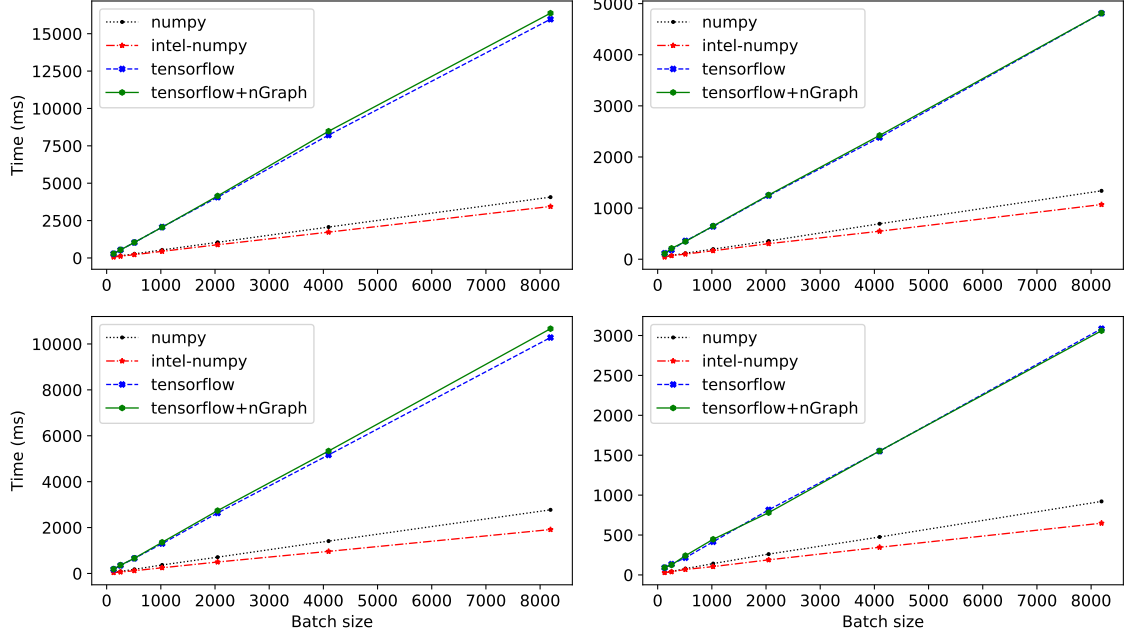


Figure 5.19: Total computation time. Top-left panel: 1 core with 128 block size. Top-right panel: 4 cores with 128 block size. Bottom-left panel: 1 core with 256 block size. Bottom-right panel: 4 cores with 256 block size. The y-axis displays total time used in milliseconds, and the x-axis shows the batch size.

of overhead in creating session & graph when using tensorflow or tensorflow+nGraph, and spawning processes and sending data when using multiple cores.

## 5.10 Summary of Design Guidelines

In general, using the block-circulant matrix-based algorithm in intel-numpy packages is the best choice since intel-numpy takes the benefit of intel MKL which is highly optimized for mathematics operations. However, there are certain cases such as when the batch size is large enough and the block size is 128 where matrix multiplication will be more beneficial. Another case is when using a single core with small batch size, using the block-circulant in numpy is the fastest only when the block size is less than 256. Due to the emphasis placed on inference phase, tensorflow and tensorflow+nGraph always perform slower than numpy and intel-numpy when applying

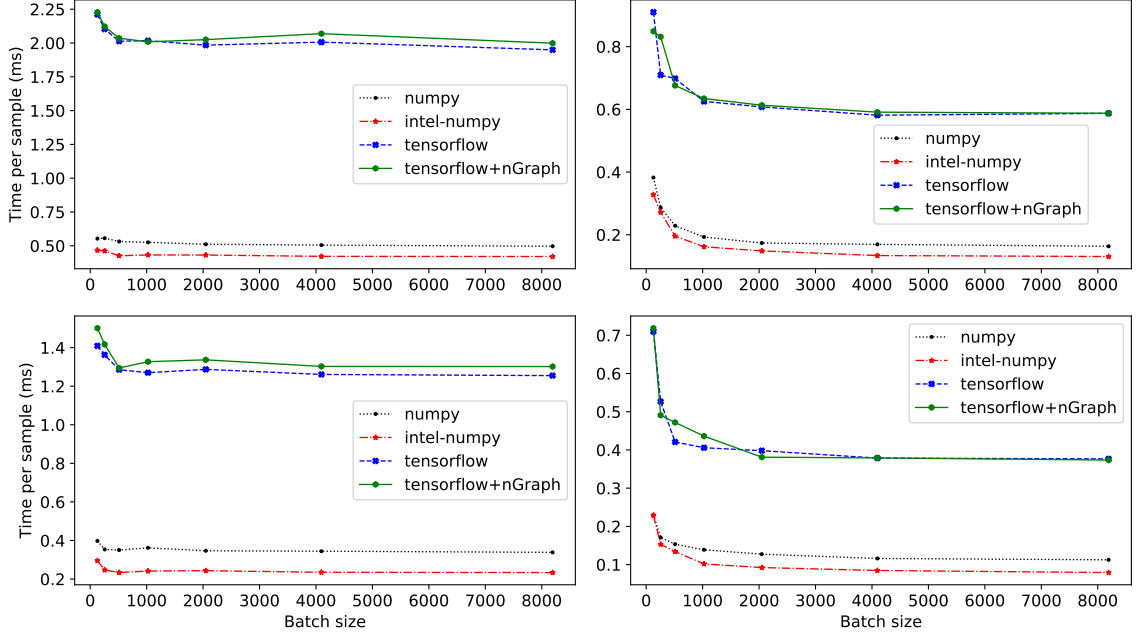


Figure 5.20: The time used per sample. Top-left panel: 1 core with 128 block size. Top-right panel: 4 cores with 128 block size. Bottom-left panel: 1 core with 256 block size. Bottom-right panel: 4 cores with 256 block size. The y-axis display time used per sample in milliseconds, and the x-axis shows the batch size.

BCM algorithm. They require time to initialize the graph and session before starting to compute the output.

When using multiple cores, larger batch size will optimize the parallelization of the algorithm which will speed-up compute time of all samples in the batch.

Although the guideline provides the best possible combination of block size, number of cores, and batch size to achieve optimal performance, it focuses mainly on the time used to compute the algorithm. The accuracy reduction must be addressed of manually in exchange for the increasing speed of the compute time.

## 5.11 Conclusion

In this chapter, we proposed a parallel design of the block-circulant based-matrix algorithm and demonstrated that this new design can achieve better performance

than previous version of algorithm. We also provide guidelines on how to select block size, batch size, and number of cores in certain situations in order to achieve optimal performance in the least amount of time. The guidelines run across popular implementation language and packages including Python, numpy, intel-numpy, tensorflow, and nGraph.

# Chapter 6

## Conclusion

### 6.1 Summary

In this thesis, we present three Deep Neural Networks that used to learn semantic information from different types of data e.g. text, image, and click locations and a design guideline to accelerate Neural Network Layer on a general purpose computing platform.

In Chapter 2, we proposed a new technique to preprocess the dataset and improve the conventional RBM based topic classification by introducing new input features based on semantically related word pairs. The technique applies semantic dependency parser to extract the word pairs from the text document, two-way TF-IDF processing to filter the data in word level and word pair level, K-means clustering algorithm to merge the similar word pairs and remove the noise from the feature dictionary. We showed that proper selection of K value and word pair generation techniques can significantly improve the topic prediction accuracy and the document retrieval performance.

In Chapter 3, we proposed a human in a loop automatic image labeling model which include combined prediction, post-processing, and adjustment model. The

model focuses on aerial images, which lack salient features for detection. We also showed that by leveraging user click information and the adjustment model, we can achieve transfer learning capability.

In Chapter 4, we proposed a MAGNet framework that utilizes the encoder-decoder language model to fuse the input image and the natural language query and train an attention distribution over the input image, which encodes both the local and global understanding of the query in relation to the input image. We also utilize the generated context from encoder-decoder language model to train an attention-assisted region proposal network to generate proposals relevant to the query phrase and train an attention-assisted region CNN to classify these proposals in a Faster-RCNN framework. With this MAGNet framework, our model is independent of external proposal generation systems and without additional information it can develop understanding of the query phrase in relation to the image to achieve respectable results.

In Chapter 5, we proposed a parallel design of the block-circulant based-matrix algorithm and demonstrated that this new design can achieve better performance than previous version of algorithm. We also provide guidelines on how to select block size, batch size, and number of cores in certain situations in order to achieve optimal performance in the least amount of time. The guidelines run across popular implementation language and packages including Python, numpy, intel-numpy, tensorflow, and nGraph.

## **6.2 Future Research Directions**

### **6.2.1 Improvement on MAGNet framework**

In Chapter 4, we present a MAGNet framework that can be used to understand the relationship between the query phrase and the image. In comparison to the state-of-the-art models, our model can achieve respectable results in three data sets which

include Flickr30k entities, ReferIt game, and Visual Genome. However, there are more data in visual grounding tasks such as RefCOCO, RefCOCO+, and RefCOCOg. We ran our model on these data sets and received comparable results, however, unfortunately, we are currently unable to outperform state-of-the-art model due to some limitations of the MAGNet framework. The MAGNet framework works better when the input phrases are concise and specific e.g. “right rocks”, “The man with red shirt”, “Person on the right”. It works less well with more complex queries like “guy in yellow dribbling the ball”, “A man with yellow shirt and black short”. The problem occurs in the Encoder-Decoder Language Model part. Given the complexity of the sentences, the attention that is generated from this model cannot focus its gaze to the relevant area of the image, resulting in the inability of the overall framework to predict the accurate location. We need to investigate different model architecture designs to achieve better attention. One of the possible architectures is the Transformer Network which has been used recently in many researches and outperform state-of-the-art sequence-to-sequence model in many areas and applications.

### **6.2.2 Accelerating MAGNet framework by applying Structure Pruning to Deep Network**

The overall MAGNet framework consists of multiple parts. Each part is implemented using Deep Neural Network which consists of multiple layers of various parameters and thousands of neurons. The large scale DNNs require computation and memory remarkably. Thus, our framework requires a huge number of resources to train/test the model in which not applicable to apply to real time application . There are multiple techniques that have been proposed to compress deep neural network size with a negligible accuracy loss. For example, [113] proposed a framework to induce different types of structured sparsity such as filter-wise, channel-wise, and shape-wise sparsity, as well as non-structured sparsity. Applying this approach to our framework

will improve the computation time and memory requirements.



# Bibliography

- [1] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. Yuan *et al.*, “Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices,” in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 395–408.
- [2] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, “Ray: A distributed framework for emerging {AI} applications,” in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 561–577.
- [3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [4] K. Wankhede, B. Wukkadada, and V. Nadar, “Just walk-out technology and its challenges: a case of amazon go,” in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2018, pp. 254–257.
- [5] D. M. Blei, “Probabilistic topic models,” *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.
- [6] M. Steyvers and T. Griffiths, “Probabilistic topic models,” *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

- [7] I. M. D. Inc., “The internet movie database,” 1990. [Online]. Available: <http://www.imdb.com/>
- [8] Q. Mei, D. Cai, D. Zhang, and C. Zhai, “Topic modeling with network regularization,” in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 101–110.
- [9] C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.
- [10] X. Wang and E. Grimson, “Spatial latent dirichlet allocation,” in *Advances in neural information processing systems*, 2008.
- [11] T. K. Landauer, *Latent semantic analysis*. Wiley Online Library, 2006.
- [12] G. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Numerische mathematik*, vol. 14, no. 5, 1970.
- [13] T. Hofmann, “Probabilistic latent semantic analysis,” in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.
- [14] D. M. Blei, A. Ng, and M. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [15] G. E. Hinton and R. R. Salakhutdinov, “Replicated softmax: an undirected topic model,” in *Advances in neural information processing systems*, 2009, pp. 1607–1614.
- [16] H. Lu, L.-Y. Xie, N. Kang, C.-J. Wang, and J.-Y. Xie, “Don’t forget the quantifiable relationship between words: Using recurrent neural network for short text topic discovery,” in *AAAI*, 2017, pp. 1192–1198.

- [17] A. B. Dieng, C. Wang, J. Gao, and J. Paisley, “Topicrnn: A recurrent neural network with long-range semantic dependency,” *arXiv preprint arXiv:1611.01702*, 2016.
- [18] S. T. Dumais, “Latent semantic analysis,” *Annual review of information science and technology*, vol. 38, no. 1, pp. 188–230, 2004.
- [19] T. Mikolov, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.
- [21] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted boltzmann machines for collaborative filtering,” in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.
- [22] N. Srivastava, R. Salakhutdinov, and G. E. Hinton, “Modeling documents with deep boltzmann machines,” *arXiv preprint arXiv:1309.6865*, 2013.
- [23] D. Chen and C. Manning, “A fast and accurate dependency parser using neural networks,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.
- [24] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. T. McDonald, S. Pyysalo, N. Silveira *et al.*, “Universal dependencies v1: A multilingual treebank collection.” in *LREC*, 2016.

- [25] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.
- [26] G. Salton and E. A. Fox, “Extended boolean information retrieval,” *Communications of the ACM*, vol. 26, no. 11, pp. 1022–1036, 1983.
- [27] G. Salton and M. J. McGill, “Introduction to modern information retrieval,” 1986.
- [28] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.
- [29] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, “Interpreting tf-idf term weights as making relevance decisions,” *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, p. 13, 2008.
- [30] B. Fritz, “Omdb api.” [Online]. Available: <http://www.omdbapi.com/>
- [31] A. M. d. J. C. Cachopo, “Improving methods for single-label text categorization,” *Instituto Superior Técnico, Portugal*, 2007.
- [32] A. Turpin and F. Scholer, “User performance versus precision measures for simple search tasks,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 11–18.
- [33] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

- [34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [35] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [38] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*. Springer, 2014, pp. 818–833.
- [39] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
- [40] R. G. Cinbis, J. Verbeek, and C. Schmid, “Weakly supervised object localization with multi-fold multiple instance learning,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 189–203, 2016.
- [41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

- [42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [44] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [45] E. Teng, R. Huang, and B. Iannucci, “Clickbait-v2: Training an object detector in real-time,” *arXiv preprint arXiv:1803.10358*, 2018.
- [46] J. Lee, S. Walsh, A. Harakeh, and S. L. Waslander, “Leveraging pre-trained 3d object detection models for fast ground truth generation,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2504–2510.
- [47] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, “We don’t need no bounding-boxes: Training object class detectors using only human verification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 854–863.
- [48] —, “Training object class detectors with click supervision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6374–6383.
- [49] C. Wah, S. Branson, P. Perona, and S. Belongie, “Multiclass recognition and part localization with humans in the loop,” in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2524–2531.

- [50] S. Bell, P. Upchurch, N. Snavely, and K. Bala, “Material recognition in the wild with the materials in context database,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3479–3487.
- [51] T. Wang, B. Han, and J. Collomosse, “Touchcut: Fast image and video segmentation using single-touch interaction,” *Computer Vision and Image Understanding*, vol. 120, pp. 14–30, 2014.
- [52] J. Liew, Y. Wei, W. Xiong, S.-H. Ong, and J. Feng, “Regional interactive image segmentation networks,” in *2017 IEEE international conference on computer vision (ICCV)*. IEEE, 2017, pp. 2746–2754.
- [53] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, “What’s the point: Semantic segmentation with point supervision,” in *European conference on computer vision*. Springer, 2016, pp. 549–565.
- [54] L. Itti, “Neovision2 annotated video datasets,” 2014.
- [55] Y. Yang, “tensorflow-yolov3,” 2019. [Online]. Available: <https://github.com/YunYang1994/tensorflow-yolo>
- [56] Y. Li, Y. Chen, N. Wang, and Z. Zhang, “Scale-aware trident networks for object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6054–6063.
- [57] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling, “Cbnet: A novel composite backbone network architecture for object detection,” *arXiv preprint arXiv:1909.03625*, 2019.
- [58] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, “Flickr30k entities: Collecting region-to-phrase correspondences

- for richer image-to-sentence models,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2641–2649.
- [59] L. Wang, Y. Li, J. Huang, and S. Lazebnik, “Learning two-branch neural networks for image-text matching tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 394–407, 2018.
- [60] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, “Referitgame: Referring to objects in photographs of natural scenes,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 787–798.
- [61] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, “Generation and comprehension of unambiguous object descriptions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 11–20.
- [62] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, “Modeling context in referring expressions,” in *European Conference on Computer Vision*. Springer, 2016, pp. 69–85.
- [63] L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg, “Mattnet: Modular attention network for referring expression comprehension,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1307–1315.
- [64] X. Liu, Z. Wang, J. Shao, X. Wang, and H. Li, “Improving referring expression grounding with cross-modal attention-guided erasing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1950–1959.



- [65] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, “Natural language object retrieval,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4555–4564.
- [66] J. Li, Y. Wei, X. Liang, F. Zhao, J. Li, T. Xu, and J. Feng, “Deep attribute-preserving metric learning for natural language object retrieval,” in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 181–189.
- [67] C. Gan, Y. Li, H. Li, C. Sun, and B. Gong, “Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1811–1820.
- [68] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, “Tell me where to look: Guided attention inference network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9215–9223.
- [69] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, “Visual7w: Grounded question answering in images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4995–5004.
- [70] K. Chen, R. Kovvuri, and R. Nevatia, “Query-guided regression network with context policy for phrase grounding,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 824–832.
- [71] C. Deng, Q. Wu, Q. Wu, F. Hu, F. Lyu, and M. Tan, “Visual grounding via accumulated attention,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7746–7755.

- [72] B. A. Plummer, P. Kordas, M. Hadi Kiapour, S. Zheng, R. Piramuthu, and S. Lazebnik, “Conditional image-text embedding networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 249–264.
- [73] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, “Generation and comprehension of unambiguous object descriptions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 11–20.
- [74] R. Yeh, J. Xiong, W.-M. Hwu, M. Do, and A. Schwing, “Interpretable and globally optimal prediction for textual grounding using image concepts,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1912–1922.
- [75] X. Chen, L. Ma, J. Chen, Z. Jie, W. Liu, and J. Luo, “Real-time referring expression comprehension by single-stage grounding network,” *arXiv preprint arXiv:1812.03426*, 2018.
- [76] A. Sadhu, K. Chen, and R. Nevatia, “Zero-shot grounding of objects from natural language queries,” *arXiv preprint arXiv:1908.07129*, 2019.
- [77] Z. Yang, B. Gong, L. Wang, W. Huang, D. Yu, and J. Luo, “A fast and accurate one-stage approach to visual grounding,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4683–4693.
- [78] J. Liu, L. Wang, and M.-H. Yang, “Referring expression generation and comprehension via attributes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4856–4864.
- [79] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

- [80] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, “Visual genome: Connecting language and vision using crowdsourced dense image annotations,” *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.
- [81] R. Kovvuri and R. Nevatia, “Pirc net: Using proposal indexing, relationships and context for phrase grounding,” in *Asian Conference on Computer Vision*. Springer, 2018, pp. 451–467.
- [82] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, “Grounding of textual phrases in images by reconstruction,” in *European Conference on Computer Vision*. Springer, 2016, pp. 817–834.
- [83] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [84] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [85] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
- [86] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
- [87] A. Babenko and V. Lempitsky, “Aggregating deep convolutional features for image retrieval,” *arXiv preprint arXiv:1510.07493*, 2015.

- [88] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [89] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [90] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International conference on machine learning*, 2015, pp. 2048–2057.
- [91] H. J. Escalante, C. A. Hernández, J. A. Gonzalez, A. López-López, M. Montes, E. F. Morales, L. E. Sucar, L. Villaseñor, and M. Grubinger, “The segmented and annotated iapr tc-12 benchmark,” *Computer vision and image understanding*, vol. 114, no. 4, pp. 419–428, 2010.
- [92] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [93] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [94] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

- [95] L. Wang, Y. Li, and S. Lazebnik, “Learning deep structure-preserving image-text embeddings,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5005–5013.
- [96] K. Chen, R. Kovvuri, J. Gao, and R. Nevatia, “Msrc: Multimodal spatial regression with semantic context for phrase grounding,” in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. ACM, 2017, pp. 23–31.
- [97] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [98] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *European conference on computer vision*. Springer, 2014, pp. 391–405.
- [99] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [100] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.
- [101] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue *et al.*, “An empirical evaluation of deep learning on highway driving,” *arXiv preprint arXiv:1504.01716*, 2015.

- [102] R. Burbidge, M. Trotter, B. Buxton, and S. Holden, “Drug design by machine learning: support vector machines for pharmaceutical data analysis,” *Computers & chemistry*, vol. 26, no. 1, pp. 5–14, 2001.
- [103] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
- [104] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, “An exploration of parameter redundancy in deep networks with circulant projections,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2857–2865.
- [105] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, “On random weights and unsupervised feature learning.” in *ICML*, vol. 2, no. 3, 2011, p. 6.
- [106] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.
- [107] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.
- [108] Z. Cai, X. He, J. Sun, and N. Vasconcelos, “Deep learning with low precision by half-wave gaussian quantization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5918–5926.
- [109] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” *arXiv preprint arXiv:1412.6115*, 2014.

- [110] M. Zhu and S. Gupta, “To prune, or not to prune: exploring the efficacy of pruning for model compression,” *arXiv preprint arXiv:1710.01878*, 2017.
- [111] M. Mathieu, M. Henaff, and Y. LeCun, “Fast training of convolutional networks through ffts,” *arXiv preprint arXiv:1312.5851*, 2013.
- [112] V. Y. Pan, *Structured matrices and polynomials: unified superfast algorithms*. Springer Science & Business Media, 2012.
- [113] T. Zhang, K. Zhang, S. Ye, J. Li, J. Tang, W. Wen, X. Lin, M. Fardad, and Y. Wang, “Adam-admm: A unified, systematic framework of structured weight pruning for dnns,” *arXiv preprint arXiv:1807.11091*, vol. 2, p. 3, 2018.

# Vita



Krittaphat Pugdeethosapol received his B.S. and M.S. degree in Computer Engineering from King Mongkut's University of Technology Thonburi, Bangkok, Thailand. He has completed the Ph.D. degree in the Department of Electrical Engineering and Computer Science, Syracuse University, in 2020. His research interests are in the fields of computer vision and autonomous system.