Visual Perception, Prediction and Understanding with Relations

by

Zheng Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2020 by the
Graduate Supervisory Committee:

Yu Cao, Chair
Chaitali Chakrabarti
Jae-sun Seo
Deliang Fan

ARIZONA STATE UNIVERSITY
December 2020
ABSTRACT

Rapid development of computer vision applications such as image recognition and object detection has been enabled by the emerging deep learning technologies. To improve the accuracy further, deeper and wider neural networks with diverse architecture are proposed for better feature extraction. Though the performance boost is impressive, only marginal improvement can be achieved with significantly increased computational overhead. One solution is to compress the exploding-sized model by dropping less important weights or channels. This is an effective solution that has been well explored. However, by utilizing the rich relation information of the data, one can also improve the accuracy with reasonable overhead.

This work makes progress toward efficient and accurate visual tasks including detection, prediction and understanding by using relations. For object detection, a novel approach, Graph Assisted Reasoning (GAR), is proposed to utilize a heterogeneous graph to model object-object relations and object-scene relations. GAR fuses the features from neighboring object nodes as well as scene nodes. In this way, GAR produces better recognition than that produced from individual object nodes. Moreover, compared to previous approaches using Recurrent Neural Network (RNN), GAR’s light-weight and low-coupling architecture further facilitate its integration into the object detection module.

For trajectories prediction, a novel approach, namely Diverse Attention RNN (DAT-RNN), is proposed to handle the diversity of trajectories and modeling of neighboring relations. DAT-RNN integrates both temporal and spatial relations to improve the prediction under various circumstances.

Last but not least, this work presents a novel relation implication-enhanced (RIE) approach that improves relation detection through relation direction and implication. With the relation implication, the SGG model is exposed to more ground truth
information and thus mitigates the overfitting problem of the biased datasets. Moreover, the enhancement with relation implication is compatible with various context encoding schemes.

Comprehensive experiments on benchmarking datasets demonstrate the efficacy of the proposed approaches.
ACKNOWLEDGMENTS

It has been my great pleasure and fortune to work with Prof. Yu (Kevin) Cao. I would like to express my sincere gratitude to Kevin, for his excellent mentorship, support, and inspiration. Kevin encourages me in the face of failures and difficulties, helps me in improving writing and speaking skills, and gives me the freedom to explore diverse research topics.

I would also like to thank my dissertation committee members, Prof. Chaitali Chakrabart, Jae-sun Seo and Deliang Fan for their time and effort to serve on my committee. I really appreciate their insightful feedback and comments throughout this process.

I would also like to show my appreciation for Jinhui, Wei for their management and guidance during my internship with Mobai Tech. I also deeply appreciate Le for his great mentorship and collaborations when I was an intern at Facebook.

I also meet amazing new colleagues and friends in Arizona State University. To name some: Abinash Mohanty, Ang Chen, Ankita Bansal, Bin Dong, Deepak Kadetotad, Devyani Patra, Gokul Krishnan, Gopikrishnan Raveendran Nair, Hidayatullah Chowdary, Jingbo Sun, Shihui Yin, Shreyas Venkataramanaiah, Srivatsava Gorthy, Yufei Ma, Zhenyu Wang, Zhezhi He, Zihan Xu. I cannot mention them all by name, but I thank you all the same.

Finally, I would like to dedicate this thesis to my family. Pursuing a PhD degree and career across the Pacific Ocean is a big challenge despite modern technologies on connection and communication. Special thank to my fiancee Xiaocong for the years of support in every battles with big challenges and difficulties.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2  BACKGROUND</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Convolutional Neural Network</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Layers of CNN</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 Loss Function</td>
<td>14</td>
</tr>
<tr>
<td>2.1.3 Optimization</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Graph Convolutional Network</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Derivation of GCN</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2 Computation of GCN</td>
<td>21</td>
</tr>
<tr>
<td>2.2.3 Application of GCN</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Recurrent Neural Networks</td>
<td>23</td>
</tr>
<tr>
<td>2.3.1 Simple RNN</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2 LSTM &amp; GRU</td>
<td>26</td>
</tr>
<tr>
<td>3  GAR: GRAPH ASSISTED REASONING FOR OBJECT DETECTION</td>
<td>28</td>
</tr>
<tr>
<td>3.1 Related Work</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Contextual Methods for Object Detection</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2 Graph on Neural Networks</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Approach</td>
<td>30</td>
</tr>
<tr>
<td>3.2.1 Relation Modeling</td>
<td>30</td>
</tr>
<tr>
<td>3.2.2 GAR Design</td>
<td>33</td>
</tr>
<tr>
<td>3.3 Experiments</td>
<td>36</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3.1 Overall Performance</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2 Design Analysis and Ablation Study</td>
<td>39</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>45</td>
</tr>
<tr>
<td>4 DAT-RNN: TRAJECTORY PREDICTION WITH DIVERSE ATTENTION</td>
<td>47</td>
</tr>
<tr>
<td>4.1 Related Work</td>
<td>47</td>
</tr>
<tr>
<td>4.1.1 Time Sequence Prediction</td>
<td>47</td>
</tr>
<tr>
<td>4.1.2 Mutual Relation Modeling</td>
<td>47</td>
</tr>
<tr>
<td>4.1.3 Attention Model</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Approach</td>
<td>49</td>
</tr>
<tr>
<td>4.2.1 Diversity-Aware Memory</td>
<td>51</td>
</tr>
<tr>
<td>4.2.2 Anomaly Attention Module</td>
<td>53</td>
</tr>
<tr>
<td>4.2.3 Temporal-spatial Fusion and Prediction</td>
<td>55</td>
</tr>
<tr>
<td>4.2.4 Implementation Details</td>
<td>57</td>
</tr>
<tr>
<td>4.3 Experiments</td>
<td>57</td>
</tr>
<tr>
<td>4.3.1 Quantitative Evaluation</td>
<td>58</td>
</tr>
<tr>
<td>4.3.2 Qualitative Evaluation</td>
<td>61</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>62</td>
</tr>
<tr>
<td>5 SCENE GRAPH GENERATION ENHANCED BY RELATION IMPLICATION</td>
<td>63</td>
</tr>
<tr>
<td>5.1 Related Work</td>
<td>63</td>
</tr>
<tr>
<td>5.1.1 Scene Graph Generation</td>
<td>63</td>
</tr>
<tr>
<td>5.1.2 Multi-label Learning</td>
<td>64</td>
</tr>
<tr>
<td>5.1.3 Knowledge Graph Embedding</td>
<td>64</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>5.2</td>
<td>Method ........................................ 66</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Problem Definition .............................. 66</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Model Architecture ............................. 67</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Multi-Label Relation Implication Module ........ 69</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Visual Rotation Embedding Module ............... 71</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiments ...................................... 73</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Datasets and Metrics ............................ 73</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Implementation ................................... 75</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Overall Performance .............................. 76</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Ablation Study .................................. 77</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Computational Overhead ........................... 78</td>
</tr>
<tr>
<td>6</td>
<td>CONCLUSION AND FUTURE WORK ..................... 79</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusion ....................................... 79</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Work ...................................... 80</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>.............................................. 81</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Detection on PASCAL VOC 2007, Trained on VOC 2007 and VOC 2012 Trainval Combined. Abbreviation: Fast R-CNN (FS-N) (Girshick, 2015), Faster R-CNN (FR-N), Aeroplane (arpl.), Bottle (bot.), Chair (chr.), Table (tbl.), Horse (hrs.), Motorbike (mbk.), Person (prs.), Plant (plt.), Sheep (shp.), Train (trn.), Tvmonitor (tvm.).</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Detection Results on PASCAL VOC 2012, Trained on VOC 2007 Trainval, 2012 Trainval and 2007 Test Combined.</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Detection on COCO 2014 Test-dev.</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Performance on VOC 2007 Validation Set Using Different $K$ for Scene Nodes Selection.</td>
<td>40</td>
</tr>
<tr>
<td>3.5</td>
<td>Top 3 Related Object/scene Entities in Terms of Co-occurrence, on VOC 2007 Trainval. From Top to Bottom: Three Categories of Mostly Co-occurred Objects, the Indoor/outdoor Label, Three Categories of Mostly Co-occurred Places and Three Mostly Related Scene Attributes.</td>
<td>41</td>
</tr>
<tr>
<td>3.6</td>
<td>Ablative Comparison with SIN on VOC 2007 Test, Trained on VOC 2007 Trainval. Abbreviation: Edge Module (E), Scene Module (S).</td>
<td>42</td>
</tr>
<tr>
<td>3.7</td>
<td>Number of FLOPS and Number of Parameters Required by GAR and SIN Modules.</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>ADE and FDE Comparison on the Five Datasets</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Number of Floating-point Operations (FLOPS) and Number of Parameters Required by Social Attention Model, the Proposed DAT-RNN as Well as DAM and AAM Decomposition.</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>SGG Performances of Various Methods.</td>
<td>74</td>
</tr>
</tbody>
</table>
5.2 Ablative Comparison of MLRI and VRotatE. MTF Is the Abbreviation of MotifNet. ......................................................... 76

5.3 Number of Floating-point Operations (FLOPS), Number of Parameters and Frame per Second (FPS) Required by MLRI and VRotatE. . . . 76
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 (a) Two Cows on the Grassland. (b) A Ship at the Seaside.</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Diverse Trajectories with Different Speed, Direction and Mutual Relation.</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Diversity of Relations: (a) Similar Visual Appearances of Objects with Different Relations; (b) Drastically Different Visual Appearances, Which Are Considered the Same Relation Class ‘sitting-on’</td>
<td>6</td>
</tr>
<tr>
<td>1.4 An Image Showing “man Holding Phone” with a Scene Graph. Positive and Negative Relation Implication in the Format of Multi-graph.</td>
<td>7</td>
</tr>
<tr>
<td>2.1 (a) A CNN in Feedforward Path, (b) Convolution in Feedforward Path, (c) Transposed Convolution in Back Propagation.</td>
<td>12</td>
</tr>
<tr>
<td>2.2 (a) Samples of Neighboring Connections with the First and Second Neighbors, (b) Aggregating Features from Neighbors, (c) Further Processing on the Fused Node Feature</td>
<td>16</td>
</tr>
<tr>
<td>2.3 (a) Recurrent Connection in One RNN Layer, (b) Unfold RNN Layer in Time Steps</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Data Flow of a LSTM Cell</td>
<td>26</td>
</tr>
</tbody>
</table>
3.1 (a) The Architecture of GAR. RPN and Associated Modules Give \( N \) Proposals (Each with a Size of \( F \)). The CRCN Generates \( N \) Cursory Class Scores (Each with a Size of \( O \)) and \( N \) Bbox Regression with a Size of 4. Meanwhile, the Scene Detector Generates Scene-related Labels with a Size of 469. Scene Pooling Picks and Concatenates \( S \) Scene Nodes from 469 of Them, Which Are Generated from Scene Embedding Module. (b) The GCR Module Merges the Instance Subgraph and the Scene Subgraph as a Heterogenous Graph. Nodes Are Proposal Features \( ins_1, \ldots, ins_N \) and Selected Scene Embeddings \( scn_1, \ldots, scn_S \). Edges Are Created from the Normalized Co-occurrence Matrix That Are Elaborated in the Following Section. .......................... 30

3.2 Qualitative Results of GAR Detection. From Top to Bottom: GAR Detection, Baseline Detection, the Indoor/outdoor Label, Place Categories with Possibilities, Scene Attributes. .............................. 43


x
4.1 (a) Illustration of the Spatial-temporal Graph, Where Temporal Edges Are Uni-directional Edges While Spatial Edges Are Bi-directional Edges. (b) Overview of the DAT-RNN: the DAM Extracts Individual Temporal Relation and the AAM Extracts Mutual Spatial Relation, the Attention Module Merges These Relation Information and the Node-RNN Makes the Final Prediction. .................................................. 50

4.2 Illustration of Detour Integral. (a) Various Trajectories with Different Detour Integral, (b) Detour Integral Calculation, (c) Path Integral Calculation. .............................................................. 52

4.3 Illustration of Temporal-spatial Fusion and Node Embedding............. 56

4.4 Comparison Between Our Model (DAT) and Previous Work, Social Attention (SA). The Ground Truth Trajectories Are in Green Color. Here We Only Showcase Several Representative Trajectories in Each Scenario. In Easy Cases of Straight Trajectories, Both Methods Can Predict Pretty Well. But Our Methods Performs Better for Detouring Trajectories.......................................................... 59

4.5 Anomaly Salience and Final Attention Weights for Target Node Are in Red Color. Anomaly Salience Is the Normalized Number While Attention Weights Are Circles Whose Radius Implies the Magnitude. . . 59

5.1 Illustration of (a) Translation Embedding with Problematic “near” Relation, (b) Rotation Embedding Handling the Symmetry Relation “near”. 65

5.2 Overview of the Scene Graph Generation Pipeline. ......................... 66
5.3 Relation Implication Visualization on Visual Genome Dataset, Nodes Are Relations, Weight Numbers on the Directed Edges Are the Conditional Ratio: (a) Positive Implication; (b) Negative Implication; (c) Multi-edge Ratio of Selected 50 Relations. ........................................ 69

5.4 Qualitative Results of RIE, Comparing with MotifNet Baseline on the Task PRDCLS. From Left to Right: Input Images, Predicate Inference of MotifNet and Predicate Inference of RIE. Green Edges Are True Positive Results, Red Edges Are False Positive Results. ................. 73
Chapter 1

INTRODUCTION

Recently, significant development of object detection has been witnessed thanks to the advance in deep Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; He et al., 2015a). The current object detection methods (Dai et al., 2016; Galleguillos et al., 2008; He et al., 2014; Ren et al., 2015) mostly follow the philosophy of anchor or region proposal introduced by R-CNN (Girshick, 2015). In these approaches, object classification and bounding box (bbox) regression are performed either on selected proposals that are generated from predefined anchors (Dai et al., 2016; Ren et al., 2015), or on these anchors (or prior boxes) (Liu et al., 2016; Redmon et al., 2015; Lin et al., 2017) directly. Recently, anchor-free methods are also a set of emerging solutions that achieve admirable performance on the detection of multi-scale, movement blurred and heavily occluded objects (Huang et al., 2015; Kong et al., 2019).

As revealed by psychological investigations (Bar, 2004; Oliva and Torralba, 2007), contextual information including scene context and object relationships plays a crit-

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{(a) Two Cows on the Grassland. (b) A Ship at the Seaside.}
\end{figure}
ical role in human capability of recognizing objects.

Take Figure 1.1a as an example. There are two cows on the grassland. Implied by the blue sky and shadows, the scene is outdoor rather than indoor with the green mat. The white objects on the grass can be recognized as cows, or closely, goats. However, the white cloud should not be wrongly detected as cows or sheep, even though an individual cloud may look like a cow or a sheep. The most obvious reason to exclude that is the blue sky because of its background on the water. Figure 1.2b is another good example. The manmade object is probably a ship instead of car or train since it is on the water. Studies in the computer vision community have attempted to boost performance of object detection by utilizing contextual information. For instance, previous studies (Carbonetto et al., 2004; Divvala et al., 2009; Galleguillos et al., 2008; Marszałek et al., 2009; Mottaghi et al., 2014) suggest that one can leverage the modeling of implicit context or explicit relation in recognition algorithms. Specifically, what categories of objects may appear in a specific scene, or what category of objects may appear simultaneously with another category of objects. However, most of the methods were proposed before the popularity of deep learning and have not been well explored in modern CNNs on object detection tasks. One of the challenges in relation modeling is the computation complexity due to the significant variations in the quantity and category of objects across different images. Another challenge is to efficiently encode and process the object relation into a CNN-based object detector.

In the recent past, Graph Convolutional Network (GCN) has been successfully applied to node classification on graph-structured data (Kipf and Welling, 2016), such as the citation network, text classification (Yao et al., 2018a) and some other Natural Language Processing (NLP) tasks including translation and semantic role identification (Marcheggiani and Titov, 2017; Bastings et al., 2017). By aggregating information from neighboring nodes, GCN produces better inference than merely
taking the features from an individual node (Kipf and Welling, 2016). This intrinsic property of GCN provides it unique advantages in handling entities with relations.

Motivated by the property of GCN, we propose a novel approach, namely Graph Assisted Reasoning (GAR), to improve the efficacy of object detection. In the graph of GAR, the object nodes are regional features generated by Regional Proposal Network (RPN) and Region of Interest (ROI) pooling. And the scene nodes are embeddings extracted from the entire image by the scene detector. The edge between two object nodes is created with object-object co-occurrence, while the edge between an object node and a scene node is built with object-scene co-occurrence. As a result, GAR generates the node scores, which act as refine items for the basic one-layer object classifier, suppressing abnormal object candidates and amplifying probable ones, leading to more reliable object detection.

To summarize, contributions of the proposed GAR are as follows:

• We propose a novel graph-assisted reasoning approach, GAR, that leverages GCN for object detection.

• The proposed GAR is an extendable scheme that encodes relations into an adjacent matrix for object detection.

• Besides the co-occurrence relation, other relations such as spatial relations and higher level semantic relations can also be incorporated into the GAR architecture.

Meanwhile, the fast development of computer vision and deep learning techniques have enabled emerging applications in robotics and autonomous vehicles. Successful trajectories prediction plays an important role in autonomous vehicles (Firl et al., 2012; Gupta et al., 2018). For example, if a self-driving car predicts an overlapping trajectory of itself and a person, it should slow down, yield, or even stop immediately
However, trajectory prediction is facing two main challenges: diversity and modeling of mutual relation, shown in Figure 1.2. Diversity refers to the fact that trajectories differ from each other by various speed, direction at different time stamps.

On the other hand, the motion of each object not only depends on itself but also depends on the objects around it. Thus, it isn’t straightforward to model this dynamic system precisely by including interactions among different objects. There has been some previous attempts (Helbing and Molnár, 1995; Alahi et al., 2016) solving the bottlenecks mentioned above.

**Diversity.** Social-GAN (Gupta et al., 2018) is the pioneering work explicitly tackling the multi-modal problem by using Generative Adversarial Network (GAN) to predict socially acceptable multi-modal trajectories. However, GAN is suffering from high computational cost and difficulty in training. Also, the traffic agent cares more about the most likely prediction rather than several possible predictions, which lead to a more complicated decision space. Due to diversity, it is hard to determine the type and property of a complex trajectory. However, from a micro perspective, the object trajectory consists of straight and detouring segments. We can make more
reliable predictions on the more straightforward segments then get the fused final results rather than using a single model to predict diverse and complex trajectories. This observation inspires the design of diversity-aware memory (DAM) in this work.

**Mutual relation.** Alahi *et al.* (2016); Gupta *et al.* (2018) leverage spatially locality to catch the relations between different interacting people. However, this method is based on the assumption that only movements in a constant local neighborhood affect each other. Such an assumption may not be accurate in real applications. Instead, we argue that sudden or abnormal movements should also receive attention. For example, a car that suddenly turns toward a person should raise more attention than a car that is running in parallel with a person. Accordingly, we design the anomaly attention module (AAM) to emulate this mechanism.

The contribution of the proposed DAT-RNN is summarized as: to tackle the diversity and mutual relation modeling challenge in trajectory prediction, we propose diverse attention RNN (DAT-RNN) with two innovative modules, *i.e.* diversity-aware memory (DAM) and anomaly attention module (AAM).

- DAM is a RNN-based module which tackles the diverse segmentation and fusion for complex trajectories in an automatic way.

- AAM is an anomaly attention-based scheme that leverages a well-designed anomaly salience as an enhancing feature for spatial attention module to improve the mutual relation capturing.

- Finally, we demonstrate that our model, DAT-RNN, outperforms the state-of-the-art approaches in prediction accuracy on human trajectories on two publicly available real-world crowd datasets. We also perform an ablation study to analyze the efficacy of two proposed modules.
In high-level visual tasks such as image captioning, text-to-image synthesis (Reed et al., 2016), and visual question answering (VQA) (Antol et al., 2015), the accurate identification of individual objects lays the foundation of the task performance. On the other hand, explicitly modeling visual contexts beyond the object-level perception has demonstrated efficacy in such visual tasks (Yao et al., 2018b; Teney et al., 2017). Visual relations naturally bridge the low-level visual features and the high-level language guided semantics by placing objects in a semantic context indicating how objects interact with each other (Zhang et al., 2017). For example, as shown in Figure 1.3a, if we can detect boy ride skateboard successfully, then answering...
the question what is the boy riding? will be straightforward and interpretable through incorporating more reasoning and mitigating data-dependent bias (Zellers et al., 2018; Zhang et al., 2017).

Scene graph generation (SGG) aims to provide explicit visual clues upon individual object detectors. The resulting scene graphs are a set of triplets in the format of <subject-predicate-object> where instances (subjects/objects) are nodes and their interactions (relation/predicate) are directed edges. Since the relation is the combination of objects and predicates, relation samples are much more sparse, leading to a much more long-tailed distribution than that of objects. Given $N$ objects and $R$ predicates, the model is required to learn $O(N^2R)$ relations with limited examples (Ramanathan et al., 2015; Sadeghi and Farhadi, 2011a). Currently, a widely-adopted strategy is to divide the learning into object classification and the subsequent relation classification (Lu et al., 2016). Most methods first classify the objects separately and then apply local graph structures to learn contextual object representations for relation classification (Chen et al., 2019; Qi et al., 2019; Xu et al., 2017).
However, it is observed that relations are subtly varying in two aspects: (1) similar or even identical subject-object pairs may have different relations. For example, a boy can either ride, hold or sit on a skateboard, as shown in Figure 1.3a. (2) The same relation may occur between different subjects and objects. For example, the actions taken by the dog, cat, man, and girl to different kinds of hats (and helmets) are all wear, as illustrated in Figure 1.3b. As a result, to distinguish different relations within a similar context and to generalize the relation for distinct contexts at the same time become a key bottleneck for the SGG task.

Besides, the multi-label property of relations makes the tasks more challenging. For example, as Figure 1.4 shows, hand holding phone implies that phone in hand is also a valid truth though the in relation is not annotated in the image. Such a problem happens in many images in a dataset. Similarly, the relation wearing of the person-shirt pair indicates a relation in or being worn by for the reversed pair shirt-person, rather than no relation that simply ignores the potential relation information in the current practice.

To mitigate the obstacles mentioned above, we propose a novel relation implication-enhanced (RIE) approach for SGG, which consists of a multi-label relation implication (MLRI) module and a visual rotation embedding (VRotatE) module. Both modules act as auxiliary components of existing scene graph generators, following the four-step (visual feature extraction, graph construction, context encoding, and relation classification) pipeline. The MLRI module is designed to harvest the ground truth information from the unlabeled edges. VRotatE module, on the other hand, makes the relation detection interpretable and extensible by modeling the relation as a vector rotating.

In summary, the contributions of our RIE method are as follows:

- We propose a novel relation implication-enhanced (RIE) approach for the scene
graph generation.

- We propose auxiliary modules, namely multi-label relation implication (MLRI) and visual relation learning (VRotatE), to leverage the ground truth information and integrate entity embedding with knowledge transfer.

- The proposed RIE approach is extendable and can be embedded into various scene graph generator with various message propagation schemes to improve the SGG performance.
Chapter 2

BACKGROUND

There have been comprehensive research and investigation about neural networks since McCulloch and Pitts (1943). As an interdisciplinary topic, neural networks or more precisely, artificial neural networks are defined in various ways. Here we adopt the classic definition: neural networks are networks composed by adaptive simple units (neuron) whose organization can mimic activity of the neural system for given stimuli (Kohonen, 1988). In other words, neural networks is contained by the intersection of machine learning and biology.

Inspired by the biological functioning of neurons in a brain, neural networks models the stacking accumulation of excitation/inhibition mathematically, i.e. M-P model. In the M-P model, a neuron receives weighted signals from $n$ input neurons, then this neuron will generate a output signal through a non-linear activation function such as Sigmoid function (Han and Moraga, 1995) and Rectified Linear Unit (ReLU) (Hahnloser and Seung, 2001). The output signal acts as a input signal for the next neuron till the final output. This layer by layer stacking organization characterizes the multi-layer feed-forward neural networks.

With the emerging of big data and powerful computing platforms, neural networks with more and more layers achieve much better performance than the conventional statistical models, indicating the approaching of the Deep Learning Era. A plausible interpretation for the power of deep networks is that the multi-layer stacking mechanism performs better feature extraction from the low-level raw input to the high-level task-oriented output feature.

People designed different networks for different feature extraction and different
tasks. For example, convolutional layers are usually employed for image processing tasks as the regularly patterned locality of image data, Graph Convolutional Networks (GCN) is designed to process structured data, and Recurrent Neural Networks (RNN) and attention models are used for sequential data to memorizing the historical state.

In this chapter, we will discuss some of the key concepts related to deep learning. We will first discuss the major structural components and the computational flow of a Convolutional Neural Network (CNN) for static images. After that, we will look at the architecture of the RNN and attention model for sequential data. Then we will briefly discuss some popular and high performing networks used in applications such as image classification, object recognition, and sequence prediction.

2.1 Convolutional Neural Network

A CNN has layered structures with current layers feeding data to successive layers as shown in Figure 2.1(a). These layers usually have different types and shapes in different feature extraction stage. In this section, we will look in detail on some of the most used layers used in the CNNs.

2.1.1 Layers of CNN

**Convolutional Layer** As the fundamental operation of CNN, convolution constitutes the most majority of the total operations in the modern CNNs (Krizhevsky et al., 2012). It involves 3-dimensional (width, height, input channel) multiplication and accumulation of N if input features with \(K \times K\) convolution filters to get an output feature neuron value as shown in Equation 2.1 and illustrated in Figure 2.1(b).

\[
O(b, c_o, w, h) = \sum_{c_i=0}^{N_i} \sum_{k_w=0}^{N_w} \sum_{k_h=0}^{N_h} W(c_o, c_i, k_w, k_h) \cdot I(b, c_i, w + k_w, h + k_h)
\]  

(2.1)

where \(b\) represents the batch index of input/output feature maps. \(O(b, c_o, w, h)\)
and $I(b, c_i, w+k_w, h+k_h)$ represent the feature pixel at location $(w, h)$ in the $c_o$ output and $c_i$ input feature map respectively. $W(c_o, c_i, k_w, k_h)$ is the weight at position $(k_w, k_h)$ that convolves input feature mao and get the output feature map. Various padding on feature maps and striding of convolution kernels may be applied. In the error back-propagation during training, gradients of output feature map are used to get gradients of the input feature map by transposed convolution (T-Conv) as shown in Figure 2.1(c).

**Activation** A non-linear activation function is used to enhance the representing ability, otherwise, only linear transformation can be learned. Commonly used activation functions in early neural networks are tanh and sigmoid. Though they are mathematically continuous and differentiable. The floating-point computing requires more time and energy for training. And they are also suffering from gradient vanishing in deep networks due to the gradient chain rule. Hence, the Rectified Linear Unit,
ReLU (Nair and Hinton (2010)), defined as $y = \max(x,0)$ has become the popular activation function among CNN models as its good gradient maintenance in training. Moreover, ReLU has less computational complexity compared to exponent functions in tanh and sigmoid, also aiding hardware design. PReLU (He et al., 2015b) is similar to ReLU except that it has a learnable slope parameter.

**Normalization**  
Normalization on feature maps is a set of transformations to boost the performance of neural networks. Local Response Normalization is proposed to mimic the biological scheme of lateral inhibition of neural systems. It creates a competition strategy for neurons, amplifying the activation of more activated neurons and thus boosting the generalization of CNNs. Recently, Batch Normalization is widely adopted in modern CNNs for its easy deployment and significant performance. By normalizing and rescaling feature maps for each layer, gradient vanishing and exploding problems can be mitigated. Instance Normalization (Ulyanov et al., 2017) and Group Normalization (Wu and He, 2018) are also proposed for the application domain where the batch size is limited, such as image style transfer and segmentation.

**Pooling Layer**  
Spatial pooling or sub-sampling is utilized to reduce the feature dimensions by taking local statistics of a moving window, including two kinds of majorly used pooling, *i.e.*, max pooling, and average pooling. For max pooling, as shown in Equation 2.4, it computes the maximum of neighboring $K \times K$ features in the same feature map, which also provides a form of translation invariance (Boureau et al., 2010). Although max-pooling is popularly used, average pooling is also used in some CNN models (Boureau et al., 2010). In the case of average pooling, we do average operation instead of max of neighboring $K \times K$ neurons in the same feature map. Reducing the dimensionality of lower-level features while preserving
the important information, the pooling layer helps to abstract higher-level features without redundancy.

**Fully Connected Layer** Fully-connected layer or inner product layer is the classification layer where all the input features \( (N_i) \) are connected to all of the output features \( (N_o) \) through synaptic weights \( (W) \). Each output neuron is the weighted summation of all the input neurons as shown in Equation 2.2.

\[
O(b, o) = \sum_{i=0}^{N_i} W(o, i) \cdot I(b, i) \tag{2.2}
\]

2.1.2 Loss Function

The loss function is a measurement for the inconsistency between prediction and ground truth. The calculated loss (error) will be back-propagated to each layer for weights updating in the training phase.

For basic classification tasks, the outputs of the last layer of feature extraction, i.e., the fully connected layer are generally traversed through a softmax function (Goodfellow et al., 2016) that converts them to the probability of each class label in the range \((0, 1)\). For inference, it simply takes the class label with the largest probability as the output. For training, cross-entropy loss (Goodfellow et al., 2016) is applied as the measurement of the distance of inferred class distribution and ground truth distribution.

\[
L = \frac{1}{N} \sum_i L_i = \frac{1}{N} \sum_i - \sum_{c=1}^{M} y_{ci} \log(p_{ci}) \tag{2.3}
\]

Where M is the number of classes in the dataset, \( y_{ci} \) is the indication variable, being 1 if predicted class and ground truth class label are the same for sample \( i \), being 0 otherwise. \( p_{ci} \) is the inferred probability of class label \( c \) for sample \( i \).
For regression tasks wherein the output is continuous, Mean Square Error (MSE) is typically used to calculate the L2 distance between inference and the ground truth values.

For higher-level tasks such as object detection, both object localization (bounding box coordinates regression) and recognition (region of interest classification) are involved (Girshick, 2015; Ren et al., 2015). Therefore, the loss function of object detection networks is a combination of both MSE and cross-entropy with predefined weights. To further relieve the penalty of mis-localizing and the training stability, smooth L1 loss (Ren et al., 2015) is proposed as an amendment to MSE.

### 2.1.3 Optimization

The training of CNNs can be treated as an optimization problem that minimizes the loss from the applied loss function. Back-propagation is an effective method to calculate gradients of trainable parameters and lays the foundation of a set of optimizers including stochastic gradient descent (SGD) (Bottou, 2004; Bertsekas, 2008) and Adaptive Moment Estimation (Adam) (Kingma and Ba, 2017), etc.

As a basic but effective optimizer, SGD selects a subset training examples (mini-batch) from the training set \(x_1, x_2, \ldots, x_N\), and then computes their averaged gradients \(\Delta L(w_t)\) in iteration \(t\). For the next iteration \(t+1\), weights are updated in the direction of negative gradient:

\[
w_{t+1} = w_t - \alpha_t \Delta L(w_t) \tag{2.4}
\]

where \(\alpha_t\) is the learning rate at iteration \(t\).

There are potential issues of oscillation at saddle points for SGD. Therefore, people propose Adagrad (Duchi et al., 2011) and Adam which incorporate the idea of
Figure 2.2: (a) Samples of Neighboring Connections with the First and Second Neighbors, (b) Aggregating Features from Neighbors, (c) Further Processing on the Fused Node Feature

moments to accelerate training and increase the stability.

2.2 Graph Convolutional Network

The key operation of Graph Convolutional Network (GCN) (Defferrard et al., 2016; Kipf and Welling, 2016) is the vertex (node) convolution, i.e. fusing embedding features in the spatial domain as illustrated in Figure 2.2. Different from the image data used in standard CNNs, graph data have arbitrary structures with various edges and orders. Therefore, standard convolution cannot apply to graph data and we need to derive convolution operation in the spectral domain then transform back to the spatial domain.

2.2.1 Derivation of GCN

Inspired by the idea of Fourier Transformation (FT) between the spatial domain and the spectral domain, Graph Fourier Transformation (spatial domain $\rightarrow$ spectral domain) and Inverse Graph Fourier Transformation (spectral domain $\rightarrow$ spatial domain) are hereby defined (Bruna et al., 2013). More specifically, we will convert the node embeddings $f(i), i \in (1, \cdots, N)$ from the spatial domain to spectral domain
through Graph Fourier Transformation, then perform convolution with the kernel $h$ in the spectral domain, where $N$ is the number of nodes in the graph. Consequently, the convolved node embeddings can be obtained by Inverse Fourier Transformation, used for the next stage feature extraction or final inference. To understand the representation of graphs in the spectral domain, we will dive into some knowledge of Spectral Graph Theory (Bruna et al., 2013), where Laplacian Matrix is a powerful tool to study the topology of graphs.

**Fourier Transformation on graphs: spatial $\rightarrow$ spectral** First, we will recap the calculation of FT on continuous time domain, shown in Equation 2.5.

$$F(\omega) = \mathcal{F}[f(t)] = \int f(t)e^{-i\omega t} dt$$  \hspace{1cm} (2.5)

which is the indefinite integral of signal function $f(t)$ and basis function $e^{-i\omega t}$. Wherein $e^{-i\omega t}$ is subject to:

$$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2} e^{-i\omega t} = -\omega^2 e^{-i\omega t}$$  \hspace{1cm} (2.6)

here $\Delta$ is the Laplacian, i.e. a second order differential $\Delta f = \nabla^2 f$, a transformation applied to function $f$. Compared to the property of the generalized eigenvalue: $Au = \lambda u$, where $A$ is the matrix under study, mathebfu is an eigenvector paired with eigenvalue $\lambda$, we can naturally treat the Laplacian $\nabla$ as a transformation matrix like $A$, $e^{-i\omega t}$ is the eigen-function of $\nabla$ and $\omega$ is related with the corresponding eigenvalue.

For FT on graph data (discrete data), once we have the Laplacian Matrix (LM)(discrete version of $\nabla$), eigen vectors $u_1, u_2, \ldots, u_n$ can be obtained by eigen decomposition. Then the FT on graphs is formulated as:
\[ F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i) u_l(i) \] (2.7)

where \( f \) is the transformation on nodes with the total number of \( N \) in the graph, its output is an embedding of a node, \( u_l(i) \) means the \( i-th \) element of the \( l-th \) eigenvector.

Equation 2.7 can also be expressed in form of matrices:

\[
\begin{pmatrix}
\hat{f}(\lambda_1) \\
\hat{f}(\lambda_2) \\
\vdots \\
\hat{f}(\lambda_N)
\end{pmatrix}
= \begin{pmatrix}
u_1(1) & u_1(2) & \ldots & u_1(N) \\
u_2(1) & u_2(2) & \ldots & u_2(N) \\
\vdots & \vdots & \ddots & \vdots \\
u_N(1) & u_N(2) & \ldots & u_N(N)
\end{pmatrix}
\begin{pmatrix}
f(1) \\
f(2) \\
\vdots \\
f(N)
\end{pmatrix}
\] (2.8)

In short, we have \( \hat{f} = U^T f \), i.e. given input embedding \( f \), left multiplication with \( U^T \) leads to the FT transformed embedding \( \hat{f} \).

**Inverse Fourier Transformation on graphs: spectral \to spatial**  
Compared to the original expression of Inverse Fourier Transformation (IFT):

\[
\mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\pi} \int F(\omega) e^{i\omega t} d\omega
\] (2.9)

we can have the analogy on graph (discrete spatial domain):

\[
f(i) = \sum_{l=1}^{N} \hat{f}(\lambda_l) u_l(i)
\] (2.10)

and convert it in form of matrices:
\[
\begin{pmatrix}
  f(1) \\
  f(2) \\
  \vdots \\
  f(N)
\end{pmatrix} =
\begin{pmatrix}
  u_1(1) & u_2(1) & \cdots & u_N(1) \\
  u_1(2) & u_2(2) & \cdots & u_N(2) \\
  \vdots & \vdots & \ddots & \vdots \\
  u_1(N) & u_2(N) & \cdots & u_N(N)
\end{pmatrix}
\begin{pmatrix}
  \hat{f}(\lambda_1) \\
  \hat{f}(\lambda_2) \\
  \vdots \\
  \hat{f}(\lambda_N)
\end{pmatrix}
\]

(2.11)

In short, we have \( f = U \hat{f} \), i.e. given embedding \( \hat{f} \) in spectral domain, embedding \( f \) in spatial domain can be obtained by left multiplication with \( U \).

**Lapalacian in Spectral Graph Theory** For a undirected graph \( G = (V, E) \), its Lapalacian Matrix (LM) is defined as \( L = D - A \), where \( D \) is the degree matrix, \( A \) is the adjacent matrix.

There are also some extension of LM:

- \( L^{sys} = D^{-1/2} \cdot L \cdot D^{1/2} \), referred as Symmetric Normalized Laplacian (Banerjee and Jost, 2008), which is prevalently used in GCNs.

- \( L^{rw} = D^{-1} \cdot L \), referred as Random Walk Normalized Laplacian, which is typically used in Random Walk algorithms (Aldous and Fill, 1995) for graph mining.

**Spectral Decomposition** As the fact that \( D \) is symmetric for undirected graphs, \( L \) is also a symmetric matrix which can be applied on Eigen Decomposition (Spectral Decomposition), gives us:

\[
L = U \begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_n
\end{pmatrix} U^{-1}
\]

(2.12)
Graph Convolution  First, check out the convolution theorem and IFT:

\[ f(t) \ast h(t) = \mathcal{F}^{-1}[\hat{f}(\omega)\hat{h}(\omega)] = \frac{1}{2\pi} \int \hat{f}(\omega)\hat{h}(\omega)e^{i\omega t} d\omega \quad (2.13) \]

where \( \ast \) represents the convolution operator, thus convolution of embedding \( f \) and convolution kernel \( h \) (in CNN, kernel is a matrix, representing a linear transformation on image data) on graph can be derived with:

\[
\begin{align*}
\hat{f} &= U^\top f \\
\hat{h} &= U^\top h = \begin{pmatrix} \hat{h}(\lambda_1) \\ \vdots \\ \hat{h}(\lambda_n) \end{pmatrix}, \quad \hat{h}(\lambda_i) = \sum_{i=1}^{N} h(i)u_i^*(i) \\
\hat{f} \hat{h} &= U^\top h \odot U^\top f = \begin{pmatrix} \hat{h}(\lambda_1) \\ \vdots \\ \hat{h}(\lambda_n) \end{pmatrix} U^\top f \\
(f \ast h)_G &= U \begin{pmatrix} \hat{h}(\lambda_1) \\ \vdots \\ \hat{h}(\lambda_n) \end{pmatrix} U^\top f
\end{align*}
\]

(2.14)

where \( U = (u_1, u_2, \cdots, u_n) \) is the matrix composed by unit column eigen vectors \( u_i, i \in (1, \cdots, n) \). While \( \lambda_i, i \in (1, \cdots, n) \) is the corresponding eigen values. Moreover, \( U \) is a orthogonal matrix, thus we have \( U \cdot U^T = I \) and \( U^T = U^{-1} \).

To sum up, convolution is essentially a kind of aggregation of embeddings of neighbor nodes. For CNN on image data, the convolution is on regularly patterned graph. To tackle the similar aggregation on arbitrary structured graph, we introduce the FT, IFT and convolution theorem, convert the embeddings from spatial to spectral domain, perform convolution in spectral domain with matrices multiplication, and get
the final convolved embeddings by IFT.

2.2.2 Computation of GCN

The graph convolution defined by:

\[
(f \star h)_g = U \begin{pmatrix} \hat{h}(\lambda_1) \\ \vdots \\ \hat{h}(\lambda_n) \end{pmatrix} U^T f
\]  

(2.15)

lays the foundation of “convolutional layer” in the first generation of graph-based network (Gori et al., 2005; Scarselli et al., 2009) which takes \(\hat{h}(\lambda_1), \ldots, \hat{h}(\lambda_n)\) as trainable parameters. Though it seems that this method is simple, the required eigen-decomposition has complexity of \(O(n^3)\), which is unacceptable when the number of nodes is large.

A lot of efforts have been taken to reduce the computational complexity by introducing additional assumption and approximation, such as Chebyshev polynomial in Defferrard et al. (2016). Moreover, in the well known GCN model (Kipf and Welling, 2016), the authors derive the simple yet effective propagation equation:

\[
Y = \sigma \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \right)
\]  

(2.16)

where \(X \in \mathbb{R}^{N \times C}\) with \(N\) nodes and \(C\) features for each node is the input node matrix, \(\Theta \in \mathbb{R}^{C \times F}\) is the parameter matrix of filters, \(Y \in \mathbb{R}^{C \times F}\) is the convolved node matrix. Normalized degree matrix is calculated as: \(\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}, \tilde{A} = A + I_N\).
2.2.3 Application of GCN

By taking both nodes feature and graph topology in to account, GCN shows advantages such accuracy and robustness over purely individual processing and traditional graph embedding methods. There have been a lot of successful application in multiple domains.

**Text Classification** Kipf and Welling (2016) builds a document graph based on reference relationship. It predicts the class for unlabeled document with unsupervised learning. While Yao et al. (2018a) constructs a heterogenous graph with both documents and words. There is no edge between documents, edges between words and documents are based on occurrence, edges between words are based on words co-occurrence within a predefined window. This method shows better robustness when labeled data is limited.

**Translation** In the application of translation, GCN is typically used as a part of encoding. In Marcheggiani et al. (2018), the graph is constructed with dependency tree while in Marcheggiani et al. (2018), the graph is constructed using semantic role parsing.

**Realtion Extraction** More tweaks and amendment are adopted in relation extraction. In Zhang et al. (2018), the graph convolution is calculated by: \( D^{-1}AXW \) instead of the standard equation in Kipf and Welling (2016). It takes GCN as embedding module and predict the relation label among neighboring nodes. In Guo et al. (2019), the authors takes soft attention weights among nodes as edges in stead of standard adjacent matrix.

In additional, GCN is also applied in event identification (Nguyen and Grishman,
2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) refer to a set of artificial neural networks with recurrent connection which enables the sequential modeling, as shown in Figure 2.3. In a RNN, the state of a hidden layer depends on both its historical state and the input. Compared to the traditional statistical models like Hidden Markov Model (HMM) (Baum and Petrie, 1966), RNN can contain larger state size and have lower computational overhead. Thus, RNN receives more and more research and investigation since its advantages.

2.3.1 Simple RNN

First RNN was proposed by Elman (1990), referred as simple RNN. In one layer of simple RNN, sequence $x_t \in \mathbb{R}^n$ is the input where $n$ is the number of input neurons, sequence $y_p \in \mathbb{R}^p$ is the output where $p$ is the number of output neurons. To illustrate the operation of RNN, we unfold the RNN layer in sequential time steps, thus we have $\{\ldots, h_{t-1}, h_t, h_{t+1}, \ldots\}$ as the hidden states in the hidden layer. The hidden state at time step $t$ is calculated by:
\[ h_t = f_H(o_t) \]  
\[ o_t = W_{IH}x_t + W_{HH}h_{t-1} + b_h \]  

(2.17)

where \( f_H(\cdot) \) is the activation function of the hidden layer, \( b_h \) is bias. And the output at time step \( t \) is calculated by:

\[ y_t = f_O(W_Hh_t + b_o) \]  

(2.18)

where \( f_O(\cdot) \) and \( b_o \) are activation function and bias of the output layer respectively. \( tanh(x) \) is typically used as the activation function in RNN.

**Issue: Long Term Dependency** As a memory of historical states and inputs, the current state can predict well when the require time step is small. However, when the predicted time step is far from the current time step, the prediction will deteriorate or even fail. Though RNN has the ability of memorizing no matter how long the sequence, it just cannot realize this in practice (Bengio et al., 1994). The qualitative reason is that by applying embedding projection \( W_{HH} \) multiple times while predicting state of further time step, the state of previous time step plays a insignificant role when \( |W_{HH}| \) is smaller than 1. And for the stability of training, parameters of \( W_{HH} \) is usually very small. Detailed derivation will be discuss in the next paragraph.

**Issue: Gradient Exploding** Another issue of RNN is the gradient exploding. Gradient exploding means gradients get larger and larger, resulting in failure of convergence. We will derive the gradients to show why the issue occurs.

In CNN, gradient vanishing refers the situation that gradients become zero during training, leading to early converge and high bias error. While gradient vanishing in
RNN is mathematically equivalent to long term dependency issue, referring to the problem of long term gradient disappearing.

Assume there are three time steps \( t = 1, 2, 3 \) and no activation function is used for simplicity:

\[
\begin{align*}
    h_1 &= W_{IH}x_1 + W_{HH}h_0 + b, 
    y_1 &= W_Hh_1 + b, \\
    h_2 &= W_{IH}x_2 + W_{HH}h_1 + b, 
    y_2 &= W_Hh_2 + b, \\
    h_3 &= W_{IH}x_3 + W_{HH}h_2 + b, 
    y_3 &= W_Hh_3 + b
\end{align*}
\]  

(2.19)

And the accumulated loss of this sequence is:

\[
\mathcal{L}(y, \hat{y}) = \sum_{t=0}^{T=3} L_t(y_t, \hat{y}_t)
\]  

(2.20)

where \( L_t(y_t, \hat{y}_t) \) is the loss on time step \( t \). And the partial derivatives of \( L_t(y_t, \hat{y}_t) \) with respect to \( W_{IH} \) and \( W_{HH} \) are:

\[
\begin{align*}
    \frac{\partial L_t}{\partial W_{IH}} &= \sum_{k=0}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left( \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W_{IH}} \\
    \frac{\partial L_t}{\partial W_{HH}} &= \sum_{k=0}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left( \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W_{HH}}
\end{align*}
\]  

(2.21)

For \( \frac{\partial L_t}{\partial W_{HH}} \), in case of activation function \( f_H(\cdot) \) is applied:

\[
\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k!}^{t} f_H'(h_{j-1}) W_{HH}
\]  

(2.22)

when \( tanh \) is used as the activation function, it is known that \( 0 \leq tanh' \leq 1 \), \( tanh'(x) = 1 \) if and only if \( x = 0 \). Therefore, we have:
Figure 2.4: Data Flow of a LSTM Cell

- when $t$ is large, $\prod_{j=k+1}^{t} f'_H(h_{j-1}) W_{HH}$ approaches 0, leading to gradient vanishing (long term gradient disappears).

- if $W_{HH}$ is large, $\prod_{j=k+1}^{t} f'_H(h_{j-1}) W_{HH}$ will approach $\infty$, resulting in gradient exploding.

Gradient exploding can be solved by gradient clipping (Pascanu et al., 2013).

While gradient vanishing issue is mitigated by long term gradient preserving of gate scheme, proposed in Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014).

2.3.2 LSTM & GRU

Compared with simple RNN, LSTM introduce another hidden variable called cell state carrying information within the unit. Cell state helps to remember the long term information and it is controlled by the gate scheme.

Gate is a key component of LSTM, it is combined by a sigmoid function and element-wise multiplication. The output of sigmoid function ranges from 0 to 1.
means all the information is passed, 0 means no information is passed. There are
three gates in a LSTM unit, i.e. forget gate, input gate and output gate.

**Forget Gate** Forget gate determines how much previous information $h_{t-1}$ need to
be forget (or remember). The forgetting factor $f_t$ is calculated by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$  (2.23)

**Input Gate** Updating of cell state involves two steps, i.e. 1) input factor obtained
from the input gate, 2) fusion of input $x_t$ and new state obtained from input via $\tanh$,
which are formulated by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$  (2.24)

**Output Gate** The output of a LSTM cell is directly or indirectly fused from the
input $x_t$, the cell state $C_{t-1}$, and the hidden state $h_{t-1}$. The update of output $o_t$ and
hidden state $h_t$ are calculated by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$  (2.25)

GRU is another popular variation of RNN. Compared to LSTM, GRU replaces
input gate $i_t$ and forget gate $f_t$ with a single update gate $z_t$, moreover, GRU uses a
hidden state only without a cell state.

Though the designs of LSTM and GRU differ from each other, they both use the
gating scheme to suppress some recent input to avoid overwriting history information,
thus solve the long term dependency problem.
3.1 Related Work

3.1.1 Contextual Methods for Object Detection

Prior to the emergence of deep learning, various approaches have explored adding contextual information to improve object detection (Alexe et al., 2012b; Heitz and Koller, 2008; Mottaghi et al., 2014; Torralba et al., 2003; Divvala et al., 2009). In a statistical method (Galleguillos et al., 2008), the detected objects are re-scored by considering object relationships such as co-occurrence, which implies how likely two categories of objects can exist in the same image. On the contrary, the presence of objects in irrelevant scenes is penalized in Torralba et al. (2003). These methods achieved moderate success in pre-deep learning era but have not been well established for deep CNNs. One of the possible reasons is that deep CNNs generally convey implicitly and hidden contextual information which is hard to use directly. Another reason is that to accommodate the contextual information within CNNs is a complicated and nontrivial work.

Recently, some approaches (Bell et al., 2016; Shrivastava and Gupta, 2016; Zeng et al., 2016) based on deep CNNs have made attempts to incorporate contextual information into object detection. The work ION (Bell et al., 2016) integrates contextual information outside the ROI using a spatial RNN. GBD-Net (Zeng et al., 2016) proposes a gated bi-directional CNN to pass messages between the features of different support regions around objects. Shrivastava and Gupta (2016) use segmentation to provide top-down context to guide region proposal generation and object detection.
Despite the aforementioned approaches that essentially exploit local context near objects and the whole image context, Chen and Gupta (2017) propose a sequential reasoning architecture that mainly utilizes object-object relationship to detect objects in an image sequentially. Similarly, Hu et al. (2017) introduce attention modules to model object-object relations. Combining both object-object and object-scene relations, SIN (Liu et al., 2018) uses Gated Recurrent Unit (GRU) for message passing. Different from the existing methods, the proposed GAR adopts a light-weight GCN with an explicit and accurate scene detection module. This property further improves the efficiency of GAR in object detection with both object-object relation and object-scene relation considered.

3.1.2 Graph on Neural Networks

The topic of Graph Neural Networks (GNN) has received growing attention recently Cai et al. (2018); Bruna et al. (2013). Kipf and Welling (2016) present a simplified yet well-behaved GNN model, i.e., GCN, which achieves state-of-the-art classification results on several benchmark graph datasets. GCN is then explored in several NLP tasks such as semantic role labeling (Marcheggiani and Titov, 2017) and machine translation (Bastings et al., 2017) to encode the syntactic structure of sentences.

In a recent work (Defferrard et al., 2016), a document or a sentence is treated as a graph of word nodes, and GCN-Text (Yao et al., 2018a) regards both the documents and words as nodes and constructs the corpus graph. In our work, proposal features and learned scene embeddings form the nodes of the heterogeneous graph, and the co-occurrence that is appropriately processed embodies the edges of the graph in GAR.
Figure 3.1: (a) The Architecture of GAR. RPN and Associated Modules Give \( N \) Proposals (Each with a Size of \( F \)). The CRCN Generates \( N \) Cursory Class Scores (Each with a Size of \( O \)) and \( N \) Bbox Regression with a Size of 4. Meanwhile, the Scene Detector Generates Scene-related Labels with a Size of 469. Scene Pooling Picks and Concatenates \( S \) Scene Nodes from 469 of Them, Which Are Generated from Scene Embedding Module. (b) The GCR Module Merges the Instance Subgraph and the Scene Subgraph as a Heterogeneous Graph. Nodes Are Proposal Features \( ins_1, \cdots, ins_N \) and Selected Scene Embeddings \( scn_1, \cdots, scn_S \). Edges Are Created from the Normalized Co-occurrence Matrix That Are Elaborated in the Following Section.

3.2 Approach

Our approach is designed upon the heterogeneous graph composed by object-object subgraph and object-scene subgraph. In this section, we will firstly derive the edges, i.e., relations of contextual information. Then we will show the relation graphs computing flow within the entire object detection network, as shown in Figure 3.1.

3.2.1 Relation Modeling

Adopting the same spirit as GCN-Text (Yao et al., 2018a), we use co-occurrence to encode the relations among objects and scenes.
There are four relations involved in GAR: (i) object-object, estimating the probability of two different categories of objects appear in the same image; (ii) object-indoor/outdoor, measuring the frequency of all types of objects appear in the indoor/outdoor scenario; (iii) object-place, wherein place represents place categories such as "living room", "museum", etc.; and (iv) object-attribute, wherein attribute represents scene attribute such as "natural light", "human-made", etc. Scene-scene relations are not required since GAR is object detection oriented. As classic datasets in object detection are typically lack of scene labels, we train a scene detector on the Place365 (Zhou et al., 2018) recognition dataset to extract scene information for generating co-occurrence matrices for object-scene relations, i.e., (ii), (iii) and (iv). The above four relations are elaborated as follows:

**Object-object relation** The value of each co-occurrence entry represents the co-occurrence number enumerating the entire training images. Multiple occurrences of objects with the same class label in a single image are counted as 1. Calculation of the 2D $O \times O$ object-object co-occurrence matrix $E_{obj}^{obj}$ can be formulated as:

$$E_{obj}^{obj}(i,j) = \sum_{x=1}^{M} \begin{cases} 
1, & \text{if } \text{DET}_x(i) \& \text{DET}_x(j) \\
0, & \text{otherwise}
\end{cases}$$

(3.1)

where $\text{DET}_x(i)$ means that the $x$-th training sample contains the object(s) with class index $i$. "&" means its left event and its right event happen at the same time. $M$ represents the size of the training set and $O$ is the number of classes in the selected dataset. It is worth noting that diagonal entries (self-loop) of $E_{obj}^{obj}$ are 0 instead of 1, as self-loop information is adaptively learned in GAR. The cumulative co-occurrence is normalized within GAR computing which is described in the following content.
**Object-indoor/outdoor relation**  Indoor/outdoor is a binary label for an image. The output of the scene detector is composed of ”indoor/outdoor” label (scalar), place categories (a vector containing 365 elements) and scene attributes (a vector containing 102 elements).

The calculation of the $O \times 2$ object-indoor/outdoor co-occurrence $E^{io}_{obj}$ is:

$$E^{io}_{obj}(i,:) = \sum_{x=1}^{M} \begin{cases} [1 \ 0], & \text{if } DET_x(i) \& \text{INOUT}_x(\text{indoor}) \\ [0 \ 1], & \text{if } DET_x(i) \& \text{INOUT}_x(\text{outdoor}) \end{cases}$$ (3.2)

Where INOUT$_x(\text{indoor})$ means that the image $x$ is classified as ”indoor”. DET$_x(i)$ is a vector composed by class indices of all objects detected in image $x$.

**Object-place relation**  The scene detector infers place labels among 365 place categories. We will calculate the $O \times 365$ object-place co-occurrence $E^{plc}_{obj}$ by:

$$E^{plc}_{obj}(i,p) = \sum_{x=1}^{M} \begin{cases} 1, & \text{if } DET_x(i) \& \text{PLACE}_x(p) \\ 0, & \text{otherwise} \end{cases}$$ (3.3)

where PLACE$_x(p)$ means that place labels with indices $p$ are detected in the image $x$. Multiple place categories are taken since they could be synonyms in the sense of ”scene”, sharing the similar scene context.

**Object-attribute relation**  The scene attributes are also generated by the scene detector among 102 classes. The $O \times 102$ object-attribute co-occurrence $E^{atr}_{obj}$ is calculated by:

$$E^{atr}_{obj}(i,q) = \sum_{x=1}^{M} \begin{cases} 1, & \text{if } DET_x(i) \& \text{ATTR}_x(q) \\ 0, & \text{otherwise} \end{cases}$$ (3.4)
Similarly, ATTR$_x(q)$ means scene attribute labels with indices $q$ are detected in image $x$.

### 3.2.2 GAR Design

Different from existing works that use implicit visual appearance context, GAR is designed to make use of explicit object-object/scene relation to reward or penalize object proposals and thus assist object detection.

GAR is composed of four major modules: (i) a backbone object detector that generates object proposals, (ii) a scene detector that generates scene labels, (iii) a cursory Regression and Classification Network (cRCN) that returns the cursory detection scores as well as a spatial adjustment vector for each object proposal, and (iv) Graph Convolutional Reasoning (GCR) module which takes cursory detection, object/scene features (nodes) and prior relation knowledge (edges) as inputs and generates the graph reasoning scores. The entire framework of GAR is illustrated in Figure 3.1.

GAR is a general method. In this work, we use Faster R-CNN (Ren et al., 2015) as the backbone object detector for demonstration purpose. Other CNN-based detectors are also compatible with it.

**Object edges to instance edges** In Faster R-CNN, thousands of region proposals that might contain objects are obtained after Region Proposal Network (RPN). Non-Maximum Suppression (NMS) (Felzenszwalb et al., 2010) is then used to select a fixed number (e.g., $N=300$) of ROIs. Next, for each ROI $i$, its visual feature $v_i$ is processed by the ROI pooling and a fully connected projection layer. Consequently, the instance feature matrix $V_{roi}$ concatenated by all $N$ ROI vectors is fed into cRCN to get $N$ cursory class scores and bbox adjustment.
Algorithm 1: cls-roi Edge

**Data:** co-occurrence matrix $E_{obj}$

cursory instance class score $Ins_{score}$

self-loop edges for instances $A$

**Result:** Instance relation edges $E_{ins}$

1. Softmax $E_{obj}$ in a row-wise manner;

2. for every instance pair: $\{i, j\}$ do
   
   3. get cursory instance class label:
      
      $$cls^i = \arg \max_{k \in O} \{Ins_{score}^{i,k}\};$$

      $$cls^j = \arg \max_{k \in O} \{Ins_{score}^{j,k}\};$$

   4. get instance relation:
      
      $$E_{ins}^{i,j} = E_{obj}(cls^i, cls^j);$$

   5. add self-loop edge:
      
      $$E_{ins}^{i,i} = E_{ins}^{i,i} + A(i), \forall i \in [0, N).$$

10. end

11. $\tilde{E}_{ins} = \text{Softmax } E_{roi}$ in a row-wise manner;

The $N \times N$ relation edges among $N$ instances, $E_{ins}$, is obtained by utilizing the prior $O \times O$ object-object co-occurrence $E_{obj}$ and the cursory detection score, as shown in Algorithm 1.

**Scene nodes and instance-scene relation** In GAR, instance nodes of the GCR module input are the instance feature matrix $V_{ins}$ in the shape of $N \times F$, which are naturally compatible with GCN. However, the latent feature of the whole image is in the shape of $512 \times 14 \times 14$ (conv5_3 of VGG-16 (Simonyan and Zisserman, 2014)). Therefore, we design a scene nodes embedding module to project the latent scene
Algorithm 2: scene-roi Edge

Data: scenic co-occurrence matrices $E_{obj}^i$, $E_{obj}^p$, $E_{attr}^i$;
cursory ROI class score $ROI_{score}$;
indoor/outdoor score $In_{score} / Out_{score}$;
place category scores $Plc_{score}$;
attribute scores $Attr_{score}$

Result: scene-ROI relation graph $E_{scene-roi}$

1 Softmax co-occurrence matrices in a row-wise manner;

2 for every ROI pair: \{i, j\} do

3 get ROI cls:
4 $R_{cls}^i = \arg \max_{k \in K} \{ROI_{score}^{i,k}\}$;
5 $R_{cls}^j = \arg \max_{k \in K} \{ROI_{score}^{j,k}\}$;

6 get ROI relation:
7 $E_{roi}^{i,j} = ROI_{score}^{R_{cls}^i, R_{cls}^j}$

8 end

9 Softmax $E_{roi}$ in a row-wise manner;

features in the same space as instance nodes. There are $S = 2 + 2K$ scene nodes
selected for scene subgraph, including an indoor node, an outdoor node, $K$ place
category nodes, and $K$ scene attribute nodes. Instance-scene relation edges $E_{scn}^{ins}$
obtained in a similar manner as Algorithm 1. By performing softmax on $E_{ins}^{ins}$ in a
row-wise manner, we are able to get a normalized relation measurement of nodes in
the heterogeneous relation graph and maintain the numeric stability while training
the GCR module.

Acquiring of instance-scene edges $E_{scn}^{ins}$ is distinct from that of instance-instance
edges in two-fold: (i) It selects $S$ scene nodes rather than all the $469 = (2+365+102)$
nodes to reduce computational complexity and to avoid over-smoothing (Li et al., 2018) induced by overwhelming irrelevant information. (ii) It normalizes complementary relations by row-wise softmax. Concretely, $E_{ins}^{dcl}$ and $E_{ins}^{atr}$ are normalized individually, while instance-indoor/outdoor relation are normalized by softmax($[E_{ins}^{in}, E_{ins}^{out}]$). Detailed computing flow is elaborated in Algorithm 2.

**GCR module** Now we get our heterogeneous graph nodes (instance nodes and scene nodes) and edges ready. It is time to perform graph reasoning.

The graph is fed into a similar two-layer GCN as used in (Kipf and Welling, 2016). In the first layer, each node has a size of 4096. For the second layer, each node has a size of 512. The output of a node is in the same size as the number of object classes. The scores of instance nodes are generated and then fused to the cursory scores weighted by learnable factors, generating the final cogitative scores:

$$Y_g = \tilde{A} \text{ReLU}(AXW_0)W_1$$  \hspace{1cm} (3.5)$$
$$Z = \text{softmax}(\frac{\exp(w_b)}{\exp(w_b) + \exp(w_g)} \cdot Y_b + \frac{\exp(w_g)}{\exp(w_b) + \exp(w_g)} \cdot Y_g)$$  \hspace{1cm} (3.6)$$

where $Y_b$ is the cursory detection scores, $\tilde{A}$ is the normalized adjacent matrix of the heterogeneous graph, $X$ are the nodes. $W_0, W_1$ are parameters of the two-layer GCN. $w_b$ and $w_p$ are fusion factors used for adding graph reasoning scores with the cursory scores.

### 3.3 Experiments

In this section, we evaluate the proposed GAR on PASCAL VOC (Everingham et al., 2010) and MS COCO (Lin et al., 2014) object detection datasets. The base detection framework is Faster R-CNN (Ren et al., 2015) whose feature extractor is by
default a VGG-16 (Simonyan and Zisserman, 2014) that is pre-trained on ImageNet classification dataset (Krizhevsky et al., 2012).

Following the same practice as SIN (Liu et al., 2018), we trained the Faster R-CNN from scratch as the baseline. We find that training backbone Faster R-CNN for several epochs and then jointly training several epochs with GAR performs better than jointly training from the beginning. This is because that GAR constructs the instance and scene edges based on the cursory detection. This training strategy is denoted as \textit{two-stage} \((M, N)\) training, where \(M\) and \(N\) represent the number of training epochs in the first and the second stage, respectively. On the contrary, training the whole system from scratch is denoted as \textit{one-stage} \((M+N)\) training wherein the network is trained for \(M+N\) epochs in total.

Specifically, when training on VOC 2007 dataset with \textit{two-stage} \((5, 5)\) strategy, we use a learning rate of \(5 \times 10^{-4}\) for the first 5 epochs, then \(5 \times 10^{-5}\) for the last 5 epochs. When training on VOC 2012 trainval with VOC 2007 trainval combined following \textit{two-stage} \((4, 6)\) strategy, we use a learning rate of \(5 \times 10^{-4}\) for the first 4 epochs, then \(5 \times 10^{-5}\) for the following 6 epochs. When training on COCO 2014 dataset with \textit{two-stage} \((4, 6)\) strategy, we use a learning rate \(5 \times 10^{-4}\) for the first 4 epochs and \(5 \times 10^{-5}\) for the last 6 epochs.

\subsection{3.3.1 Overall Performance}

\textbf{PASCAL VOC.} There are 20 classes of objects in the VOC dataset. The VOC 2007 dataset consists of about 5k training and validation combined (trainval) images and 5k testing images, while VOC 2012 dataset includes about 11k trainval images and 11k test images. We set two kinds of training datasets. The evaluations that are performed on the VOC 2007 and VOC 2012 testing sets are shown in Table 3.1 and Table 3.2, respectively. By applying GAR, we get the mAP of 76.1\% on VOC 2007
<table>
<thead>
<tr>
<th>Net</th>
<th>mAP</th>
<th>arpl.</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bot.</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chr.</th>
<th>cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS-N</td>
<td>70.0</td>
<td>77.0</td>
<td>78.1</td>
<td>69.3</td>
<td>59.4</td>
<td>38.3</td>
<td>81.6</td>
<td>78.6</td>
<td>86.7</td>
<td>42.8</td>
<td>78.8</td>
</tr>
<tr>
<td>FR-N</td>
<td>73.2</td>
<td>76.5</td>
<td>79.0</td>
<td>70.9</td>
<td>65.5</td>
<td>52.1</td>
<td>83.1</td>
<td>84.7</td>
<td>86.4</td>
<td>52.0</td>
<td>81.9</td>
</tr>
<tr>
<td>SSD500</td>
<td>75.1</td>
<td><strong>79.8</strong></td>
<td>79.5</td>
<td>74.5</td>
<td>63.4</td>
<td>51.9</td>
<td>84.9</td>
<td>85.6</td>
<td>87.2</td>
<td>56.6</td>
<td>80.1</td>
</tr>
<tr>
<td>ION</td>
<td>75.6</td>
<td>79.2</td>
<td><strong>83.1</strong></td>
<td><strong>77.6</strong></td>
<td>65.6</td>
<td>54.9</td>
<td>85.4</td>
<td>85.1</td>
<td>87.0</td>
<td>54.4</td>
<td>80.6</td>
</tr>
<tr>
<td>SIN</td>
<td>76.0</td>
<td>77.8</td>
<td>73.6</td>
<td>61.2</td>
<td>61.7</td>
<td><strong>83.6</strong></td>
<td><strong>85.9</strong></td>
<td>85.1</td>
<td>56.6</td>
<td><strong>83.4</strong></td>
<td>67.2</td>
</tr>
<tr>
<td><strong>GAR</strong></td>
<td><strong>76.1</strong></td>
<td>77.4</td>
<td><strong>81.3</strong></td>
<td>74.8</td>
<td><strong>65.9</strong></td>
<td>59.9</td>
<td>85.0</td>
<td><strong>86.6</strong></td>
<td><strong>88.6</strong></td>
<td>56.2</td>
<td><strong>84.6</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Net</th>
<th>mAP</th>
<th>tbl.</th>
<th>dog</th>
<th>hrs.</th>
<th>mbk.</th>
<th>prs.</th>
<th>plt.</th>
<th>shp.</th>
<th>sofa</th>
<th>trn.</th>
<th>tvm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS-N</td>
<td>70.0</td>
<td>68.9</td>
<td>84.7</td>
<td>82.0</td>
<td>76.6</td>
<td>69.9</td>
<td>31.8</td>
<td>70.1</td>
<td>74.8</td>
<td>80.4</td>
<td>70.4</td>
</tr>
<tr>
<td>FR-N</td>
<td>73.2</td>
<td>65.7</td>
<td>84.8</td>
<td>84.6</td>
<td>77.5</td>
<td>76.7</td>
<td>38.8</td>
<td>73.6</td>
<td>73.9</td>
<td>83.0</td>
<td>72.6</td>
</tr>
<tr>
<td>SSD500</td>
<td>75.1</td>
<td>70.0</td>
<td>85.4</td>
<td>84.9</td>
<td>80.9</td>
<td>78.2</td>
<td>49.0</td>
<td><strong>78.4</strong></td>
<td>72.4</td>
<td><strong>84.6</strong></td>
<td>75.5</td>
</tr>
<tr>
<td>ION</td>
<td>75.6</td>
<td>73.8</td>
<td>85.3</td>
<td>82.2</td>
<td><strong>82.2</strong></td>
<td>74.4</td>
<td>47.1</td>
<td>75.8</td>
<td>72.7</td>
<td>84.2</td>
<td><strong>80.4</strong></td>
</tr>
<tr>
<td>SIN</td>
<td>76.0</td>
<td><strong>81.7</strong></td>
<td>83.9</td>
<td>78.4</td>
<td>78.2</td>
<td>47.5</td>
<td><strong>73.3</strong></td>
<td>68.3</td>
<td><strong>77.8</strong></td>
<td>76.7</td>
<td>70.0</td>
</tr>
<tr>
<td><strong>GAR</strong></td>
<td><strong>76.1</strong></td>
<td>72.2</td>
<td><strong>86.9</strong></td>
<td><strong>86.2</strong></td>
<td>77.3</td>
<td><strong>79.2</strong></td>
<td>46.8</td>
<td>77.4</td>
<td>4.6</td>
<td>83.2</td>
<td>77.3</td>
</tr>
</tbody>
</table>

**Table 3.1:** Detection on PASCAL VOC 2007, Trained on VOC 2007 and VOC 2012 Trainval Combined. Abbreviation: Fast R-CNN (FS-N) (Girshick, 2015), Faster R-CNN (FR-N), Aeroplane (arpl.), Bottle (bot.), Chair (chr.), Table (tbl.), Horse (hrs.), Motorbike (mbk.), Person (prs.), Plant (plt.), Sheep (shp.), Train (trn.), Tvmonitor (tvm.).

<table>
<thead>
<tr>
<th>Net</th>
<th>mAP</th>
<th>arpl.</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bot.</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chr.</th>
<th>cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS-N</td>
<td>68.3</td>
<td>82.3</td>
<td>78.4</td>
<td>70.8</td>
<td>52.3</td>
<td>38.7</td>
<td>77.8</td>
<td>71.6</td>
<td>89.3</td>
<td>44.2</td>
<td>73.0</td>
</tr>
<tr>
<td>FR-N</td>
<td>70.4</td>
<td><strong>84.9</strong></td>
<td>79.8</td>
<td>74.3</td>
<td>53.9</td>
<td>49.8</td>
<td>77.5</td>
<td>75.9</td>
<td>88.5</td>
<td>45.6</td>
<td>77.1</td>
</tr>
<tr>
<td>SIN</td>
<td><strong>73.1</strong></td>
<td>84.8</td>
<td>79.5</td>
<td>74.5</td>
<td><strong>59.7</strong></td>
<td><strong>55.7</strong></td>
<td>79.5</td>
<td>78.8</td>
<td><strong>89.9</strong></td>
<td><strong>51.9</strong></td>
<td>76.8</td>
</tr>
<tr>
<td><strong>GAR</strong></td>
<td><strong>73.1</strong></td>
<td><strong>84.9</strong></td>
<td><strong>80.1</strong></td>
<td><strong>74.6</strong></td>
<td>58.0</td>
<td>53.9</td>
<td><strong>80.0</strong></td>
<td><strong>78.9</strong></td>
<td>89.5</td>
<td>49.8</td>
<td><strong>77.6</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Net</th>
<th>mAP</th>
<th>tbl.</th>
<th>dog</th>
<th>hrs.</th>
<th>mbk.</th>
<th>prs.</th>
<th>plt.</th>
<th>shp.</th>
<th>sofa</th>
<th>trn.</th>
<th>tvm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS-N</td>
<td>68.3</td>
<td>55.0</td>
<td>87.5</td>
<td>80.5</td>
<td>80.8</td>
<td>72</td>
<td>35.1</td>
<td>68.3</td>
<td><strong>65.7</strong></td>
<td>80.4</td>
<td>64.2</td>
</tr>
<tr>
<td>FR-N</td>
<td>70.4</td>
<td>55.3</td>
<td>86.9</td>
<td>81.7</td>
<td>80.9</td>
<td>79.6</td>
<td>40.1</td>
<td>72.6</td>
<td>60.9</td>
<td>81.2</td>
<td>61.5</td>
</tr>
<tr>
<td>SIN</td>
<td><strong>73.1</strong></td>
<td>58.2</td>
<td>87.8</td>
<td>82.9</td>
<td><strong>81.8</strong></td>
<td>81.6</td>
<td><strong>51.2</strong></td>
<td>75.2</td>
<td>63.9</td>
<td>81.8</td>
<td><strong>67.8</strong></td>
</tr>
<tr>
<td><strong>GAR</strong></td>
<td><strong>73.1</strong></td>
<td><strong>58.4</strong></td>
<td><strong>88.3</strong></td>
<td><strong>83.2</strong></td>
<td>81.5</td>
<td><strong>82.4</strong></td>
<td>50.7</td>
<td><strong>76.4</strong></td>
<td>63.6</td>
<td><strong>82.2</strong></td>
<td>67.7</td>
</tr>
</tbody>
</table>

**Table 3.2:** Detection Results on PASCAL VOC 2012, Trained on VOC 2007 Trainval, 2012 Trainval and 2007 Test Combined.

testing set and mAP of 73.1% on VOC 2012 testing set.
Table 3.3: Detection on COCO 2014 Test-dev.

MS COCO To validate the efficacy of GAR on a larger dataset, we conduct experiments on COCO and summarize the results in Table 3.3. COCO dataset involves 80 object categories. Different from VOC, COCO dataset uses AP as its evaluation metric. The overall performance AP averages mAP over different intersection over union (IOU) thresholds from 0.5 to 0.95, placing more weight on localization. In this more challenging dataset, GAR achieves 23.1% on test-dev score and brings about 2.1% improvement over baseline detector, again verifying the advantage of its efficacy.

3.3.2 Design Analysis and Ablation Study

Top K place labels and scene attributes As aforementioned, a lot of place labels are synonyms which can be hardly differentiated. For example, ”cafeteria”, ”restaurant” and ”dining hall” are all places for dining and share a lot of common features. Though making use of more possible place labels and scene attributes tends to provide more information about the scene. However, too much irrelevant information involved aggravates over-smoothing problem (Li et al., 2018) of GCN. To find the optimal design hyper-parameter $K$, we conduct evaluations on VOC 2007 validation set with different $K$ by tuning $K$, as shown in Table 3.4. It is observed that $K = 3$ achieves the optimal mAP for GAR.
Table 3.4: Performance on VOC 2007 Validation Set Using Different $K$ for Scene Nodes Selection.

To get a better understanding of object-object/scene relation, we summarize the top 3 related entities (object classes, place categories, scene attribute, and indoor/outdoor labels) in terms of co-occurrence for each object class, as shown in Table 3.5. Some interesting phenomena are observed: First, the object ”person” is highly correlated with other objects in the VOC dataset; Second, besides ”person”, ”car” usually appears with ”bus” and ”motorbike”. Meanwhile, these three methods of transportation are all labeled as ”outdoor” usually appear at ”street” and ”parking lot” which are featured by ”man-made”, ”natural-light” and ”open-area”. In Table 3.5, ”NA” in the indoor/outdoor field means that neither the probability of ”indoor” nor ”outdoor” exceeds 30%.

Scene/Object Ablative Comparison  We evaluate the effectiveness of object-object reasoning (edge) and object-scene reasoning (scene) separately and compare
Table 3.5: Top 3 Related Object/scene Entities in Terms of Co-occurrence, on VOC 2007 Trainval. From Top to Bottom: Three Categories of Mostly Co-occurred Objects, the Indoor/outdoor Label, Three Categories of Mostly Co-occurred Places and Three Mostly Related Scene Attributes.

<table>
<thead>
<tr>
<th>arpl.</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bot.</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chr.</th>
<th>cow</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>person</td>
<td>person</td>
<td>person</td>
<td>person</td>
<td>car</td>
<td>person</td>
<td>chair</td>
<td>person</td>
<td>person</td>
</tr>
<tr>
<td>car</td>
<td>car</td>
<td>cow</td>
<td>car</td>
<td>diningtable</td>
<td>person</td>
<td>bus</td>
<td>person</td>
<td>diningtable</td>
<td>horse</td>
</tr>
<tr>
<td>boat</td>
<td>bus</td>
<td>boat</td>
<td>bird</td>
<td>chair</td>
<td>bicycle</td>
<td>motorbike</td>
<td>sofa</td>
<td>sofa</td>
<td>bird</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>outdoor</th>
<th>outdoor</th>
<th>outdoor</th>
<th>outdoor</th>
<th>indoor</th>
<th>outdoor</th>
<th>outdoor</th>
<th>indoor</th>
<th>indoor</th>
<th>outdoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>airfield</td>
<td>raceway</td>
<td>wtr-hl</td>
<td>harbor</td>
<td>pub</td>
<td>bus-st</td>
<td>park-lot</td>
<td>vtr-ofc</td>
<td>din-rm</td>
<td>corral</td>
</tr>
<tr>
<td>runway</td>
<td>crosswalk</td>
<td>pond</td>
<td>boat-dk</td>
<td>br-hall</td>
<td>street</td>
<td>street</td>
<td>pet-shp</td>
<td>din-hl</td>
<td>pasture</td>
</tr>
<tr>
<td>sky</td>
<td>street</td>
<td>field</td>
<td>ocean</td>
<td>bar</td>
<td>park-lot</td>
<td>raceway</td>
<td>kennel</td>
<td>liv-rm</td>
<td>farm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>natr-lt</th>
<th>natr-lt</th>
<th>natr-lt</th>
<th>no-hrz</th>
<th>man-md</th>
<th>man-md</th>
<th>no-hrz</th>
<th>no-hrz</th>
<th>no-hrz</th>
<th>natr-lt</th>
</tr>
</thead>
<tbody>
<tr>
<td>open-ar</td>
<td>man-md</td>
<td>no-hrz</td>
<td>open-ar</td>
<td>enclosed</td>
<td>ntr-lt</td>
<td>ntr-lt</td>
<td>enclosed</td>
<td>enclosed</td>
<td>open-ar</td>
</tr>
<tr>
<td>man-md</td>
<td>no-hrz</td>
<td>open-ar</td>
<td>man-md</td>
<td>man-md</td>
<td>open-ar</td>
<td>man-md</td>
<td>man-md</td>
<td>man-md</td>
<td>no-hrz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tbl.</th>
<th>dog</th>
<th>hrs.</th>
<th>mbk.</th>
<th>prs.</th>
<th>plt.</th>
<th>shp.</th>
<th>sofa</th>
<th>trn.</th>
<th>tvm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>chair</td>
<td>person</td>
<td>person</td>
<td>person</td>
<td>car</td>
<td>person</td>
<td>person</td>
<td>person</td>
<td>person</td>
<td>chair</td>
</tr>
<tr>
<td>person</td>
<td>sofa</td>
<td>car</td>
<td>car</td>
<td>chair</td>
<td>chair</td>
<td>dog</td>
<td>chair</td>
<td>car</td>
<td>person</td>
</tr>
<tr>
<td>bottle</td>
<td>chair</td>
<td>dog</td>
<td>bicycle</td>
<td>horse</td>
<td>sofa</td>
<td>cow</td>
<td>tvmonitor</td>
<td>boat</td>
<td>sofa</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>indoor</th>
<th>NA</th>
<th>outdoor</th>
<th>outdoor</th>
<th>NA</th>
<th>NA</th>
<th>outdoor</th>
<th>indoor</th>
<th>outdoor</th>
<th>indoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>din-hall</td>
<td>vtr-ofc</td>
<td>corral</td>
<td>raceway</td>
<td>street</td>
<td>rf-grdn</td>
<td>pasture</td>
<td>liv-rm</td>
<td>rl-tck</td>
<td>hm-ofc</td>
</tr>
<tr>
<td>restaurant</td>
<td>outdoor</td>
<td>racecourse</td>
<td>street</td>
<td>indoor</td>
<td>vege-gd</td>
<td>farm</td>
<td>wait-rm</td>
<td>platform</td>
<td>office</td>
</tr>
<tr>
<td>din-rm</td>
<td>pet-shp</td>
<td>stable</td>
<td>highway</td>
<td>nurs-hm</td>
<td>liv-rm</td>
<td>hayfield</td>
<td>drm-rm</td>
<td>platform</td>
<td>ct-rm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no-hrz</th>
<th>no-hrz</th>
<th>ntr-lt</th>
<th>ntr-lt</th>
<th>no-hrz</th>
<th>no-hrz</th>
<th>ntr-lt</th>
<th>no-hrz</th>
<th>man-md</th>
<th>no-hrz</th>
</tr>
</thead>
<tbody>
<tr>
<td>enclosed</td>
<td>enclosed</td>
<td>open-ar</td>
<td>man-md</td>
<td>man-md</td>
<td>man-md</td>
<td>open-ar</td>
<td>enclosed</td>
<td>ntr-lt</td>
<td>enclosed</td>
</tr>
<tr>
<td>man-md</td>
<td>man-md</td>
<td>man-md</td>
<td>no-hrz</td>
<td>ntr-lt</td>
<td>enclosed</td>
<td>grass</td>
<td>man-md</td>
<td>open-ar</td>
<td>man-md</td>
</tr>
</tbody>
</table>

their performance with the previous work SIN (Liu et al., 2018). As shown in Table 3.6, all methods are trained on VOC 2007 trainval and testing set on VOC 2007 testing set. GAR-scene module achieves better mAP of 70.29% as compared to SIN-scene with mAP of 70.23%. SIN-edge module provides higher mAP of 70.31% than GAR-edge with mAP of 70.29%. The reason is that SIN-edge takes more compli-
Table 3.6: Ablative Comparison with SIN on VOC 2007 Test, Trained on VOC 2007 Trainval. Abbreviation: Edge Module (E), Scene Module (S).

cated spatial and geometric relations, which might contain more information than co-occurrence relation used in GAR-edge.

Interestingly, it is observed that the edge/scene module boosts mAP in some categories, such as "boat", "cow", "horse", "sheep", "tvmonitor", etc. This is expected since such categories are generally correlated with scene context and other objects occurrence. However, we observed that the mAP of "table" is suffering from degradation. One possible reason is that "table" is so similar to "chair" and "sofa". Therefore, the possibility of mislabeling as well as the IOU loss are largely increased due to similar relations.

Qualitative Analysis We show representative qualitative results in Figure 3.2 to present how GAR with graph reasoning helps object detection. GAR benefits object detection in two folds:

(1) It detects obscure objects better with reliable scene inference. For example, Figure 3.2a depicts a car in front of a gas station. With the detected scene and prior
knowledge that “car” is highly correlated with “person”, GAR successfully detects the driver inside the car. Similar reasoning is applied to the dog in Figure 3.2b and the person at the left-bottom corner of Figure 3.2d.

(2) It helps to drop irrelevant objects which are, in some sense, ridiculous. For example, the baseline detector detects the car door as a “tvmonitor” in Figure 3.2c. While based on the prior knowledge in Table 3.5, we know “tvmonitor” is typically related with ”indoor”, ”enclosed area” and frequently appears in ”home office”, ”office” and ”computer room”. Thus, GAR drops this wrong detection correctly. Other similar cases also demonstrate the efficacy of GAR, Another example is that the ”boat” detected by the baseline detector in Figure 3.2d is successfully eliminated by GAR.

Sensitivity of Object Characteristics  To further quantitively measure the improvement achieved by GAR, we look at a detailed breakdown of results of VOC 2007
Figure 3.3: Summary of Sensitivity of Object Characteristics. It presents the average (over 7 categories) normalized AP (APN (Hoiem et al., 2012)) of the highest score and lowest score subsets in each characteristic group (occlusion, truncation, bounding box area, aspect ratio, viewpoint, part visibility). Overall APN is indicated by the dashed line. The difference between max and min indicates sensitivity. The difference between max and overall indicates the impact. Red: Scene. Green: Baseline.

Table 3.7: Number of FLOPS and Number of Parameters Required by GAR and SIN Modules.

<table>
<thead>
<tr>
<th>Net</th>
<th>GAR</th>
<th>SIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mode</td>
<td>Edge</td>
</tr>
<tr>
<td>#FLOPS</td>
<td>540M</td>
<td>17.4M</td>
</tr>
<tr>
<td>#params</td>
<td>&lt;1K</td>
<td>2.35M</td>
</tr>
</tbody>
</table>

Using the detection analysis tool (Hoiem et al., 2012). Figure 3.3 provides a compact summary of the sensitivity to each characteristic group and the potential impact of improving robustness on seven categories selected by Hoiem et al. (2012), which are 'aeroplane', 'bicycle', 'bird', 'boat', 'cat', 'chair' and 'diningtable'. Overall, our method is more robust than baseline and SIN method against occlusion, truncation and area size.
Computational Overhead  The proposed GAR is efficient for both training and inference thanks to its paralleled computing flow and small model size. Use the same feature extractor network (VGG-16) as the backbone object detection, we take the output feature of conv5.3 and re-train the fully connected layer for the scene detector on Place365 scene recognition dataset (Zhou et al., 2018). Table 3.7 demonstrates the number of floating point operations (FLOPs) as well as the number of parameters required by GAR and compares it with the previous work SIN (Liu et al., 2018). It is observed that SIN requires much more computing and parameter memory than our GAR due to its complicated edge calculation and sequential GRU propagation. We compare the training and inference speed of baseline Faster R-CNN, SIN and GAR on a single Nvidia RTX 2080 GPU. For sake of the fair comparison, we implement the SIN with Pytorch (Paszke et al., 2017) framework. We also optimize its edge calculation with parallel tensor operation instead of iterative loops where used in its original Tensorflow (Abadi et al., 2015) implementation. For training, frame per second (FPS) of baseline is 6.3, SIN is 2.2 and GAR is 4.0. For inference, FPS of baseline is 15.5, SIN is 8.8 and GAR is 14.1. It can be observed that the overhead of GAR module is much lower than SIN.

3.4 Summary

In this work, we propose a graph-assisted detection method, GAR, that leverages object-object and object-scene relations in object detection. Experiments show prominent accuracy improvement, especially on the categories which are highly correlated to scene context. Moreover, our GAR method has the advantage of computation efficiency: it requires less FLOPs and parameter memory than previous RNN-based methods, making GAR a practical solution in real-time applications. Last but not the least, the framework of GAR can also be extended to model more sophisticated
relations than the demonstrated co-occurrence, such as spatial relation with localization/scale and semantic relation with knowledge graphs.
DAT-RNN: TRAJECTORY PREDICTION WITH DIVERSE ATTENTION

4.1 Related Work

4.1.1 Time Sequence Prediction

Before the prevalent of deep neural networks, many classical statistical models are designed to solve the trajectory prediction problem, such as Hidden Markov Model (Firl et al., 2012) and Kalman Filters (Kalman, 1960). These methods are sufficient in simple and well-defined scenarios. However, in a real application, it is challenging to model trajectories as a formulated distribution because of the complexity of object motions and interactions. Recently, the deep recurrent neural networks show their potential in solving sequence-related problems, such as speech recognition (Miao et al., 2015), natural language processing (NLP) (Yin et al., 2017), etc. Several RNN-based methods (Alahi et al., 2016; Gupta et al., 2018; Vemula et al., 2017) have been proposed for trajectory prediction, in which trajectories are treated as time sequences: historical positions are the input of the models and predicted positions are the output of the models. Beyond the trajectory sequence-only method, which this work is categorized into, there are some works emerging with image processing networks to help trajectory prediction (Sadeghian et al., 2019; Xue et al., 2018).

4.1.2 Mutual Relation Modeling

Early works (Thompson et al., 2009; Large et al., 2004) in the domain of trajectory prediction model individual human motion patterns in crowds individually without
There have been many attempts on human behavior and interactions. The social force model (Helbing and Molnár, 1995) presents a pedestrian motion model with attractive and repulsive forces. Another similar method uses continuum dynamics (Treuille et al., 2006) has also been proposed. Most of these models design handcraft energy potentials based on relative distances and rules for specific scenes. Meanwhile, it is not straightforward to integrate all factors that influence trajectory into one single model. These drawbacks limit the application of model-based methods for trajectory prediction in crowded scenes. Afterward, recently proposed RNN-based methods take latent features from RNN to calculate the relations in a more generic data-driven fashion. For example, Alahi et al. (2016) uses a social pooling layer that models nearby pedestrians. In Social-Pooling (Gupta et al., 2018), a Multi-Layer Perceptron (MLP) followed by max pooling is used to incorporate neighbors’ information. Vemula et al. (2017) use a spatial attention module to generate social interaction embeddings. Ma et al. (2018) design a large model for different types of traffic agents. All these methods are simply taking distance or position difference as the measurement of relations, though distance cannot imply the strength of a relation accurately. In this work, we take richer relation information from anomaly detection than simple distance.

4.1.3 Attention Model

The attention module (Rush et al., 2015) is able to capture the long-term dependencies in these problems well. Recently, it has been successfully applied in the NLP field (Gehring et al., 2017; Vaswani et al., 2017) and sequential system modeling (Hoshen, 2017; Battaglia et al., 2016; Watters et al., 2017). In the domain of trajectory prediction, Vemula et al. (2017) proposed to use an attention model to
capture the relative importance of each person when navigating in the crowd. While the input feature of the attention model is simply position differences. The performance of the attention module may be limited by the less informative feature, though the attention module may generate various weights based on distances. In this paper, we design the AAM that takes both positions and the anomaly salience as the input feature to capture the importance of neighbors.

4.2 Approach

For the traffic agents, their movement patterns have temporal locality. In other words, a person who is walking straight probably keeps walking straight in a short period; similarly, a person who is detouring probably keeps turning in a short period until finishing the detouring process. Moreover, a complicated trajectory can be separated into several segments so that each one consists of either straight sub-trajectories or detouring sub-trajectories in a coarse criterion. By fusing the prediction of these two categories of segments, the prediction is more accurate and efficient. The proposed DAM is designed based on these observations.

As discussed in the previous section that distance may not be a proper measurement for the strength of relations. In a crowd scenario, people may behave distinctly or identically no matter they are closed to each other or not. In this case, anomaly events matter in the attention mechanism. Accordingly, we design the RNN-based AAM module to compute an anomaly salience and then obtain the importance for all surrounding nodes.

**Problem Definition**  Our goal is to jointly reason and predict the future trajectories of all the agents involved in a scene.

Following the same convention as previous works (Alahi et al., 2016; Vemula et al.,
Figure 4.1: (a) Illustration of the Spatial-temporal Graph, Where Temporal Edges Are Uni-directional Edges While Spatial Edges Are Bi-directional Edges. (b) Overview of the DAT-RNN: the DAM Extracts Individual Temporal Relation and the AAM Extracts Mutual Spatial Relation, the Attention Module Merges These Relation Information and the Node-RNN Makes the Final Prediction.

2017), we assume that each snippet is pre-processed to obtain the spatial coordinates of all people at different time stamp in the pixel space.

The input trajectory of a person $i$ is defined as $\text{OBS}_i = (p_{t_{xi}}, p_{t_{yi}})$ from time steps $t = 1, \ldots, t_{obs}$ and the future trajectory (ground truth) can be defined similarly as $\text{PRED}_i = (p_{t_{xi}}^f, p_{t_{yi}}^f)$ from time steps $t = t_{obs} + 1, \ldots, t_{pred}$.

Model Overview In this paper, we formulate the task of human trajectory prediction with the spatial-temporal graph paradigm proposed in (Jain et al., 2016) and (Vemula et al., 2017), as illustrated in Figure 4.1a. In spatial-temporal graphs, nodes are associated with the node-RNN, and edges are associated with edge-RNNs. More specifically, the DAM models temporal edges conveying the temporal dynamics of individual motion of each human, and the AAM models spatial edges conveying the spatial dynamics of human-human interactions in the crowd.

The overview of the entire model is illustrated in Figure 4.1b. DAM and AAM extract temporal and spatial dynamics, then the importance of adaption from the attention module is used to generate future prediction by the node-RNNs.
4.2.1 Diversity-Aware Memory

Unlike the same parameters are shared by temporal edge-RNN in (Vemula et al., 2017), two modules, i.e., straight temporal RNN and detouring temporal RNN are used to extract dynamics of the two types of trajectory correspondingly. Then another RNN-based module is employed to differentiate and fuse the two kinds of dynamics.

**Detour Integral** To divide the trajectories into simple segments, we need to set a criterion. In this work, we proposed to use the feature named detour integral (DI). It is the area constrained by the observed trajectory and a straight line from the starting point to the ending point for a given sequence, as shown in Figure 4.2a. Furthermore, Figure 4.2b gives an example to compute DI. In the sequence starting from point o and ending at point e, all the position points are scattered on the given trajectory. For instance, DI within the small interval around a is:

\[
\left\| \vec{o}a - (\vec{o}a, \vec{ae}) \cdot \frac{\vec{oe}}{\|\vec{oe}\|_2} \right\| \cdot \frac{\|\vec{oe}\|_2}{m} \tag{4.1}
\]

Here \( m \) is the number of points within the given trajectory. Consequently, the DI value of the given trajectory is obtained by summing up small DI of all points. To eliminate the ambiguity of long straight trajectory and short detouring trajectory, we add path integral \( p_i^t \) (Blau et al., 1990) shown in Figure 4.2c and the velocity vector \( \gamma_i^t \) into the DI feature \( di_i^t \) for node \( i \) at time stamp \( t \):

\[
pi_i^t = \sum_{i=1}^{m} \|p_i^t - p_i^{t-1}\|_2 \\
g_i^t = p_i^t - p_i^{t-1} \\
di_i^t = [di_i^t, pi_i^t, \gamma_i^t] \tag{4.2}
\]

Where \( p_i^t \) is the position vector of node \( i \) at time stamp \( t \). \( di_i^t \) contains the information of straight or detouring property of the trajectory of node \( i \). The DI-embedding layer
will take this feature to extract the weights for straight and detouring dynamics, and integrate these two hidden states for the final spatial-temporal edge fusion.

**Temporal Edge Representation** Based on the idea to capture straight and detouring dynamics of the trajectories, there are two sub-modules within the temporal edge module, i.e., straight temporal RNN ($R_{t-s}$) and detouring temporal RNN ($R_{t-d}$). Note that there is no explicit assignment for dynamics learning. Their intrinsic learning and adaption is automated during training. Therefore, their inputs and structures are the same while outputs are different, and their outputs will be fused with the DAM module. We will take $R_{t-s}$ as an example to explain the structure and computing flow. The input feature of $R_{t-s}$ for the temporal edge of node $i$ at the time stamp $t$ is $x_{ii}^t$, the vector difference between position vectors of node $i$ at time stamp $t$ and $t-1$. Here subscript $ii$ is to keep same form as inter-node spatial edge $ij$. The hidden state $h_{ii-s}^t$ is obtained as follows:

\[
\begin{align*}
    e_{ii-s}^t &= \phi(x_{ii}^t; W_{t-s}^{emb}) \\
    h_{ii-s}^t &= RNN(h_{ii-s}^{t-1}, e_{ii-s}^t; W_{t-s}^{rnn})
\end{align*}
\] (4.3)
Where $\phi$ is the embedding layer embeds the input feature $x^t_{ii}$ into fixed-length embeddings $e^t_{ii-s}$. $h^t_{ii-s}$ is the hidden state of the RNN at time stamp $t$. $W^{emb}_{t-s}$ and $W^{rnn}_{t-s}$ are trainable parameters of the embedding layer and RNN respectively.

Meanwhile, $h^t_{ii-d}$, the hidden state of $R_{t-d}$ sub-module, can be obtained in a similar manner. The final integrated hidden state of the temporal edge module can be computed with DAM module:

$$e^t_{ii-dam} = \phi(d_i^t, W^{emb}_{di}),$$

$$w^t_{ii-dam} = \text{softmax}(e^t_{ii-dam}),$$  \hspace{1cm} (4.4)

$$h^t_{ii} = w^t_{ii-dam}[; 0] \odot h^t_{ii-s} + w^t_{ii-dam}[; 1] \odot h^t_{ii-d},$$

first, the hidden features $e^t_{ii-dam}$ of DI-embedding layer is obtained. Then $e^t_{ii-di}$ is normalized to $N \times 2$ fusion weights $w^t_{ii-dam}$ through a softmax function. Finally, the integrated hidden state $h^t_{ii}$ of temporal edge module is obtained from the element-wise weighted sum operation.

### 4.2.2 Anomaly Attention Module

As discussed in the previous sections, the position difference vector between nodes conveys limited information as the input feature of the spatial edge module. In a real traffic scenario, the same vector difference maybe from human pairs in disparate conditions. We argue that anomaly events are also very important when people make movement plan. Therefore, we design the anomaly salience that is generated by a anomaly detection sub-module as an augmented feature for the spatial edge module.

**Anomaly Salience** Anomaly is something that out of expectation. In this work, we adopt the anomaly detection using the prediction error distribution method presented in (Malhotra et al., 2015)

We construct a batch of sequences consisted of joint position tuples for all node
pairs from the same batch of input sequences. For instance, \([p_{x1}^t, p_{y1}^t, p_{x2}^t, p_{y2}^t]\) is the tuple at the time stamp \(t\) for node pair 1 and 2. A prediction module \(R_{pred}\) with the similar structure as \(R_{t-s}\) and \(R_{t-d}\) is used to forecast the future joint position by observing the historical sequence. Then the resulting prediction errors are modeled with a multivariate Gaussian distribution, which is used to estimate the possibility of anomalous neighbors.

Now we will discuss the prediction error calculation in detail. Initially, the joint position sequence is fed into \(R_{pred}\) to get the dynamics of the current movements. Given the prediction length of \(l_{pred}\), \(R_{pred}\) will generate \(l_{pred}\) predicted joint positions. As a result, we get the prediction error by comparing the ground truth positions and the corresponding predictions. The prediction error vector \(e_{i,j}^t\) for joint position composed by node \(i\) and node \(j\) at time stamp \(t\) is represented as a flatten vector, \([e_{i,j,1}^t, \cdots, e_{i,j,l_{pred}}^t]\).

Based on the assumption that the error vector is a random variable following a multi-variate Gaussian distribution \(N(\mu, \Sigma)\), the possibility of observing an error vector \(e_{i,j}^t\) is given by:

\[
 p_{i,j}^t \propto |\Sigma|^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (e_{i,j}^t - \mu) \Sigma^{-1} (e_{i,j}^t - \mu)^T \right)
\]

(4.5)

where the parameters \(\mu\) and \(\Sigma\) of the Gaussian distribution is calculated with unbiased estimation.

Here \(-p_{i,j}^t\) defines the anomaly salience \(as_{i,j}^t\) that measures the anomalous significance of neighborhood node \(j\) to node \(i\) and vice versa. The lower possibility to observe an error vector, the more anomalous for a neighborhood node appears for a given node.

Then the augmented feature \(x_{ij}^t\), i.e., \([dx_{ij}^t, dy_{ij}^t, ss_{ij}^t]\) will be fed into the spatial edge module to get the soft attention weights for neighborhood nodes. Where the
Table 4.1: ADE and FDE Comparison on the Five Datasets

<table>
<thead>
<tr>
<th>Metric (m)</th>
<th>Datasets</th>
<th>LSTM</th>
<th>Social-LSTM</th>
<th>Social-Attention</th>
<th>Social-GAN</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ETH</td>
<td>1.21</td>
<td>1.09</td>
<td>0.69</td>
<td>0.81</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>HOTEL</td>
<td>0.75</td>
<td>0.79</td>
<td>0.43</td>
<td>0.72</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>ZARA-1</td>
<td>0.61</td>
<td>0.67</td>
<td>0.32</td>
<td>0.60</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>ZARA-2</td>
<td>0.55</td>
<td>0.47</td>
<td>0.38</td>
<td>0.34</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>UCY</td>
<td>0.52</td>
<td>0.56</td>
<td>0.33</td>
<td>0.42</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>0.73</td>
<td>0.72</td>
<td>0.43</td>
<td>0.58</td>
<td>0.40</td>
</tr>
<tr>
<td>FDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ETH</td>
<td>2.64</td>
<td>2.35</td>
<td>1.82</td>
<td>1.52</td>
<td>1.43</td>
</tr>
<tr>
<td></td>
<td>HOTEL</td>
<td>1.91</td>
<td>1.76</td>
<td>1.50</td>
<td>1.61</td>
<td>1.55</td>
</tr>
<tr>
<td></td>
<td>ZARA-1</td>
<td>1.31</td>
<td>1.40</td>
<td>1.03</td>
<td>1.26</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>ZARA-2</td>
<td>0.93</td>
<td>1.00</td>
<td>0.89</td>
<td>0.69</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>UCY</td>
<td>1.07</td>
<td>1.17</td>
<td>0.95</td>
<td>0.84</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1.57</td>
<td>1.54</td>
<td>1.24</td>
<td>1.18</td>
<td>1.15</td>
</tr>
</tbody>
</table>

three numbers are minmax normalized in the input batch to keep the numeric scale consistent.

Spatial Edge Representation The hidden state of the spatial RNN ($R_s$) is calculated in a similar way as $R_{t-d}$ and $R_{t-s}$:

$$e_{ij}^t = \phi(x_{ij}^t, W_{embd}^s)$$

$$h_{ij}^t = RNN(h_{ij}^{t-1}, e_{ij}^t, W_{rnn}^s)$$

(4.6)

Where $i$ and $j$ represent different nodes. Similarly, there are an embedding layer and a RNN module within the spatial RNN.

4.2.3 Temporal-spatial Fusion and Prediction

Temporal-spatial fusion Besides the individual temporal information for each node, we also want to take advantage of the spatial relation information from neighborhood nodes. In this work, the fusion of temporal and spatial information is conducted with the soft attention mechanism described in (Vemula et al., 2017), as shown
in Figure 4.3. The attention weights for all spatial edges associated with node $i$ is calculated by:

$$w(h^t_{ij}) = \frac{m}{\sqrt{d_e}} (W_i h^t_{ii}) \cdot (W_{ij} h^t_{ij}) \quad (4.7)$$

where $W_i$ and $W_{ij}$ are parameters of embedding layers whose output is a vector in length of $d_e$, $m$ is the number of nodes in the given sequence, $\frac{m}{\sqrt{d_e}}$ is the scaling factor for stable training (Vaswani et al., 2017).

And the fused temporal-spatial vector $H^t_i$ is computed as a weighted sum of $h^t_i$ with the attention weights $w(h^t_{ij})$.

Now we are ready to get the hidden state $h^t_i$ of node-RNN for the position prediction. The input feature $x^t_i$ of node-RNN is the positions of node $i$ at time stamp $t$. Calculation of $h^t_i$ is as follows:

$$e^t_i = \phi(x^t_i; W_{node}^{emb})$$

$$a^t_i = \phi(concat(h^t_{ii}, H^t_i); W_{node}^{a}) \quad (4.8)$$

$$h^t_i = RNN(h^{t-1}_i, concat(e^t_i, a^t_i); W_{rnn}^{s})$$
Position Prediction  Based on the same assumption as (Alahi et al., 2016) that the position of a node in the next time stamp follows a bivariate Gaussian distribution with parameters including the mean \( \mu^t_i = [\mu^t_{xi}, \mu^t_{yi}] \), standard deviation \( \sigma^t_i = [\sigma^t_{xi}, \sigma^t_{yi}] \) and correlation coefficient \( \rho^t_i \). These parameters can be obtained by embedding fused state \( h^t_i \):

\[
[\mu^t_i, \sigma^t_i, \rho^t_i] = \phi(h^t_i; W^{embd}_{pred})
\]  

(4.9)

Accordingly, the loss \( L_i \) is the summation of negative log likelihood for all predicted time stamps from \( T_{obs} + 1 \) to \( T_{pred} \):

\[
L_i = \sum_{t=T_{obs}+1}^{T_{pred}} -\log(P(x^t_i, y^t_i|\mu^t_i, \sigma^t_i, \rho^t_i))
\]  

(4.10)

The loss is computed over trajectories of all nodes in the training dataset. All trainable parameters involved will be updated to minimize the loss during the training process.

4.2.4 Implementation Details

We use LSTM as the RNN in our model for both DAM and AAM modules. The dimension of hidden state of DAM/AAM and node-RNN are set to 256 and 128 respectively. The output size of all the embedding layers in the network are 64. ReLU activation is used after linear layers. The training batch size is 8 and the network is trained for 200 epochs using Adam with an initial learning rate of 0.001 and decay of 0.5 for every 60 epoches. The global norm of gradients are clipped at a value of 10 to ensure stable training. The model was trained on a single GTX 1080-TI GPU.

4.3 Experiments

Evaluation Datasets and Metrics  We evaluate our method on two publicly available datasets: ETH (Pellegrini et al., 2009) and UCY (Lerner et al., 2007). These
datasets consist of human trajectories with interaction in various scenarios, including crowded settings with challenging scenarios like group behavior, people crossing each other, collision avoidance, and groups forming and dispersing. There are 1536 pedestrians in 5 sets of data (ETH and HOTEL from ETH; ZARA-1, ZARA-2, and UCY from UCY). We extract all coordinates and perform linear interpolate to obtain sequences with a step of 0.4 seconds.

We apply the same metrics used in previous works (Vemula et al., 2017; Gupta et al., 2018), i.e. average displacement error (ADE), and Final Displacement Error (FDE).

1) ADE: Similar to the metric used in (Lerner et al., 2007), this computes the mean Euclidean distance overall estimated points at each time-step in the predicted trajectory and true trajectory.

2) FDE: Introduced in Helbing and Molnár (1995), this metric computes the mean Euclidean distance between the final predicted location and the final true location after $T_{pred}$ time-steps.

Following the similar practice as Helbing and Molnár (1995), we take the leave-one-out approach where we train and validate the model on 4 sets and test on the remaining set. We repeat this for all the 5 sets iteratively. For validation, within each set, we divide the set of trajectories in an 80-20 split for training and validation data. The observation length $T_{obs}$ for the DAT-RNN is set to 8 (corresponding to 3.2 seconds), and the prediction length $T_{pred}$ is set to 12 (corresponding to 4.8 seconds).

4.3.1 Quantitative Evaluation

Overall performance Though there are some state-of-the-art methods with visual features (Sadeghian et al., 2019; Zhang et al., 2019; Zhao et al., 2019) achieving better
Figure 4.4: Comparison Between Our Model (DAT) and Previous Work, Social Attention (SA). The Ground Truth Trajectories Are in Green Color. Here We Only Showcase Several Representative Trajectories in Each Scenario. In Easy Cases of Straight Trajectories, Both Methods Can Predict Pretty Well. But Our Methods Performs Better for Detouring Trajectories.

Figure 4.5: Anomaly Salience and Final Attention Weights for Target Node Are in Red Color. Anomaly Salience Is the Normalized Number While Attention Weights Are Circles Whose Radius Implies the Magnitude.

performance on trajectory prediction, here our goal is to better explore trajectory dynamics itself. Therefore, we perform comparison with the typical trajectory feature
The overall performance of different methods in terms of ADE and FDE are shown in Table 4.1. The base LSTM approach suffers from high prediction errors, as it cannot capture human-human interactions unlike other methods. However, in some cases, performance of the independent LSTM is close to other methods, especially for ZARA. The potential reasons are ZARA are less crowded settings where there are few interactions involved. In these cases, a simple LSTM model is sufficient to predict. Attention models, including Social Attention and DAT-RNN outperforms others, especially in HOTEL and ZARA data. Since for stationary pedestrians, Social LSTM and Social GAN consider them essential if they are within the local neighborhood, but actually, they do not affect others’ motion in a significant way, while DAT-RNN and Social Attention capture this information effectively. For curly trajectories, when people are detouring in the crowded scenarios, DAT-RNN can make relatively closer predictions while predictions of other methods diverge from the ground truth due to frequent turning. By learning more accurate importance of each pedestrian in the crowd and balancing straight and detouring trajectories, DAT-RNN results in more accurate predictions.

**Ablation Study**  The ablation study is shown in the last two columns of Table 4.1. It is noticed that the DAM module contributes more error reduction than the AAM

<table>
<thead>
<tr>
<th>item</th>
<th>Social Attention</th>
<th>DAT-RNN</th>
<th>DAM</th>
<th>AAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>#params</td>
<td>13.5M</td>
<td>22.3M</td>
<td>19.2M</td>
<td>16.6M</td>
</tr>
<tr>
<td>#FLOP</td>
<td>1.7G</td>
<td>3.5G</td>
<td>2.9G</td>
<td>2.3G</td>
</tr>
</tbody>
</table>

*Table 4.2: Number of Floating-point Operations (FLOPS) and Number of Parameters Required by Social Attention Model, the Proposed DAT-RNN as Well as DAM and AAM Decomposition.*
module. There are two main reasons:

1) In many cases, the dynamics of a moving agent depends highly on itself. For example, when people walking along the pedestrian lane, even though these people might have relations since they have very similar speed direction, they will very likely keep the same movement, i.e., walking along the lane if there are no other people.

2) The majority displacement errors come from the offset in detouring trajectories, while the differentiating and fusion scheme of DAM helps to mitigate these errors. In other words, the detouring trajectories introduce more complicated patterns, and DAT-RNN has a larger model capacity to fit this complexity compared with other methods using a single RNN.

On the other hand, the computing overhead is compared in Table 4.2. DAM is more computationally expensive as it use also as twice temporal RNN as the Social Attention model. One potential way to reduce the overhead of DAT-RNN is to take weight sharing for the introduced RNNs, such as to share certain layers within RNN.

4.3.2 Qualitative Evaluation

In Figure 4.4, we illustrate the prediction results of our DAT-RNN as well as other baseline methods. The images are from UCY-ZARA-1 dataset, and there are three people on the road. DAT-RNN has the more accurate predictions when people make turns. Generally, DAT-RNN has the least direction mismatch at the turning point of the given trajectory, and the mismatch is amended when the model observes longer sequences.

Visualized anomaly salience and attention weights of AAM module are shown in Figure 4.5 based on a toy data we created. Note that the two values are not
always directly correlated, as anomaly salience is an input feature to obtain the final attention weights. In Figure 4.5a, the node moves towards the target has higher anomaly salience. That anomalous node and the node closed to the target node have relatively higher attention weight. This result conforms to the intuition.

However, in Figure 4.5b, the model assigns higher attention weight on two stationary nodes even they have relatively lower anomaly salience. A potential reason is that the relative distance of these stationary nodes to the target node is larger, affecting the hidden state fusion process.

4.4 Summary

In this work, we propose a trajectory prediction method, DAT-RNN, that handles the diversity of trajectories and accurate modeling of neighboring relations. By differentiating diverse segments of the complicated trajectories, the DAM module makes more confident predictions separately and then fuses the features to obtain a balanced one. Furthermore, AAM uses augmented attention features with anomaly salience and captures mutual relations more accurately. Experiments on several datasets validate the improvement of the prediction under various circumstances.
Chapter 5

SCENE GRAPH GENERATION ENHANCED BY RELATION IMPLICATION

5.1 Related Work

5.1.1 Scene Graph Generation

As a critical step of the high-level visual understanding, using scene graph generation to extract the interactions between objects receives increasing interest. Originally, Visual Phrases (Sadeghi and Farhadi, 2011b) models the scene graphs as a set of visual phases and use separated detectors to detect them from the image directly. However, due to the curse of dimensionality (Bellman, 1966), making each visual phase as a distinct class fails to extend to moderate datasets since the number of visual phase triples can be huge even with a small number of object classes. A scalable paradigm is to separate the scene graphs into isolated object classes and relation classes. Consequently, most of these methods focus on aggregated representation learning by modeling the dependencies among objects and relations. DR-Net (Dai et al., 2017) embeds the statistical inference procedure into the deep neural network via structural learning. IMP (Xu et al., 2017) constructs bipartite sub-graphs of scene graphs and use RNNs to refine the visual features with an iterative message-passing scheme. Recently, Graph-RCNN (Yang et al., 2018) adds a preliminary relation proposal to get a sparse graph, then uses graph neural networks to learn contextual visual representation. MotifNet (Zellers et al., 2018) introduces the statistical prior to improving the performance with a vanilla bidirectional LSTM-based model. VC-Tree (Tang et al., 2019) takes one step future by constructing a dynamic tree rather than a simple chain for context encoding, improving the nodes feature aggregation.
Hose-Net (Wei et al., 2020) introduces hierarchical semantic aggregation to leverage higher level patterns in the overall contexts. However, explicitly modeling the multi-label property and relation implication remains as an open problem.

5.1.2 Multi-label Learning

A straightforward way for multi-label recognition is to train separated binary classifiers for each class, referred as the ‘binary relevance’ method or the “first-order” strategy (Boutell et al., 2004). To further consider relationships among labels, the second or even higher-order strategy is proposed to model relevance between labels explicitly with higher computational cost (Elisseeff and Weston, 2002; Fürnkranz et al., 2008).

In scene graph generation, the multi-labeling is not completely annotated. The aforementioned relation triplet of hand holding phone is an example. Therefore, it is hard to directly use the correlation between labels as the lack of annotation, limiting the ability of multi-label learning. In this paper, we take the first-order method with great simplicity and high efficiency for the relation prediction.

5.1.3 Knowledge Graph Embedding

As a powerful tool to learn knowledge representations from knowledge graph bases and infer new knowledge, knowledge graph embedding (KGE) (Bianchi et al., 2020) has been extensively investigated. Similar to scene graphs, knowledge graphs are collections of factual triplets \((h, r, t)\), depicting that the relation \(r\) exists between the head entity \(h\) and tail entity \(t\). Formally, let \(\mathcal{E}\) denote the set of entities and \(\mathcal{R}\) denote the set of relations, then entities \(h, t, \in \mathcal{E}\) and relation \(r \in \mathcal{R}\). Considering that entity embeddings are usually represented as vectors, KGE methods define different score function in form of \(f_r(h, t)\) as a measurement of vector transformation, where \(h\) and
Figure 5.1: Illustration of (a) Translation Embedding with Problematic “near” Relation, (b) Rotation Embedding Handling the Symmetry Relation “near”.

\( h \) and \( t \) are embedding vectors of head and tail entities respectively. In other words, the validity of a relation triplet is measured by the score. The goal of the optimization is usually to reward true triplets \((h, r, t)\) and penalize false triplets \((\hat{h}, r, \hat{t})\) with incorrect subjects or objects (Mikolov et al., 2013). After training, semantically closed relations tend to have a similar transformation in the embedding space. TransE (Bordes et al., 2013) represents relation as a translation in the embedding space: \( h + r \approx t \) when the relation holds, and \( h + r \neq t \) otherwise, as illustrated in Figure 5.1a. Both cat and man embeddings have the same translation to the object hat (may have different appearance) embedding. However, TransE model will be problematic if symmetric relation occurs. For example, both car near tree and tree near car are valid relation triplets. However, the translation of relation near is contradicting itself.

RotatE (Sun et al., 2018) models the transformation as embedding rotation in a complex space, theoretically solves the symmetric relation, as shown in Figure 5.1. In this paper, we extend the RotatE model for visual relation modeling by mapping subjects and objects into a low-dimensional relation space, consistent to recent works (Akata et al., 2016; Frome et al., 2013).
5.2 Method

5.2.1 Problem Definition

In the scope of network computation, the scene graph is formulated as $G = \{B, O, R\}$. $O = \{o_i | i \in \{1, 2, \ldots, n\}\}$ is the object (entity) set and $o_i$ denotes the $i_{th}$ object in the image $I$. And similarly, $B = \{b_i | i \in \{1, 2, \ldots, n\}\}$ is the bounding box set associated to the objects. $R = \{r_k(o_i, o_j) | k \in \{1, 2, \ldots, m\}, i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, n\}, i \neq j\}$ is the relation set of the image $I$, $r_k(o_i, o_j)$ represents the existence or absence of $k_{th}$ relation between subject $o_i$ and object $o_j$.

Then the scene graph generation process is decomposed as:

$$P(G|I) = P(B|I)P(O|B, I)P(R|O, B, I),$$

(5.1)

where $P(G|I)$ is the posterior probability of generated scene graph $G$ given the image $I$. $P(B|I)$ and $P(O|B, I)$ represent the regional proposal and bounding box classification of the object detection module. While $P(R|O, B, I)$ is the critical relation prediction component in a scene graph parser.

Elaborations of building blocks are discussed in the following sections. Sec-
tion 5.2.2 provides an overview of the scene graph generation model and explains the four-step pipeline. Section 5.2.3 introduces the MLRI module with relation implication in the way of ternary multi-label learning. Then the VRotatE module is discussed in detail in Section 5.2.4.

5.2.2 Model Architecture

Though various SGG methods are proposed with distinct structures, they are generalized into four consecutive steps: (1) object detection and feature extraction. This step is invariant for different scene graph generation models. The object detector extracts visual and spatial features of each region of interest (ROI) after the proposal, non-maximum suppression (NMS), and ROIAlign operators (He et al., 2017). Spatial features include normalized locations and sizes of bounding boxes. (2) graph construction/visual context structuring. For the constructed graphs, nodes are typically detected objects which can be sparsely or densely connected by generated edges. MotifNet (Zellers et al., 2018) and IMP (Xu et al., 2017) methods convert the graph as a chain to feed into vanilla RNNs. While VCTree (Tang et al., 2019) converts the graph as a binary tree and uses TreeLSTMs (Tai et al., 2015) to process it. (3) context feature encoding to learn aggregated features for every node and (4) relation prediction with context encoded features. Figure 5.2 illustrates the four steps. In this paper, the proposed MLRI and VRotatE modules are incorporated in step (4), and we will take the existing step (1), (2), and (3) designed in the state-of-the-art works.

Context Graph Construction In a coarse granular view, three context graph construction methods have been proposed to make context propagation feasible in current deep learning frameworks: (1) Chain. It is compatible with sequential RNN-based models by simply traversing all object nodes in a specific order. For example,
IMP (Xu et al., 2017) takes the confidence score after NMS as the sorting criterion. MotifNet tries different criteria and applies the left-to-right ordering method with the best performance. Chain structure is efficient and requires less porting effort on current RNNs. Nevertheless, it oversimplifies the complex interactions among different nodes. Some semantically related neighbors may not be appropriately aggregated during the sequential propagation. (2) Tree. Compared with the flattened chain, the hierarchical tree structure preserves more topological information of the graph. VCTree (Tang et al., 2019) designs Prim’s algorithm-based tree construction method (Prim, 1957) and uses TreeLSTM (Tai et al., 2015) to recursively perform the feature aggregation. (3) Graph. Graph R-CNN (Yang et al., 2018) introduces a relation proposal network to obtain a sparse graph and uses attention as edges’ weight. However, the sparse graph may prune important connections in the early stage, limiting the quality of down-streaming stages such as context encoding and final relation prediction.

The proposed MLRI and VRotatE modules are orthogonal to the context graph construction. We integrate MLRI and VRotatE into a representative SGG method i.e. MotifNet for the demonstration purpose.

**Context Encoding** Context encoding is a feature propagation process fusing neighbors’ information into each node. The propagation routing is guided by the context graph $G_c$ described in the previous step. Conceptually, the context encoding is expressed as:

$$D = \text{RNN}(G_c \{ f_i | i \in 0, 1, \cdots, N \}), \quad (5.2)$$

where $D = [d_1, d_2, \cdots, d_N]$ is the encoded context representation for all objects, $d_i$ is the resulting context encoding feature for the $i_{th}$ object in the dimension of 512, and $f_i$ is feature vector of $i_{th}$ node (object/entity). Particularly, the feature vector
Figure 5.3: Relation Implication Visualization on Visual Genome Dataset, Nodes Are Relations, Weight Numbers on the Directed Edges Are the Conditional Ratio: (a) Positive Implication; (b) Negative Implication; (c) Multi-edge Ratio of Selected 50 Relations.

is concatenated by the visual feature \( v_i \), spatial feature \( s_i \) and context embedding indexed by object label \( l_i \):

\[
f_i = [v_i; s_i; Wl_i], \quad (5.3)
\]

\( v_i \in \mathbb{R}^{4096} \) is ROI features after ROIAlign and flatten operation. \( s_i \in \mathbb{R}^{128} \) is embedded spatial features from the normalized coordinates of the corresponding bounding box:

\[
\begin{bmatrix}
\frac{x}{w_{img}}, & \frac{y}{h_{img}}, & \frac{x + w}{w_{img}}, & \frac{y + h}{h_{img}}, & \frac{wh}{w_{img}h_{img}}
\end{bmatrix}. \quad (5.4)
\]

While \( W \in \mathbb{R}^{200 \times N_o} \) is a learnable look-up table to embed sparse object labels into dense features.

5.2.3 Multi-Label Relation Implication Module

The relationship prediction is based on the pair-wise object features after context encoding. Therefore, the input instances of relation prediction is the Carte-
sian product of the object instances. Three pair-wise features are collected for each subject-object pair: (1) context encoding pair $d_{ij} = MLP([d_i, d_j]) \in \mathbb{R}^{2048}$, (2) spatial feature pair $s_{ij} = MLP([x_i - x_j, y_i - y_j, \log \frac{w_i}{w_j}, \log \frac{h_i}{h_j}]) \in \mathbb{R}^{128}$ and visual union box feature $v_{ij} = \text{ROIAlign}(B_{ij}) \in \mathbb{R}^{4096}$. The final pair-wise feature is fused as:

$$g_{ij} = MLP([d_{ij}; s_{ij}]) \odot v_{ij},$$

(5.5)

where $\odot$ is the element-wise product. Note that ROIAlign here represents ROIAlign and a fully-connected layer.

Then the common practice of state-of-the-art methods to classify predicates is:

$$P(r_{ij}|B, O) = \text{softmax}(W_r g_{ij}),$$

(5.6)

where $W_r \in \mathbb{R}^{N_r \times 4096}$ is the projection matrix. However, as aforementioned, scene graphs are multi-graphs, which means two nodes can have more than one edge. Therefore, the exclusive softmax function may not properly model the multi-edge cases such as the previously discussed hand holding phone example. The current practice is randomly select one edge every time in the training process to let the model learn the multi-edge property during the entire training process. However, the true but uns-elected predicate classification will apply unwanted errors to the model. To further understand the interaction among multi-edge cases, we perform a quantitative analysis on the Visual Genome dataset (Krishna et al., 2017). The results are summarized in Figure 5.3.

There are two types of implication within the multi-edges in this paper: (1) positive implication $P_p(r_m|r_n) = P(r_m(o_i, o_j)|r_n(o_i, o_j))$, which means given the subject $o_i$ and object $o_j$ and one existing relation $r_n$, the conditional probability of another relation $r_m$ occurs in the same subject and object pair; (2) negative implication $P_n(r_m|r_n) = P(r_m(o_j, o_i)|r_n(o_i, o_j))$ defined similarly but with reversed subject and
object pair. As shown in Figure 5.3a, spatial relation on is positively implied by other relations such as riding, looking at and etc with a certain probability. The number on the directed edge riding → on represents the conditional probability \( P(r_{on}(\text{sub}, \text{obj}) \vert r_{ride}(\text{sub}, \text{obj}), \text{multi}(\text{sub}, \text{obj})) \). Similarly, Figure 5.3b shows the negative implication graph. The bar chart of Figure 5.3c reveals that almost all relations can have multi-edge instances. Around 25\% says edges are multi-edges. The most “isolated” relation is growing on which has a 2.2\% multi-edge ratio.

On the other hand, in the conventional softmax classifier, \( \arg\max(\text{softmax}(W_{rij})) \) == background means that the model simply treats the corresponding subject \( i \) and object \( j \) as irrelevant entities. However, the model will fail to capture the potential interaction if there is a relation edge from subject \( j \) to object \( i \), referred as a negative relation in this paper. We hereby introduce the MLRI module as an auxiliary multi-label classifier to exploit the relation information of every subject-object pair. Each predicate/relation \( r \) has a ternary classifier to infer the three possible relations: background \( (\bot \iff r_r(o_i, o_j)) \land (\bot \iff r_r(o_j, o_i)) \), positive relation \( \top \iff r_r(o_i, o_j) \) and negative relation \( \top \iff r_r(o_j, o_i) \).

5.2.4 Visual Rotation Embedding Module

Scene graphs are usually incomplete and biased. The KGE methods can generalize decent relation transformation according to the observed knowledge facts, so that semantically same predicates learned on a set of subject-object pairs can help determine their existence on another rare on even un-seen subject-object pairs.

Rotation Embedding (RotatE) (Sun et al., 2018) method models “subject-relation-object” in semantic vectors \( s, r, \) and \( o, \) respectively, and the relation is represented as a rotation in the complex embedding space: \( s \circ r \approx o \) when the relation exists, and \( s \circ r \neq o \) otherwise. Without introducing computational complexity, RotatE
better models various relation types, especially the symmetric relations compared with TransE. To bridge the RotatE method with the SGG problem, we need to map the features of the detected objects into the complex relation space in a similar manner of recent work VtransE (Zhang et al., 2017):

$$W_s d_s \circ r_r \approx W_o d_o,$$

(5.7)

where $W_s \in \mathbb{R}^{200 \times 512}$ and $W_o \in \mathbb{R}^{200 \times 512}$ are projection matrices for context encoded features of subject and object. $r_r \in \mathbb{C}^{100}$ is the rotation vector associated with relation $r$. To keep the modulus constant after rotation, $r_r$ is enforced as an unit complex vector calculated with Euler’s formula:

$$r_r^\theta = \cos \theta + i \sin \theta,$$

(5.8)

where the rotation angle $\theta \in \mathbb{R}^{100}$ is the learnable vector.

The loss function of VRotatE is similar to original RotatE model:

$$\mathcal{L} = -\sum_{i=1}^{n} \log(\gamma - d(W_s s_i \circ r_r, W_o o_i))$$

$$-\sum_{j=1}^{m} \log(d(W_s s'_j \circ r_r, W_o o'_j) - \gamma),$$

(5.9)

where $d$ is L2 distance by default, $s_i, o_i$ belong to a true relation triplet while $(s'_i, o'_i)$ are from a false relation triplet. $\gamma = 1$ is a distance margin factor.

However, due to the incompleteness of scene graphs, there are a lot of relations logically hold but not annotated as true samples, as we discussed in the previous sections. Thus we perform negative sampling only from false relations that do not exist in both (subject, object) and (object, subject) cases, mitigating the noise introduced by incompleteness.

Finally, the learned complex representations are fused into context encoded fea-
Figure 5.4: Qualitative Results of RIE, Comparing with MotifNet Baseline on the Task PRDCLS. From Left to Right: Input Images, Predicate Inference of MotifNet and Predicate Inference of RIE. Green Edges Are True Positive Results, Red Edges Are False Positive Results.

5.3 Experiments

5.3.1 Datasets and Metrics

Visual Genome (Krishna et al., 2017) is a large-scale dataset dedicated for structural vision tasks with 75729 object classes and 40480 relation classes. There are several reduced versions for scene graph generation. In this paper, we follow the same train/val splits as Xu et al. (2017) with 70% training samples and 30% testing samples in which the most frequent 150 objects and 50 relations are chosen. We followed Tang et al. (2019) to sample a 5k validation set from the training set for parameter tuning. The SGG method on VG is estimated in three tasks:
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Graph Constraint</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SGDET</td>
<td>SGCLS</td>
<td>PRDCLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R@</td>
<td></td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>VRD</td>
<td>-</td>
<td>0.3</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>AssocEmbed</td>
<td>6.5</td>
<td>8.1</td>
<td>8.2</td>
<td>18.2</td>
</tr>
<tr>
<td>IMP</td>
<td>14.6</td>
<td>20.7</td>
<td>24.5</td>
<td>31.7</td>
</tr>
<tr>
<td>G-RCNN</td>
<td>-</td>
<td>11.4</td>
<td>13.7</td>
<td>29.6</td>
</tr>
<tr>
<td>Frequency</td>
<td>-</td>
<td>23.5</td>
<td>27.6</td>
<td>-</td>
</tr>
<tr>
<td>HOSE-Net</td>
<td>-</td>
<td>28.9</td>
<td>33.3</td>
<td>36.3</td>
</tr>
<tr>
<td>MotifNet</td>
<td>24.6</td>
<td>32.3</td>
<td>36.8</td>
<td>34.9</td>
</tr>
<tr>
<td>MOTIFS+RIE</td>
<td><strong>25.0</strong></td>
<td><strong>32.9</strong></td>
<td><strong>37.2</strong></td>
<td><strong>35.5</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>No Graph Constraint</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SGDET</td>
<td>SGCLS</td>
<td>PRDCLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R@</td>
<td></td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>VRD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AssocEmbed</td>
<td>-</td>
<td>9.7</td>
<td>11.3</td>
<td>-</td>
</tr>
<tr>
<td>IMP</td>
<td>-</td>
<td>22.0</td>
<td>27.4</td>
<td>-</td>
</tr>
<tr>
<td>G-RCNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Frequency</td>
<td>-</td>
<td>25.3</td>
<td>30.9</td>
<td>-</td>
</tr>
<tr>
<td>HOSE-Net</td>
<td>-</td>
<td>30.5</td>
<td>36.3</td>
<td>-</td>
</tr>
<tr>
<td>MotifNet</td>
<td>26.8</td>
<td>36.0</td>
<td>42.7</td>
<td>40.0</td>
</tr>
<tr>
<td>MOTIFS+RIE</td>
<td><strong>27.1</strong></td>
<td><strong>36.7</strong></td>
<td><strong>43.3</strong></td>
<td><strong>40.6</strong></td>
</tr>
</tbody>
</table>

Table 5.1: SGG Performances of Various Methods.

(1) Predicate classification (PRDCLS): $P(B|I) = 1$ and $P(O|B,I) = 1$. The ground truth object classes and bounding boxes are provided to predict each object pair’s relation type.
(2) Scene graph classification (SGCLS): \( P(B|I) = 1 \). Given the ground truth annotations of object bounding boxes, predict the object classes and the relation type of each object pair.

(3) Scene graph detection (SGDET): Predict the bounding boxes, the object classes and the relation type of each object pair.

The evaluation metrics is recall @ 20/50/100 (Zellers et al., 2018; Tang et al., 2019; Alexe et al., 2012a). Recall @ \( K \) computes the fraction of correct hits in the top \( K \) confident relationship predictions. Note that no more than one edge is generated by the model for a given pair of objects while calculating recall (Xu et al., 2017; Lu et al., 2016), referred as recall with graph constraint.

5.3.2 Implementation

Following the same practice as the previous works, we use a pre-trained Faster-RCNN (Ren et al., 2017) to detect object bounding boxes and extract ROI features. We reimplement the MotifNet model and add MLRI as well as VRotatE in Pytorch (Paszke et al., 2019). The entire model is trained with SGD optimizer. The batch size is 2 to fit in a single RTX 2080 GPU.

In the training phase, the MLRI module introduces the ternary multi-label classification loss, and the VRotatE module introduces the L2 loss. In the inference phase, MLRI does not contribute the inference directly. We test to add a multi-label score onto the original softmax score, no gain is observed. We argue that the ternary multi-label loss already makes the difference in the weight optimization in training, explicitly modifying the inference score may lead to more semantically plausible but not annotated predictions.
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Graph Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>SGDET</td>
</tr>
<tr>
<td>R@ 20 50 100</td>
<td>20 50 100</td>
</tr>
<tr>
<td>MotifNet</td>
<td>24.6 32.3 36.8</td>
</tr>
<tr>
<td>MTF+MLRI</td>
<td>24.7 32.5 37.0</td>
</tr>
<tr>
<td>MTF+VRotatE</td>
<td>24.9 32.7 37.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Mean Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>SGDET</td>
</tr>
<tr>
<td>R@ 20</td>
<td>50 100 20</td>
</tr>
<tr>
<td>MotifNet</td>
<td>4.3 6.2 7.3</td>
</tr>
<tr>
<td>MTF+MLRI</td>
<td>4.7 6.6 7.8</td>
</tr>
<tr>
<td>MTF+VRotatE</td>
<td>4.8 6.8 7.9</td>
</tr>
</tbody>
</table>

Table 5.2: Ablative Comparison of MLRI and VRotatE. MTF Is the Abbreviation of MotifNet.

<table>
<thead>
<tr>
<th>item</th>
<th>MotifNet</th>
<th>MLRI</th>
<th>VRotatE</th>
</tr>
</thead>
<tbody>
<tr>
<td>#params</td>
<td>60.3M</td>
<td>626K</td>
<td>210K</td>
</tr>
<tr>
<td>#FLOP</td>
<td>2.2G</td>
<td>94.8M</td>
<td>3.8M</td>
</tr>
<tr>
<td>#FPS</td>
<td>9.6</td>
<td>9.4</td>
<td>9.6</td>
</tr>
</tbody>
</table>

Table 5.3: Number of Floating-point Operations (FLOPS), Number of Parameters and Frame per Second (FPS) Required by MLRI and VRotatE.

5.3.3 Overall Performance

Quantitative Analysis. Representative methods are compared in terms of the above three task metrics. The results are summarized in Table 5.1. We integrate the proposed RIE module upon the previous state-of-the-art MotifNet. Improvements on the three tasks with and without graph constraint are observed. Graph constraint recall is the standard metric enforcing at most one edge between two objects. No
graph constraint recall allows multiple relations occur between objects. HOSE-NET achieves the best performance on PRDCLS as it uses a deeper structure embedded network for hierarchical semantic embedding.

**Qualitative Analysis.** Qualitative comparison in Figure 5.4 gives a straightforward view of how the proposed RIE method improves the baseline MotifNet method. It is observed that RIE does a better job at relation retrieval. For example, in the first image about a bag of fruits, the RIE method successfully detects the relation 1-banana in 0-bag and 4-handle on 0-bag while MotifNet fails. On the other hand, RIE will increase the possibility of “false positive” as the 7-orange on 10-table and 2-glove on 6-man indicate, limiting the quantitative recall performance.

### 5.3.4 Ablation Study

We also perform the ablative study to investigate how the two proposed modules affects the base scene graph parser, MotifNet, as shown in Table 5.2. Besides the traditional recall R@K performance that is dominated by some frequent relations, we also take the comprehensive metric mean recall mR@K (Chen et al., 2019; Tang et al., 2019) which computes the mean value of the recall for all relations. We observe that VRotatE helps more on SGDET performace while MLRI achieves more improvement on SGCLS and PRDCLS for both R@K and mR@K. The potential reason is that SGDET does not provide accurate object labels and bounding boxes which are a part of MLRI features. While VRotatE learns auxiliary embeddings to mitigate the influence of inaccurate object detection.
5.3.5 Computational Overhead

The proposed RIE is efficient for both training and inference. Using MotifNet as the context encoding network, MLRI takes the final encoded feature for ternary multi-label classification. VRotatE learns a low-dimensional semantic representation for objects. Table 5.3 shows the number of floating point operations (FLOPs) of a single image inference as well as the number of parameters required by RIE modules, in comparison to the base MotifNet (Zellers et al., 2018). We investigate the test set of Visual Genome split and use the mean value of object number in an image, i.e.12.3, as the batch size of object and sequence length of LSTMs. Here we only calculate the parameters and FLOPs for context encoding module, as the up-streaming modules including feature extractor and object detector are much larger. Compared to the MotifNet context encoding network, the MLRI and VRotatE introduce marginal overhead on parameter memory and computation resources.
Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we present a comprehensive study of deep learning based perception and prediction algorithms and enhancement with relation modeling. We demonstrate the our designs for object detection and trajectory prediction with relations incorporated.

First, we propose a graph-assisted detection method, GAR, that leverages object-object and object-scene relations in object detection. Experiments show prominent accuracy improvement, especially on the categories which are highly correlated to scene context. Moreover, our GAR method has the advantage of computation efficiency: it requires less FLOPs and parameter memory than previous RNN-based methods, making GAR a practical solution in real-time applications. Last but not the least, the framework of GAR can also be extended to model more sophisticated relations than the demonstrated co-occurrence, such as spatial relation with localization/scale and semantic relation with knowledge graphs.

Second, on the other hand, we propose a trajectory prediction method, DAT-RNN, that handles the diversity of trajectories and accurate modeling of neighboring relations. By differentiating diverse segments of the complicated trajectories, the DAM module makes more confident predictions separately and then fuses the features to obtain a balanced one. Furthermore, AAM uses augmented attention features with anomaly salience and captures mutual relations more accurately. Experiments on several datasets validate the improvement of the prediction under various circumstances.
Last but not least, we propose a relation implication enhanced approach for scene graph generation, RIE, that aims to better leverage the ground truth information in the dataset. Particularly, MLRI module tackles the multi-label issue of relations and takes advantage of the intrinsic negative relation which is ignored by traditional relation classifiers. VRotatE learns a semantic representations by using the knowledge graph embedding method, which enriches the object features and solves the symmetric relation modeling. Experiments show observable accuracy improvement on the three main scene graph generation tasks: PRCLS, SGCLS and SGDET. Furthermore, our RIE method has the advantage of computation efficiency. It requires marginal FLOPs and parameters compared with the base context encoding network, making RIE a practical solution in real-time applications.

6.2 Future Work

In the future, it is promising to continue the work about relation mining, especially semantic relations for visual understanding in a higher level. With the semantic relations, we can infer the content and activity as well as associated objects and subjects, so that we can make better prediction of the movement or future activities. It will requires more knowledge and technology of Natural Language Processing (NLP) and Visual Question Answering (VQA), etc.
REFERENCES


