Graph Neural Networks for Multimodal Learning and Representation

by

Mahmoud Khademi

M.Sc., McMaster University, 2013
M.Sc., Sharif University of Technology, 2009
B.Sc., Isfahan Azad University, 2003

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2019

Approval

Name:	**Mahmoud Khademi**
Degree:	**Doctor of Philosophy (Computer Science)**
Title:	**Graph Neural Networks for Multimodal Learning and Representation**
Examining Committee:	**Chair:** Fred Popowich Professor
	Oliver Schulte Senior Supervisor Professor
	Martin Ester Supervisor Professor
	James Delgrande Internal Examiner Professor
	Abdolreza Mirzaei External Examiner Assistant Professor Department of Electrical and Computer Engineering Isfahan University of Technology
Date Defended:	**December 19, 2019**

Abstract

Recently, several deep learning models are proposed that operate on graph-structured data. These models, which are known as graph neural networks, emphasize on methods for reasoning about non-Euclidean data. By combining end-to-end and handcrafted learning, graph neural networks can supply both relational reasoning and compositionality which are extremely important in many emerging tasks. This new paradigm is also consistent with the attributes of human intelligence: a human represents complicated systems as compositions of simple units and their interactions. Another important feature of graph neural networks is that they can often support complex attention mechanisms, and learn rich contextual representations by sending messages across different components of the input data.

The main focus of this thesis is to solve some multimodal learning tasks by either introducing new graph neural network architectures or extending the existing graph neural network models and applying them to solve the tasks. I address three tasks: visual question answering (VQA), scene graph generation, and automatic image caption generation. I show that graph neural networks are effective tools to achieve better performance on these tasks.

Despite all the hype and excitements about the future influence of graph neural networks, an open question about graph neural networks remains: how can we obtain the (structure of) the graphs that graph neural networks perform on? That is, how can we transform sensory input data such as images and text into graphs. A second main emphasis of this thesis is, therefore, to introduce new techniques and algorithms to address this issue. We introduce a generative graph neural network model based on reinforcement learning and recurrent neural networks (RNNs) to extract a structured representation from sensory data. The specific contributions are the following:

We introduce a new neural network architecture, Multimodal Neural Graph Memory Networks (MN-GMN), for the VQA task. A key issue for VQA is how to reason about information from different image regions that is relevant for answering the question. Our novel approach uses graph structure with different region features as node attributes and applies a recently proposed powerful graph neural network model, Graph Network (GN), to reason about objects and their interactions in the scene context. The flexibility of GNs allows us to integrate bimodal sources of local information, text and visual, both within and across

each modality. Experiments show MN-GMN outperforms the state-of-the-art on Visual7W and VQA v2.0 datasets and achieves comparable to the state-of-the-art results on CLEVR dataset.

We propose a new algorithm, called Deep Generative Probabilistic Graph Neural Networks (DG-PGNN), to generate a scene graph for an image. The input to DG-PGNN is an image, together with a set of region-grounded captions (RGCs) and object bounding-box proposals for the image. To generate the scene graph, DG-PGNN constructs and updates a new model, called a Probabilistic Graph Network (PGN). A PGN can be thought of as a scene graph with uncertainty: it represents each node and each edge by a CNN feature vector and defines a probability mass function (PMF) for node-type (object category) of each node and edge-type (predicate class) of each edge. The DG-PGNN sequentially adds a new node to the current PGN by learning the optimal ordering in a Deep Q-learning framework, where states are partial PGNs, actions choose a new node, and rewards are defined based on the ground-truth. After adding a node, DG-PGNN uses message passing to update the feature vectors of the current PGN by leveraging contextual relationship information, object co-occurrences, and language priors from captions. The updated features are then used to fine-tune the PMFs. Our experiments show that the proposed algorithm significantly outperforms the state-of-the-art results on the Visual Genome dataset for the scene graph generation.

We present a novel context-aware attention-based deep architecture for image caption generation. Our architecture employs a Bidirectional Grid LSTM, which takes visual features of an image as input and learns complex spatial patterns based on a two-dimensional context, by selecting or ignoring its input. The Grid LSTM can be seen as a graph neural network model with a grid structure. The Grid LSTM has not been applied to the image caption generation task before. Another novel aspect is that we leverage a set of local RGCs obtained by transfer learning. The RGCs often describe the properties of the objects and their relationships in an image. To generate a global caption for the image, we integrate the spatial features from the Grid LSTM with the local region-grounded texts, using a two-layer Bidirectional LSTM. The first layer models the global scene context such as object presence. The second layer utilizes a novel dynamic spatial attention mechanism, based on another Grid LSTM, to generate the global caption word-by-word while considering the caption context around a word in both directions. Unlike recent models that use a soft attention mechanism, our dynamic spatial attention mechanism considers the spatial context of the image regions. Experimental results on the MS-COCO dataset show that our architecture outperforms the state-of-the-art.

Keywords: graph neural networks, deep generative models, deep reinforcement learning, scene graph generation, visual question answering, automatic image caption generation

Dedication

This thesis is dedicated to my mother and father.

Acknowledgements

I am very grateful to my senior supervisor Oliver Schulte for his guidance in this thesis, bringing energy and helpful insight into our discussions. I would also like to thank Greg Mori and Anoop Sarkar, for numerous discussions and motivating comments. My research has been punctuated by two internships at Microsoft Research in Cambridge and Montreal Institute for Learning Algorithms (Quebec Artificial Intelligence Institute). I would like to thank Miltiadis Allamanis, Marc Brockschmidt, and Jian Tang for interesting and often valuable discussions. Moreover, I would like to thank Martin Ester and James Delgrande. Last but not least, I would like to thank my family for their love and care.

Table of Contents

Approval ii

Abstract iii

Dedication v

Acknowledgements vi

Table of Contents vii

List of Tables ix

List of Figures x

1 Introduction 1

1.1 Application of Graph Neural Networks . . . 3
1.2 Contributions . . . 3
1.3 Multimodal Learning Tasks . . . 5
1.4 The Organization of the Thesis . . . 9

2 Graph Neural Network Models 11

2.1 Graph Neural Networks (GNNs) . . . 11
2.2 Non-local Neural Networks (NLNN) . . . 12
2.3 Gated Graph Neural Networks . . . 13
2.4 Message Passing Neural Networks . . . 14
2.5 Graph Attention Network (GAT) . . . 15
2.6 Graph Networks . . . 16
2.7 Graph Auto-encoders (GAEs) . . . 17
2.8 Graph Convolutional Networks (GCNs) . . . 18
2.9 Variational Graph Auto-encoders (VGAEs) . . . 19
2.10 Generative Graph Networks . . . 20

3 Image Caption Generation with Hierarchical Contextual Visual Spatial Attention 21

3.1 Introduction . . . 21
3.2 Related Work . . . 22
3.3 Proposed Model . . . 23
3.3.1 Deep CNN for Extracting 2D Feature Maps . . . 24
3.3.2 Bidirectional Grid LSTM . . . 24
3.3.3 Deep (Two-Layer) Bidirectional LSTM with a Dynamic Visual Spatial Attention . . . 27
3.3.4 Integrating with Region-Grounded Texts . . . 31
3.3.5 Training Details . . . 31
3.4 Experiments . . . 32
3.4.1 Data, Metrics and Experimental Setup . . . 32
3.4.2 Results and Discussion . . . 33

4 Multimodal Neural Graph Memory Networks for Visual Question Answering 38
4.1 Related Work . . . 41
4.2 Our Proposed Architecture . . . 42
4.2.1 Answer Module . . . 47
4.3 Experiments . . . 48

5 Graph Neural Networks for Scene Graph Generation 56
5.1 Deep Generative Probabilistic Graph Neural Networks . . . 56
5.1.1 Related Work . . . 59
5.1.2 Proposed DG-PGNN Algorithm . . . 60
5.1.3 Experiments . . . 69
5.1.4 Examples for VQA and Scene Graph Generation Tasks . . . 71
5.2 Proposed Algorithm for Dynamic Gated Graph Neural Networks . . . 75
5.2.1 New Reinforcement Learning Architecture for Graph Structure Generation . . . 76
5.2.2 Experiments . . . 82
5.3 Discussion: Deep Generative Probabilistic Graph Neural Networks versus Dynamic Gated Graph Neural Networks . . . 84

6 Discussion and Conclusion 88

Bibliography 90

List of Tables

Table 1.1 The designs examined in this thesis. We move from lower-level to higher-level representations. Novel components that originate in our research are underlined. Abbreviations are RGC for region-grounded captions, GN for Graph Networks (Battaglia et al., 2018), GGNN for gated graph neural networks (Li et al., 2015), ICG for image caption generation, and SG for scene graph generation. 10

Table 2.1 Notations used throughout this chapter. 11

Table 3.1 BLEU-1,2,3,4 and METEOR scores on MS-COCO test set. We only compare with the results that have been officially published. For fairness, we only compare with the results which employ comparable CNNs such as GoogLeNet, VGGNet and ResNet. The - denotes that the result is not reported. Models that use ResNet are denoted by °. The † denotes the results are reported on validation set, since the results on the test set are not available. 35

Table 4.1 Accuracy percentage of various models on VQA-v2.0. 50
Table 4.2 Accuracy Percentage on Visual7W. 50
Table 4.3 Accuracy Percentage on CLEVR. 51

Table 5.1 Recall for predicate classification, scene graph classification, and scene graph generation tasks on VG dataset. 70
Table 5.2 Accuracy Percentage on Visual7W. 71
Table 5.3 Accuracy Percentage on CLEVR. 72
Table 5.4 Experimental Results of the relationship phrase detection and relationship detection tasks on the VG1.4-b dataset. 84

List of Figures

Figure 1.1 An image and its ground truth scene graph. The nodes represent the objects in the image, and the edges show the relationships between them. 4

Figure 1.2 An image and its 5 captions from MS-COCO dataset. The visual contextual information such as co-occurrence of *dog* and *frisbee* and the spatial relationship between these two objects help annotators to recognize the *frisbee*. All annotators ignore the trees in the background, instead, they attend to the *dog* and *frisbee*. 6

Figure 1.3 An image from the Visual Genome dataset with the region-grounded captions, object attributes, relationships triplets, and visual questions. The region-grounded captions and relationships are helpful to answer the questions. 8

Figure 3.1 The architecture of our model 25

Figure 3.2 The architecture of a Grid LSTM cell. 26

Figure 3.3 Example captions generated by our model. 34

Figure 3.4 Visualization of the attention. The 7×7 heatmaps represents value of weight $\upsilon_{r,s}^{t}$ for t-th word in the generated caption. Each value corresponds to a specific region in the image. 36

Figure 3.5 Two example of mistakes of our model. 37

Figure 4.1 An example from Visual Genome. The region-grounded captions provide useful clues to answer questions. 39

Figure 4.2 Multimodal Neural Graph Memory Networks for VQA. The visual features/ region-grounded captions are extracted from white/black bounding-boxes and are used as node features to construct the visual/textual Graph Network. 43

Figure 4.3 N-GMN versus MN-GMN. The MN-GMN provides the correct answer using *a cloudy blue sky*. 51

Figure 4.4 N-GMN versus MN-GMN. The MN-GMN provides the correct answer using *white and black tennis shoes*. 52

Figure 4.5 Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention. 53
Figure 4.6 Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention. 54
Figure 4.7 Example VQA with N-GMN on CLEVR dataset. 55
Figure 4.8 Example VQA with N-GMN on CLEVR dataset. 55

Figure 5.1 Scene graph represents semantic content of an image via a graph. We leverage region-grounded captions to construct a scene graph. Scene graph is useful to answer visual questions such as *What color is the umbrella?* (features of node *umbrella*) and *What is the woman in white pants holding?* (relationship triplet *woman-holding-umbrella*). 57
Figure 5.2 Scene graph generation using DG-PGNN. 61
Figure 5.3 Examples of scene graphs generated on Visual Genome dataset. . . 72
Figure 5.4 Examples of VQA by DG-PGNN† on CLEVR dataset. For each question, the nodes and edges with the highest attention weight are specified. 73
Figure 5.5 An example of scene graph generation and VQA by DG-PGNN on Visual7W dataset. For each question, the nodes and edges with the highest attention weight are specified. DG-PGNN can correctly detect spatial relationships *wall beside zebra* and *zebra behind fence*. Also, DG-PGNN can correctly recognize *zebra in snow* (instead of *zebra in grass*). 74
Figure 5.6 Dynamic Gated Graph Neural Networks for scene graph generation task. Given an image and its candidate object bounding-boxes, we simultaneously build the graph and assign node-types and edge-types to the nodes and edges in a Deep Q-Learning framework. We use two separate Q-networks to select the actions, i.e. node-types and edge-types. We use two fully-connected layers with ReLU activation function to implement each Q-network. The input to the Q-networks is the concatenation of the GGNN representation of the current graph and a global feature vector from the image. The search for the scene graph continues with the next node in a breadth-first search order. 80
Figure 5.7 Some scene graphs generated by our model. 85
Figure 5.8 Some scene graphs generated by our model. 86

Chapter 1

Introduction

Over the last decade, deep learning has revolutionized artificial intelligence (AI) and machine learning areas. By leveraging new computational resources and large training datasets, models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders achieved great success in many machine learning tasks such as object detection (Ren et al., 2015), machine translation (Luong et al., 2015), speech recognition (Hinton et al., 2012), and image caption generation (Donahue et al., 2015). Given no opening theory except the game rules, AlphaZero, the chess engine created by Google DeepMind based on deep reinforcement learning, has beaten the world's best chess-engine (Silver et al., 2017). These standard deep learning models often focus on an "end-to-end" design principle and try to keep away from "handcrafted" learning and explicit structure. For instance, in machine translation sequence-to-sequence models have shown efficient results without applying parse trees (Sutskever et al., 2014; Bahdanau et al., 2014).

As an alternative to a complete end-to-end design, this thesis explores intermediate representations of the information in an image that support reasoning for advanced vision tasks like visual question answering and image caption generation. The intermediate representations have a graph structure, where nodes correspond to local regions of the image and edges capture interactions, either spatial or abstract. The graph structures extracted from the image are passed to neural graph processing technique that extract rich latent features useful for downstream reasoning. Our main finding is that there is an accuracy-usefulness trade-off for graph-structured representations: the higher-level representations are the most useful when they are accurate (e.g. an accurate scene graph for VQA in the CLEVR dataset, described in Chapter 5). On the other hand, higher-level representations are more difficult to generate accurately, which is why we introduce a new formalism for capturing uncertainty in a scene graph (Chapter 5). Lower-level representations can be more reliably extracted from the image. For example, the grid cells used as nodes in Chapter 3 can be deterministically computed from the input image. The multi-modal graph structure is intermediate in both accuracy and usefulness. The thesis explains this trade-off in details and explores options for resolving it.

Despite recent advances, a huge gap between human and AI systems remains in many areas. Several critical essays (Marcus, 2018a; Yuille and Liu, 2018; Marcus, 2018b; Lake and Baroni, 2017; Pearl, 2018) and analyses have emphasized important problems that deep learning models encountered in tasks which need reasoning about relations and graph-structured data such as scene and language understanding.

The standard end-to-end deep learning models such as multi-layer perceptron (MLP), CNNs and RNNs avoid compositionality and imposing explicit structure on the data. Also, they do not assume pairwise interactions between different components of the input data. Moreover, these models are not permutation-invariant which is a crucial feature in many tasks. For instance, consider the simple task of predicting the center of a set of points in a plane. The order of the points is not important for this task. But, if we use an MLP to solve this task, the MLP needs all possible permutations of the points to learn the task.

CNNs can process images efficiently and extract rich features by imposing locality and translation invariance. However, they can only perform on grids and Euclidean data such as image pixels, but not on a set of object bounding-boxes. Similarly, RNNs can learn complex temporal patterns by imposing temporal invariance to the input sequence. But, they can only perform on data with linear temporal structure such as natural language sentences.

For instance, consider the task of learning an embedding for each token in a source code or predicting the name of a variable given the context around the variable in a source code. Unlike learning word embedding and "fill in the blank" task in natural language processing, for this task an effective model needs to exploit the rich structure of the source code, i.e. the pairwise interactions between different (possibly distant) tokens. As a result, applying an RNN such as a bidirectional long short-term memory (LSTM) for this task is not efficient.

Recently, several methods tried to tackle these issues by proposing deep learning models and architectures that operate on graph-structured data (Bruna et al., 2013; Li et al., 2015; Defferrard et al., 2016; Battaglia et al., 2018; Kipf and Welling, 2016b). These models, which are known as *graph neural networks*, emphasize methods for reasoning about non-Euclidean data and learning to represent the pair-wise interactions between different components of the input data. For example, to learn an embedding for each token in a file of source code, a graph can be constructed using different edge types to model syntactic and semantic interactions between various tokens (nodes), using the program's abstract syntax tree (Allamanis et al., 2017). The resulting graph can encode both the text and the semantic information of the source code. Then, a graph neural network may be applied to the graph to obtain a rich embedding for each token.

By combining end-to-end and handcrafted learning, graph neural networks can supply both relational reasoning and compositionality which are extremely important in many emerging tasks. This new paradigm is also consistent with the attributes of human intelligence: a human represents complicated systems as compositions of simple units and their interrelationships. Another important feature of graph neural networks is that they can

often support complex attention mechanisms, and learn rich contextual representations by sending messages across different components of the input data. For the aforementioned example, to learn an embedding for a given token in a source code, some tokens around the given token may be more important. A graph neural network model with an attention mechanism which learns an attention weight for each node (token) in the constructed graph may be applied to this task efficiently.

1.1 Application of Graph Neural Networks

There are many real-world applications which involve analyzing graph-structured data. For example, in drug discovery a molecule is represented by a set of atoms (nodes) and bounds (edges) between the atoms. A scene graph is a visually-grounded digraph for an image, where the nodes represent the objects and the edges show the relationships between them (see Figure 1.1). The scene graph is a very useful representation in many tasks such as image caption generation and visual question answering.

In natural language processing, a Knowledge Graph represents a set of interconnected descriptions of entities which can be useful for tasks such as textual question answering. In e-commence, a graph neural network model can produce precise recommendations by learning relationships between customers and products. In a social network, individuals, organizations, and their social interactions can be effectively represented by a graph. Moreover, graph neural networks may be applied to create new drug molecules and predict drug-target interactions.

1.2 Contributions

The main focus of this thesis is to solve some real-world multimodal learning tasks by either introducing new graph neural network architectures or extending the existing graph neural network models and applying them to solve the tasks. I address three multimodal learning tasks: visual question answering (VQA), scene graph generation, and image caption generation. I show that graph neural networks are effective tools to achieve better performance on these tasks.

Despite all the hype and excitements about the future influence of graph neural networks, an open question about graph neural networks remains: how can we obtain the (structure of) graphs that graph neural networks perform on? That is, how can we transform sensory input data such as images and text into graphs. This is an active research area. A second main emphasis of this thesis is, therefore, to introduce new techniques and algorithms to address this issue by applying reinforcement learning and recurrent neural networks (RNNs) to extract a structured representation (scene graph) from sensory data (image).

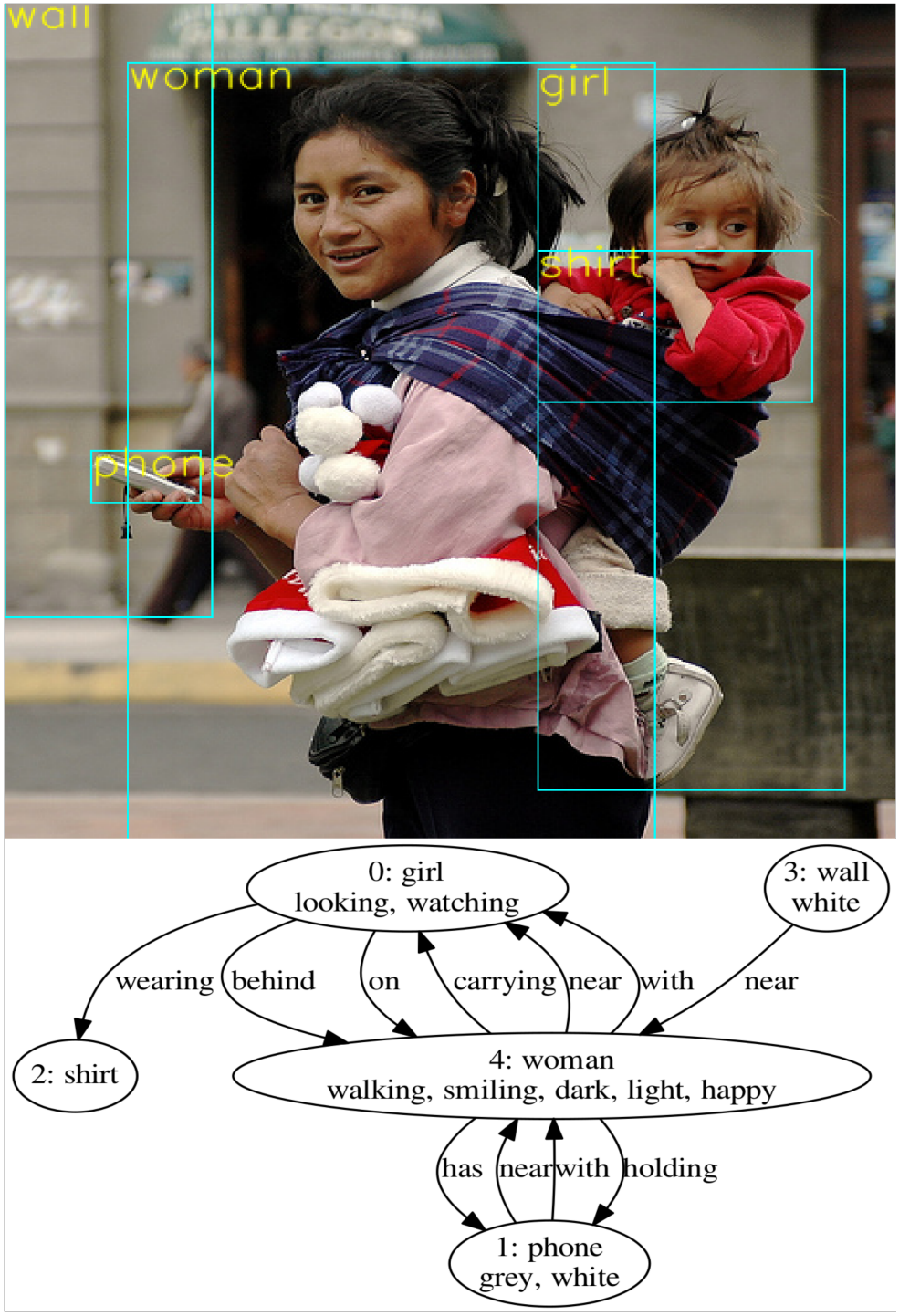


Figure 1.1: An image and its ground truth scene graph. The nodes represent the objects in the image, and the edges show the relationships between them.

1.3 Multimodal Learning Tasks

In this section, I briefly discuss the three multimodal learning tasks that I address in this thesis. For each task, I provide the intuitions behind applying graph neural networks to these tasks.

Recently, multimodal learning for both vision and natural language has become an active research area of AI. In particular, many studies have made rapid progress on the task of automatic image caption generation (Karpathy and Fei-Fei, 2015; Xu et al., 2015; Chen and Zitnick, 2014; Vinyals et al., 2014; Donahue et al., 2015; Mao et al., 2014b; You et al., 2016; Mao et al., 2014a). Most of them are developed based on combining Recurrent Neural Networks (RNNs) and Convolutional Neural Network (CNNs). Given an image, the automatic image caption generation is to generate a natural language sentence which describes the image. This task is very challenging since not only must a trained model represent the semantic information of the given image, but also it must express that information in natural language.

Recently, several research groups have developed visual recognition models based on Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) that significantly improve the quality of the generated captions (Chen and Zitnick, 2014; Mao et al., 2014a; Vinyals et al., 2014; Fang et al., 2015; Donahue et al., 2015; Lebret et al., 2015; You et al., 2016; Yao et al., 2016; Wu et al., 2016; Yang et al., 2016). However, there is still a long way to go for an intelligent system to match the accuracy of human image descriptions. Two important aspects in automatic image caption generation are incorporating the *visual contextual information* and the *visual spatial attention* (see Figure 1.2). The visual contextual information such as scene context (e.g. a park) and object co-occurrences may serve as a contextual cue to facilitate recognizing objects and describing the image. The visual spatial is directing the attention to a particular location in a scene or image. We propose a graph neural network architecture which has the capacity to model these two aspects and can improve the quality of the generated captions.

To advance the frontiers of AI research, a new task called visual question answering (VQA) (Antol et al., 2015) or image question answering (Image QA) (Malinowski and Fritz, 2014a,b, 2015) has been recently proposed. Given an image and a freeform natural language question about the image, the VQA task is to produce an accurate natural language answer. This task is similar to real-world situations, such as helping blind people, both the questions and answers are open-ended.

The VQA has many potential applications. For example, a VQA system may provide navigation for vision-less people, or when a blind user reads his Facebook or Twitter page via an image caption generation system, he may use a VQA system to ask about the images iteratively and get more information. VQA has also application in image retrieval. For instance, to find all images in a snowy scene, we may ask *is it snowing?* from the images in

Airborne dog catching a frisbee in the park.
A dog jumping and catching a frisbee in his mouth.
A black dog jumping to catch a frisbee.
A large black and white dog catching an orange frisbee in a park.
A black and white dog jumps to catch a frisbee.

Figure 1.2: An image and its 5 captions from MS-COCO dataset. The visual contextual information such as co-occurrence of *dog* and *frisbee* and the spatial relationship between these two objects help annotators to recognize the *frisbee*. All annotators ignore the trees in the background, instead, they attend to the *dog* and *frisbee*.

the database. Other potential applications are image search, human-robot interaction and early education.

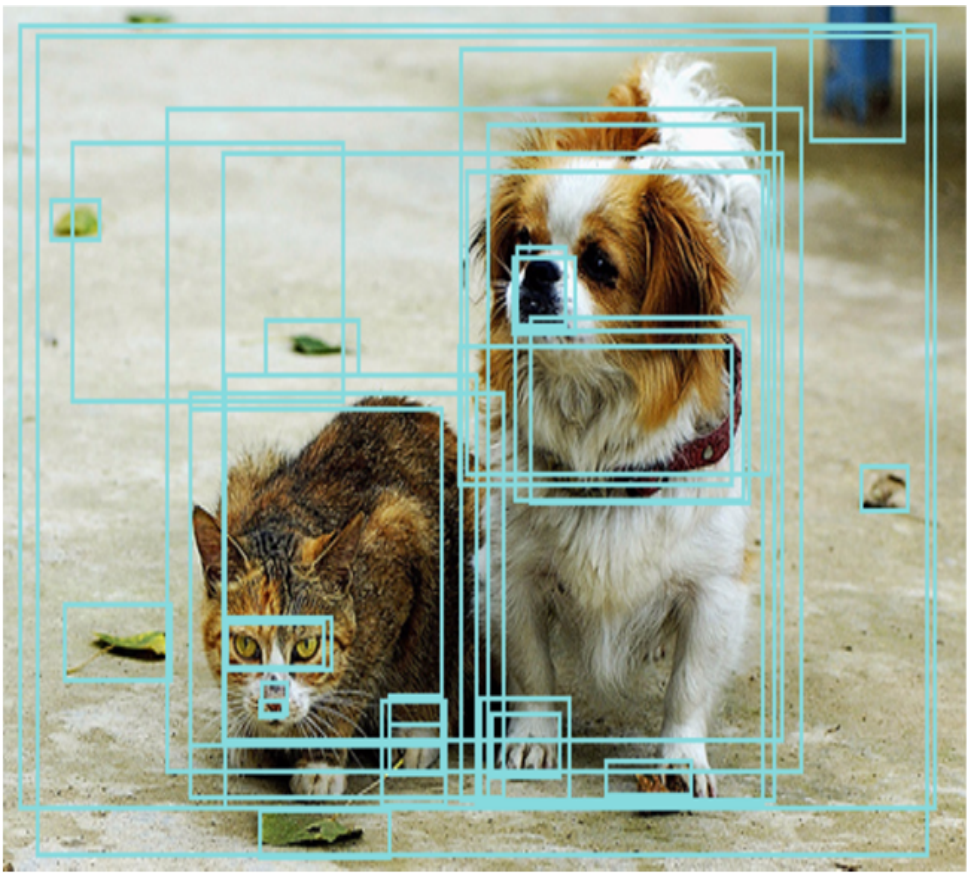
Unlike automatic image caption generation which only requires a generic caption of an image, the VQA task needs more interaction between the image and a question, since an answer provided by a VQA model requires being conditioned on both the image and question. Visual questions selectively target various areas of an image, including background details and underlying context. As a result, a system that succeeds at VQA typically needs a more detailed understanding of an image and sophisticated reasoning than a system producing generic image captions. Such a system must be able to recognize the objects and their visual relationships in the image, identify the attributes of these objects, reason about the role of each object in the scene context, and provide a natural language answer based on the given question.

The VQA can involve lots of sub-problems in computer vision, for instance

- Object recognition - *what objects are in this image?*
- Activity recognition - *what is this girl doing?*
- Object detection - *where is the bird?*
- Scene classification - *is it sunny or cloudy?*
- Image caption generation - *how do you describe this image?*
- Fine-grained recognition - *what kind of boat is the far left blue boat?*
- Knowledge base reasoning - *is the glass full?*
- Counting - *how many dogs are there?*
- Common-sense reasoning - *what street will we be on if we turn left?*

Since an ideal VQA system must solve many of these computer vision sub-problems, it may be regarded as a part of a Turing Test for computer vision systems (Malinowski and Fritz, 2014b; Geman et al., 2015). A Visual Turing Test examines the ability of an intelligent system to probe whether it can understand an image like a human (Malinowski and Fritz, 2014b; Geman et al., 2015). The VQA can be seen as a Visual Turing Test that also needs to understand simple natural language questions about an image. If a VQA system passes this test on sufficiently large datasets, then possibly much of computer vision will be solved. Despite the difficulties in achieving the above ambitious goal, recently, an exciting set of models has been developed by the research community for the VQA task.

In spite of recent advances in VQA, current VQA models often fail on sufficiently new samples, converge on an answer after listening to only a few words of the question, and do not alter their answers across different images (Agrawal et al., 2016; Kafle and Kanan, 2016). Our point of departure is that success at VQA task requires a rich representation of the objects and their visual relationships in an image, which identifies the attributes of



Regions: a green leaf, a red dog collar, a small brown and white dog, a tabby cat, green cat's eyes, etc.
Attributes: leaf is green, collar is red, dog is small, cat is tabby, eyes is green, paw is white, nose is pink, etc.
Relationships: collar OF dog, leaf ON ground, pole behind animals, leaf next to cat, collar around neck, dog ON sidewalk, etc.
QA examples:
Q1: what animals are present? A: dog and cat.
Q2: who is wearing a collar? A: dog.
Q3: where is the cat? A: left.

Figure 1.3: An image from the Visual Genome dataset with the region-grounded captions, object attributes, relationships triplets, and visual questions. The region-grounded captions and relationships are helpful to answer the questions.

these objects, and supports reasoning about the role of each object in the scene context. A key issue for VQA is how to reason about information from different image regions that is relevant for answering the question. Our novel approach uses a graph structure with different region features as node attributes and applies a recently proposed powerful graph neural network model, Graph Network (GN), to reason about objects and their interactions in the scene context. The flexibility of GNs allows us to integrate bimodal sources of local information, text and visual, both within and across each modality (see Figure 1.3).

Given an image, the scene graph generation task is to predict a directed graph where the nodes represent the objects bounding boxes with an object category such as *man* and *horse*, and an edge represent a predicate with a predicate class such as *riding* and *above* which shows a relationship between two nodes. A scene graph is a very useful rich semantic representation of an image which can facilitate tasks such as image caption generation, image generation from textual descriptions of the images, and visual question answering.

A way to obtain a scene graph is to train two separate detectors: an object detector and a predicate detector. Given a testing image, the trained detectors are applied to generate a set of relationship triplets. However, this approach leads to poor results, since it does not consider contextual information of the input image. To generate accurate scene graphs, a model needs to incorporate visual contextual information such as global scene context, object co-occurrences, and spatial relationships between the objects that occur in the image as well as language priors. That is, to obtain a rich embedding for an object (node), an efficient model needs to consider the presence of the other objects and their pair-wise relationships in the image. For example, if we know that a node has an incoming edge from another node which represents a *man* and the type of the edge is *wearing*, then the type of the node is likely *coat* or *shirt*, but not *book*; the predicate class of an edge from a node with object category *woman* to another node with object category *bicycle* is likely *riding* or *beside*, but not *eating*. Graph neural networks are tailored to this task, since they can efficiently compute a contextualized representation (embedding) by sending massages across different nodes (objects bounding-boxes).

1.4 The Organization of the Thesis

The organization of this thesis is as follows: In Chapter 2, we provide a review of graph neural network models and discuss the limitations of them. This review mainly focuses on the graph neural network models that we either extend or apply them to the multimodal tasks. In Chapter 3, we present a novel, context-aware, attention-based deep architecture for image caption generation. Our architecture employs a Bidirectional Grid LSTM, which takes visual features of an image as input and learns complex spatial patterns based on two-dimensional context, by selecting or ignoring its input. The Grid LSTM can be seen as a graph neural network model with a grid structure. In Chapter 4, we introduce a new

neural network architecture, Multimodal Neural Graph Memory Networks (MN-GMN), for visual question answering (VQA) task.

Chapter 5 proposes two generative graph neural networks architectures for scene graph generation task: Deep Generative Probabilistic Graph Neural Networks (DG-PGNN), and Dynamic Gated Graph Neural Networks (D-GGNN). In particular, we focus on scene graph generation to evaluate these new architectures. However, the proposed architectures can be seen as a general framework for generative graph neural networks. We also show that the scene graphs constructed by DG-PGNN improve performance on the visual question answering task for questions that need reasoning about objects and their interactions in the scene context. Chapter 6 provides a discussion, limitations of this work, conclusions and future research directions. Table 1.1 summarizes the structure of this thesis.

Chapter	Graph Structure	Nodes	Features	Technique	Task
Chapter 3	Grid	Grid cells	Visual	Grid LSTM Dynamic Attention	ICG
Chapter 4	Relational Multi-modal	Bounding-Boxes	Visual/RGC	GN	VQA
Chapter 5	Probabilistic Graph	Objects	Visual/RGC	Generative Reinforcement Learning GN	SG

Table 1.1: The designs examined in this thesis. We move from lower-level to higher-level representations. Novel components that originate in our research are underlined. Abbreviations are RGC for region-grounded captions, GN for Graph Networks (Battaglia et al., 2018), GGNN for gated graph neural networks (Li et al., 2015), ICG for image caption generation, and SG for scene graph generation.

We focus on the Graph Network (Battaglia et al., 2018) formalism, which is a state-of-the-art graph deep processing technique. A graph neural network is not optimal for a grid graph structure, because it is designed for flexible graph topologies, so we apply a Grid LSTM (Kalchbrenner et al., 2015) structure in Chapter 3.

Chapter 2

Graph Neural Network Models

In this chapter, I provide a review of graph neural networks. These are neural network models that take as input a graph with initial features for the nodes (possible also edge features), and return rich features that aggregate information from across the graph. There are several reviews on graph neural networks (Zhang et al., 2018b; Zhou et al., 2018; Wu et al., 2019; Battaglia et al., 2018). This chapter mainly focuses on the recently developed graph neural network models that are extended in later chapters. Table summarizes the notations that I use throughout this chapter.

Notation	Description	Notation	Description
$\mathcal{G}$	A graph	$\mid \mathcal{E} \mid$	Number of edges in $\mathcal{E}$
$\mathcal{E}$	A set of edges in a graph	N, $\mid \mathcal{V} \mid$	Number of nodes in $\mathcal{V}$
$\mathcal{V}$	A set of nodes in a graph	$\mathbf{v}_i$, $\mathbf{h}_i$, $\mathbf{x}_i$	Node attributes of node i
$\|\|$	Concatenation operator	$\mathcal{N}_v$	Neighbours of node v
$\mathbf{A}$	Adjacency matrix of a graph	K	Number of edge-types
M	Number of node-types	$\mathbf{P}$	Transition matrix of a graph

Table 2.1: Notations used throughout this chapter.

2.1 Graph Neural Networks (GNNs)

Gori et al. (2005); Scarselli et al. (2008) proposed one of the first works on graph neural networks called Graph Neural Networks (GNNs). A GNN is defined based on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The graph may have a node label (node-type) $l_v \in \{1, \ldots, M\}$ for each node v, and an edge label (edge-type) $l_e \in \{1, \ldots, K\}$ for each edge $e = (u, v)$, where M is the number of node-types, and K is the number of edge types. Each node v has a node embedding (node representation) $\mathbf{h}_v \in \mathbb{R}^D$. Let $\mathbf{h}_{\mathcal{A}} = \{\mathbf{h}_a \mid a \in \mathcal{A}\}$, where $\mathcal{A}$ is a set of nodes, and $l_{\mathcal{B}} = \{l_b \mid b \in \mathcal{B}\}$, where $\mathcal{B}$ is a set of nodes or edges. Also, let $\text{IN}(v)$ be the set of of all nodes incoming to node v, and $\text{OUT}(v)$ be the set of all nodes outgoing from node v. Moreover, let $\mathcal{N}_v$ be the set of all nodes neighboring v, and $\text{CO}(v)$

be the set of all edges incoming to or outgoing from v. That is, $\mathcal{N}_v = \text{IN}(v) \cup \text{OUT}(v)$ and $\text{CO}(v) = \{(a, b) \in \mathcal{E} \mid a = v \text{ or } b = v\}$.

With initial node representations (node embeddings) $\mathbf{h}_v^0$ for each node v, the GNNs use a propagation model to iteratively update the node representations until convergence as

$$\mathbf{h}_v^t = f\Big(l_v, l_{\text{CO}(v)}, l_{\mathcal{N}_v}, \mathbf{h}_{\mathcal{N}_v}^{t-1}\Big) \tag{2.1}$$

where f is a function. To guarantee the convergence, f must be contraction mapping. That is, there must be some nonnegative real number $0 \leq \alpha < 1$ such that for all $\mathbf{x}$ and $\mathbf{y}$,

$$|| f(\mathbf{x}) - f(\mathbf{y}) ||\leq \alpha || \mathbf{x} - \mathbf{y} || . \tag{2.2}$$

To guarantee the convergence, when f is implemented by a neural network, a penalty term on the Jacobian matrix of the model parameters should be used. Scarselli et al. (2008) proposed the following form for f,

$$f\Big(l_v, l_{\text{CO}(v)}, l_{\mathcal{N}_v}, \mathbf{h}_{\mathcal{N}_v}^{t-1}\Big) = \sum_{u \in \text{IN}(v)} \hat{f}(l_v, l_{(u,v)}, l_u, \mathbf{h}_u^{t-1}) + \sum_{u \in \text{OUT}(v)} \hat{f}(l_v, l_{(v,u)}, l_u, \mathbf{h}_u^{t-1}) \tag{2.3}$$

where $\hat{f}$ is a linear function. The parameters of $\hat{f}$ are determined by the structure of the labels as

$$\hat{f}(l_v, l_{(u,v)}, l_u, \mathbf{h}_u^{t-1}) = \mathbf{W}^{(l_v, l_{(u,v)}, l_u)} \mathbf{h}_u^{t-1} + \mathbf{b}^{(l_v, l_{(u,v)}, l_u)} \tag{2.4}$$

where $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters. Let T be the number of iterations for convergence. An output model $o_v = g(\mathbf{h}_v^T, l_v)$ is used to obtain an output o_v for each node v, where g is a neural network layer. The advantage of GNNs is that to compute the gradients there is no need to store intermediate states. However, imposing the penalty term may restrict the expressivity of the GNN model (Li et al., 2015).

2.2 Non-local Neural Networks (NLNN)

RNNs are used to capture long-term dependencies in sequential data such as language, and CNNs are used to learn long-distance dependencies in images using a stack of large receptive fields. However, both of these deep learning models operate on a local neighborhood in time or space. To learn long-term/long-distance dependencies, these local operations must be repeated through time or space. This is computationally inefficient and may lead to optimization difficulties. To address this issue, Wang et al. (2018) introduced a graph neural network model called Non-local Neural Networks (NLNN) to represent long-range dependencies in space or time. The non-local operation can be seen as an extension to the non-local mean operation which is used in computer vision. This operation computes an

output signal at a position (in space or time) using a weighted sum of the input signals at all different positions. Formally, given a set of input signals (feature vectors) $\mathbf{h}_j$, the general non-local operation is defined as follows

$$\mathbf{h}'_i = \frac{1}{c(\mathbf{h})} \sum_j s(\mathbf{h}_i, \mathbf{h}_j) f(\mathbf{h}_j) \tag{2.5}$$

where s is a scalar valued function that computes the "similarity" between a pair of input signals, $f(\mathbf{h}_j)$ is a transformation of $\mathbf{h}_j$ which can be implemented with a neural network layer, and $\frac{1}{c(\mathbf{h})}$ normalizes the resulting output signal. This operation is called no-local since all possible positions indexed by j are considered to compute the output signal.

Wang et al. (2018) suggest multiple instantiations of the general non-local operation with different functions s. A choice for s is Gaussian function defined as $s(\mathbf{h}_i, \mathbf{h}_j) = \exp(\mathbf{h}_i^\top \mathbf{h}_j)$. An embedded Gaussian is defined as $s(\mathbf{h}_i, \mathbf{h}_j) = \exp\left(\phi(\mathbf{h}_i)^\top \theta(\mathbf{h}_j)\right)$, where $\phi(\mathbf{h}_i) = \mathbf{W}_\phi \mathbf{h}_i$ and $\theta(\mathbf{h}_j) = \mathbf{W}_\theta \mathbf{h}_j$. Here, $\mathbf{W}_\phi$ and $\mathbf{W}_\theta$ are trainable weight matrices. For both of these choices $c(\mathbf{h}) = \sum_j s(\mathbf{h}_i, \mathbf{h}_j)$.

Other choices for s are dot-product and concatenation defined as $s(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{h}_i^\top \mathbf{h}_j$ and $s(\mathbf{h}_i, \mathbf{h}_j) = \text{ReLU}\Big(\mathbf{W}_s^\top (\phi(\mathbf{h}_i) \mid\mid \theta(\mathbf{h}_j))\Big)$, respectively, where $\mathbf{W}_s$ is a trainable weight matrix, and ReLU is the rectified linear unit activation function. For both of these choices $c(\mathbf{h})$ is set to the number of possible positions indexed by j.

2.3 Gated Graph Neural Networks

Formally, a Gated Graph Neural Network (GGNN) (Li et al., 2015) takes as input a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each directed edge $e = (v, u) \in \mathcal{E}$ has an edge-type $l_e \in \{1, \ldots, K\}$, where K is the number of edge-types (classes, labels). The given graph may also contain node-types $l_v \in \{1, \ldots, M\}$ for each node v, where M is the number of node-types. Given the graph structure and the edge types, a GGNN iteratively computes a new node embedding $\mathbf{h}_v^t \in \mathbb{R}^d$, $t = 1, \ldots, T$, for each node v via a propagation model, where d is the embedding size.

For each node v, a node representation must take into account the information from every linked neighbor of v. Let $\mathbf{A}_k$ be the adjacency matrix for edge-type k. That is, $\mathbf{A}_k(u, v) = 1$, if there is an edge with type k from node u to node v, otherwise $\mathbf{A}_k(u, v) = 0$. The matrix $\mathbf{A}_k$ determines how nodes in the graph communicate with each other via edges of type k.

Let $\mathbf{x}_v \in \mathbb{R}^M$ be the one-hot representation of node v's type. The recurrence of the propagation for computing node embeddings $\mathbf{h}_v^t \in \mathbb{R}^d$ is initialized with the padded one-hot

representation of node v's type, i.e. $\mathbf{h}_v^0 = (\mathbf{x}_v, \mathbf{0}) \in \mathbb{R}^d$. The update equations are

$$\mathbf{a}_v^t = \sum_{k=1}^{K} W_k(\mathbf{H}^{t-1}\mathbf{A}_{k:v}) \tag{2.6}$$

$$\mathbf{h}_v^t = \texttt{GRU}(\mathbf{a}_v^t, \mathbf{h}_v^{t-1}). \tag{2.7}$$

Here, the matrix $H^t \in \mathbb{R}^{d\times|\mathcal{V}|}$ has node representations $\mathbf{h}_v^t$, $v = 1, \ldots, |\mathcal{V}|$, as its column vectors, $W_k \in \mathbb{R}^{d\times d}$ is a weight matrix for edges of type k that we learn, and $\mathbf{A}_{k:v}$ is the v'th column of $\mathbf{A}_k$. The term $\mathbf{H}^{t-1}\mathbf{A}_{k:v}$ represents the sum of latent feature representations for all nodes u that are linked to node v by an edge of type k. Thus, $\mathbf{a}_v^t$ combines activations from incoming edges of all types for node v, aggregating *messages* from all nodes that have an edge to v. The GGNN first computes the node activations $\mathbf{a}_v^t$ for each node v. Then, a Gated Recurrent Unit (GRU)(Cho et al., 2014) is used to update node representations for each node by incorporating information from the previous time-step. Other RNNs such as LSTMs may also be applied. The GRU computation is defined as

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{a}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.8}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{a}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.9}$$

$$\tilde{\mathbf{h}}_t = \psi\big(\mathbf{W}_h\mathbf{a}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\big) \tag{2.10}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{2.11}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h, \mathbf{b}_z, \mathbf{b}_r, \mathbf{b}_h$ are trainable parameters, σ is the sigmoid activation function, and ψ is the hyperbolic tangent activation function. The computations defined by the equations (2.6) and (2.7) are repeated for a fixed number of time steps T. The choice of T depends on the task. For each node, the hidden state from the last time step is used as the node representation.

Johnson (2017) proposed an extension to Gated Graph Neural Network which can learn to construct and modify a graph using textual input. They applied the proposed extension to solve some bAbI tasks successfully. A main problem, however, is that the model selects an ordering to add the nodes to the graph which may not be optimal. To resolve this issue, in Chapter 4, we propose a generative extension to the GGNN, called Dynamic Gated Graph Neural Networks, which constructs a graph from the input data in a reinforcement learning framework.

2.4 Message Passing Neural Networks

Gilmer et al. (2017) proposed a graph neural networks framework called Message Passing Neural Networks (MPNNs). The MPNNs composed of two stages, a propagation (message passing) stage and a readout stage. The propagation stage, which is repeated for T time

steps, is based on a message function M_t and a node update function U_t

$$\mathbf{a}_v^t = \sum_{w \in \mathcal{N}_v} M_t(\mathbf{h}_v^{t-1}, \mathbf{h}_w^{t-1}, \mathbf{e}_{vw}) \tag{2.12}$$

$$\mathbf{h}_v^t = U_t(\mathbf{a}_v^t, \mathbf{h}_v^{t-1}) \tag{2.13}$$

The readout stage computes a global representation for the input graph as

$$\mathbf{u} = R(\{\mathbf{h}_v^T \mid v \in V\}) \tag{2.14}$$

The message passing function M_t and node update function U_t may take various forms such as RNN and MLP. The readout function R can be implemented as

$$R(\{\mathbf{h}_v^T \mid v \in V\}) = \sum_{v \in \mathcal{V}} \sigma(f(\mathbf{h}_v^T, \mathbf{h}_v^0)) \odot g(\mathbf{h}_v^T) \tag{2.15}$$

where f and g are two neural networks, and σ is the sigmoid activation function.

2.5 Graph Attention Network (GAT)

In Veličković et al. (2017), the authors proposed Graph Attention Network (GAT) by stacking several graph attention layers. An attention layer computes an attention weight for each node pair (i, j) as

$$\alpha_{ij} = \frac{\exp\Big(\texttt{LeakyReLU}(\mathbf{a}^\top(\mathbf{W}\mathbf{h}_i \,||\, \mathbf{W}\mathbf{h}_j))\Big)}{\sum_{k \in \mathcal{N}_i} \exp\Big(\texttt{LeakyReLU}(\mathbf{a}^\top(\mathbf{W}\mathbf{h}_i \,||\, \mathbf{W}\mathbf{h}_k))\Big)} \tag{2.16}$$

where $||$ is the concatenation operator, and `LeakyReLU` is the leaky rectified linear unit activation function which allows a small, positive gradient when the unit is not active (Maas et al., 2013). The attention weight α_{ij} computes the contribution of neighbour j to node i. The input to the attention layer is $\{\mathbf{h}_i \mid i = 1 \dots N\}$, where $\mathbf{h}_i \in \mathbb{R}^D$, and N is the number of nodes in the graph. The weight matrix $\mathbf{W} \in \mathbb{R}^{D \times D'}$ is a linear transformation, and $\mathbf{a} \in \mathbb{R}^{2D'}$ is a weight vector. The new node features are obtained via

$$\mathbf{h}_i' = \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\Big) \tag{2.17}$$

To stabilize the learning algorithm, the authors also proposed to apply a multi-head attention mechanism. This uses K separate attention mechanisms to obtain the outputs and

then concatenates them or takes the average of them by

$$\mathbf{h}'_i = ||_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j \Big) \tag{2.18}$$

$$\mathbf{h}'_i = \sigma\Big(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j \Big) \tag{2.19}$$

where α_{ij}^k is the attention weight obtained by k-th attention mechanism.

2.6 Graph Networks

Battaglia et al. (2018) proposed a graph neural network framework called graph networks (GN). The GN extend several other graph neural networks such as message-passing neural networks (MPNN) (Gilmer et al., 2017), and non-local neural networks (NLNN) (Wang et al., 2018).

In a GN framework, a graph is represented by a 3-tuple $\mathcal{G} = (\mathbf{u}, \mathcal{V}, \mathcal{E})$, where $\mathbf{u}$ is a graph-level (global) attribute. The notation $\mathcal{V} = \{\mathbf{v}_i\}_{i=1:N}$ denotes a set of node attributes, where $\mathbf{v}_i$ is a node attribute of node i, and N is the number of nodes. The notation $\mathcal{E} = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:M}$ denotes a set of edges, where $\mathbf{e}_k$ is an edge attribute for the edge going from node s_k to node r_k, and M is the number of edges. A GN block has three update functions ϕ and three aggregation functions ρ defined as

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) \tag{2.20}$$

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) \tag{2.21}$$

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) \tag{2.22}$$

$$\bar{\mathbf{e}}'_i = \rho^{e \to v}(\mathcal{E}'_i) \tag{2.23}$$

$$\bar{\mathbf{e}}' = \rho^{e \to u}(\mathcal{E}') \tag{2.24}$$

$$\bar{\mathbf{v}}' = \rho^{v \to u}(\mathcal{V}') \tag{2.25}$$

where $\mathcal{E}'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i, k=1:M}$ is the set of resulting per-edge outputs for each node i, $\mathcal{V}' = \{\mathbf{v}'_i\}_{i=1:N}$, and $\mathcal{E}' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:M}$. Given an input graph, a GN block updates the graph using the update and aggregation functions. The computational steps in a GN are represented in Algorithm 1. The ϕ^e is mapped over entire edges to calculate per-edge updates, ϕ^v is mapped over entire nodes to calculate per-node updates, and ϕ^u is used to update the global attribute. The function ρ's should be unvarying to permutations of their inputs, and must be flexible to a varying number of arguments, such as maximum, summation, etc. In Chapter 4, we propose a probabilistic generative extension to the GN model.

Algorithm 1: Computational steps in a Graph Network block (Battaglia et al., 2018)

Input : A graph $G = (\mathbf{u}, \mathcal{V}, \mathcal{E})$
Output: The updated graph $G' = (\mathbf{u}', \mathcal{V}', \mathcal{E}')$
Function GraphNetwork($\mathcal{E}$, $\mathcal{V}, \mathbf{u}$)
for $k \leftarrow 1$ **to** M **do**
  $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$ ▷ *Compute new edge attributes*
end
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{E}'_i \leftarrow \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i, k=1:M}$
  $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}(\mathcal{E}'_i)$ ▷ *Aggregate edge attributes for each node*
  $\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$ ▷ *Compute new node attributes*
end
$V' \leftarrow \{\mathbf{v}'_i\}_{i=1:N}$
$\mathcal{E}' \leftarrow \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:M}$
$\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}(\mathcal{E}')$ ▷ *Aggregate edge attributes for the whole graph*
$\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}(\mathcal{V}')$ ▷ *Aggregate node attributes for the whole graph*
$\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$ ▷ *Compute new global attribute*
return $(\mathbf{u}', \mathcal{V}', \mathcal{E}')$

2.7 Graph Auto-encoders (GAEs)

Autoencoders (Vincent et al., 2008) are broadly applied to unsupervised learning tasks. Tian et al. (2014) applied autoencoders to graph-structured data by considering the adjacency matrix or the transition matrix (an $N \times N$ square matrix that provides the probabilities of going from one node to another) as node features. In their work, they used a reconstruction loss to obtain a low-dimensional representation for each node as follows

$$\min_{\mathbf{\Theta}} \mathcal{L} = \sum_{i=1}^{N} || \mathbf{P}(i,:) - \hat{\mathbf{P}}(i,:) ||_2 \tag{2.26}$$

$$\hat{\mathbf{P}}(i,:) = g(\mathbf{h}_i), \mathbf{h}_i = f(\mathbf{P}(i,:)) \tag{2.27}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ is the low-dimensional representation of node i, $\mathbf{P}$ is the transition matrix, $\hat{\mathbf{P}}$ is the reconstructed transition matrix, f is the encoder, g is the decoder, $\mathbf{\Theta}$ are trainable parameters, and $d \ll N$. The functions f and g are implemented using a multi-layer perceptron with several hidden layers.

Wang et al. (2016) changed the loss function in (2.27) using the adjacency matrix instead of the transition matrix, applying different weights for zero and non-zeros elements of the adjacency matrix, and adding a regularization term as

$$\min_{\mathbf{\Theta}} \mathcal{L} = \sum_{i=1}^{N} || \left(\mathbf{A}(i,:) - g(\mathbf{h}_i)\right) \odot \mathbf{b}_i ||_2 + \alpha \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{A}(i,j) || \mathbf{h}_i - \mathbf{h}_j ||_2 \tag{2.28}$$

where $b_{ij} = 1$ if $\mathbf{A}(i,j) = 0$, otherwise $b_{ij} = \beta$. Here, α and β are hyper-parameters. Intuitively, the regularization term is used to obtain similar low-dimensional representations for two nodes if the nodes are directly connected.

2.8 Graph Convolutional Networks (GCNs)

Bruna et al. (2013) proposed to extend the convolution operation for graph-structured inputs by leveraging the graph Laplacian matrix. The normalized graph Laplacian matrix of an undirected graph is defined as $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal degree matrix ($\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$), and $\mathbf{A}$ is the adjacency matrix of the graph. The normalized graph Laplacian matrix is symmetric and positive-semidefinite with eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix of eigenvalues, and $\mathbf{U}$ is the matrix of eigenvectors of the normalized graph Laplacian matrix. Let $\mathbf{x} \in \mathbb{R}^N$ be a signal. The graph Fourier transform of $\mathbf{x}$ is defined as $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$, with inverse graph Fourier transform $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. Concretely, the graph convolution operation $\star_G$ of signal $\mathbf{x}$ and filter $\mathbf{g}_\theta \in \mathbb{R}^N$ is defined as

$$\mathbf{x} \star_G \mathbf{g}_\theta = \mathbf{U}(\mathbf{U}^\top\mathbf{x} \odot \mathbf{U}^\top\mathbf{g}_\theta) = \mathbf{U}\mathbf{\Theta}\mathbf{U}^\top\mathbf{x} \tag{2.29}$$

where $\odot$ is the Hadamard product, and $\mathbf{\Theta}$ is an $\mathbb{R}^{N\times N}$ diagonal matrix defined as $\mathbf{\Theta} = \mathbf{\Theta}(\mathbf{\Lambda}) = \text{diag}(\mathbf{U}^\top\mathbf{g})$. The $\mathbf{\Theta}$ can be thought of as a function of the eigenvalues of the normalized graph Laplacian matrix. In Bruna et al. (2013), the authors proposed a graph convolutional layer defined by

$$\mathbf{X}^{k+1}_{:,j} = \sigma(\sum_{i=1}^{f_{k-1}} \mathbf{U}\mathbf{\Theta}^k_{i,j}\mathbf{U}^\top\mathbf{X}^k_{:,i}), j = 1, \ldots f_k \tag{2.30}$$

where $\mathbf{X}^k_{:,i}$ is the input node embeddings, N is number of nodes, f_{k-1} is the number of input channels, f_k is the number of output channels, k is the layer index, and $\mathbf{\Theta}^k_{i,j}$ is a trainable diagonal matrix.

Computing the graph convolution using (2.29) is computationally expensive, specially for large graphs, since computing the eigendecomposition of $\mathbf{L}$ is expensive. Also, the multiplication with the matrix $\mathbf{U}$ is in $\mathcal{O}(N^2)$. To overcome these issues, Defferrard et al. (2016) proposed to approximate the diagonal matrix $\mathbf{\Theta}$ using Chebyshev polynomials as

$$\mathbf{\Theta} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{\Lambda}}) \tag{2.31}$$

where $\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I}_N$, in which λ_{max} is the largest eigenvalue of the normalized graph Laplacian matrix, and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^K$ is a vector of coefficients. The Chebyshev polynomials are defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$.

Since $(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top)^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\top$, the graph convolution can now be computed as

$$\mathbf{x} \star_G \mathbf{g}_\theta \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x} \tag{2.32}$$

where $\tilde{\mathbf{L}} = \frac{2}{\lambda_{max}}\mathbf{L} - \mathbf{I}_N$.

Unlike (2.29), the complexity of the above equation is in $\mathcal{O}(M)$, where M is the number of edges in $\mathcal{E}$. Moreover, (2.32) is K-localized. That is, it only depends on the nodes that are less than $K+1$ steps away from the center node. To prevent overfitting, Kipf and Welling (2016a) proposed a first-order approximation of (2.32) ($K = 1$). They also approximate $\lambda_{max} \approx 2$, assuming the neural network trainable parameters can be adjusted to this change. To further reduce the number of parameters, they also set $\theta = \theta_0 = \theta_1$. This results in the following equation for the graph convolution operation

$$\mathbf{x} \star_G \mathbf{g}_\theta = \theta(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{x}. \tag{2.33}$$

For multi-dimensional graph signals, Kipf and Welling (2016a) proposed the following graph convolution layer

$$\mathbf{X}^{k+1} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}^k\mathbf{\Theta} \tag{2.34}$$

2.9 Variational Graph Auto-encoders (VGAEs)

Kipf and Welling (2016b) introduced a frameworks called variational graph auto-encoders (VGAE) for unsupervised learning on graph-structured data by integrating GCN into a variational auto-encoder (VAE) framework (Rezende et al., 2014; Kingma and Welling, 2013). Let $\mathbf{Z}$ be an $N \times F$ matrix of latent variables $\mathbf{z}_i \in \mathbb{R}^F$, and $\mathbf{X}$ be an $N \times D$ matrix of node features. The VGAE introduces the following inference model which is parameterized via a two-layer GCN as an encoder

$$q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}) \tag{2.35}$$

$$\tag{2.36}$$

where $q(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$, $\mathbf{M} = \mathrm{GCN}_\mu(\mathbf{X}, \mathbf{A})$ is a matrix of vectors $\boldsymbol{\mu}_i$, and $\log \mathbf{\Sigma} = \mathrm{GCN}_\sigma(\mathbf{X}, \mathbf{A})$. The VGAE model uses an inner product between each pair of the latent variables to define a generative model as

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(\mathbf{A}_{ij} \mid \mathbf{z}_i, \mathbf{z}_j), \tag{2.37}$$

where $p(\mathbf{A}_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$, and σ denotes the sigmoid activation function. The VGAE optimizes the following cost function to find node embeddings $\mathbf{Z}$:

$$\mathcal{L}(\mathbf{Z}) = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \mid\mid p(\mathbf{Z})], \tag{2.38}$$

where $\mathrm{KL}[q(.) \mid\mid p(.)]$ is the Kullback-Leibler divergence between $q(.)$ and $p(.)$.

2.10 Generative Graph Networks

You et al. (2018) introduced a model, called Graph Convolutional Policy Network (GCPN), based on reinforcement learning to generate molecular graphs. The GCPN can incorporate complicated and non-differentiable domain-specific rules to generate a graph. This model formulates the graph generation as a Markov Decision Process where an action predicts a link. Experiments show the effectiveness of GCPN in molecular graph generation task.

In De Cao and Kipf (2018), the authors proposed a model called MolGAN to generate small molecular graphs by adjusting the generative adversarial networks (GANs) to generate graph structures. They merge their approach with reinforcement learning to generate molecules with certain desired chemical attributes. In Chapter 4, we introduce two generative graph neural networks for scene graph generation task.

Chapter 3

Image Caption Generation with Hierarchical Contextual Visual Spatial Attention

3.1 Introduction

This chapter uses the lowest-level graph-structure representation in the thesis. The nodes are fixed-cell grids. Edges represent deterministic spatial adjacency relationships. The graph processing technique applied is the Grid LSTM (Kalchbrenner et al., 2015). The task is generating an image caption.

Automatically generating a description for an image is a fundamental problem in computer vision and scene understanding. This task is very challenging since not only a trained model must recognize objects in an image, but also it must represent the properties of the objects and their relationships in natural language. Recently, several research groups have developed visual recognition models based on Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) that significantly improved the quality of the generated captions (Chen and Zitnick, 2014; Mao et al., 2014a; Vinyals et al., 2014; Fang et al., 2015; Donahue et al., 2015; Lebret et al., 2015; You et al., 2016; Yao et al., 2016; Wu et al., 2016; Yang et al., 2016). However, there is still a long way to go for an intelligent system to match the accuracy of human image descriptions.

One of the most important aspects of our visual recognition system is incorporating *contextual information* such as scene context (e.g. a farm), object presence, and object co-occurrence to describe a scene. For example, if we are asked to predict a caption for an image knowing only that it includes a *farm*, a *horse*, and a *human*, we will probably produce a sentence such as *a boy is riding the horse*, or *a man is next to the horse*. If we also consider the spatial relationship between the horse and the human, we can describe the image more accurately. Another crucial capability of our visual system is *visual spatial attention* (Posner, 1980), which directs attention to a particular location in a scene or image. Spatial attention enables us to give priority to a region within our visual field. The visual

spatial attention is most important when the scene contains background clutter; humans do not describe everything in a scene, but instead look at the important regions and objects.

Inspired by these remarkable capabilities of the human visual system, we propose a hierarchical contextual attention-based deep architecture for image caption generation. The major components of our architecture function as follows. (1) Learn complex spatial patterns in a scene that aggregate local information from regions in both spatial directions. This component utilizes a recent model, called Grid LSTM, adding the advantages of a two-dimensional LSTM to a deep CNN. (2) Generate region-grounded texts for image regions using transfer learning from a region-grounded caption dataset. The region-grounded texts often describe the properties of the objects and their relationships in an image. (3) The Grid LSTM and region-grounded text generator provide informative spatial and textual features. Our main component integrates these two input modalities, to generate a caption using a novel dynamic spatial attention mechanism. This component utilizes a Deep Bidirectional LSTM. The Deep Bidirectional LSTM incorporates a new attention mechanism that selects relevant regions dynamically while generating a caption. This attention mechanism is implemented by (another) Grid LSTM. The Deep Bidirectional LSTM has a hierarchical structure: the first layer models the global scene context such as scene class (e.g. a farm), presence of objects, and co-occurrence of the words which facilitates the image caption generation, while the second layer generates a caption which describes the image. Unlike recent attention-based models, which learn a simple fixed weight for each region of the image, the Grid LSTM allows our attention mechanism to take into account the two-dimensional spatial context and order of the image regions. Our components are carefully designed and connected to be effective for the image caption generation task. Our model outperforms the state-of-the-art performance on MS-COCO dataset. Lesion studies show the value added by the separate components. The content of this chapter has been published in the proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (Khademi and Schulte, 2018b).

3.2 Related Work

Since image caption generation requires a comprehensive understanding of an image and capability to communicate that information via natural language, it is related to different areas in computer vision, machine learning, and natural language processing. On the language side, models based on RNNs have been shown to produce state-of-the-art results on various tasks. RNNs are suitable for sequential data of varying lengths and can learn complicated temporal dynamics. But, because of the vanishing gradient problem (Hochreiter and Schmidhuber, 1997), they have difficulties in learning long-term dependencies. This drawback has been overcome by introducing LSTMs (Hochreiter and Schmidhuber, 1997). On the image side, CNNs such as ResNet, GoogLeNet and VGGNet have recently shown great

success for visual recognition tasks such as image classification and object detection. These models are pre-trained on large image datasets such as ImageNet and are widely accessible. Recently, Kalchbrenner et al. (2015) proposed an extension to LSTM, called Grid LSTM, which can encode 2D signals such as images. They used Grid LSTM to classify digits. Our architecture introduces an extension of Grid LSTM which takes into account spatial context around an image region in all directions.

Several recent papers leverage the power of CNNs and RNNs for image caption generation problem (Fang et al., 2015; Chen and Lawrence Zitnick, 2015; Chen and Zitnick, 2014; Vinyals et al., 2014; Donahue et al., 2015; Lebret et al., 2015; You et al., 2016; Mao et al., 2014a; Lu et al., 2017; Chen et al., 2017). Most of these works represent an image using a feature vector at the very top layer of a pre-trained CNN. This approach may lose spatial information relevant to the caption. Moreover, since these models represent a whole image with a single feature vector, they are not robust to background clutter. Karpathy and Fei-Fei (2015) instead proposed a model based on a bidirectional RNN which scores the similarity between snippets of the caption, and the image regions generated by Region CNN object detectors (Girshick et al., 2014). However, they found that feeding the detected region features, instead of full image features, to their model deteriorates the caption generation performance. The main challenge is that some of the regions are not participating in the target caption. Also, it is difficult to find the right order to feed the image regions to the model at the test time. In this work, we provide a solution to these problems by representing an image with 2D feature maps, and introducing a new attention mechanism which can learn to focus on important regions of the image and ignore the other parts.

Closely related to our work, Xu et al. (2015) proposed a model which can learn to fix its gaze on salient objects while generating the corresponding words. They computed a positive attention weight for each location in the input image using a multilayer perceptron conditioned on the previous generated word. In another related work, You et al. (2016) employed an attention model to combine visual features and visual concepts such as words and objects in an RNN that generates the caption. Unlike these works, instead of applying a simple soft attention, we leverage the power of Grid LSTMs to dynamically attend to the important regions of an image. Also, our model considers spatial context along both vertical and horizontal directions in its attention mechanism.

3.3 Proposed Model

Our model has three components (see Figure 3.1): (1) A deep CNN for extracting image features. (2) A Bidirectional Grid LSTM (BiGrid LSTM) finds complex spatial patterns. This component considers only the image, not the words that already have been generated. (3) A caption generator which includes a Deep Bidirectional LSTM with a dynamic spatial attention mechanism, a word detector to represent global scene context, a region-grounded

caption encoder, and a softmax layer to generate the next word in the caption. This component dynamically attends to different regions of the image while it generates the caption. We firstly describe the details of the model components. Then, we discuss training details.

3.3.1 Deep CNN for Extracting 2D Feature Maps

We apply a CNN to extract visual features. We experiment with the 16-layer VGGNet (Simonyan and Zisserman, 2014) and ResNet (He et al., 2015). For VGGNet, we extract 512 $R \times S$ $(R, S = 7)$ feature maps of the last max pooling layer. The input to this CNN is an image of size 224×224. Thus, each $(224/R) \times (224/S)$ region (r, s) $(r = 1, ..., R)$ and $(s = 1, ..., S)$ is represented by a feature vector of size 512. For ResNet, we extract 1024 $R \times S$ $(R, S = 14)$ feature maps of layer *res4b35x*. Then, we project the feature vector that the CNN extracts from region (r, s) to obtain two vectors $\mathbf{x}_{r,s}$ and $\mathbf{x}'_{r,s}$ of size m, where m is 256 for VGGNet, and 512 for ResNet. These vectors are used as input to our Bidirectional Grid LSTM.

3.3.2 Bidirectional Grid LSTM

The outputs of the CNN, i.e. $\mathbf{x}_{r,s}$ and $\mathbf{x}'_{r,s}$, only describe location (r, s) of an input image, without considering the global information of the image in two spatial directions. Also, some of the image regions are not important for generating the caption. To address these issues, we propose to apply a Grid LSTM to an $R \times S$ grid, corresponding to the regions of the input image. A Grid LSTM is a grid of LSTM cells that can be applied to encode an image. The cells on the grid share the same trainable parameters.

Figure 3.2 shows the basic architecture of a Grid LSTM cell. The cell executes computation with two LSTM cells along two spatial directions. The Grid LSTM with m' hidden states gets as input two input feature vectors $\mathbf{x}_{r,s}$ and $\mathbf{x}'_{r,s} \in \mathbb{R}^m$, two hidden vectors $\mathbf{h}_{r,s-1}$, $\mathbf{h}'_{r-1,s} \in \mathbb{R}^{m'}$, and two memory vectors $\mathbf{c}_{r,s-1}$, $\mathbf{c}'_{r-1,s} \in \mathbb{R}^{m'}$. It gets $\mathbf{h}_{r,s-1}$ and $\mathbf{c}_{r,s-1}$ from the previous Grid LSTM cell in horizontal direction. Also, $\mathbf{h}'_{r-1,s}$ and $\mathbf{c}'_{r-1,s}$ are coming from the previous Grid LSTM cell in vertical direction. A Grid LSTM cell consists of two input gates $\mathbf{i}_{r,s}$,$\mathbf{i}'_{r,s}$, two forget gates $\mathbf{f}_{r,s}$, $\mathbf{f}'_{r,s}$, two output gates $\mathbf{o}_{r,s}$,$\mathbf{o}'_{r,s}$, two input modulation gates $\mathbf{g}_{r,s}$, $\mathbf{g}'_{r,s}$, and two memory cells $\mathbf{c}_{r,s}$,$\mathbf{c}'_{r,s}$, corresponding to horizontal and vertical directions, respectively. The forget gates learn how much of the previous memory should be kept, and the input gates controls how much of the input should be read. Similarly, the output gates control how much of the memory cell should be carried to the hidden states. These gates enable the Grid LSTM to learn complicated distant spatial dynamics and attend to the important regions of the image by reading, writing and erasing the information from the memory cells.

Formally, let $\sigma : \mathbb{R} \mapsto (0, 1)$, $\sigma(x) = (1 + \exp(-x))^{-1}$ and $\phi : \mathbb{R} \mapsto (-1, 1)$, $\phi(x) = 2\sigma(2x) - 1$ be the *sigmoid* and *hyperbolic tangent* nonlinearity, respectively. At each step,

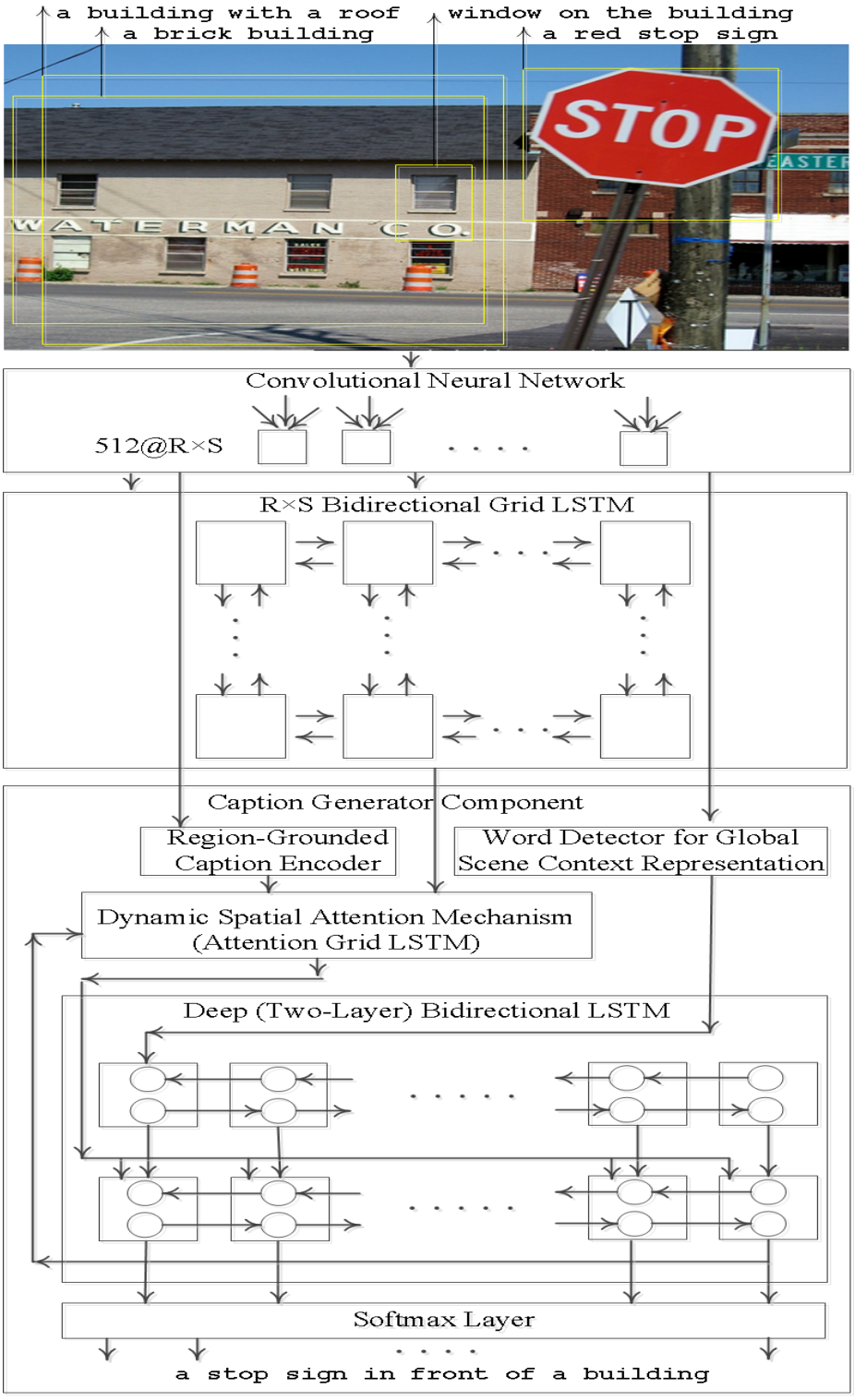


Figure 3.1: The architecture of our model

the Grid LSTM outputs two new hidden vectors $\mathbf{h}_{r,s}$, $\mathbf{h}'_{r,s}$, and two new memory vectors $\mathbf{c}_{r,s}$, $\mathbf{c}'_{r,s}$ as follows

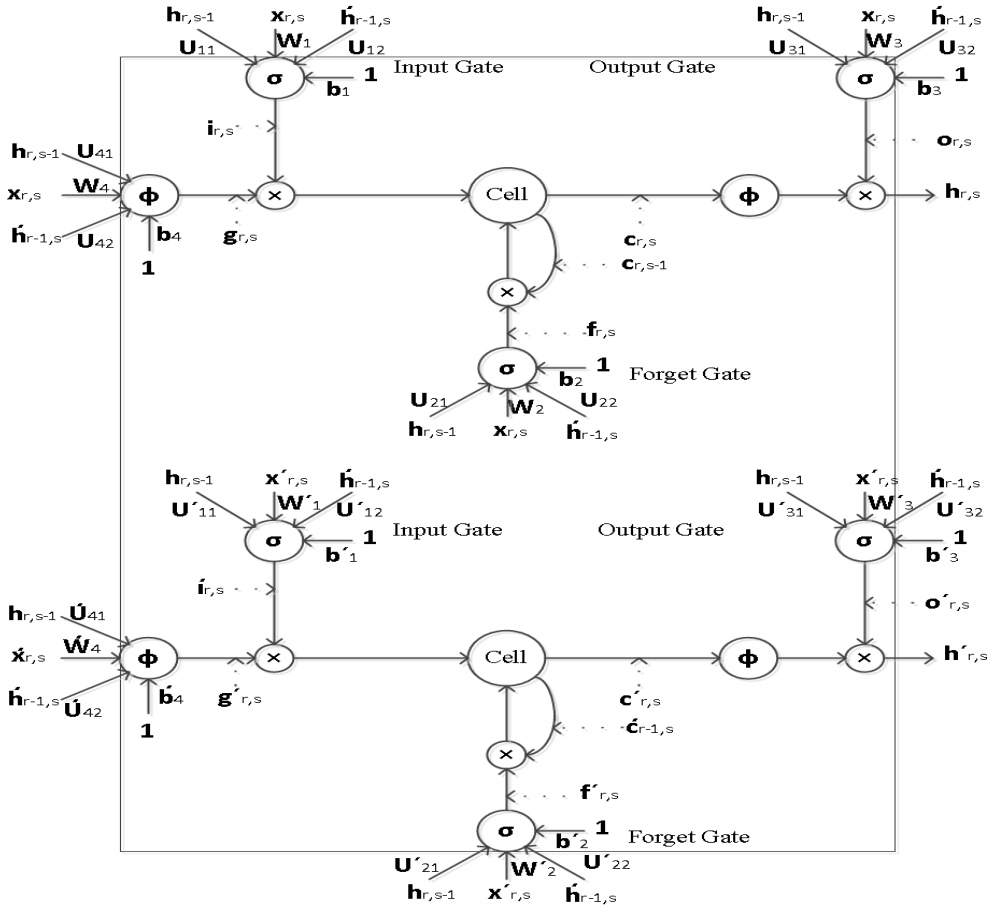


Figure 3.2: The architecture of a Grid LSTM cell.

$$\mathbf{i}_{r,s} = \sigma(\mathbf{W}_1\mathbf{x}_{r,s} + \mathbf{U}_{11}\mathbf{h}_{r,s-1} + \mathbf{U}_{12}\mathbf{h}'_{r-1,s} + \mathbf{b}_1) \tag{3.1}$$

$$\mathbf{f}_{r,s} = \sigma(\mathbf{W}_2\mathbf{x}_{r,s} + \mathbf{U}_{21}\mathbf{h}_{r,s-1} + \mathbf{U}_{22}\mathbf{h}'_{r-1,s} + \mathbf{b}_2) \tag{3.2}$$

$$\mathbf{g}_{r,s} = \phi(\mathbf{W}_4\mathbf{x}_{r,s} + \mathbf{U}_{41}\mathbf{h}_{r,s-1} + \mathbf{U}_{42}\mathbf{h}'_{r-1,s} + \mathbf{b}_4) \tag{3.3}$$

$$\mathbf{c}_{r,s} = \mathbf{f}_{r,s} \odot \mathbf{c}_{r,s-1} + \mathbf{i}_{r,s} \odot \mathbf{g}_{r,s} \tag{3.4}$$

$$\mathbf{i}'_{r,s} = \sigma(\mathbf{W}'_1\mathbf{x}'_{r,s} + \mathbf{U}'_{11}\mathbf{h}_{r,s-1} + \mathbf{U}'_{12}\mathbf{h}'_{r-1,s} + \mathbf{b}'_1) \tag{3.5}$$

$$\mathbf{f}'_{r,s} = \sigma(\mathbf{W}'_2\mathbf{x}'_{r,s} + \mathbf{U}'_{21}\mathbf{h}_{r,s-1} + \mathbf{U}'_{22}\mathbf{h}'_{r-1,s} + \mathbf{b}'_2) \tag{3.6}$$

$$\mathbf{g}'_{r,s} = \phi(\mathbf{W}'_4\mathbf{x}'_{r,s} + \mathbf{U}'_{41}\mathbf{h}_{r,s-1} + \mathbf{U}'_{42}\mathbf{h}'_{r-1,s} + \mathbf{b}'_4) \tag{3.7}$$

$$\mathbf{c}'_{r,s} = \mathbf{f}'_{r,s} \odot \mathbf{c}'_{r-1,s} + \mathbf{i}'_{r,s} \odot \mathbf{g}'_{r,s} \tag{3.8}$$

$$\mathbf{o}_{r,s} = \sigma(\mathbf{W}_3\mathbf{x}_{r,s} + \mathbf{U}_{31}\mathbf{h}_{r,s-1} + \mathbf{U}_{32}\mathbf{h}'_{r-1,s} + \mathbf{b}_3) \tag{3.9}$$

$$\mathbf{h}_{r,s} = \mathbf{o}_{r,s} \odot \phi(\mathbf{c}_{r,s}) \tag{3.10}$$

$$\mathbf{o}'_{r,s} = \sigma(\mathbf{W}'_3\mathbf{x}'_{r,s} + \mathbf{U}'_{31}\mathbf{h}_{r,s-1} + \mathbf{U}'_{32}\mathbf{h}'_{r-1,s} + \mathbf{b}'_3) \tag{3.11}$$

$$\mathbf{h}'_{r,s} = \mathbf{o}'_{r,s} \odot \phi(\mathbf{c}'_{r,s}) \tag{3.12}$$

where, $\mathbf{h}_{1,0} = \mathbf{h}'_{0,1} = \mathbf{c}_{1,0} = \mathbf{c}'_{0,1} = \mathbf{0}$, $\odot$ denotes element-wise multiplication, $\mathbf{W}_i$,$\mathbf{W}'_i$,$\mathbf{U}_{ij}$, $\mathbf{U}'_{ij}$, $\mathbf{b}_i$, and $\mathbf{b}'_i$ ($i = 1, 2, 3, 4$, $j = 1, 2, 3$) are trainable parameters. The computation begins at the upper-left region of the input image, continues along horizontal and vertical spatial directions and ends at the bottom-right region.

One restriction of a Grid LSTM is that it can only benefit from the spatial context on upper left side of the current region. But, the spatial context in all directions around an image area is crucial to represent the visual meaning of that area. To resolve this problem, we introduce a Bidirectional Grid LSTM (BiGrid LSTM) by processing the input image in four directions (from top-left to bottom-right and vice versa, and from top-right to bottom-left and vice versa) with four separate Grid LSTMs. The equations for other Grid LSTMs are similar. For example, the equations for the Grid LSTM which its computation starts from bottom-right to top-left are as before, except that they are applied to the successor hidden states $\mathbf{h}_{r,s+1}$ and $\mathbf{h}'_{r+1,s}$ instead of the predecessor hidden states $\mathbf{h}_{r,s-1}$ and $\mathbf{h}'_{r-1,s}$. The initial hidden states for the bottom-right to top-left traversal are $\mathbf{h}_{R,S+1} = \mathbf{h}'_{R+1,S} = \mathbf{0}$. Since the visual meaning of a region in all Grid LSTMs must be the same, we share the weight matrices $\mathbf{W}_i$ and $\mathbf{W}'_i$, across all Grid LSTMs. This technique will reduce the number of the trainable parameters properly. We abbreviate the computation of the Grid LSTM which starts at top-left corner as

$$(\overset{\ulcorner}{\mathbf{c}}_{r,s}, \overset{\ulcorner}{\mathbf{c}}'_{r,s}, \overset{\ulcorner}{\mathbf{h}}_{r,s}, \overset{\ulcorner}{\mathbf{h}}'_{r,s}) = \overset{\ulcorner}{\texttt{GrLSTM}}\,(\mathbf{x}_{r,s}, \mathbf{x}'_{r,s}, \overset{\ulcorner}{\mathbf{h}}_{r,s-1}, \overset{\ulcorner}{\mathbf{h}}'_{r-1,s}, \overset{\ulcorner}{\mathbf{c}}_{r,s-1}, \overset{\ulcorner}{\mathbf{c}}'_{r-1,s}).$$

With similar notations for other Grid LSTMs, the BiGrid LSTM computes two new hidden states and two new memory states as

$$\overset{\square}{\mathbf{h}}_{r,s} = \overset{\ulcorner}{\mathbf{h}}_{r,s} + \overset{\lrcorner}{\mathbf{h}}_{r,s} + \overset{\urcorner}{\mathbf{h}}_{r,s} + \overset{\llcorner}{\mathbf{h}}_{r,s} \tag{3.13}$$

$$\overset{\square}{\mathbf{h}}'_{r,s} = \overset{\ulcorner}{\mathbf{h}}'_{r,s} + \overset{\lrcorner}{\mathbf{h}}'_{r,s} + \overset{\urcorner}{\mathbf{h}}'_{r,s} + \overset{\llcorner}{\mathbf{h}}'_{r,s} \tag{3.14}$$

$$\overset{\square}{\mathbf{c}}_{r,s} = \overset{\ulcorner}{\mathbf{c}}_{r,s} + \overset{\lrcorner}{\mathbf{c}}_{r,s} + \overset{\urcorner}{\mathbf{c}}_{r,s} + \overset{\llcorner}{\mathbf{c}}_{r,s} \tag{3.15}$$

$$\overset{\square}{\mathbf{c}}'_{r,s} = \overset{\ulcorner}{\mathbf{c}}'_{r,s} + \overset{\lrcorner}{\mathbf{c}}'_{r,s} + \overset{\urcorner}{\mathbf{c}}'_{r,s} + \overset{\llcorner}{\mathbf{c}}'_{r,s} \tag{3.16}$$

We use $\mathbf{v}_{r,s} = (\overset{\square}{\mathbf{h}}_{r,s}||\overset{\square}{\mathbf{c}}_{r,s}||\overset{\square}{\mathbf{h}}'_{r,s}||\overset{\square}{\mathbf{c}}'_{r,s})$ as a feature vector extracted from region (r, s) of the image, where $||$ denotes concatenation. Note that unlike the CNN features, $\mathbf{v}_{r,s}$ considers the global scene context around region (r, s).

3.3.3 Deep (Two-Layer) Bidirectional LSTM with a Dynamic Visual Spatial Attention

We propose a Deep Bidirectional LSTM with a novel spatial attention mechanism to predict the t-th word of the caption, after it has seen the image and all preceding words. Intuitively, the first layer provides some contextual information about the image such as scene context

(e.g. a farm), presence of objects, and co-occurrence of the words which facilitates the image caption generation. Then, the second layer generates the caption which describes the image in more detail. The motivation for a Bidirectional LSTM is that it can exploit the previous context of the input signal. The context around a word in both directions plays an important role in natural language description of the scene, e.g. the word *watching* is more likely to follow *TV* than *studying*.

Word Embedding. We map the *one hot* representation of word i, denoted by $\mathbf{e}_i$, to a semantic space of dimensionality d via $\mathbf{u}^i = \mathbf{L}\mathbf{e}_i$, where $\mathbf{L}$ denotes a $d \times n$ word embedding matrix, and $\mathbf{u}^i$ is the semantic representation of word i. We randomly initialize $\mathbf{L}$ and fine-tune during training our model. We denote by $Y = (Y_1, ..., Y_T)$, the caption of training image I, where Y_t $(t = 1, ..., T)$, is the word at time step t, Y_1 is a special start word, and Y_T is a special end word which define the start and end of the caption respectively. We represent Y_t by a vector of size d denoted by $\mathbf{u}_t$ using the word embedding matrix $\mathbf{L}$.

Global Scene Context. We initialize the Bidirectional LSTM with contextual information from the image. The information is represented by a feature vector denoted by $\mathbf{z}$ of size 1000, where $\mathbf{z}_i$ is the probability of word i (in a dictionary of 1000 most common words) occurring in the caption of image I. The feature vector is computed by a word detector code from `https://github.com/s-gupta/visual-concepts`. The words are detected by applying the VGGNet to image regions and integrating the information with a Multiple Instance Learning framework (Fang et al., 2015). The word detector is trained only using captions, not word bounding-boxes. We feed $\mathbf{A}\mathbf{z} + \mathbf{a}$ to the first layer of the Deep Bidirectional LSTM at $t = 0$, where $\mathbf{A}$ is a $d \times 1000$ matrix, and $\mathbf{a}$ is a trainable bias. This layer also makes our model more robust to background clutter, since the word detector detect the words based on small local regions, not the whole image. After informing the Bidirectional LSTM about the image context, we ignore the output at time $t = 0$. Then, the embedded words $(\mathbf{u}_1, ..., \mathbf{u}_{T-1})$ are fed into the first layer.

Dynamic Spatial Attention Mechanism. For the second layer, we use a dynamic representation of the relevant patch of the image at time t denoted by $\mathbf{v}_t$. To produce $\mathbf{v}_t$, we first compute a weight for each encoded visual feature $\mathbf{v}_{r,s}$ using a two-layer feed forward neural network as follows

$$\hat{\upsilon}^t_{r,s} = \phi(\mathbf{M}^{(1)}\phi(\mathbf{M}^{(2)}\hat{\mathbf{v}}^t_{r,s} + \mathbf{b}^{(1)}) + b^{(2)}) \tag{3.17}$$

$$\upsilon^t_{r,s} = \frac{\exp(\hat{\upsilon}^t_{r,s})}{\sum_{r=1}^{R}\sum_{s=1}^{S}\exp(\hat{\upsilon}^t_{r,s})} \tag{3.18}$$

where, $\mathbf{M}^{(1)}$, $\mathbf{M}^{(2)}$, $\mathbf{b}^{(1)}$, $b^{(2)}$ are trainable parameters, and $\hat{\mathbf{v}}^t_{r,s}$ is the concatenation of the visual feature at location (r, s) and hidden state of the second layer of the Deep Bidirectional

LSTM at time $t-1$. That is, $\hat{\mathbf{v}}_{r,s}^{t} = \mathbf{v}_{r,s} \mid\mid \mathbf{h}_{t-1}^{(2)}$. Intuitively, $\upsilon_{r,s}^{t}$ is a positive weight for the location (r,s) which can be interpreted as the relative importance to give to location (r,s) at time t. The weight $\upsilon_{r,s}^{t}$ depends on the visual features and the previous word that has already been generated.

After computing the attention weights, we need an attention mechanism to compute $\mathbf{v}_t$. A straightforward attention mechanism is soft attention mechanism which is used by recent attention-based caption generation models (Xu et al., 2015). It computes $\mathbf{v}_t$ as a weighted summation of visual features $\mathbf{v}_{r,s}$, $(r = 1, ..., R)$ and $(s = 1, ..., S)$. That is, $\mathbf{v}_t = \sum_{r=1}^{R}\sum_{s=1}^{S} \upsilon_{r,s}^{t}\mathbf{v}_{r,s}$. However, a disadvantage of this method is that it does not consider the spatial context and order of the visual features. To resolve this issue, Xiong et al. (2016); Kumar et al. (2016) proposed an attention mechanism based on a Gated Recurrent Unit. By extending this idea, we introduce a new 2D spatial attention mechanism based on a Grid LSTM. In a Grid LSTM, the forget gates learn how much of the previous memory should be kept, and the input gates control how much of the input should be read. Therefore, we use another Grid LSTM, called *attention Grid LSTM*, whose input and forget gates are replaced with the attention weights that we computed.

More precisely, the equations for the attention Grid LSTM are as before, except that 3.4, and 3.8 are substituted by

$$\hat{\mathbf{c}}_{r,s} = (1 - \upsilon_{r,s}^{k}) \odot \hat{\mathbf{c}}_{r,s-1} + \upsilon_{r,s}^{k} \odot \hat{\mathbf{g}}_{r,s} \tag{3.19}$$

$$\hat{\mathbf{c}}'_{r,s} = (1 - \upsilon_{r,s}^{k}) \odot \hat{\mathbf{c}}'_{r-1,s} + \upsilon_{r,s}^{k} \odot \hat{\mathbf{g}}'_{r,s}, \tag{3.20}$$

where, $\hat{}$ denote the computation of the attention Grid LSTM. The inputs to the attention Grid LSTM at step (r,s) are $\mathbf{x}_{r,s} = (\hat{\square}_{r,s} || \hat{\square}_{r,s})$ and $\mathbf{x}'_{r,s} = (\hat{\square}'_{r,s} || \hat{\square}'_{r,s})$. The concatenation of the vertical and horizontal hidden and memory states of the attention Grid LSTM at the last spatial step is used as the relevant patch of the image at time t. That is, $\mathbf{v}_t = \Big(\hat{\mathbf{h}}_{R,S}^{t} \mid\mid \hat{\mathbf{c}}_{R,S}^{t} \mid\mid \hat{\mathbf{h}}_{R,S}^{\prime t} \mid\mid \hat{\mathbf{c}}_{R,S}^{\prime t}\Big)$. Our experiments show utilizing the spatial attention mechanism rather than soft attention mechanism will improve the performance. With $\upsilon_{r,s}^{0} = 1/RS$, $\mathbf{u}_0^{(1)} = \mathbf{A}\mathbf{z} + \mathbf{a}$, $\mathbf{v}_t^{(1)} = \mathbf{0}$, $\mathbf{v}_t^{(2)} = \mathbf{v}_t$, $\mathbf{u}_t^{(1)} = \mathbf{u}_t$, $\mathbf{u}_t^{(2)} = \mathbf{h}_t^{(1)}$ $(t = 0, ..., T)$, the equations for cell update and the output of the left-to-right layer l $(l = 1, 2)$ of the Deep Bidirectional LSTM are as

follows:

$$\mathbf{r}_t^{(l)} = \sigma\big(\mathbf{r}_1^{(l)}\mathbf{u}_t^{(l)} + \mathbf{r}_1^{(l)}\mathbf{r}_{t-1}^{(l)} + \mathbf{V}_1\mathbf{v}_t^{(l)} + \mathbf{r}_1^{(l)}\big) \tag{3.21}$$

$$\mathbf{r}_t^{(l)} = \sigma\big(\mathbf{r}_2^{(l)}\mathbf{u}_t^{(l)} + \mathbf{r}_2^{(l)}\mathbf{r}_{t-1}^{(l)} + \mathbf{V}_2\mathbf{v}_t^{(l)} + \mathbf{r}_2^{(l)}\big) \tag{3.22}$$

$$\mathbf{r}_t^{(l)} = \sigma\big(\mathbf{r}_3^{(l)}\mathbf{u}_t^{(l)} + \mathbf{r}_3^{(l)}\mathbf{r}_{t-1}^{(l)} + \mathbf{V}_3\mathbf{v}_t^{(l)} + \mathbf{r}_3^{(l)}\big) \tag{3.23}$$

$$\mathbf{r}_t^{(l)} = \phi\big(\mathbf{r}_4^{(l)}\mathbf{u}_t^{(l)} + \mathbf{r}_4^{(l)}\mathbf{r}_{t-1}^{(l)} + \mathbf{V}_4\mathbf{v}_t^{(l)} + \mathbf{r}_4^{(l)}\big) \tag{3.24}$$

$$\mathbf{r}_t^{(l)} = \mathbf{r}_t^{(l)} \odot \mathbf{r}_{t-1}^{(l)} + \mathbf{r}_t^{(l)} \odot \mathbf{r}_t^{(l)} \tag{3.25}$$

$$\mathbf{r}_t^{(l)} = \mathbf{r}_t^{(l)} \odot \phi(\mathbf{r}_t^{(l)}) \tag{3.26}$$

where, $\mathbf{r}_{-1}^{(l)} = \mathbf{0}$, $\mathbf{V}_i$, $\mathbf{r}_i^{(l)}$, $\mathbf{r}_i^{(l)}$, and $\mathbf{r}_i^{(l)}$ $(i = 1, 2, 3, 4)$ are trainable parameters. The equations for the right-to-left computation are similar except that $t-1$ is replaced by $t+1$ and $\overleftarrow{\mathbf{h}}_{T+1}^{(l)} = \mathbf{0}$. Intuitively, $\mathbf{r}_i^{(l)}$ models grammar and left-to-right context, while $\mathbf{r}_i^{(l)}$ encodes the words. Since the meaning of a word form left-to-right or right-to-left is the same, we set $\mathbf{r}_i^{(l)} = \overleftarrow{\mathbf{W}}_i^{(l)}$. This technique also helps to prevent overfitting. The output of the l-th layer of the Deep Bidirectional LSTM is computed as $\mathbf{h}_t^{(l)} = \mathbf{r}_t^{(l)} + \overleftarrow{\mathbf{h}}_t^{(l)}$. The final output of the Deep Bidirectional LSTM at time t is fed to a softmax layer to produce a probability distribution $\mathbf{p}_t$ over the dictionary words

$$\mathbf{p}_t(i) = \frac{\exp\left(\mathbf{m}_i^\top \mathbf{h}_t^{(2)} + b_i\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{m}_j^\top \mathbf{h}_t^{(2)} + b_j\right)} \tag{3.27}$$

where, $\mathbf{p}_t(i)$ is the probability of Y_{t+1} being i-th word in the dictionary given $\mathbf{h}_t^{(2)}$, b_i is i-th entry of a trainable bias $\mathbf{b}$, and $\mathbf{m}_i$ specifies i-th row of a trainable $n \times d$ matrix $\mathbf{M}$. Intuitively, this matrix decodes the dense word representation into a pseudo one-hot word representation which is the inverse function of the word embedding matrix. Thus, matrix $\mathbf{M}$ is shared with the transpose of the word embedding matrix $\mathbf{L}$. This technique will effectively reduce the number of parameters of the model (Mao et al., 2015).

To generate a caption for a new image at the test time, ideally we need to find a caption $\hat{Y}$ such that

$$\hat{Y} = \arg\max_{Y} \sum_{t=1}^{T-1} \log\left(\mathbf{p}_{t+1}(Y_{t+1})\right). \tag{3.28}$$

However, since the exhaustive search is intractable, we use *beam search* with size $k = 20$ to find $\hat{Y}$. The beam search algorithm iteratively considers the k best captions up to time t as candidates to generate new captions of size $t+1$.

3.3.4 Integrating with Region-Grounded Texts

In this section, we propose a transfer learning technique which incorporates region-grounded texts of the input image, e.g. *a red stop sign*, *a cloudy sky*, *boy on horse*, to boost the performance. For this purpose, we use a dense captioning model from `https://github.com/jcjohnson/densecap` to extract a set of descriptions for the input image (Johnson et al., 2015a). This model has been trained on Visual Genome region caption dataset (Krishna et al., 2016). This enables our model to transfer *learning* from a region-grounded caption dataset and produce more precise captions, since the grounded textual information often describes the properties of the objects and their relationships in an image which may not be represented properly by visual features on small datasets. Each region-grounded text has a bounding-box and a confidence score. Our goal is to summarize local region-grounded texts into a single global caption by attending to the important bounding-boxes.

For each image, we select all region-grounded texts with a confidence score greater than 1.0. The words are encoded by the same embedding matrix $\mathbf{L}$. We applied a simple Bag of Words to encode a region-grounded text. LSTMs may be also applied, but they need more computations and parameters. Each region (r, s) is then represented by a textual feature vector of size d by taking the average of the encoded region-grounded texts whose bounding-box covers region (r, s). Then, we obtain vertical and horizontal textual features $\bar{\mathbf{x}}_{r,s}$ and $\bar{\mathbf{x}}'_{r,s}$ by projecting the resulting feature vector to $d/2$ dimensions. The inputs to the attention Grid LSTM are now computed as $\mathbf{x}_{r,s} = (\square\mathbf{h}_{r,s}||\square\mathbf{e}_{r,s}) \mid\mid \bar{\mathbf{x}}_{r,s}$ and $\mathbf{x}'_{r,s} = (\square'\mathbf{h}_{r,s}||\square'\mathbf{e}_{r,s}) \mid\mid \bar{\mathbf{x}}'_{r,s}$, and $\hat{\mathbf{v}}^t_{r,s}$ is computed as $\hat{\mathbf{v}}^t_{r,s} = (\mathbf{v}_{r,s} \mid\mid \mathbf{h}^{(2)}_{t-1}) \mid\mid (\bar{\mathbf{x}}_{r,s} \mid\mid \bar{\mathbf{x}}'_{r,s})$.

3.3.5 Training Details

The sum of the negative log likelihood of the correct word at each time step is chosen as the loss, that is $-\sum_{t=1}^{T-1} \log\left(\mathbf{p}_{t+1}(\texttt{idx}(Y_{t+1}))\right)$, where $\texttt{idx}(Y_{t+1})$ is the index of word Y_{t+1} in the dictionary. This loss is minimized using RMSprop with minibatches of size 50 and learning rate 0.0001. To prevent overfitting, dropout with probability 0.6 and early-stopping are used. During training, all parameters are tuned except for the weights of the word detector and CNN components which we keep fixed to prevent overfitting. In VGGNet experiments, we use $m' = 128$ hidden states for the BiGrid LSTM and 256 hidden states for the Deep Bidirectional LSTM, respectively. In ResNet experiments, we use $m' = 256$ hidden states for the BiGrid LSTM and 512 hidden states for the Deep Bidirectional LSTM, respectively. The word embedding size is set to $d = 256$ for VGGNet, and $d = 512$ for ResNet.

Since at the test time a caption must be generated word by word from left to right, the backward LSTM needs to see the reverse of the sub-captions during training. To resolve this problem we train our model with all sub-captions of a training sample, that is $Y = (Y_1, ..., Y_{t'})$ $(t' = 1, ...T)$. We reset the Bidirectional LSTM after feeding each sub-caption. This increases the number of training samples and the training time by the average of the

caption lengths in the training data, which is 10 for MS-COCO dataset. However, we find fewer epochs are required for convergence. Our model took around four days to train on two NVIDIA Titan X GPUs.

3.4 Experiments

In this section, we firstly introduce the dataset and evaluation metrics that we use in our experiments. Then, we explain our experimental set up and methodology. Finally, the experimental results are presented and discussed.

3.4.1 Data, Metrics and Experimental Setup

Our results are reported on the MS-COCO dataset. MS-COCO dataset contains $82,783$ training images, $40,504$ validation images, and $40,775$ test images. The dataset is annotated with 5 sentences using Amazon Mechanical Turk. The captions for the test set are not publicly available. We follow Karpathy and Fei-Fei (2015) to preprocess the captions with basic tokenization by converting all sentences to lower case, throwing away non-alphanumeric characters, and filtering the words to those that occur at least 5 times in the training set. This results in a dictionary of size $8,791$. We use METEOR (Denkowski and Lavie, 2014) and BLEU (Papineni et al., 2002) as evaluation metrics, which are popular in the machine translation literature and used in recent image caption generation papers. The BLEU score is based on n-gram precision of the generated caption with respect to the references. The METEOR is based on the harmonic mean of unigram precision and recall, and produces a good correlation with human judgment.

To analyze the effect of various improvements, we implemented a few lesion models. In Visual Concepts+LSTM model, we feed word probabilities $\mathbf{z}$ to a single layer LSTM. The CNN+LSTM uses a feature vector of size $4,096$ at the very top layer of VGGNet. This vector is mapped to a space of dimensionality 256. Then, the resulting vector is fed to a single layer LSTM at time $t = 0$. The CNN+LSTM2 stacks two single layers of unidirectional LSTM in the first and second lesion models. These lesion models do not apply the Grid LSTM. Thus, they cannot exploit spatial information efficiently.

The Gr+LSTM2+soft uses a Grid LSTM and a deep LSTM for caption generation with a soft attention mechanism. The Gr+LSTM2+sp is similar to Gr+LSTM2+soft but it utilizes our dynamic spatial attention mechanism. The BiGr+LSTM2+sp employs a BiGrid LSTM and a deep LSTM for caption generation with a dynamic spatial attention mechanism. The BiGr+BiLSTM2+sp uses a BiGrid LSTM and a Deep Bidirectionl LSTM for caption generation with dynamic spatial attention mechanism. We also implemented this last model with a more powerful CNN, ResNet (He et al., 2015), instead of VGGNet. Finally, BiGr+BiLSTM2+sp+rg is the same as BiGr+BiLSTM2+sp with ResNet but it uses region-grounded texts.

Our results are reported in Table 3.1. The full model outperforms the baseline and previous works in all cases. The Visual Concepts+LSTM model does not use CNN features of the image. It only uses the word detector to generate a caption. Conversely, CNN+LSTM model only uses the CNN features. CNN+LSTM2 outperforms both of these models by exploiting both CNN features and word detector. Models which use a Grid LSTM outperform others that use global image features, which is due to the power of Grid LSTM to represent spatial information of an image. Gr+LSTM2+sp outperforms Gr+LSTM2+soft which shows dynamic spatial attention is more efficient than soft attention. BiGr+LSTM2+sp outperforms Gr+LSTM2+sp since it can scan an image from all directions (from top-left to bottom-right and vice versa, and from top-right to bottom-left and vice versa). Also, BiGr+BiLSTM2+sp outperforms BiGr+LSTM2+sp since it can consider the surrounding context of a word from both sides. Finally, BiGr+BiLSTM2+sp+rg with ResNet outperforms BiGr+BiLSTM2+sp with ResNet which shows incorporating region-grounded texts boost the performance. Since our model has more parameters than the other state-of-the-art models, it is extremely data driven. As a result, we believe that even with these promising results, the capability of our model will grow in the future, as the image captioning datasets extend.

3.4.2 Results and Discussion

Figure 3.3 shows example captions generated by our model. We also visualize the attention weights in Figure 3.4. The 7×7 heatmaps represents the value of the visual attention weights $\upsilon_{r,s}^{t}$ for each generated word. The examples shows our visual spatial attention model can attend to the right concept, even in the presence of background clutter, especially for the words which have a well-defined bounding box such as *bear*. Moreover, our model generates captions that are grammatically correct. This shows the power of our Deep Bidirectional LSTM to model the context from both sides, and thereby generate grammatically correct sentences. Finally, Figure 3.5 shows two example of mistakes of our model which are probably due to small size data.

Figure 3.3: Example captions generated by our model.

Model	B-1	B-2	B-3	B-4	MET
BiRNN† (Karpathy and Fei-Fei, 2015)	62.5	45.0	32.1	23.0	19.50
mRNN† (Mao et al., 2014a)	67.0	49.0	35.0	25.0	-
LRCN† (Donahue et al., 2015)	62.8	44.2	30.4	21.0	-
Google NIC† (Vinyals et al., 2014)	66.6	46.1	32.9	24.6	-
Log Bilinear† (Kiros et al., 2014)	70.8	48.9	34.4	24.3	20.03
Soft-Attention† (Xu et al., 2015)	70.7	49.2	34.4	24.3	23.90
Hard-Attention† (Xu et al., 2015)	71.8	50.4	35.7	25.0	23.04
ATT-FCN† (You et al., 2016)	70.9	53.7	40.2	30.4	24.30
Review Networks (Yang et al., 2016)	-	-	-	29.0	23.20
SCA-CNN° (Chen et al., 2017)	71.9	54.8	41.1	31.1	25.00
ACVT (Wu et al., 2016)	73.0	56.0	41.0	31.0	25.00
Boosting with Attributes° (Yao et al., 2016)	73.0	56.5	42.9	32.5	25.10
Adaptive Attention° (Lu et al., 2017)	74.2	58.0	43.9	33.2	26.60
Visual Concepts (Fang et al., 2015)+LSTM	65.1	48.3	35.0	24.6	21.00
CNN+LSTM	67.0	50.0	36.2	26.0	21.90
CNN+LSTM2	68.7	52.1	38.2	28.0	22.90
Gr+LSTM2+soft	73.2	56.1	41.1	31.3	25.14
Gr+LSTM2+sp	73.6	56.4	41.5	31.7	25.25
BiGr+LSTM2+sp	74.2	56.8	41.9	32.0	25.48
BiGr+BiLSTM2+sp	74.7	57.5	42.1	32.3	25.71
BiGr+BiLSTM2+sp°	74.9	58.9	44.0	33.9	26.25
BiGr+BiLSTM2+sp+rg°	**76.2**	**60.1**	**45.1**	**35.0**	**27.02**

Table 3.1: BLEU-1,2,3,4 and METEOR scores on MS-COCO test set. We only compare with the results that have been officially published. For fairness, we only compare with the results which employ comparable CNNs such as GoogLeNet, VGGNet and ResNet. The - denotes that the result is not reported. Models that use ResNet are denoted by °. The † denotes the results are reported on validation set, since the results on the test set are not available.

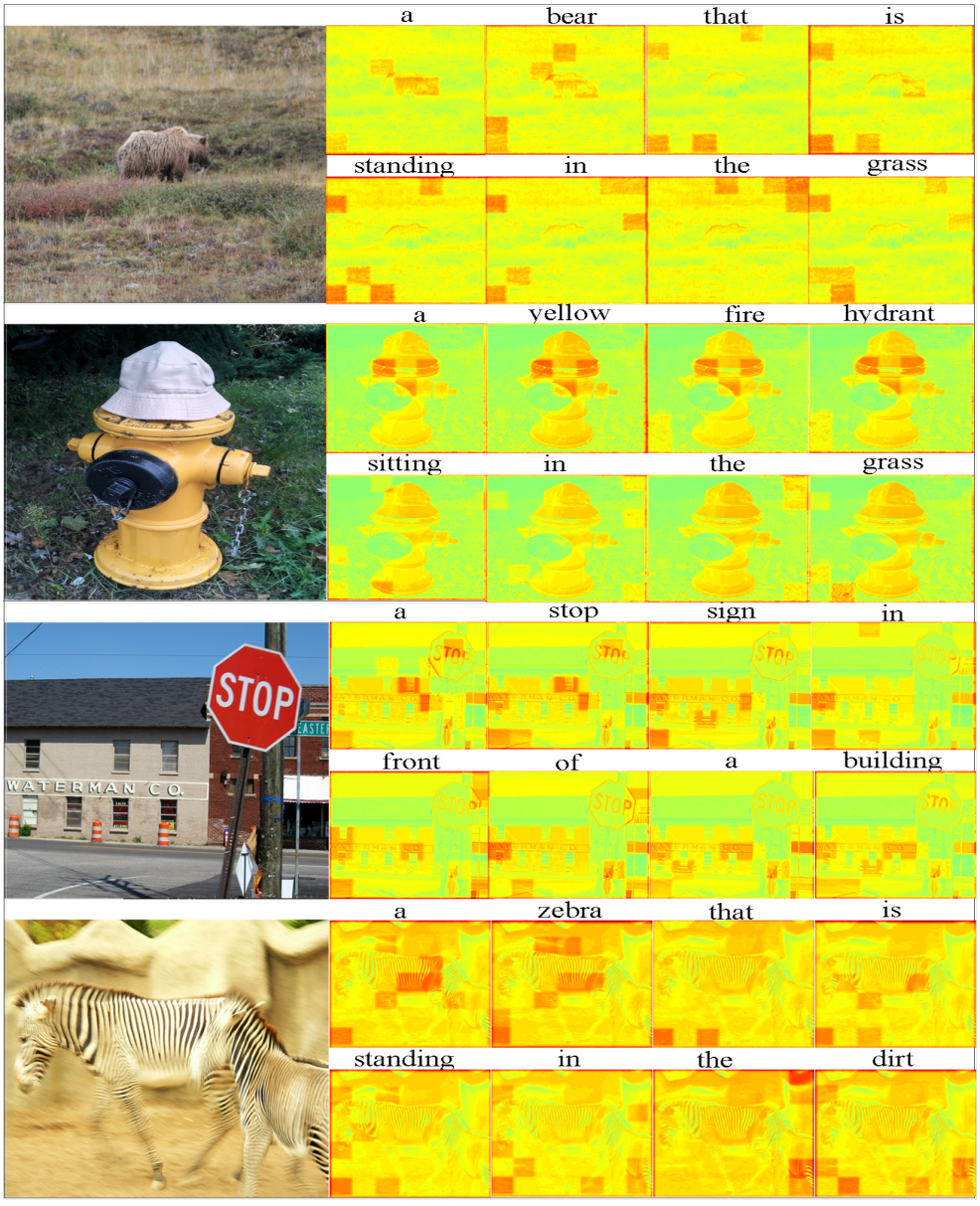


Figure 3.4: Visualization of the attention. The 7×7 heatmaps represents value of weight $v_{r,s}^t$ for t-th word in the generated caption. Each value corresponds to a specific region in the image.

Figure 3.5: Two example of mistakes of our model.

Chapter 4

Multimodal Neural Graph Memory Networks for Visual Question Answering

This chapter introduces an intermediate graph-structured representation for image information. The nodes are bounding-boxes inferred from existing bounding-box detectors. The edges represent spatial adjacency relationships. We use two different graphs to capture information from two different modalities: visual features and textual features derived from image captions. The graph processing technique is Graph Network, augmented with a spatial memory network that aggregates information from both modalities. The task considered is Visual Question Answering (VQA).

Visual question answering has been recently introduced as a grand challenge for AI. Given an image and a freestyle question about it, the VQA task is to produce an accurate natural language answer. VQA has many applications such as image retrieval and search. In this chapter, we propose a new neural network architecture for VQA based on the recent Graph Network (GN) (Battaglia et al., 2018); see also Chapter 2.

The pair-wise interactions between various regions of an image and spatial context in both horizontal and vertical directions are important to answer questions about objects and their interactions in the scene context. For example, to answer *How many cats are in the picture?* a model may need to aggregate information from multiple, possibly distant, regions; hence applying a convolutional neural network may not be sufficient to perform reasoning over the regions. Our new architecture (see Figure 4.2), Multimodal Neural Graph Memory Network (MN-GMN), uses a graph structure to represent pairwise interactions between visual/textual features (nodes) from different regions of an image. GNs provide a context-aware neural mechanism for computing a feature for each node that represents complex interactions with other nodes. This enables our MN-GMN to answer questions which need reasoning about complex arrangements of objects in a scene.

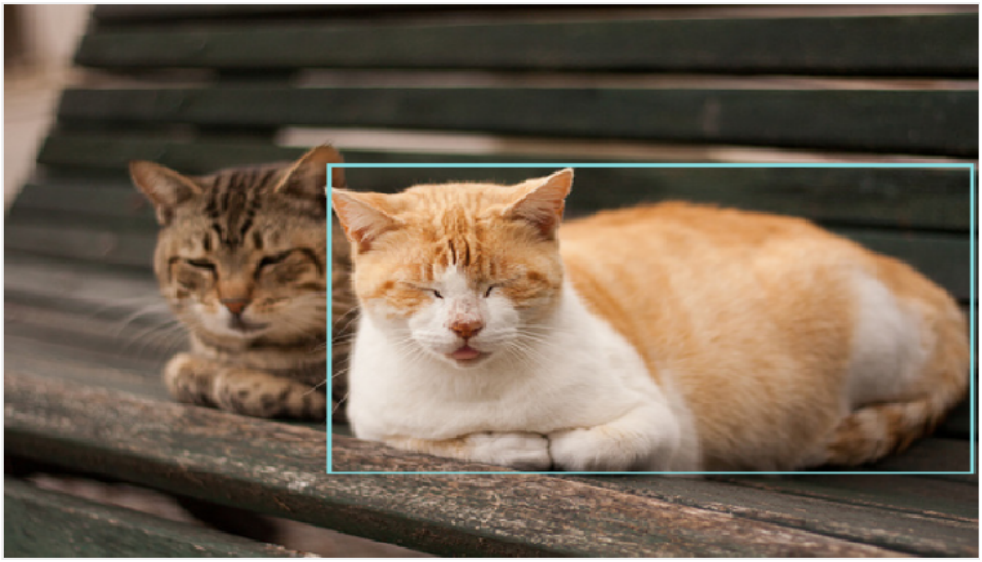


Figure 4.1: An example from Visual Genome. The region-grounded captions provide useful clues to answer questions.

Previous approaches such as Memory Networks (MN) (Sukhbaatar et al., 2015) and Dynamic Memory Networks (DMNs) (Kumar et al., 2015) combined a memory component and an attention mechanism to reason about a set of inputs. The DMN was first proposed

for text QA. A given text QA task is composed of a question, and a set of statements, called facts, in the order that describes a short story. For each question, only a subset of the facts is required to answer the question. DMN includes four modules: input, question, episodic memory, and answer. The input and question modules encode the question and the facts. Then, the episodic memory takes as input the question and aggregates the facts to produce a vector representation of the relevant information. This vector is passed to the answer module to predict an answer. Previous applications of the MN and DMN for VQA either represent each image region independently as a single visual fact (Xu and Saenko, 2015) or represent the regions of an image like facts of a story with a linear sequential structure, for example in row-major order (Xiong et al., 2016). But, whereas a linear order may be sufficient for text QA, it is insufficient to represent the 2D context of an image.

The major novel aspect of our approach is that we exploit the flexibility of GNs to combine information from two different sources: visual features from different image regions as in previous work, and textual features based on region-grounded captions (RGCs). An RGC detector is learned by transfer learning from a dataset with region-grounded captions (Visual Genome). Like visual features, an RGC is specified with a bounding-box. The RGCs capture object attributes and relationships that are often useful to answer visual questions, but may not be represented by low-level visual features. For example, in Figure 4.2, to answer *Is the water calm?*, the caption *a wave in the ocean* is informative; the caption *the water is blue* specifies an attribute of *water*; the captions *a sailboat in the water* and *surfer riding a wave* describe interactions between objects. Captions also incorporate commonsense knowledge that may not be represented by an end-to-end model especially with currently available small VQA datasets. Our multimodal graph memory network comprises a visual GN and a textual GN, one for each information source. Each node of the two GNs iteratively computes a question-guided contextualized representation of the visual/textual information at the bounding-box assigned to it.

The third component in our architecture is an external spatial memory which is designed to combine information across the modalities: each node writes the updated representations to the external spatial memory, which is composed of memory cells arranged in a 2D grid. The final state of the memory cells are then fed into the answer module to predict an answer. The external spatial memory resolves the redundancy introduced by overlapping bounding-boxes, which causes difficulties for example with counting questions. To summarize, our main contributions are the following: i) We introduce a new memory network architecture, based on graph neural networks, which can reason about complex arrangements of objects in a scene to answer visual questions, ii) To the best of our knowledge, this is the first work that explicitly incorporates local textual information (RGCs) of the image via a transfer learning technique into a multimodal memory network to answer visual question, iii) The proposed architecture improves the accuracy of the state-of-the-art models by about 2.5% on VQA and Visual7W datasets, which is substantial given the maturity of the existing approaches. The

content of this chapter has appeared in the NeurIPS 2019 Graph Representation Learning Workshop.

4.1 Related Work

Our work addresses several issues that have been explored in computer vision, machine learning, and natural language processing. An important part of the VQA task is to understand the given question. Most approaches utilize a neural network architecture that can handle sequences of flexible length and learn complex temporal dynamics using a sequence of hidden states. Such architectures include Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and the Gated Recurrent Unit (GRU). To encode a given image, most VQA approaches employ a Convolutional Neural Network (CNN) pretrained on ImageNet such as VGGNet, and ResNet to extract visual information from an image. These two recent trends of applying CNNs and RNNs have been successfully applied to image captioning and visual grounding (Johnson et al., 2015a) tasks. Grounding connects words to their visual meaning. Our approach sees the VQA as first grounding the question in the image and then predicting an answer.

Most early deep neural-based VQA models produce an answer conditioned on a global visual feature vector and the embedded question. However, since many questions and answers relate to a specific region in an image, these models often cannot predict a precise answer. To overcome this issue, many attention-based models are proposed. The attention-based models compute an attention weight of spatially localized CNN features based on the question to predict an answer (Xu and Saenko, 2015; Xiong et al., 2016). Teney et al. (2018) used the Bottom-Up Attention model (Anderson et al., 2018) to obtain a set of features at different regions of the image and computed an attention weight for each region based on the encoded question to predict an answer. In Lu et al. (2016b), the authors proposed a hierarchical co-attention model that jointly implements both image-guided question attention and question-guided visual attention. Nam et al. (2016) proposed a similar joint attention-based model for both image and question information. In Fukui et al. (2016), authors proposed a VQA model based on multimodal compact bilinear (MCB) pooling to get a joint representation for image and question. Similarly, Yu et al. (2018); Kim et al. (2018) utilized higher-order fusion techniques to combine the question with visual features more efficiently. Cadene et al. (2019) proposed a bilinear fusion algorithm to represent interactions between question and image regions.

Recently, a few models are proposed which use graph structures can learn the interactions between image regions. The graph learner model Norcliffe-Brown et al. (2018), merges a graph representation of the image based on the question, with a graph convolutional network, to learn visual features that can represent question specific interactions. Yang et al. (2018b) proposed to reason over a visual representation of the input image called scene

graph which represents object and their relationships explicitly. Li et al. (2019) introduced a VQA model called Relation-aware Graph Attention Network (ReGAT). Guided by the input question, ReGAT encodes an image into a graph that represents relations among visual objects. The ReGAT is trained on Visual Genome dataset (Krishna et al., 2016). Jabri et al. (2016) introduced a graph neural network model called Relation Networks which uses multilayer perceptron models to reason over all pairs of local image features extracted from a spatially localized grid of image regions. Dynamic tree structures have been used in VQA to capture the visual context of image objects Tang et al. (2019). Unlike all of the aforementioned work, our VQA architecture exploits the rich textual information of the image via incorporating the RGCs to learn the attributes of an image region and the interactions between a set of image regions enclosed by an RGC bounding-box. Previous work either explicitly use features extracted from a set of object bounding-boxes of the image or extract the features from a spatially localized grid of image regions; the former approach suffers from redundancy introduced by overlapping bounding-boxes; the latter approach cannot represent the local context properly, since objects often cross fixed region boundaries of a grid. We resolve this issues by introducing an external spatial memory.

4.2 Our Proposed Architecture

Figure 4.2 shows our MN-GMN architecture which is composed of four modules: input, question, multimodal graph memory network, and answer. In this section, we describe these modules emphasizing the intuition behind our design.

Input Module

The input module has two components: A deep CNN (e.g., Bottom-Up Attention (Anderson et al., 2018), VGGNet, ResNet, etc.), and a region-grounded caption generation model which incorporates a set of RGCs to answer questions about object attributes and their relationships. The RGCs are generated by a state-of-the-art dense captioning model. Then, they are encoded with a GRU and a parser (Schuster et al., 2015) which takes a caption and parses it into a set of objects with their attributes and a set of relationship triplets. We now describe the details and motivation for these components.

Visual Feature Extraction

To extract visual features, we use the Bottom-Up Attention model (Anderson et al., 2018). The features are obtained via a Faster R-CNN framework and 101-layer ResNet which attend to specific image regions. Using a fixed threshold on object detection, we extract N 2048-dimensional image features from N different regions of the image. The value of N depends on the image, and ranges from 10 to 100. Each feature vector $\mathbf{v}_i$ has a bounding-box specified by its coordinates $\mathbf{r} = (r_x, r_y, r_{x'}, r_{y'})$, where (r_x, r_y) and $(r_{x'}, r_{y'})$ are the top-

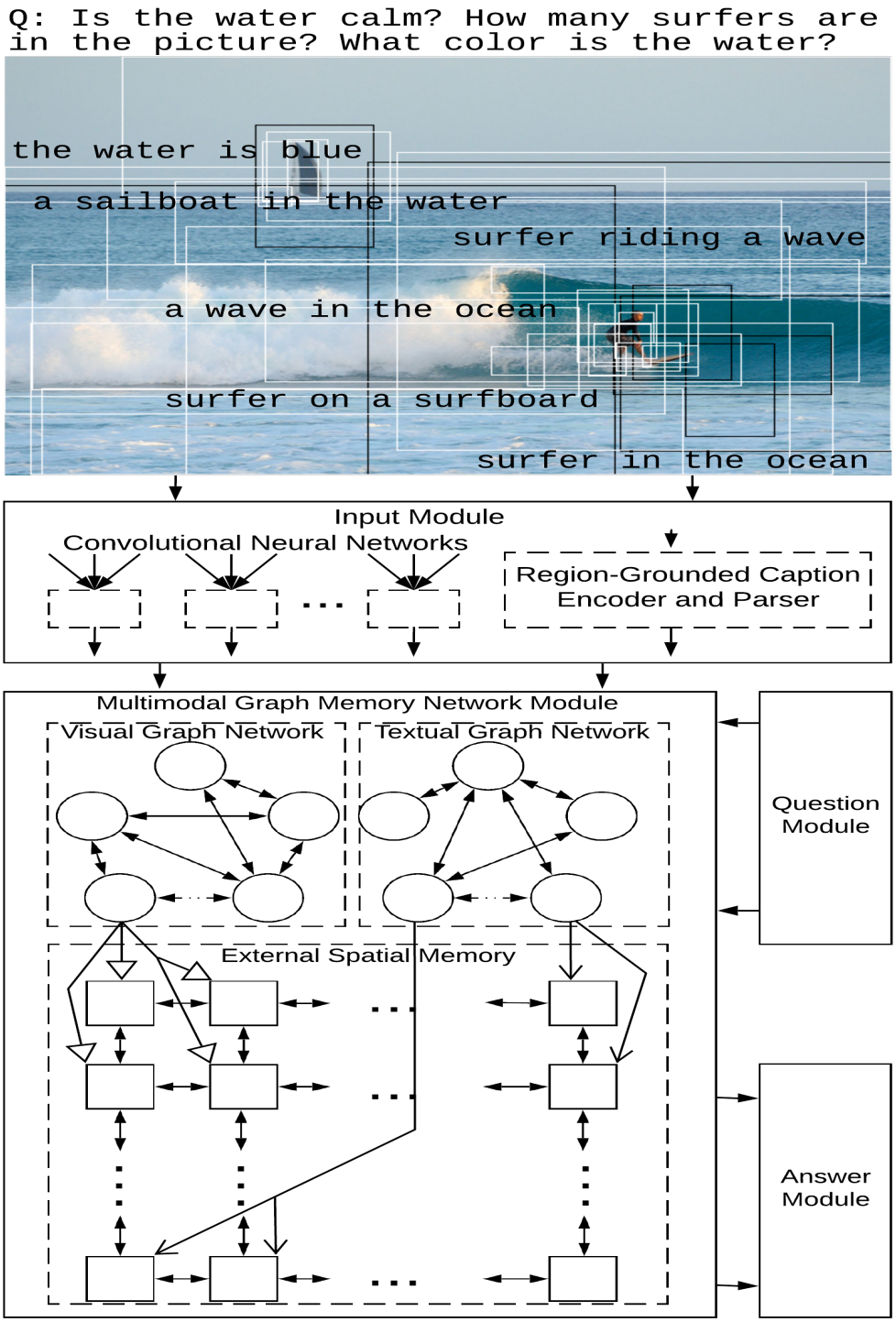


Figure 4.2: Multimodal Neural Graph Memory Networks for VQA. The visual features/ region-grounded captions are extracted from white/black bounding-boxes and are used as node features to construct the visual/textual Graph Network.

left and bottom-right corners of the bounding-box which are normalized to have a values between 0 and 1 based on the height and width of the image. We concatenate each feature

vector with its bounding-box to obtain a vector denoted by $\mathbf{x}_i$, $(i = 1, \ldots, N)$. Note that $\mathbf{x}_i$ only describes the image at its bounding-box, without exploiting global spatial context.

Region-Grounded Captions

The second component of the input module applies a dense captioning model from `https://github.com/jcjohnson/densecap` to extract a set of RGCs for the input image (Johnson et al., 2015a). This model contains a CNN (VGGNet), a dense localization layer, and an RNN language model that generates the captions. The model has been trained on RGCs from the Visual Genome dataset. Through transfer learning, our model is leveraging the caption annotations. In contrast with other datasets which only include full image captions, or ground words of image captions in regions, the Visual Genome provides individual region captions. Each RGC has a caption, a bounding-box, and a confidence score. For each image, we select all RGCs with a confidence score greater than 1.5. To encode a caption, we first create a dictionary using all words in the captions and questions. We preprocess the captions and questions with basic tokenization by converting all sentences to lower case and throwing away non-alphanumeric characters. We map the words to a dense vector representation using a trainable word embedding matrix $\mathbf{L} \in L \times D$, where D is the dimensionality of the semantic space, and L is the size of the dictionary. To initialize the word embeddings, we use the pretrained GloVe vectors. The words not occur in the pretrained word embedding model are initialized with zeros. We encode a caption using a GRU. Given a caption, the hidden state of the GRU for the i-th word is computed by $\mathbf{c}_i = \text{GRU}(\mathbf{s}_i, \mathbf{c}_{i-1})$, where $\mathbf{s}_i$ is the semantic representation of the i-th word in the caption. The final hidden state of the GRU, denoted by $\mathbf{c} \in \mathbb{R}^D$, is used as a vector representation of the caption. We reset the GRU after feeding each caption.

The encoded captions may not properly represent the objects in an image, the relationships between them and the object attributes, specially for caption with many words and a complex parse tree. Thus, we enrich this representation by utilizing a parser (Schuster et al., 2015) that takes a single caption and parses it into a set of relationship triplets (possibly empty), a set of objects and their attributes. For example, given caption *orange and white cat laying on a wooden bench*, the parser outputs a triplet *cat-lay on-bench*, objects *cat* and *bench*, and attributes *cat-white*, *cat-orange* and *bench-wooden*. For Visual Genome dataset, the parser produces less than three relationships for about %98 of the captions. We obtain a fixed length representation for the output of the parser, denoted by $\tilde{\mathbf{c}} \in \mathbb{R}^{14D}$ by allocating the embedding of 14 words: 6 words for up to two relationship triplets and 8 words for up to 4 objects and their attributes. For example, for the aforementioned example, the fixed length representation is the concatenation of the embedding of each word in sequence $\prec$*cat-lay on-bench,x-x-x,bench-wooden,cat-orange,cat-white,x-x*$\succ$ where x is a special token to represent an empty slot. We use an arbitrary order but fixed for all RCGs, to create the sequences. Each RGC has also a bounding-box specified by its coordinates

$\tilde{\mathbf{r}} = (\tilde{r}_x, \tilde{r}_y, \tilde{r}_{x'}, \tilde{r}_{y'})$, where $(\tilde{r}_x, \tilde{r}_y)$ and $(\tilde{r}_{x'}, \tilde{r}_{y'})$ are the top-left and bottom-right corners of the bounding-box which are normalized to have a value between 0 and 1 based on the height and width of the image. For each RGC, we project the concatenation of $\tilde{\mathbf{r}}$, $\mathbf{c}$ and $\tilde{\mathbf{c}}$ to a space of dimensionality D using a densely-connected layer with ReLU activation function to obtain a vector representation denoted by $\tilde{\mathbf{x}} \in \mathbb{R}^D$.

Question Module

We encode a question using a GRU and the same dictionary as we use for captions. This enables our model to match the words in a caption with the words in a question and attend to the relevant caption. The final hidden state of the question GRU, denoted by $\mathbf{q}$, is used as the representation of the question.

Multimodal Graph Memory Network Module

Given a set of visual feature vectors, a set of encoded RGCs and the encoded question, the multimodal graph memory network module produces a vector representation of the relevant information based on the encoded question. The memory chooses which parts of the inputs to focus on using an attention mechanism. Unlike previous works (Xu and Saenko, 2015; Xiong et al., 2016), our memory network module is multimodal and relational. That is, it employs both textual and visual information of the input image regions, and it exploits pair-wise interactions between each pair of visual/textual features using a visual/textual GN. Similar to visual features, most of the RGCs may be irrelevant to the given question. Thus, the memory module needs to learn an attention mechanism for focusing on the relevant RGCs. Formally, the multimodal graph memory network is composed of a visual GN $\mathcal{G} = (\mathbf{u}, \mathcal{V}, \mathcal{E})$ with N nodes, a textual GN $\tilde{\mathcal{G}} = (\tilde{\mathbf{u}}, \tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ with $\tilde{N}$ nodes, and an external spatial memory. Each node of the visual GN represents a visual feature with an associated bounding-box. Similarly, each node of the textual GN has a bounding-box corresponds to a detected RGC of the image. In both GNs, we connect two nodes via two forward and backward edges if they are nearby (i.e., the Euclidean distance between the normalized center of their bounding-boxes is less than γ). We set the γ to 0.5 in our experiments. Note that even if two nodes of a GN are not neighbors, they may still communicate via massage passing mechanism of the GN. The external memory is a network of memory cells arranged in a $P \times Q$ two-dimensional grid. Each cell has a fixed location corresponds to a specific $(H/P) \times (W/Q)$ region in the image, where H and W are height and width of the image. Each node of the visual/textual GN sends its information to a memory cell, if its bounding-box covers the location of the cell. Since the bounding-boxes may overlap, a cell may get information from multiple nodes. The external memory network is responsible for aggregating the information from both GNs and eliminating redundancy introduced by overlapping bounding-boxes. This make our architecture less sensitive to the number of detected bounding-boxes.

Initialization

To initialize each node attribute of the visual GN, we combine a visual feature vector extracted from a region of the image with the encoded question using multimodal compact bilinear (MCB) pooling (Fukui et al., 2016) as $\mathbf{v}_i = \mathbf{q} \star \mathbf{x}_i$, where $\star$ represents the MCB pooling. Similarly, we initialize each node attribute of the textual GN as $\tilde{\mathbf{v}}_i = \mathbf{q} \odot \tilde{\mathbf{x}}_i$, where $\odot$ is the element-wise multiplication. We use the MCB to combine the visual features with the encoded question, since the question and visual features are from different modalities. The global attribute $\mathbf{u}$ is initialized by a global feature vector of the image extracted from the last layer of the 101-layer ResNet, and the global attribute $\tilde{\mathbf{u}}$ is initialized with the encoded question. The edge features of the GNs and memory cells are initialized with zero vectors.

Updates

At each iteration, we first update the GNs. Then, we we update the content of the memory cells. We update edge attributes, node attributes and the global attribute of both GNs as described in Algorithm 1. For each GN, we use three different GRUs to implement the functions ϕ^e and ϕ^v, ϕ^u. The $\rho^{e\to v}$ is a simple element-wise summation. The $\rho^{v\to u}$ and $\rho^{e\to u}$ for visual GN are implemented as

$$\bar{\mathbf{v}}' = \psi\big(\textstyle\sum_{i=1}^{N} \sigma(\mathbf{W}_1\mathbf{v}_i + \mathbf{b}_1) \odot \psi(\mathbf{W}_2\mathbf{v}_i + \mathbf{b}_2)\big) \tag{4.1}$$

$$\bar{\mathbf{e}}' = \psi\big(\textstyle\sum_{k=1}^{M} \sigma(\mathbf{W}_3\mathbf{e}_k + \mathbf{b}_3) \odot \psi(\mathbf{W}_4\mathbf{e}_k + \mathbf{b}_4)\big) \tag{4.2}$$

where, σ and ψ are the sigmoid and tangent hyperbolic activation functions, and $\mathbf{W}_i$, $\mathbf{b}_i, i = 1, \ldots, 4$, are trainable parameters. This allows to incorporate information from the question for computing the attention weights using the sigmoid function for each node/edge. The $\rho^{v\to u}$ and $\rho^{e\to u}$ for the textual GN are implemented in a similar way. Let, $\bar{\mathbf{v}}_{p,q} = \frac{1}{|\mathcal{N}_{p,q}|}\sum_{i\in\mathcal{N}_{p,q}} \mathbf{v}_i$ and $\bar{\tilde{\mathbf{v}}}_{p,q} = \frac{1}{|\tilde{\mathcal{N}}_{p,q}|}\sum_{i\in\tilde{\mathcal{N}}_{p,q}} \tilde{\mathbf{v}}_i$, where $\mathcal{N}_{p,q}$ and $\tilde{\mathcal{N}}_{p,q}$ are the set of nodes which are connected to the memory cell (p, q) in the visual and textual GNs, respectively. Each memory cell is updated as

$$\bar{\mathbf{m}}_{p,q} = f(\mathbf{m}_{p-1,q}, \mathbf{m}_{p,q-1}, \mathbf{m}_{p,q+1}, \mathbf{m}_{p+1,q}) \tag{4.3}$$

$$\mathbf{m}'_{p,q} = \mathrm{GRU}([\bar{\mathbf{v}}_{p,q}, \bar{\tilde{\mathbf{v}}}_{p,q}, \bar{\mathbf{m}}_{p,q}], \mathbf{m}_{p,q}) \tag{4.4}$$

where f is a neural network layer which aggregates the memories from the neighboring cells. We repeat these steps for two iterations. As observed by Kumar et al. (2015), iterating over the inputs allows the memory network to take several reasoning steps, which some questions require.

4.2.1 Answer Module

The answer module predicts an answer using a GN called answer GN. The nodes of the answer GN are the external spatial memory cells. However, there is an edge between every ordered pair of the nodes (cells), hence the answer GN is a complete graph. This supports reasoning across distant regions of the image. Let $\mathbf{m}^{\circ}_{p,q}$ be the final state of the memory cell at location (p, q). We initialize the node attributes of the answer GN denoted by $\mathbf{v}^{\circ}_{p,q}$ as $\mathbf{v}^{\circ}_{p,q} = \mathbf{m}^{\circ}_{p,q}$. The edge attributes are initialized using the one-hot representation of the location of the sender and receiver memory cells. That is, the edge attribute of the edge going from memory cell at location (p, q) to (p', q'), is initialized with a vector of size $2P+2Q$ which is computed by concatenating the one-hot representation of p, q, p', and q'. The global attribute of the answer GN is initialized with a vector of zeros. Then, we update the edge attributes, the node attributes and the global attribute of the answer GN as described in Algorithm 1. As before, we use three different GRUs to implement functions ϕ^e and ϕ^v, ϕ^u. The $\rho^{e\to v}$ is a simple element-wise summation. The $\rho^{v\to u}$ and $\rho^{e\to u}$ are implemented as 4.1 and 4.2, but with different set of parameters. The answer module predicts an answer as

$$\hat{\mathbf{p}} = \sigma(\mathbf{W}g(\mathbf{u}^{\circ}) + \tilde{\mathbf{W}}\tilde{g}(\mathbf{u}^{\circ}) + \mathbf{b}) \tag{4.5}$$

where, $\mathbf{u}^{\circ}$ is the updated global attribute of the answer GN, $\mathbf{W} \in \mathbb{R}^{Y\times 2048}$, $\tilde{\mathbf{W}} \in \mathbb{R}^{Y\times 300}$, $\mathbf{b} \in \mathbb{R}^{Y}$ are trainable parameters, g, $\tilde{g}$ are non-linear layers, and Y is the number of possible answers. Following Teney et al. (2018), to exploit prior linguistic information about the candidate answers, the GloVe embeddings of the answer words are used to initialize the rows of the $\tilde{\mathbf{W}}$. Similarly, to utilize prior visual information about the candidate answers, a visual embedding is used to initialize the rows of $\mathbf{W}$. The visual embedding is obtained by retrieving 10 image from Google Images for each word. Then, the images are encoded using the ResNet-101 pretrained on ImageNet to obtain a feature vector of size 2048. For each word, the average of the feature vectors is used to initialize a row of $\mathbf{W}$. The loss for a single sample is defined as $\mathcal{L} = -\sum_{i=1}^{Y} p_i \log(\hat{p}_i) + (1 - p_i)\log(1 - \hat{p}_i)$ where, $\hat{p}_i$ is the ith element of $\hat{\mathbf{p}}$, and p_i is the ith element of the ground-truth vector $\mathbf{p}$ ($p_i = 1.0$ if $A \geq 3$ annotators give the ith answer word, otherwise $p_i = A/3$).

For multiple choice task, the candidate answers are encoded by the last state of a GRU and concatenated with $\mathbf{u}^{\circ}$ using a neural network layer as $\acute{p} = \sigma(\acute{\mathbf{w}}\acute{f}([\mathbf{u}^{\circ}, \mathbf{a}]) + \acute{b})$ where, $\mathbf{a}$ is an encoded answer choice, $\acute{f}$ is a non-linear layer, and $\acute{\mathbf{w}}, \acute{b}$ are trainable parameters. For multiple choice task, the binary logistic loss $-p\log(\acute{p}) - (1-p)\log(1-\acute{p})$ is used, where p is 1.0 for an (image,question,answer) triplet, if the answer choice is correct, otherwise p is 0.

Training Details and Optimization

The MN-GMN is implemented in TensorFlow framework. We use a library from `https://github.com/deepmind/graph_nets` to implement the GNs. To minimize the loss, we

apply RMSprop optimization algorithm with learning rate 0.0001 and minibatches of size 100. To prevent overfitting, dropout with probability 0.5 and early stopping are applied. During training, all parameters are tuned except for the weights of the CNN and RGC detector components to avoid overfitting. The output dimension of the MCB is set to 512. The dimension of the hidden layer in both RGC and question GRUs is set to 512. Also, we set $P, Q = 14$ and $D = 512$. Our model takes around 6 hours to train on two NVIDIA Titan X GPUs. To apply an ensemble technique, 20 instances of the model is trained with various initial random seeds. For test images, the scores for the answers by all models are summed, and the answer is predicted using the highest summed score.

4.3 Experiments

In this section, we explain the datasets, baseline models and evaluation metric that we use in our experiments. Then, the experimental results are presented and discussed.

Datasets

VQA v2.0 (Antol et al., 2015) includes $82,783$ training images, $40,504$ validation images, and $81,434$ testing images. There are $443,757$ training questions, $214,354$ validation questions, and $447,793$ test questions in this dataset. A subset of the standard test set, called test-dev, contains $107,394$ questions. Each question has 10 candidate answers generated by humans. We choose correct answers that appear more than 8 times. This makes $Y = 3,110$ candidate answers. We use the standard metric (Antol et al., 2015): an answer is correct if at least 3 people agree.

Visual7W dataset (Zhu et al., 2015) includes $47,300$ images which occur in both Visual Genome and MS-COCO datasets. We train and evaluate our model on *telling* questions of the Visual7W which includes $28,653$ images. This set uses six types of questions: *what* (6%), *where* (48%), *when* (16%), *who* (5%), *why* (10%), *how* (15%). The training, validation and test splits, contain 50%, 20%, 30% of the QA pairs, respectively. For evaluation, Visual7W provides four candidate answers. The Visual7W has fewer language biases compared to VQA.

We also experiment on CLEVR dataset (Johnson et al., 2017a). CLEVR evaluates different aspects of visual reasoning such as attribute recognition, counting, comparison, logic, and spatial relationships. Each object in an image has the following attributes: shape (*cube*, *sphere*, or *cylinder*), size (*large* or *small*), color (8 colors), and material (*rubber* or *metal*). An object detector with 96 classes is trained using all combinations of the attributes by the Tensorflow Object Detection API. Given an image, the output of the object detector is a set of object bounding-boxes with their feature vectors. For CLEVR, we omit the textual GN, since the CLEVR images do not have rich textual information.

Baseline Methods

For VQA dataset, we compare our model with several architectures developed recently including the state-of-the-art models ReGAT, BAN, VCTREE, and MuRel. The ReGAT uses Visual Genome relationships as external training data. MAN (Ma et al., 2018) is a memory-augmented neural network which attends to each training exemplar to answer visual questions, even when the answers happen infrequently in the training set. The Count (Zhang et al., 2018a) is a neural network model designed to count objects from object proposals. For Visual7W, we compare our models with Zhu et al. (2015), MCB, MAN, and MLP. MCB leverages the Visual Genome QA pairs as additional training data and the 152-layer ResNet as a pretrained model. This model took the first place in the VQA Challenge workshop 2016. The MLP method uses (image,question,answer) triplets to score answer choices. For CLEVR, we compare our models with several baselines proposed by Johnson et al. (2017a) as well as state-of the art models PROGRAM-GEN (Johnson et al., 2017b) and CNN+LSTM+RN (Santoro et al., 2017). N2NMN (Hu et al., 2017) learns to predict a layout based on the question and compose a network using a set of neural modules. CNN+LSTM+RN learns to infer a relation using a neural network model called Relation Networks. PROGRAM-GEN exploits supervision from functional programming which is used to generate CLEVR questions.

Ablation Study

We implement several lesion architectures. The MN+Res model does not use any GNs and is designed to evaluate the effect of using GN. This model is similar to Memory Networks (Sukhbaatar et al., 2015): Soft attention for the 14×14 ResNet feature maps (the last 14×14 pooling layer) generates a representation $\mathbf{u}^{\circ} = h(\mathbf{q}) \star h'(\sum_{i=1}^{196} \alpha_i \mathbf{x}_i)$. Here h, h' are non-linear layers, and α_i is an attention weight computed as $\alpha_i = \text{softmax}(\mathbf{w}h''([\mathbf{x}_i, \mathbf{q}]))$, where $\mathbf{w}$ is a learned parameter vector and h'' is a non-linear layer. Then, using Eq. 4.5 an answer is predicted. The N-GMN model only uses the visual GN (no textual GN nor spatial memory). This model evaluates the effect of incorporating RGCs. After two iterations, the global feature vector of the visual GN is used as $\mathbf{u}^{\circ}$ in Eq. 4.5 to generate an answer. The N-GMN^{+} model only uses the visual GN and the external spatial memory components (no textual GN). This model is used for CLEVR dataset, since CLEVR images do not have rich textual information. The MN-GMN^{-} model does not use the external spatial memory. After two iterations, the global feature vector of the visual and textual GNs are concatenated and fed into a non-linear layer to generate $\mathbf{u}^{\circ}$ in Eq. 4.5. Finally, MN-GMN is our full model.

Results and Discussion

Our experimental results on VQA dataset are reported in Table 4.1. Across all question types, N-GMN outperforms MN+Res. This shows that applying the visual GN with ex-

plicit object bounding-boxes provides a usefully richer representation than a grid of fixed visual features. MN-GMN$^-$ outperforms N-GMN. This shows that RGCs help to improve the accuracy. RGCs are especially useful for answering the Other and Yes/No question types. Our full model MN-GMN outperforms MN-GMN$^-$. This shows that applying external spatial memory is effective, especially for Number questions. The full model's accuracy is higher than the baselines.

	Test-dev				Test-std			
Model	Y/N	Num	Other	All	Y/N	Num	Other	All
MAN (Ma et al., 2018)	-	-	-	-	79.2	39.5	52.6	62.1
Count (Zhang et al., 2018a)	83.1	51.6	59.0	68.1	83.6	51.4	59.1	68.4
MFH (Yu et al., 2018)	84.3	50.7	60.5	68.8	-	-	-	-
Buttom-Up (Teney et al., 2018)	81.8	44.2	56.1	65.3	-	-	-	65.7
G-learner (Norcliffe-Brown et al., 2018)	-	-	-	-	82.9	47.1	56.2	66.2
v-AGCN (Yang et al., 2018b)	82.4	45.9	6.5	65.9	82.6	45.1	56.7	66.2
MuRel (Cadene et al., 2019)	84.8	49.8	57.9	68.0	-	-	-	68.4
VCTREE (Tang et al., 2019)	84.3	47.8	59.1	68.2	84.6	47.4	59.3	68.5
BAN (Kim et al., 2018)	85.4	54.0	60.5	70.0	-	-	-	70.4
ReGAT (Li et al., 2019)	86.1	54.4	60.3	70.3	-	-	-	70.6
MN+Res	84.2	43.4	58.1	67.3	84.5	44.0	58.1	67.5
N-GMN	86.1	53.5	61.2	70.6	86.7	53.6	61.8	71.2
MN-GMN$^-$	88.0	53.5	63.8	72.6	88.5	53.7	64.2	73.1
MN-GMN	**88.2**	**56.0**	**64.2**	**73.2**	**88.3**	**56.1**	**64.5**	**73.5**

Table 4.1: Accuracy percentage of various models on VQA-v2.0.

Model	What	Where	When	Who	Why	How	Avg
Human (Zhu et al., 2015)	96.5	95.7	94.4	96.5	92.7	94.2	95.7
LSTM-ATT (Zhu et al., 2015)	51.5	57.0	75.0	59.5	55.5	49.8	54.3
Concat+ATT (Fukui et al., 2016)	47.8	56.9	74.1	62.3	52.7	51.2	52.8
MCB+ATT (Fukui et al., 2016)	60.3	70.4	79.5	69.2	58.2	51.1	62.2
MAN (Ma et al., 2018)	62.2	68.9	76.8	66.4	57.8	52.9	62.8
MLP (Jabri et al., 2016)	64.5	75.9	82.1	72.9	68.0	56.4	67.1
N-GMN	66.2	77.2	83.3	74.0	69.2	58.5	68.6
MN-GMN$^-$	67.1	77.4	84.0	**75.1**	70.1	59.2	69.3
MN-GMN	**67.3**	**77.4**	**84.0**	75.0	**70.3**	**59.4**	**69.5**

Table 4.2: Accuracy Percentage on Visual7W.

Our results on Visual7W are reported in Table 4.2. Our N-GMN, MN-GMN$^-$, and MN-GMN outperform the baselines MLP, MAN and MCB+ATT. The results for our N-GMN$^+$ on CLEVR in Table 4.3 are competitive with the state-of-the-art PROGRAM-GEN. We emphasize that, unlike PROGRAM-GEN, our algorithm does not exploit supervision from functional programming.

Qualitative Evaluation

Figure 4.3 and Figure 4.4 show how MN-GMN can answer a question correctly by incorporating RGCs, whereas N-GMN gives the wrong answer. Figure 4.5 and Figure 4.6 illustrate

Model	All	Exist	Count	Cmp-Int	Q-At	Cmp-At
HUMAN (Johnson et al., 2017a)	92.6	96.6	86.7	86.5	95.0	96.0
Q-TYPE MODE (Johnson et al., 2017a)	41.8	50.2	34.6	51.0	36.0	51.3
LSTM (Johnson et al., 2017a)	46.8	61.1	41.7	69.8	36.8	51.3
CNN+BOW (Johnson et al., 2017a)	48.4	59.5	38.9	51.1	48.3	51.8
CNN+LSTM (Johnson et al., 2017a)	52.3	65.2	43.7	67.1	49.3	53.0
CNN+LSTM+MCB (Johnson et al., 2017a)	51.4	63.4	42.1	66.4	49.0	51.0
CNN+LSTM+SA (Johnson et al., 2017a)	68.5	71.1	52.2	73.5	85.3	52.3
N2NMN (Hu et al., 2017)	83.3	85.7	68.5	85.0	90.0	88.8
CNN+LSTM+RN (Santoro et al., 2017)	95.5	97.8	90.1	93.6	97.9	97.1
PROGRAM-GEN (Johnson et al., 2017b)	**96.9**	97.1	**92.7**	**98.7**	**98.2**	**98.9**
N-GMN$^-$	95.6	97.7	90.3	93.5	98.0	97.3
N-GMN$^+$	96.3	**98.0**	91.8	94.8	98.1	98.1

Table 4.3: Accuracy Percentage on CLEVR.

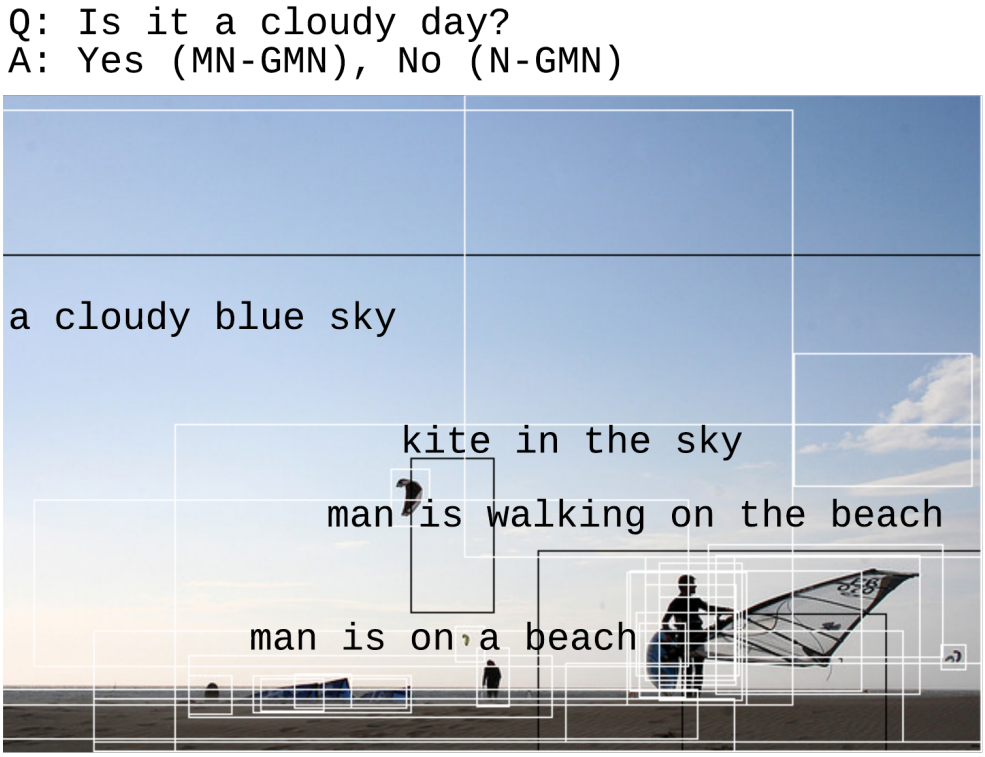


Figure 4.3: N-GMN versus MN-GMN. The MN-GMN provides the correct answer using *a cloudy blue sky*.

visualization of the attention weights with MN-GMN to answer a Number question. For this example, we compute the attention weights which are used to obtain $\bar{\mathbf{v}}'$ for each memory cell. (More precisely, the magnitude of the sigmoid output which implements $\rho^{v \to u}$ for the external spatial memory; cf. Eq. 4.1). Each attention weight shows the importance of a fixed region in a 14×14 grid of cells to the question. Figure 4.7 and Figure 4.8 show example VQA on CLEVR dataset.

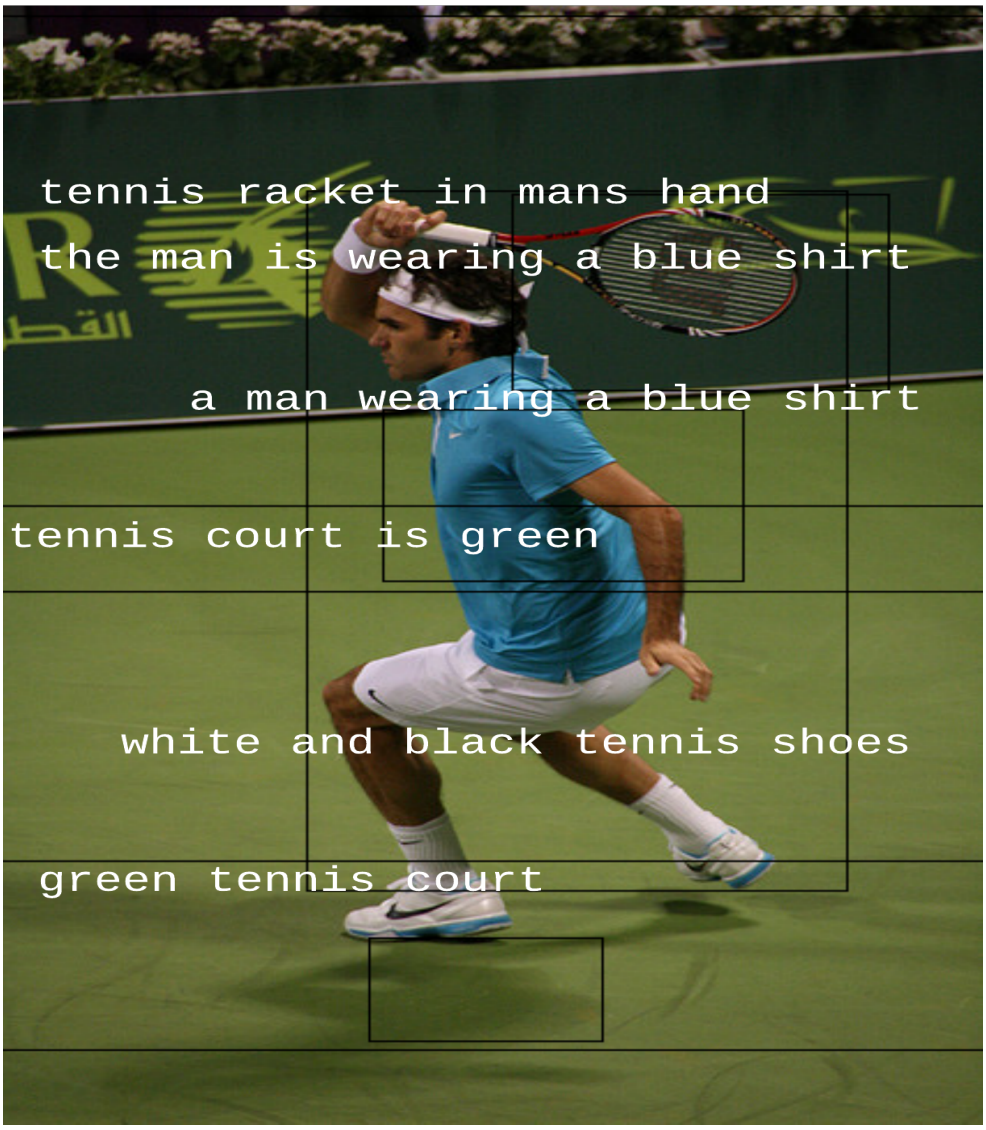


Figure 4.4: N-GMN versus MN-GMN. The MN-GMN provides the correct answer using *white and black tennis shoes*.

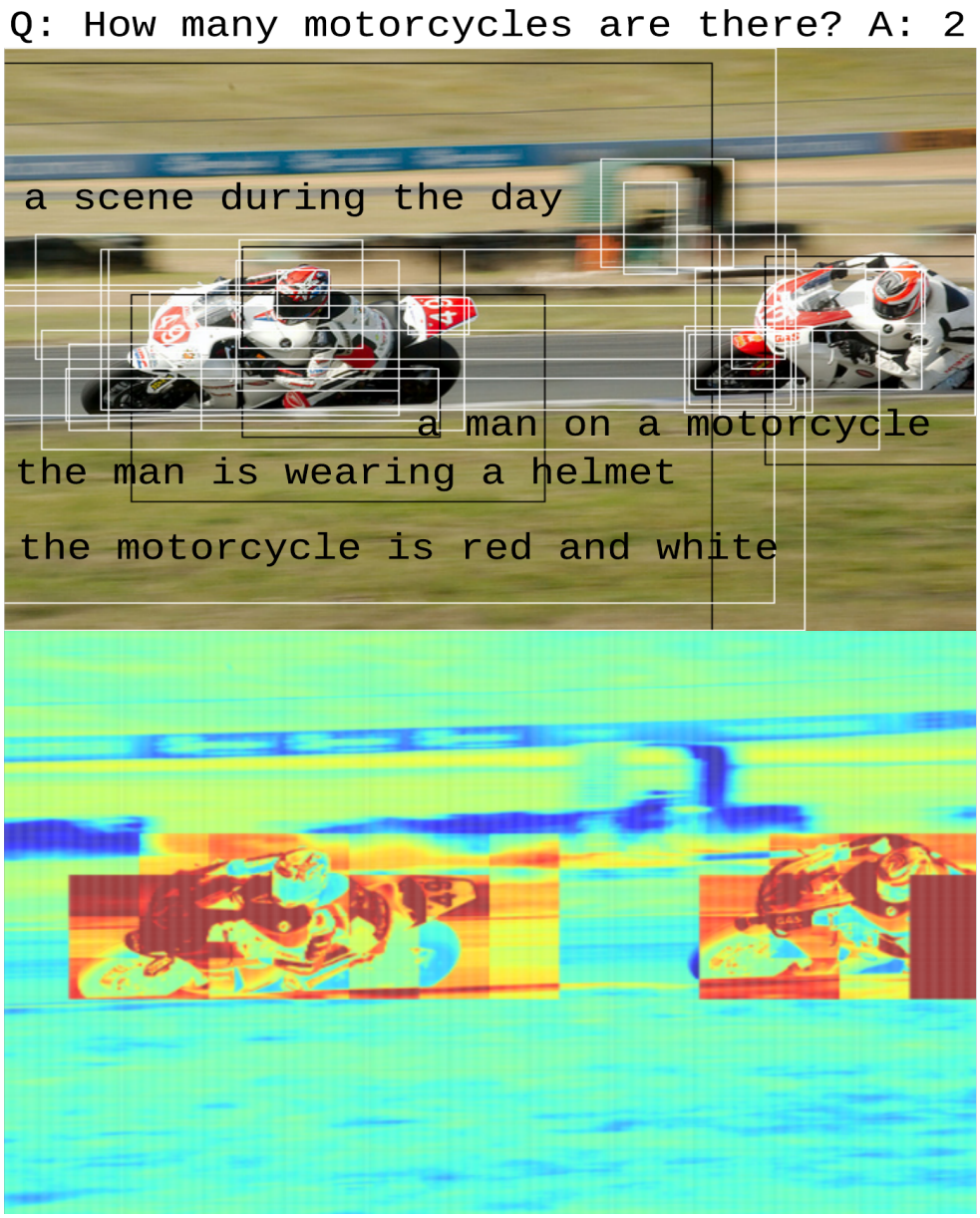


Figure 4.5: Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention.

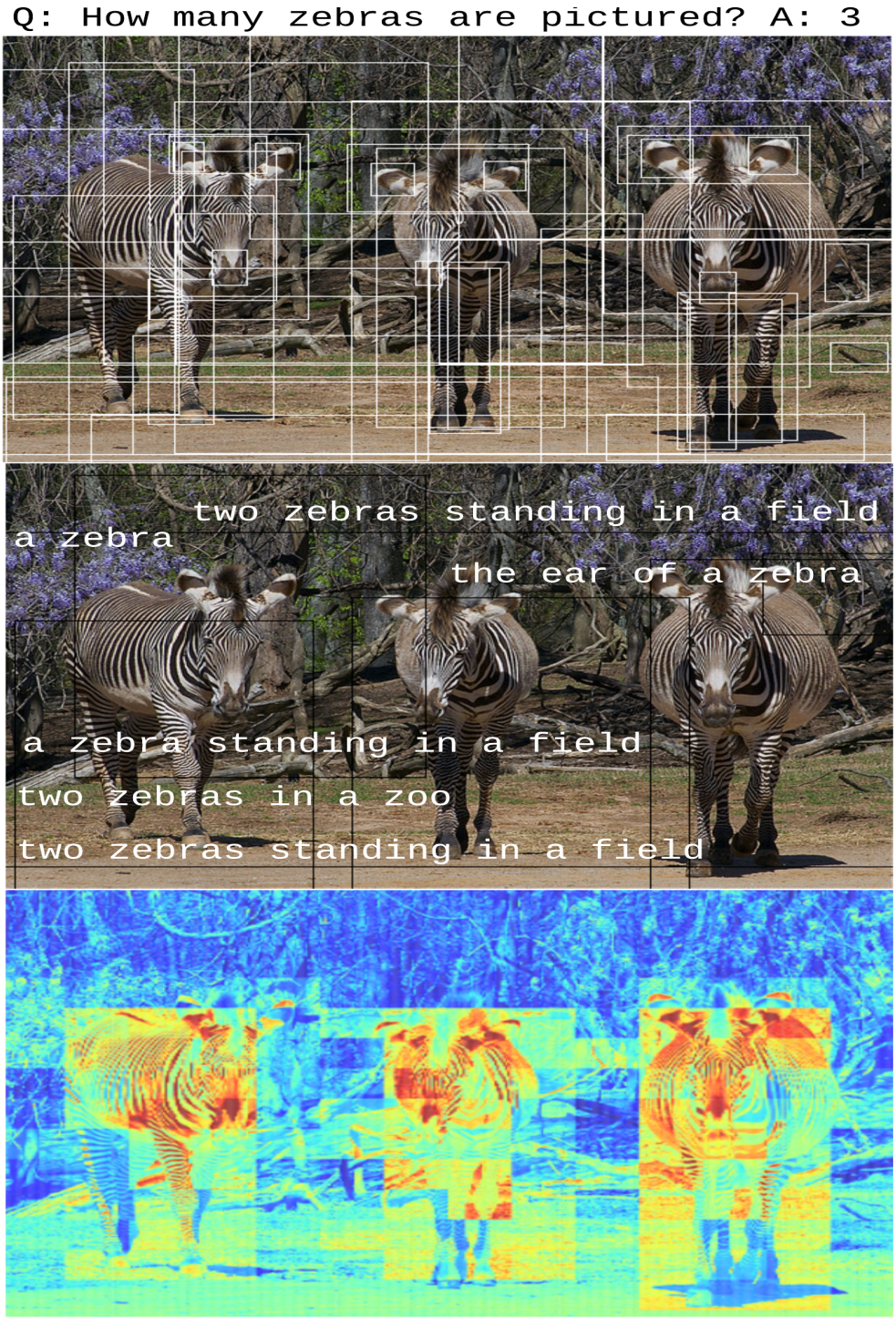


Figure 4.6: Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention.

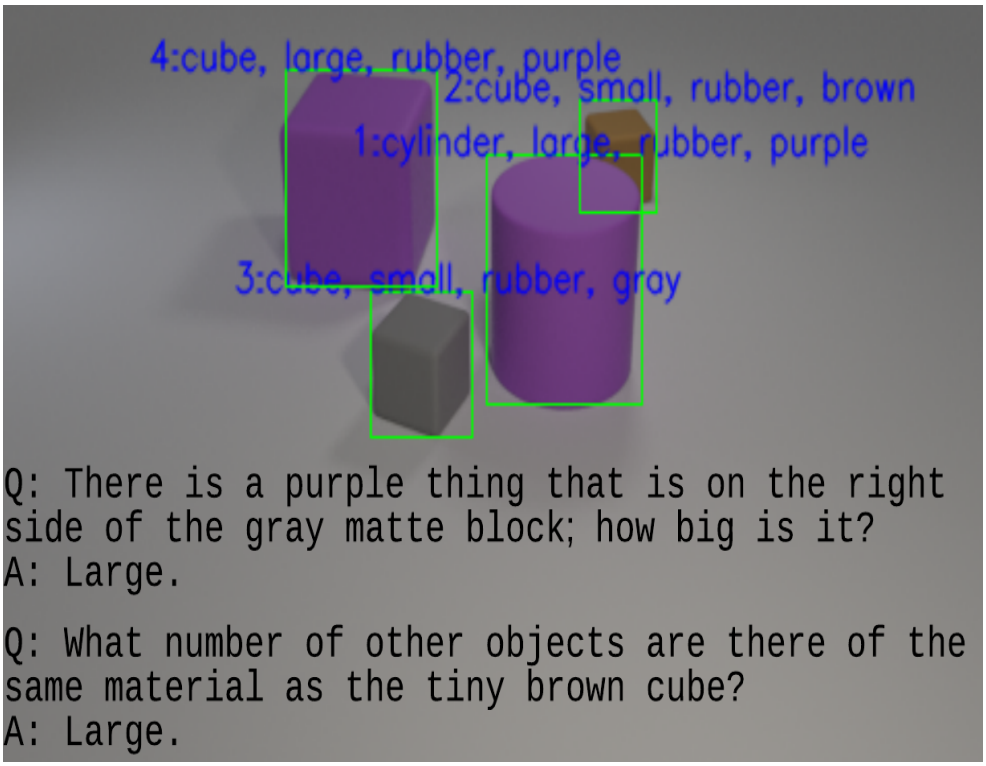


Figure 4.7: Example VQA with N-GMN on CLEVR dataset.

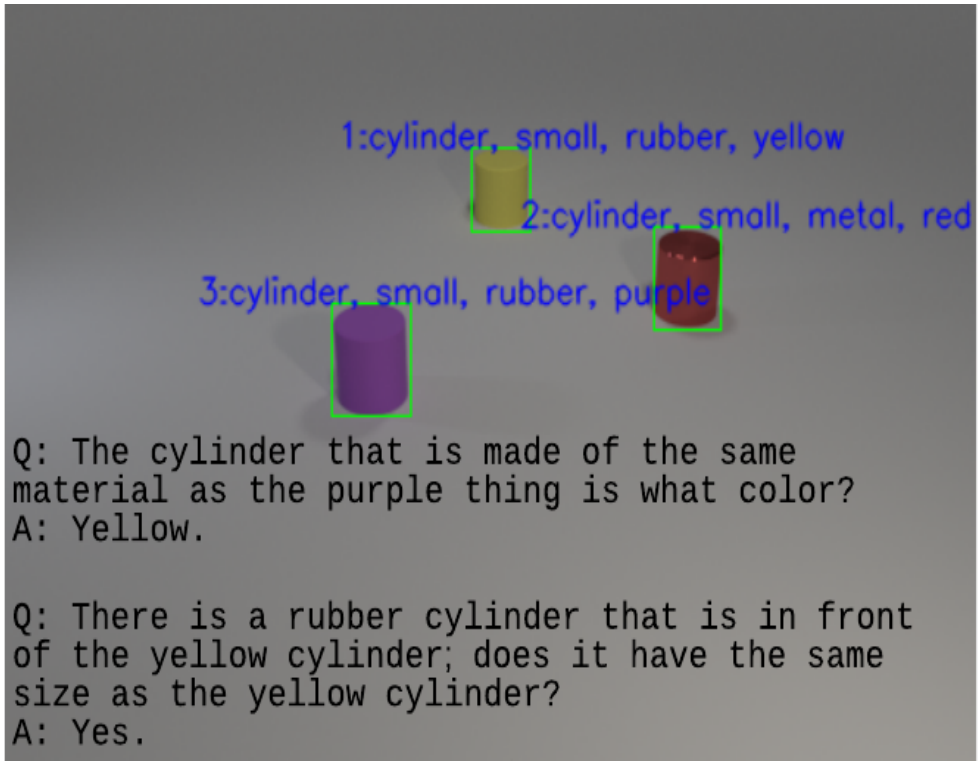


Figure 4.8: Example VQA with N-GMN on CLEVR dataset.

Chapter 5

Graph Neural Networks for Scene Graph Generation

In this chapter, we introduce the highest-level representation: a scene graph. The nodes in a scene graph represent semantically meaningful objects and the edges represent semantically meaningful abstract predicates. For example, a scene graph may contain two objects *man* and *horse* linked by an edge representing the predicate *riding*. Because this high-level representation can be inferred only subject to uncertainty, we augment a scene graph by estimating probabilities for its components.

To generate a scene graph, we develop two novel approaches. 1) Deep Generative Probabilistic Graph Neural Networks (DG-PGNN) combine probabilistic scene graphs with reinforcement learning and graph neural network processing for feature generation. 2) Dynamic Gated Graph Neural Networks (D-GGNN) combine breadth-first search with reinforcement learning and gated graph neural network techniques for feature representation. As a validation task, we apply the generated scene graphs to VQA task. We now describe these two algorithms and compare them.

5.1 Deep Generative Probabilistic Graph Neural Networks

Visual understanding of a scene is one of the most important objectives in computer vision. Over the past decade, there have been great advances in relevant tasks such as image classification and object detection. However, understanding a scene is not only recognizing the objects in the scene. The interactions between objects also play a crucial role in visual understanding of a scene. To represent the semantic content of an image, previous work proposed to build a graph-structured representation called scene graph (Krishna et al., 2016; Lu et al., 2016a; Li et al., 2017), where the nodes represent the objects and the edges show the relationships between them (see Figure 5.1). The visual information represented by a scene graph is beneficial in applications such as visual question answering (Teney et al.,

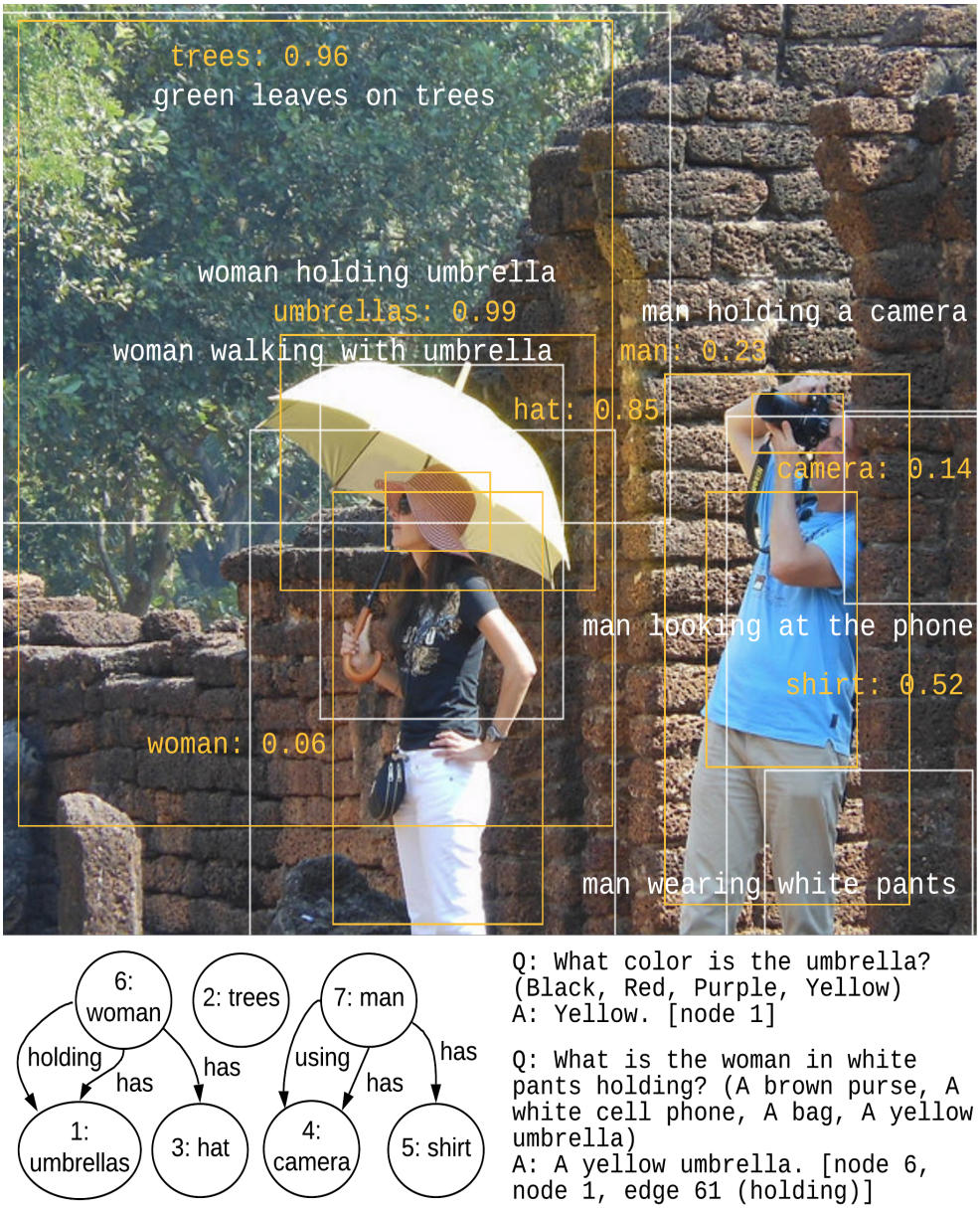


Figure 5.1: Scene graph represents semantic content of an image via a graph. We leverage region-grounded captions to construct a scene graph. Scene graph is useful to answer visual questions such as *What color is the umbrella?* (features of node *umbrella*) and *What is the woman in white pants holding?* (relationship triplet *woman-holding-umbrella*).

2016). This work therefore introduces a new algorithm, Deep Generative Probabilistic Graph Neural Networks (DG-PGNN), to generate a scene graph for an image.

An efficient scene graph generation model needs to leverage visual contextual information as well as language priors. For example, if we know that a node has an incoming edge from another node which represents a *man* and the type of the edge is *riding*, then the type of the node is likely *horse* or *bicycle*, but not *dog*. Previous work proposed to learn a

contextualized representation for nodes and edges either by sending messages across a fixed complete graph (Xu et al., 2017), where each node is potentially an object, or by assuming a fixed linear ordering for the bounding-boxes and applying a bidirectional LSTM (Zellers et al., 2018). The node and edge representations are then used to infer the node-types and edge-types. In these models, potential objects are equated with detected bounding-boxes. However, a detected box often does not represent an actual object, and relationship prediction presents an imbalanced classification problem since there is no relationship between most of the detected boxes. Also, since these models assume a fixed structure, they cannot leverage the rich latent graph-structured nature of the input. Unlike these approaches, our method is well-designed for latent graph-structured input data; we simultaneously build the graph and fine-tune the predictions, as we get new information about the scene.

We introduce a new graph neural network model, Probabilistic Graph Network (PGN), to represent a scene graph with uncertainty. The PGN is a probabilistic extension to Graph Network (Battaglia et al., 2018); see also Chapter 2. In a PGN, each node and each edge is represented by a CNN feature vector, and the degree of belief about node-type (object category) of a node and edge-type (predicate class) of an edge is represented by a probability mass function (PMF). Our novel DG-PGNN algorithm constructs a PGN as follows. It sequentially adds a new node to the current PGN by learning the optimal ordering in a Deep Q-learning framework, where states are partial PGNs, actions choose a new node, and rewards are defined based on the ground-truth scene graphs. After adding a node, DG-PGNN uses message passing to update the feature vectors of the current PGN by leveraging contextual relationship information, object co-occurrences, and language priors from captions. The updated features are then used to fine-tune the PMFs. Our experiments show that DG-PGNN substantially outperforms the state-of-the-art models for scene graph generation on Visual Genome dataset. Our model can scale to predict hundreds of predicate classes and thousands of object categories.

We also evaluate the usefulness of DG-PGNN by applying it to the visual question answering (VQA) task. In spite of recent advances in VQA, current VQA models often fail on sufficiently new samples, converge on an answer after listening to only a few words of the question, and do not alter their answers across different images. Our point of departure is that success at VQA task requires a rich representation of the objects and their visual relationships in an image, which identifies the attributes of these objects, and supports reasoning about the role of each object in the scene context (Figure 5.1). To apply DG-PGNN to the VQA task, we first construct a scene graph for the image. Then, the predicted scene graph is used to produce a vector representation of the image and question information by attending to the nodes and edges which are relevant to the question. This enables our model to answer questions which need reasoning about the objects and their interactions.

In summary, our contributions are the following: i) A new graph neural network model for representing the uncertainty associated with a scene graph, ii) A new scene graph con-

struction algorithm that combines deep feature learning with probabilistic message passing in a completely differentiable probabilistic framework. The DG-PGNN sequentially adds a new node to the graph by learning the optimal ordering in a reinforcement learning (RL) framework. The motivation for using RL is that a graph can be generated through sequentially choosing components, and Markov decision process is our most effective framework for sequential decision making. Also, RL is a scalable and differentiable approach in structure prediction tasks. This is a new approach to the optimal ordering problem for generating a graph neural network. Although we focus on scene graph generation task, the DG-PGNN can be seen as a generic generative graph neural network algorithm for feature learning from latent graph-structured data, when the graph structure is unknown in advance, e.g. knowledge graph construction from texts. iii) To the best of our knowledge, this is the first work that explicitly exploits textual information of an image to build a scene graph for the image. The textual information of an image provide a rich set of information such as object attributes and their visual relationships to construct a precise scene graph. For example, caption *a man is riding a horse* increases the chance of presence of objects *man* and *horse*, and predicate *riding* in the scene graph of the image. The region-grounded caption detector which is used by our DG-PGNN may be trained on any available region-grounded caption datasets. This introduces a new transfer learning technique for scene graph generation task. The content of this section has appeared in the NeurIPS 2019 Graph Representation Learning Workshop. A detailed version of this paper will appear in the proceedings of the thirty-fourth AAAI Conference on Artificial Intelligence (AAAI-2020).

5.1.1 Related Work

Our work is related to several areas in computer vision, natural language processing and deep learning.

Scene Graph Generation

Schuster et al. (2015) introduced a rule-based and a classifier based method for scene graph generation from a natural language scene description. Johnson et al. (2015b) proposed a model based on a conditional random field that reasons about various groundings of potential scene graphs for an image. Liang et al. (2017) proposed a model for visual relationship and attributes detection based on reinforcement learning. In their method, to extract a state feature, they used the embedding of the last two relationships and attributes that were searched during the graph construction. This will lead to limited representational power, since the resulting representation depends on the order that the algorithm selects node and edges. Hence, this method may generate different representations for the same graph. Recently, Li et al. (2017) proposed a neural network model, called Multi-level Scene Description Network (MSDN), to address the object detection, region captioning, and scene graph generation tasks jointly. They aligned object, phrase, and caption regions by apply-

ing a dynamic graph using their spatial and semantic links. Then, they applied a feature refining schema to send massages across features of the phrase, objects, and caption nodes via the graph. Lu et al. (2016a) proposed a model to detect a relatively large set of relationships using language priors from semantic word embeddings. Zellers et al. (2018) proposed to learn a contextualized representation for nodes and edges by assuming a fixed linear ordering for the bounding-boxes and applying a bidirectional LSTM. In Newell and Deng (2017), authors proposed to train a CNN which takes in an image and produces a scene graph in an end-to-end framework. In Tang et al. (2019), authors introduced a model which composes dynamic tree structures that put the objects in an image into a visual context to improve scene graph generation accuracy. In Xu et al. (2017), authors developed a model for scene graph generation which uses an RNN and learns to improves its predictions iteratively through message passing across the scene graph.

Using Graph Structures for VQA

Teney et al. (2016) introduced a VQA model based on structured representations of both scene contents and questions for abstract scenes with a limited number of object classes, a small set of discrete features for each objects, and no relationship between them, except the spatial relationships. While highly effective, a limitation is that the scene graph must be given at test time.

5.1.2 Proposed DG-PGNN Algorithm

In this section, we first introduce the Probabilistic Graph Network (PGN). Then, we explain how to construct a complete PGN with N nodes using all bounding-box proposals of the input image, where N is number of bounding-boxes. This complete PGN provides initial node feature vectors, edge feature vectors, and PMFs for the DG-PGNN algorithm. Next, we explain the DG-PGNN algorithm to generate a scene graph for an image. Staring from a null PGN, the algorithm sequentially add a new node to the current PGN (see Figure 5.2 and supplementary materials). At each step, the algorithm has two major parts: i) a Q-learning framework which is responsible for selecting the next node. ii) PGN updates which update the current PGN after adding the new node and edges. Whenever we add a new node to the current PGN, we connect the new node to each node of the current PGN using two directed edges. The PMFs between the new node and each node in the current PGN, the feature vector for the new node, and the feature vector for the new edges are initialized by copying the corresponding feature vectors and PMFs from the complete PGN. Then, the node features, edge features, and PMFs of the current graph are updated by exploiting visual contextual information as well as the region-grounded captions of the image, hence reducing uncertainty about objects and predicate classes. We use a GRU to implement an update. But, neural networks and LSTMs may also be applied. After completion of the algorithm, we eliminate the edges which have a probability less than 0.5 and the nodes

with background node-type from the last PGN. The remaining graph is the predicted scene graph. In the following, we explain the algorithm components and provide intuitions for their usage.

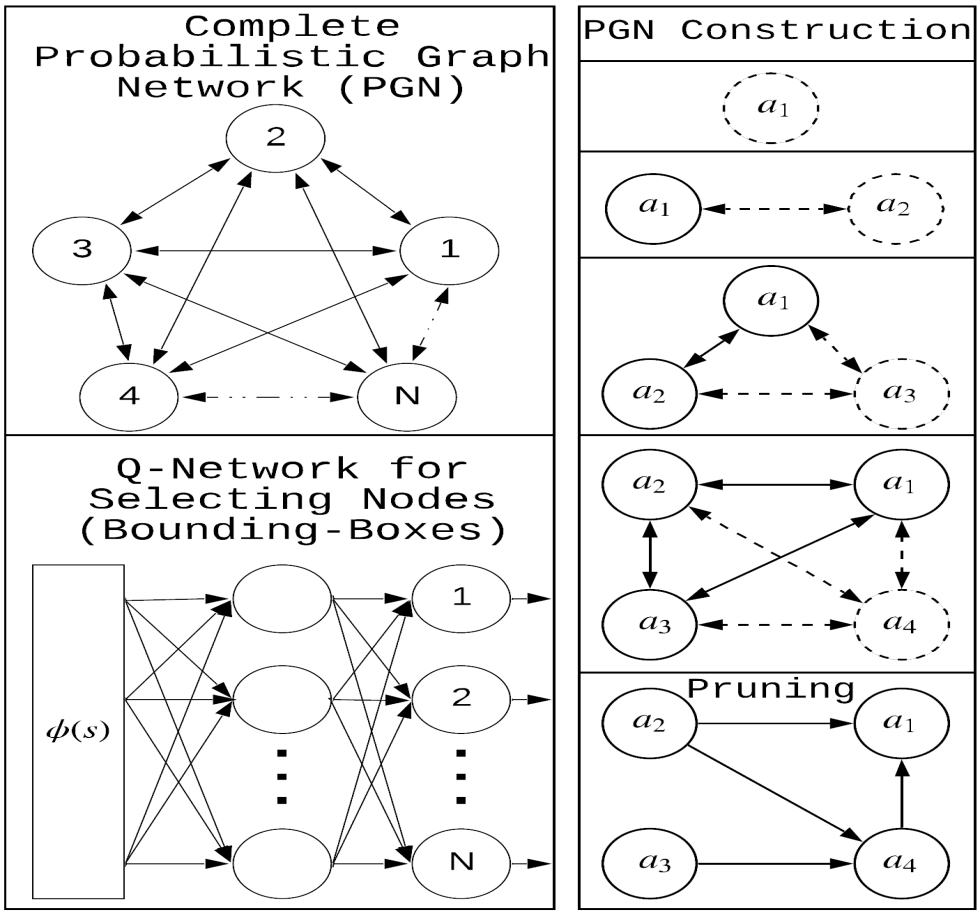


Figure 5.2: Scene graph generation using DG-PGNN.

Probabilistic Graph Networks

A Probabilistic Graph Network (PGN) is defined based on a given (scene) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes (bounding-boxes), and $\mathcal{E}$ is a set of probabilistic adjacency matrices. Each $\mid \mathcal{V} \mid \times \mid \mathcal{V} \mid$ matrix $E_k \in \mathcal{E}$ represents a probabilistic adjacency matrix for edge-type (predicate class) k. That is, for each $k \in \{1, \ldots, K\}$, $E_k(u, v)$ is the probability that there exists an edge of type k going from node u to node v, where K is the number of edge-types. We write $\mathbf{e}_{u,v} = [E_1(u, v), \ldots, E_K(u, v)]^\top \in \mathbb{R}^K$ for the PMF from u to v. The sum of the entries of a PMF must equal 1. We relax this condition for edges, to allow zero or multiple edges with various edge-types from u to v. Each edge $e = (u, v)$ has an edge representation (feature) $\mathbf{h}_{u,v} \in \mathbb{R}^D$. All edges from u to v with different edge-types share the same edge representation. Also, each node v has a node representation $\mathbf{h}_v \in \mathbb{R}^D$ and a node-type PMF

$\mathbf{n}_v \in \mathbb{R}^M$, where M is the number of node-types (including "background"), and ith entry of $\mathbf{n}_v$ is the probability that node v has type i.

Constructing a Complete PGN

Given an image dataset with ground-truth annotations, we first train an object detector. For this purpose, we use the Tensorflow Object Detection API. We use `faster_rcnn_nas` trained on MS-COCO dataset as the pretrained model. Given an image, the output of the object detector is a set of object bounding-boxes with their objectness scores, classification scores, bounding-box coordinates, and feature vectors. We denote the feature vector of bounding-box v by $\mathbf{h}^\circ_v$. Each bounding-box v is specified by its coordinates $\mathtt{B}(v) = (v_x, v_y, v_{x'}, v_{y'})$ and a confidence score $p(v)$, where (v_x, v_y) and $(v_{x'}, v_{y'})$ are the top-left and bottom-right corners of the bounding-box. We use $N = 256$ bounding-boxes per image with the highest objectness scores. We also can extract a feature vector for each edge $e = (u, v)$ denoted by $\mathbf{h}^\circ_{u,v}$ from the union of bounding-boxes u, and v. For this purpose, we first feed the image to 152-layer ResNet, pretrained on ImageNet, and obtain 1024 feature maps from the last 14×14 pooling layer. Then, we apply a Region of Interest pooling layer (Girshick, 2015), based on the coordinates of each bounding-box, to get 1024 features maps of size 7×7. Next, we use two fully-connected layers with ReLU activation to get a 512-d feature vector denoted by $\mathbf{h}^\circ_{u,v}$.

We may obtain a node-type PMF $\mathbf{n}^\circ_v$ for each candidate bounding-box v based on the classification scores of v. However, to incorporate global context, we sort the bounding-boxes based on their confidence score, and apply a bidirectional LSTM in an Encoder-Decoder architecture (similar to Zellers et al. (2018)) to obtain a node-type PMF $\mathbf{n}^\circ_v$ for each node, and an edge-type PMF $\mathbf{e}^\circ_{u,v}$ for each pair of the nodes. The Encoder-Decoder architecture is explained in the following. Note that the order of the bounding-boxes are not crucial here, since the $\mathbf{e}^\circ_{u,v}$, $\mathbf{n}^\circ_v$ are just initial values and they will be updated later during the execution of the DG-PGNN algorithm.

We obtain a node-type PMF $\mathbf{n}^\circ_v$ for each candidate bounding-box v as follows. We first sort the bounding-boxes based on their confidence score, and apply a bidirectional LSTM as

$$\mathbf{f}_k = \mathtt{BiLSTM}([\mathbf{h}^\circ_k, W^{(1)}\hat{\mathbf{n}}^\circ_k], \mathbf{f}_{k-1}) \tag{5.1}$$

where $\mathbf{f}_0 = \mathbf{0}$, $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times M}$ is a trainable matrix, and $\hat{\mathbf{n}}^\circ_k \in \mathbb{R}^M$ is the node-type PMF for candidate bounding-box v which is obtained base on the classification scores of bounding-box v. Then, we use an LSTM to decode a node-type distribution $\mathbf{n}^\circ_k$ as

$$\hat{\mathbf{f}}_k = \mathtt{LSTM}([\mathbf{f}_k, \mathbf{n}^\circ_{k-1}], \hat{\mathbf{f}}_{k-1}) \tag{5.2}$$

where $\hat{\mathbf{f}}_0 = \mathbf{0}$, $\mathbf{n}_k^\circ = \texttt{softmax}(\mathbf{W}^{(2)}\hat{\mathbf{f}}_k)$, and $\mathbf{W}^{(2)} \in \mathbb{R}^{M \times d'}$ is a trainable matrix. Similarly, to get a contextualized representation for edges, we use another bidirectional LSTM as

$$\mathbf{f}_k^\dagger = \texttt{BiLSTM}([\mathbf{f}_k, \mathbf{W}^{(3)}\mathbf{n}_k], \mathbf{f}_{k-1}^\dagger) \tag{5.3}$$

where $\mathbf{f}_0^\dagger = \mathbf{0}$, and $\mathbf{W}^{(3)} \in \mathbb{R}^{d'' \times M}$ is a trainable matrix. Then, we extract a contextualized feature vector from edge (u, v) as

$$\mathbf{e}_{u,v}^\dagger = \mathbf{W}^{(4)}\mathbf{f}_u^\dagger \odot \mathbf{h}_{u,v}^\circ + \mathbf{W}^{(5)}\mathbf{f}_v^\dagger \odot \mathbf{h}_{u,v}^\circ \tag{5.4}$$

where $\mathbf{W}^{(4)}, \mathbf{W}^{(5)} \in \mathbb{R}^{d^\circ \times d'^\circ}$. Next, we use a sigmoid layer to decode an edge-type distribution $\mathbf{e}_{u,v}^\circ$ as

$$\mathbf{e}_{u,v}^\circ = \sigma(\mathbf{W}^{(6)}\mathbf{e}_{u,v}^\dagger + \mathbf{b}^{(1)}) \tag{5.5}$$

where $\mathbf{W}^{(6)} \in \mathbb{R}^{K \times d^\circ}$, and $\mathbf{b}^{(1)}$ is a trainable bias specific to the decoded object classes of u and v. The d, d', d'', d°, and d'° are set to 512. We train the model by minimizing the sum of the cross entropy for edge-types and cross entropy for node-types. The output of this section is an initial node-type distribution $\mathbf{n}_v^\circ$ for each node v, and an initial edge-type distribution $\mathbf{e}_{u,v}^\circ$ for each edge $e = (u, v)$.

Caption Encoding

A region-grounded captioning model from `https://github.com/jcjohnson/densecap` is used to extract a set of descriptions for the input image, e.g. *a cloudy sky*, *woman holding umbrella*. Each caption c has a bounding-box $\texttt{B}(c)$ and a confidence score. We map the one hot representation of each word to a semantic space of dimensionality D via a $D \times n$ word embedding matrix, where n is the size of a dictionary which is created using all training captions (and questions for VQA task). To initialize the word embeddings, we apply skip-gram training to all questions and captions, and concatenate the skip-gram vectors with the pretrained GloVe vectors. The last hidden state of a GRU is used to encode a caption. We reset the GRU after presenting each caption.

Deep Q-Learning for Selecting the Nodes

Starting from a null graph, we sequentially add a new node to the current graph in a Deep Q-learning framework, where a state is a PGN, each action chooses a new node, and rewards are defined based on the ground-truth graph of the input image. We use two fully-connected layers with a ReLU activation function to implement the Q-network. Let the current state be a partial PGN denoted by s. The input to the Q-network is defined as

$$\phi = \phi(s) = [\mathbf{g}, \mathbf{p}, \mathbf{d}, \mathbf{h}_1, \ldots, \mathbf{h}_N, \mathbf{n}_1, \ldots, \mathbf{n}_N, \mathbf{o}^{\text{n}}, \mathbf{o}^{\text{e}}] \tag{5.6}$$

where $\mathbf{g}$ is a global feature vector extracted from the last layer of 152-layer ResNet, $\mathbf{o}^{\mathrm{n}}$ and $\mathbf{o}^{\mathrm{e}}$ are graph-level representations of the current PGN initialized to $\mathbf{0}$, $\mathbf{p} \in \mathbb{R}^N$ is a confidence score vector defined as $\mathbf{p} = [p(1), \ldots, p(N)]$, and vector $\mathbf{d} \in \mathbb{R}^N$ is defined as: $\mathbf{d}(v) = 1$ if v is selected before, otherwise $\mathbf{d}(v) = 0$. The feature vector ϕ provides a history about the nodes, node-types, and edge-types which were selected before. Thus, the Q-network can exploit information about co-occurrence of the objects for selecting the next node. For example, if the current graph has a node with type *street*, it is likely that the image contains object *car*; so the Q-network is more likely to select a bounding-box with a high node-type probability for *car*.

At each step, we select an action from the set $\mathcal{A} = \{v : \mathbf{d}(v) = 0\} \cup \{\texttt{STOP}\}$, where `STOP` is a special action that indicates the end of graph construction. The reward function for taking action a at state s is defined as: $r(a, s) = 1$ if there exists a ground-truth box v such that $\texttt{IoU}(\texttt{B}(a), \texttt{B}(v)) \geq 0.5$, otherwise $r(a, s) = -1$, where `IoU` stands for Intersection over Union. After each 3000 steps, the Q-network trainable parameters $\boldsymbol{\theta}$ are copied to $\hat{\boldsymbol{\theta}}$ which are used to compute the target Q-values. This helps stabilize the optimization. To reduce correlation between samples and keep experiences from the past episodes, we use an experience replay technique (Mnih et al., 2013, 2015). To update the parameters of the Q-networks, we choose a random minibatch from the replay memory. Let $(\hat{\phi}, \hat{s}, \hat{a}, \hat{r}, \hat{\phi}', \hat{s}')$ be a random transition sample. The target Q-value for the network is obtained as

$$\hat{y} = \hat{r} + \gamma \max_{v \in \hat{\mathcal{A}}'} Q\big(\phi(\hat{s}' + v); \hat{\boldsymbol{\theta}}\big) \tag{5.7}$$

where $\hat{\mathcal{A}}'$ is a set of actions that can be taken in state $\hat{s}'$, and $\hat{s}' + v$ is obtained by adding node v to $\hat{s}'$. Finally, the parameters of the model are updated as follows:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha\big(\hat{y} - Q(\phi(\hat{s} + \hat{a}); \boldsymbol{\theta})\big) \nabla_{\boldsymbol{\theta}} Q\big(\phi(\hat{s} + \hat{a}); \boldsymbol{\theta}\big) \tag{5.8}$$

To train the model, we use *ϵ-greedy learning*, that is, with probability ϵ a random action is selected, and with probability $1 - \epsilon$ an optimal action is selected, as indicated by the Q-networks. For test images, we construct the graph by sequentially selecting the optimal actions (bounding-boxes) based on the highest Q-values until it selects the `STOP`.

Node Representation Update

We compute a new node representation $\mathbf{h}'_v$ for each node v, every time that we extend the graph. This allows to exploit information about co-occurrence of objects. For example, if the current graph has a node with type *building*, it is likely that the graph contains *window*. For each node v in the current PGN, an ideal node representation must take into account the information from every node which has an edge to v based on the strength of the edge.

The update equations are as follows

$$\mathbf{a}_v = \sum_{k=1}^{K} \sum_{u \in \mathcal{V}} E_k(u, v) \mathbf{h}_{u,v} \tag{5.9}$$

$$\mathbf{h}'_v = \texttt{GRU}(\mathbf{a}_v, \mathbf{h}_v) \tag{5.10}$$

where $\mathcal{V}$ is the set of the nodes in the current PGN. Intuitively, $\mathbf{a}_v$ is the aggregation of *massages* from all nodes to v weighted by the strength of their outgoing edge to v. The update mechanism first computes the node activations $\mathbf{a}_v$ for each node v. Then, a (one-step) GRU is used to update node representation for each node by incorporating information from the previous node representation $\mathbf{h}_v$. Note that unlike Li et al. (2017); Xu et al. (2017), our message propagation mechanism is based on strength of the edges, not the predefined structure of the graph, since E is probabilistic. Also, our message passing scheme is part of a larger RL-based architecture. Thus, unlike these work which use all candidate boxes, in our DG-PGNN the massages propagate through the partial graph that has been constructed by DG-PGNN, not the complete graph.

Edge Representation Update

After updating the node representations, we compute a new edge representation $\mathbf{h}'_{u,v}$ for each edge $e = (u, v)$. This allows to exploit contextual relationship information. For example, if the type of node u is *man*, and the type of node v is *horse*, it is likely that the edge-type of $e = (u, v)$ be *riding*. We use a (one-step) GRU to update the edge representations as

$$\mathbf{h}'_{u,v} = \texttt{GRU}([\mathbf{h}_u, \mathbf{h}_v], \mathbf{h}_{u,v}) \tag{5.11}$$

Caption-Guided Updates

If a word of a caption referrers to a node-type, then it is useful to update the node representation of a node which is located at the region of the caption based on the encoded caption. For example, for region-grounded caption *a yellow umbrella*, it is useful to update the node representation of a node that is located at the region of the caption, such that the new representation increases the degree of the belief that the node-type of the node is *umbrella*; or *man is riding a horse* increases the chance that an edge that is located at the region of the caption has type *riding*. Given a new node v, if there is a caption c such that $\texttt{IoU}(\texttt{B}(c), \texttt{B}(v)) \geq 0.5$, we update the node representation of v as $\mathbf{h}'_v = \texttt{GRU}([\mathbf{c}, \mathbf{n}_v], \mathbf{h}_v)$, where $\mathbf{c}$ is the encoded caption. Similarly, if there is a caption c and a node u such that $\texttt{IoU}(\texttt{U}(u, v), \texttt{B}(c)) \geq 0.5$, where $\texttt{U}$ stands for union of two boxes, we update the edge representation of (u, v) and (v, u) as

$$\mathbf{h}'_{u,v} = \texttt{GRU}([\mathbf{c}, \mathbf{e}_{u,v}], \mathbf{h}_{u,v}) \tag{5.12}$$

$$\mathbf{h}'_{v,u} = \texttt{GRU}([\mathbf{c}, \mathbf{e}_{v,u}], \mathbf{h}_{v,u}) \tag{5.13}$$

Graph-Level Representation

We obtain a graph-level representation $\mathbf{o}^{\mathrm{n}}$ using the node representations as

$$\mathbf{o}^{\mathrm{n}} = \psi\big(\textstyle\sum_v \sigma\big(f^{\mathrm{n}}(\mathbf{n}_v, \mathbf{h}_v)\big) \odot \psi\big(g^{\mathrm{n}}(\mathbf{n}_v, \mathbf{h}_v)\big)\big) \tag{5.14}$$

where f^{n} and g^{n} are two neural networks, ψ is the hyperbolic tangent function, and $\sigma\big(f(\mathbf{n}_v, \mathbf{h}_v)\big)$ is an attention weight vector that decides which nodes are related to the current graph-level representation. Similarly, we obtain a graph-level representation $\mathbf{o}^{\mathrm{e}}$ using edge representations as

$$\mathbf{o}^{\mathrm{e}} = \psi\big(\textstyle\sum_{u,v} \sigma\big(f^{\mathrm{e}}(\mathbf{e}_{u,v}, \mathbf{h}_{u,v})\big) \odot \psi\big(g^{\mathrm{e}}(\mathbf{e}_{u,v}, \mathbf{h}_{u,v})\big)\big) \tag{5.15}$$

These two representations provide a summary of the current graph for the Q-network.

Node-Type Probability Update

To compute a new node-type PMF for node v, we use a GRU and a softmax layer

$$\mathbf{n}'_v = \texttt{softmax}\big(\texttt{GRU}([\mathbf{h}_v, \mathbf{o}^{\mathrm{n}}, \mathbf{o}^{\mathrm{e}}], \mathbf{n}_v)\big) \tag{5.16}$$

where the softmax function guarantees that sum of the probability of all node-types for node v, is 1.

Edge-Type Matrix Update

We update the edge-type probability vectors as follows

$$\mathbf{r}_{u,v} = \sigma\big(\mathbf{W}^{(1)}\psi\big(\mathbf{W}^{(2)}\mathbf{x} + \mathbf{b}^{(1)}\big) + \mathbf{b}^{(2)}\big) \tag{5.17}$$

$$\mathbf{z}_{u,v} = \sigma\big(\mathbf{W}^{(3)}\psi\big(\mathbf{W}^{(4)}\mathbf{x} + \mathbf{b}^{(3)}\big) + \mathbf{b}^{(4)}\big) \tag{5.18}$$

$$\mathbf{e}'_{u,v} = \mathbf{e}_{u,v} \odot (1 - \mathbf{r}_{u,v}) + (1 - \mathbf{e}_{u,v}) \odot \mathbf{z}_{u,v} \tag{5.19}$$

where $\mathbf{x} = [\mathbf{h}_{u,v}, \mathbf{n}_v, \mathbf{h}_v, \mathbf{n}_u, \mathbf{h}_u]$, and the weight matrices and biases are trainable parameters. This update grantees that the updated probabilities are between 0 and 1.

Parameter Update

We update the parameters of the proposed model after updating the node-type PMFs and the edge-type matrices. For these updates, we minimize the following loss functions

$$\mathcal{L}^{\mathrm{n}} = -\textstyle\sum_v \mathbf{n}^{\star}_v \cdot \log(\mathbf{n}'_v) + (1 - \mathbf{n}^{\star}_v) \cdot \log(1 - \mathbf{n}'_v) \tag{5.20}$$

$$\mathcal{L}^{\mathrm{e}} = -\textstyle\sum_{u,v} \mathbf{e}^{\star}_{u,v} \cdot \log(\mathbf{e}'_{u,v}) + (1 - \mathbf{e}^{\star}_{u,v}) \cdot \log(1 - \mathbf{e}'_{u,v}) \tag{5.21}$$

where $\mathbf{n}_v^\star$ and $\mathbf{e}_{u,v}^\star$ denote the ground-truth for PMF of node v, and edge-type probabilities of edge $e = (u, v)$. That is, $\mathbf{n}_v^\star(m) = 1$ if there is a ground-truth bounding-box u of node-type m such that $\texttt{IoU}(u, v) \geq 0.5$, otherwise $\mathbf{n}_v^\star(m) = 0$. Similarly, $\mathbf{e}_{u,v}^\star(k) = 1$ if there is a ground-truth edge $e = (u', v')$ of type k such that $\texttt{IoU}(\texttt{U}(u, v), \texttt{U}(u', v')) \geq 0.5$, otherwise $\mathbf{e}_{u,v}^\star(k) = 0$.

DG-PGNN for VQA

We use the last hidden state of a GRU to obtain an encoded question denoted by $\mathbf{q}$. Then, we obtain question-guided graph-level representations as

$$\mathbf{o}_{\mathrm{q}}^{\mathrm{n}} = \psi\big(\textstyle\sum_v \sigma(\mathbf{h}_v \star \mathbf{q}) \odot \psi(\mathbf{h}_v \star \mathbf{q})\big) \tag{5.22}$$

$$\mathbf{o}_{\mathrm{q}}^{\mathrm{e}} = \psi\big(\textstyle\sum_{u,v} \sigma(\mathbf{h}_{u,v} \star \mathbf{q}) \odot \psi(\mathbf{h}_{u,v} \star \mathbf{q})\big) \tag{5.23}$$

where we used MCB to implement $\star$. This allows to incorporate information from the question for computing the attention weights for each node and edge using the sigmoid function. The candidate answers are also encoded by the last hidden state of a GRU and concatenated with $\mathbf{o}_{\mathrm{q}}^{\mathrm{n}}$ and $\mathbf{o}_{\mathrm{q}}^{\mathrm{e}}$ using a neural network layer as $\hat{p} = \sigma(\mathbf{w} f([\mathbf{o}_{\mathrm{q}}^{\mathrm{n}}, \mathbf{o}_{\mathrm{q}}^{\mathrm{e}}, \mathbf{a}]) + b)$ where $\mathbf{a}$ is the encoded answer choice, f is a ReLU non-linear layer, and $\mathbf{w}, b$ are trainable parameters. The binary logistic loss $-p \log(\hat{p}) - (1 - p) \log(1 - \hat{p})$ is used, where p is 1.0 for an image-question-answer triplet, if the answer choice is correct, otherwise p is 0.

Training Details and Optimization

The discount factor γ is set to 0.85 and ϵ is annealed from 1 to 0.05 during the first 50 epochs, and is fixed after epoch 50. For VQA, the loss is minimized using RMSprop optimization algorithm with learning rate 0.0001 and minibatches of size 100. To prevent overfitting, dropout with probability 0.5 and early stopping are applied. D is set to 512. For each image, we use up to 10 captions with a confidence score greater than 1.0. During training, all parameters are tuned except for the weights of the CNN and caption generation component to avoid overfitting. We use default values for all hyper-parameters of the object detector API. Our model takes around two days to train on two NVIDIA Titan X GPUs.

Algorithm 2: Deep Generative Graph Neural Networks for Scene Graph Generation

Input : A set of image captions $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$
Input : A set of N bounding-boxes
Input : A global image feature vector $\mathbf{g}$
Output : Updates the parameters of the models
for $u \leftarrow 1$ **to** N **do**
  $\mathbf{d}(u) = 0$
end
Let s be an empty graph
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{A} \leftarrow \{v : \mathbf{d}(v) = 0\} \cup \{\texttt{STOP}\}$
  Generate a random number $z \in (0, 1)$
  if $z < \epsilon$ **then**
    Select a random node a from $\mathcal{A}$
  else
    $a \leftarrow \arg\max_{v \in \mathcal{A}} Q\big(\phi(s + v); \boldsymbol{\theta}\big)$
  end
  if a is not `STOP` **then**
  Compute reward r, $s' \leftarrow s + a$, $\mathbf{d}(a) \leftarrow 1$
  else break
  $\phi' \leftarrow [\mathbf{g}, \mathbf{p}, \mathbf{d}, \mathbf{h}_1, \ldots, \mathbf{h}_N, \mathbf{n}_1, \ldots, \mathbf{n}_N, \mathbf{o}^{\mathrm{n}}, \mathbf{o}^{\mathrm{e}}]$
  Store transition $(\phi, s, a, r, \phi', s')$
  Sample minibatch of transitions $(\hat{\phi}, \hat{s}, \hat{a}, \hat{r}, \hat{\phi}', \hat{s}')$
  Compute target Q-value $\hat{y}$ ▷ *Eq. 5.7*
  Update the parameters of the Q-network ▷ *Eq. 5.8*
  $s \leftarrow s'$, Let s be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  for $u \leftarrow 1$ **to** $|\mathcal{V}|$ **do**
    Update $\mathbf{h}_v$ ▷ *Eq. 5.10*
  end
  for $e = (u, v) \in \mathcal{V} \times \mathcal{V}$ **do**
    Update $\mathbf{h}_{u,v}$ ▷ *Eq. 5.11*
  end
  for $j \leftarrow 1$ **to** $|\mathcal{C}|$ **do**
    Obtain $\mathbf{c}$ by encoding caption c_j
    if $\texttt{IoU}(\texttt{B}(c_j), \texttt{B}(a)) \geq 0.5$ **then**
      update $\mathbf{h}_a$
    end
    for $u \leftarrow 1$ **to** $|\mathcal{V}|$ **do**
      if $\texttt{IoU}(\texttt{U}(u, a), \texttt{B}(c_j)) \geq 0.5$ **then**
        update $\mathbf{h}_{u,a}$ and $\mathbf{h}_{a,u}$ ▷ *Eq. 5.12, 5.13*
      end
    end
  end
  Obtain graph-level representations $\mathbf{o}^{\mathrm{n}}$, $\mathbf{o}^{\mathrm{e}}$ ▷ *Eq. 5.14*
  for $u \leftarrow 1$ **to** $|\mathcal{V}|$ **do**
    Update $\mathbf{n}_v$ ▷ *Eq. 5.16*
  end
  for $e = (u, v) \in \mathcal{V} \times \mathcal{V}$ **do**
    Update $\mathbf{e}_{u,v}$ ▷ *Eq. 5.19*
  end
  Update parameters of the PGN model ▷ *Eq. 5.20, 5.21*
end

5.1.3 Experiments

For each task, we introduce the datasets, baseline models and evaluation metric that we use in our experiments. Then, the results are presented and discussed.

Scene Graph Generation Task

Dataset. The Visual Genome (VG) dataset (Krishna et al., 2016) contains 108,077 images. We used Visual Genome version 1.4 release. This release contains cleaner object annotations (Xu et al., 2017). Annotations provide subject-predicate-object triplets. A triplet means that there is an edge between the subject and the object and the type of the edge is indicated by the predictate. Following Xu et al. (2017), we use the most frequent 150 object categories and 50 predicates for scene graph prediction task. This results in a scene graph of about 11.5 objects and 6.2 relationships per image. The training and test splits contains 70% and 30% of the images, respectively.

Metrics. Top-K recall (`Rec@K`) is used as the metric, which is the fraction of the ground truth relationship triplets subject-predicate-object hit in the top-K predictions in an image. Predictions are ranked by the product of the node-type probability of the subject, the node-type probability of the object, and the edge-type probability of the predicate. Following Xu et al. (2017), we evaluate our model based on three tasks as follows: i) Predicate classification (PRED-CLS) task: to predict the predicates of all pairwise relationships of a set of objects, where the location and object categories are given. ii) Scene graph classification (SG-CLS) task: to predict the predicate and the object categories of the subject and object in all pairwise relationships, where the location of the objects are given. iii) Scene graph generation (SG-GEN) task: to detect a set of object bounding-boxes and predict the predicate between each pair of the objects, at the same time. An object is considered to be properly detected, if it has at least 0.5 Intersection over Union overlap with a bounding-box annotation with the same category.

Baseline Models. We compare our model with several baseline models including the state-of-the-art models (Tang et al., 2019; Zellers et al., 2018; Newell and Deng, 2017). Lu et al. (2016a) uses language priors from semantic word embeddings, while Xu et al. (2017) uses an RNN and learns to improves its predictions iteratively through message passing across the scene graph. After a few steps the learned features are classified. Moreover, three ablation models help evaluate the impact of each component of DG-PGNN: i) DG-PGNN$^\circ$ is the initial complete PGN without applying the DG-PGNN ($e = (u, v)$ is pruned if $\mathbf{e}^\circ_{u,v}$ is less than 0.5, and v is deleted if $\mathbf{n}^\circ_v$ is less than 0.5), ii) DG-PGNN$^-$ is the same as DG-PGNN except that it does not use captions. iii) DG-PGNN† is the same as the full model (DG-PGNN), but it uses the VGG features instead of ResNet.

Results and Discussion. Our experimental results are reported in Table 5.1. The results show that DG-PGNN outperforms the state-of-the art models for scene graph gen-

eration task. Both DG-PGNN and Xu et al. (2017) leverage the power of RNNs to learn a feature vector for each bounding-box. DG-PGNN incorporates captions and contextual information of the image via information propagation to construct precise scene graphs. However, Xu et al. (2017) suffers from imbalanced classification problem (often there is no edge between many pairs of objects). DG-PGNN† results (comparing to DG-PGNN) show the effect of backbone CNN (VGG versus ResNet) is insignificant. This is because our algorithm uses strong contextual information to compensate for the detection errors caused by applying the VGG. Figure 5.1 and Figure 5.4 illustrate some scene graphs generated by our model. The DG-PGNN predicts a rich semantic representation of the given image by recognizing objects, their locations, and relationships between them. For example, DG-PGNN can correctly detect spatial relationships (*cube behind sphere*) and interactions (*man using camera*, *man has shirt*). DG-PGNN can correctly recognize *woman* (instead of *girl*). More examples are provided in supplementary materials.

	PRED-CLS		SG-CLS		SG-GEN	
Model	R@50	R@100	R@50	R@100	R@50	R@100
SG-LP (Lu et al., 2016a)	27.88	35.04	11.79	14.11	00.32	00.47
SG-IMP (Xu et al., 2017)	44.75	53.08	21.72	24.38	03.44	04.24
D-GGNN	46.9	55.6	23.8	26.8	06.4	07.5
MSDN (Li et al., 2017)	63.12	66.41	19.30	21.82	7.73	10.51
GRCNN (Yang et al., 2018a)	54.2	59.1	29.6	31.6	11.4	13.7
PIX2GR (Newell and Deng, 2017)	68.0	**75.2**	26.5	30.0	9.7	11.3
MOTIF (Zellers et al., 2018)	65.2	67.1	35.8	36.5	27.2	30.3
VCTREE (Tang et al., 2019)	66.4	68.1	38.1	38.8	27.9	31.3
DG-PGNN$^{\circ}$	63.5	65.3	34.2	34.8	26.1	28.8
DG-PGNN^{-}	67.3	70.1	38.6	39.2	30.2	31.8
DG-PGNN†	69.0	72.1	39.3	40.1	31.2	32.5
DG-PGNN	**70.1**	73.0	**39.5**	**40.8**	**32.1**	**33.1**

Table 5.1: Recall for predicate classification, scene graph classification, and scene graph generation tasks on VG dataset.

VQA Task

Datasets. Visual7W (Zhu et al., 2015) includes 47,300 images which occurs in both Visual Genome and MS-COCO datasets. The training, validation and test splits, contains 50%, 20%, 30% of the QA pairs, respectively. We train and evaluate our model on *telling* questions of the Visual7W. This set uses six types of questions: *what*, *where*, *when*, *who*, *why*, *how*. For evaluation, Visual7W provides four candidate answers. We also evaluated our model on the test set of CLEVR dataset (Johnson et al., 2017a) (`https://cs.stanford.edu/people/jcjohns/clevr/`). CLEVR evaluates different aspects of visual reasoning such as attribute recognition, counting, comparison, logic, and spatial relationships. Each object in an image has the following attributes: shape (*cube*, *sphere*, or *cylinder*), size (*large* or *small*), color (8 colors), and material (*rubber* or *metal*). An object detector with 96 classes is trained using all combinations of the attributes. The predicates are *left*, *right*, *in front*, *behind*, *same size*,

same color, *same shape*, and *same material*. We created a scene graph for each training image using the provided annotations. We concatenate the normalized coordinates of each bounding-box with the features vector that we extract from Region of Interest pooling layer. This helps to detect spatial relationships between objects.

Baseline Models. For Visual7W, we compare our models with Zhu et al. (2015), MCB (Fukui et al., 2016), MAN (Ma et al., 2018), and MLP (Jabri et al., 2016). MCB leverages the Visual Genome QA pairs as additional training data and the 152-layer ResNet as a pretrained model. This model took the first place in the VQA Challenge workshop 2016. MAN utilized a memory-augmented neural network which attend to each training exemplar to answer visual questions, even when the answers happen infrequently in the training set. The MLP method uses image-question-answer triplets to score answer choices. For CLEVR, we compare our DG-PGNN$^-$ with several baselines proposed by Johnson et al. (2017a) as well as state-of the art models PROGRAM-GEN (Johnson et al., 2017b) and CNN+LSTM+RN (Santoro et al., 2017). N2NMN (Hu et al., 2017) learns to predict a layout based on the question and compose a network using a set of neural modules. CNN+LSTM+RN learns to infer a relation using a neural network model called Relation Networks. PROGRAM-GEN exploits supervision from functional programming which is used to generate CLEVR questions.

Results. Tables 5.2 and 5.3 reports our experimental results on Visual7W and CLEVR datasets, respectively. For Visual7W, the results show that our algorithm outperforms MCB, MAN, and MLP for VQA task. This is because the scene graph provides a rich intermediate representation to answer questions involving objects and their relationships. For CLEVR, our algorithm achieves comparable results with the state of the art. We emphasize that, unlike PROGRAM-GEN, our algorithm does not exploit supervision from functional programming.

Model	What	Where	When	Who	Why	How	Avg
HUMAN (Zhu et al., 2015)	96.5	95.7	94.4	96.5	92.7	94.2	95.7
LSTM-ATT (Zhu et al., 2015)	51.5	57.0	75.0	59.5	55.5	49.8	54.3
CONCAT+ATT (Fukui et al., 2016)	47.8	56.9	74.1	62.3	52.7	51.2	52.8
MCB+ATT (Fukui et al., 2016)	60.3	70.4	79.5	69.2	58.2	51.1	62.2
MAN (Ma et al., 2018)	62.2	68.9	76.8	66.4	57.8	52.9	62.8
MLP (Jabri et al., 2016)	64.5	75.9	82.1	72.9	68.0	56.4	67.1
DG-PGNN$^-$	65.4	77.1	83.0	74.1	68.8	57.3	68.0
DG-PGNN	**66.8**	**78.3**	**84.4**	**75.6**	**70.0**	**58.2**	**69.3**

Table 5.2: Accuracy Percentage on Visual7W.

5.1.4 Examples for VQA and Scene Graph Generation Tasks

Figure 5.3 shows two examples of scene graphs generated by DG-PGNN on Visual Genome dataset. Figure 5.4 shows two examples of VQA by DG-PGNN† on CLEVR dataset. Fig-

Model	All	Exist	Count	Cmp-Int	Q-At	Cmp-At
HUMAN (Johnson et al., 2017a)	92.6	96.6	86.7	86.5	95.0	96.0
Q-TYPE MODE (Johnson et al., 2017a)	41.8	50.2	34.6	51.0	36.0	51.3
LSTM (Johnson et al., 2017a)	46.8	61.1	41.7	69.8	36.8	51.3
CNN+BOW (Johnson et al., 2017a)	48.4	59.5	38.9	51.1	48.3	51.8
CNN+LSTM (Johnson et al., 2017a)	52.3	65.2	43.7	67.1	49.3	53.0
CNN+LSTM+MCB (Johnson et al., 2017a)	51.4	63.4	42.1	66.4	49.0	51.0
CNN+LSTM+SA (Johnson et al., 2017a)	68.5	71.1	52.2	73.5	85.3	52.3
N2NMN (Hu et al., 2017)	83.3	85.7	68.5	85.0	90.0	88.8
CNN+LSTM+RN (Santoro et al., 2017)	95.5	97.8	90.1	93.6	97.9	97.1
PROGRAM-GEN (Johnson et al., 2017b)	**96.9**	97.1	**92.7**	**98.7**	**98.2**	**98.9**
DG-PGNN$^-$	96.1	**98.0**	91.2	94.2	98.1	97.8

Table 5.3: Accuracy Percentage on CLEVR.

ures 5.5 shows examples of VQA by DG-PGNN on Visual7W dataset. For each question, the nodes and edges with the highest attention weight are specified.

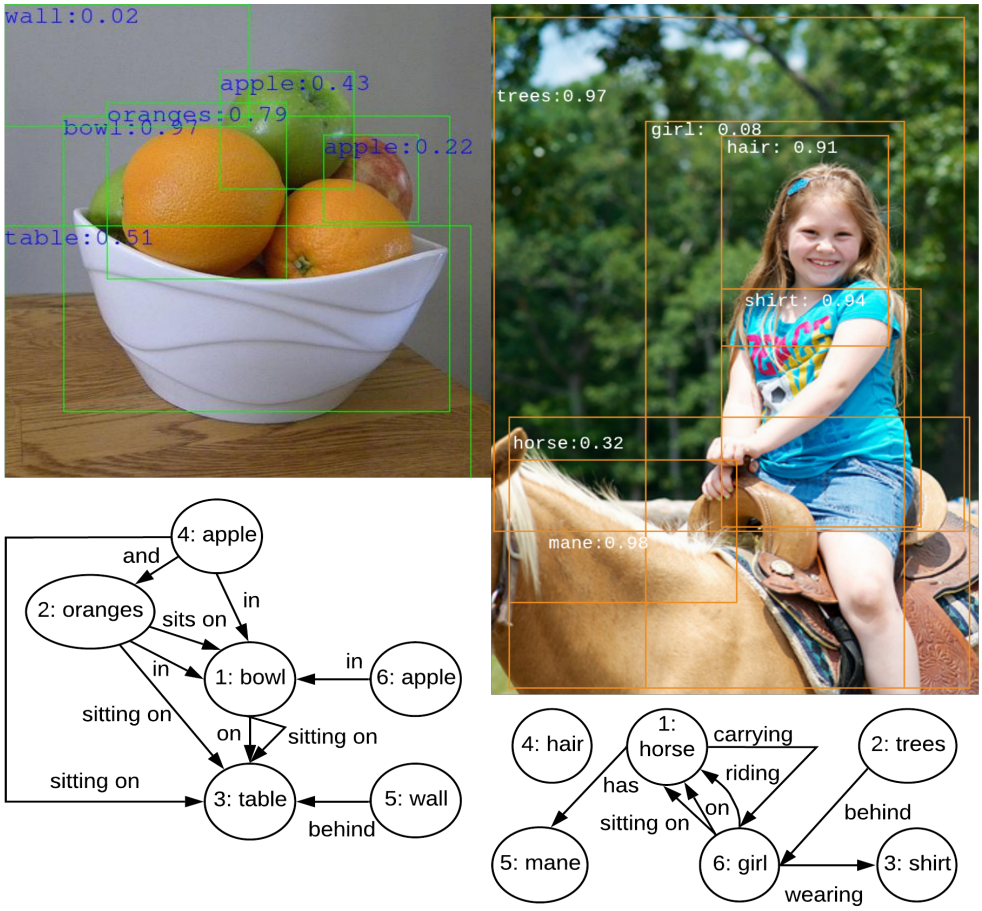


Figure 5.3: Examples of scene graphs generated on Visual Genome dataset.

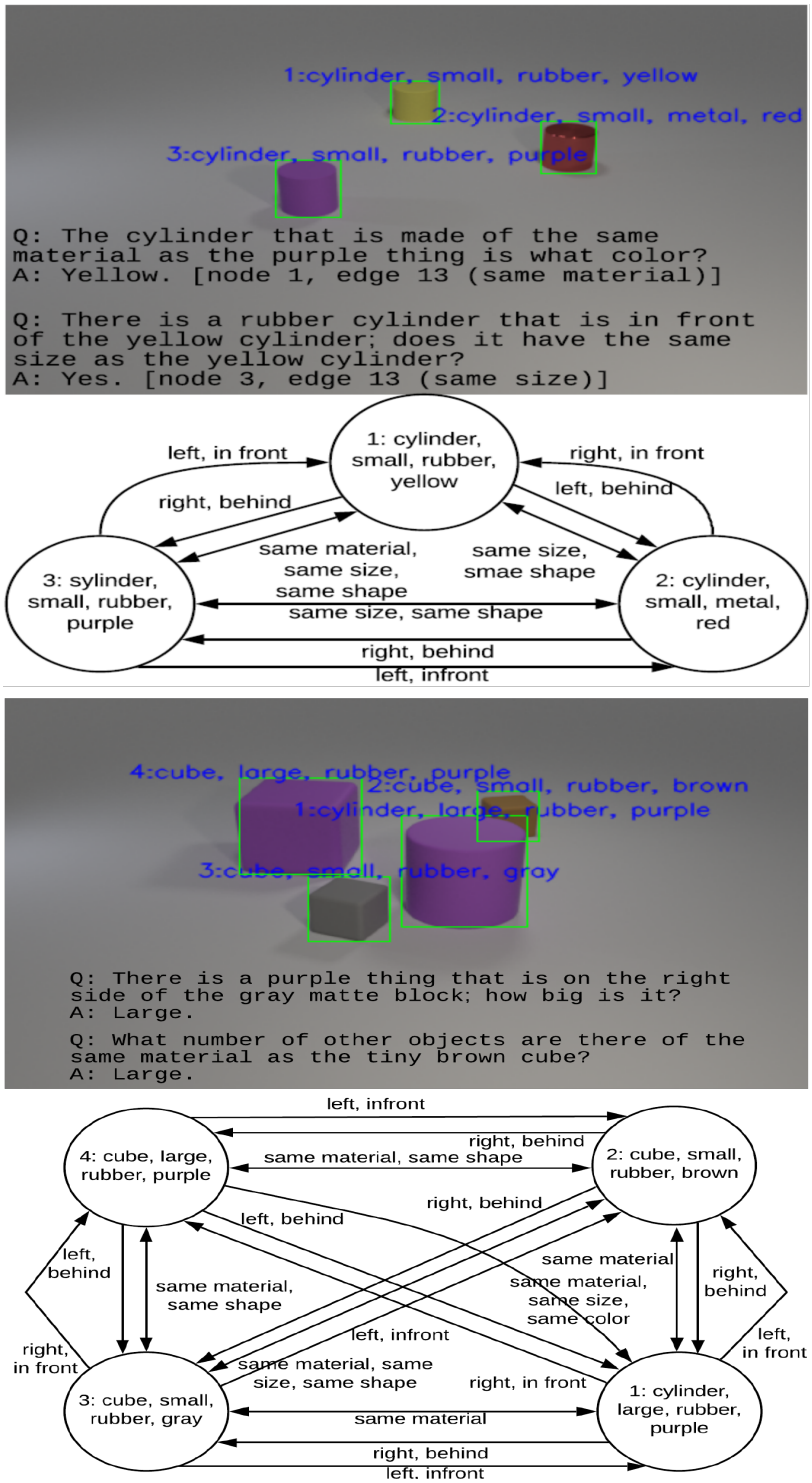


Figure 5.4: Examples of VQA by DG-PGNN† on CLEVR dataset. For each question, the nodes and edges with the highest attention weight are specified.

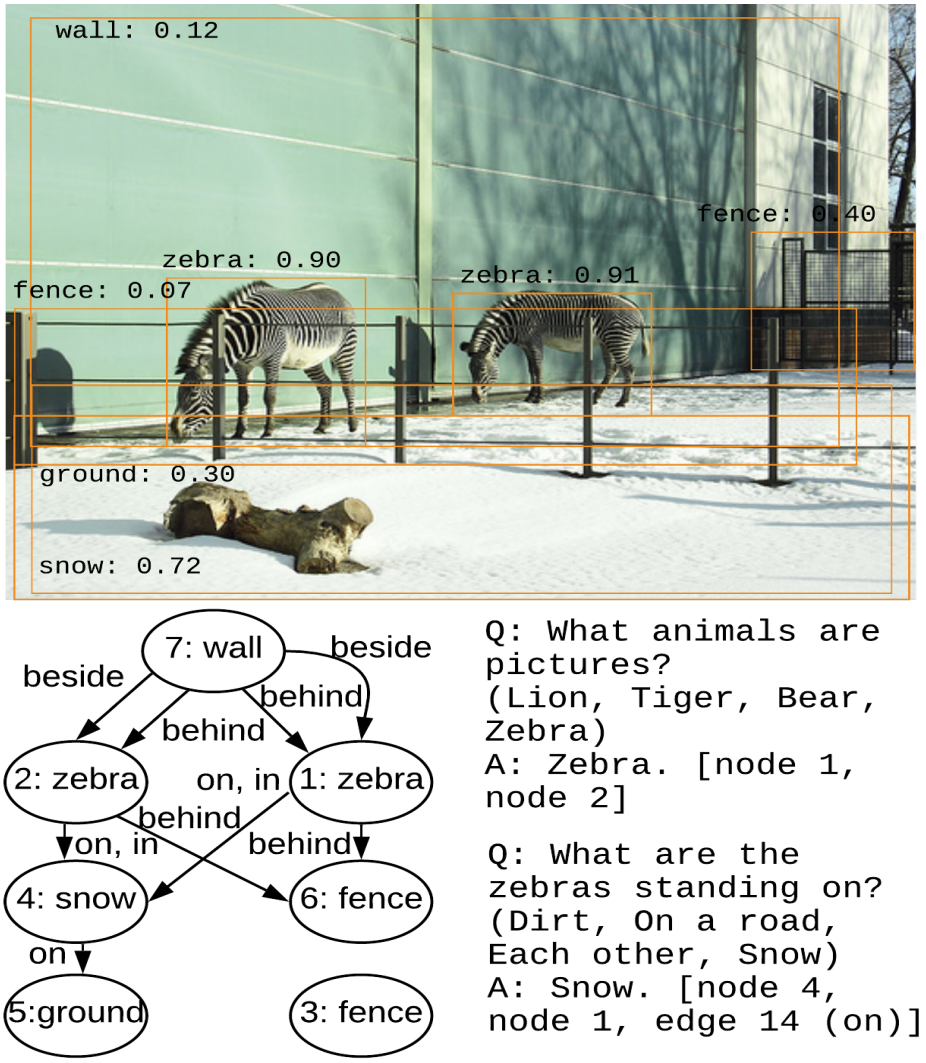


Figure 5.5: An example of scene graph generation and VQA by DG-PGNN on Visual7W dataset. For each question, the nodes and edges with the highest attention weight are specified. DG-PGNN can correctly detect spatial relationships *wall beside zebra* and *zebra behind fence*. Also, DG-PGNN can correctly recognize *zebra in snow* (instead of *zebra in grass*).

5.2 Proposed Algorithm for Dynamic Gated Graph Neural Networks

An effective scene graph generation model needs to consider visual contextual information as well as domain priors. For example, if we know that a node has an incoming edge from another node which represents a *man* and the type of the edge is *riding*, then the type of the node is likely *horse* or *bicycle*. Clues from the relational context can be represented by leveraging models such as the recently proposed Gated Graph Neural Networks (GGNN) (Li et al., 2015), which can learn a latent feature vector (embedding, representation) for each node from graph-structured inputs. However, to apply the GGNN model, the structure of the input digraph and the type of each edge must be known, whereas the structure and edge-types must be *inferred* in the scene graph generation task. In this work, we propose a new deep generative architecture, called Dynamic Gated Graph Neural Networks (D-GGNN), to perform this inference. Unlike GGNN, the D-GGNN can be applied to an input image to build a scene graph, without assuming that the structure of the input digraph is known.

In each training episode, the D-GGNN constructs a candidate graph structure for a given training input image by sequentially adding new nodes and edges. The D-GGNN builds the graph in a *deep reinforcement learning* (RL) framework. In each training step, the graph built so far is the current state. To encode the current graph in a state feature vector, the D-GGNN leverages the power of a GGNN to exploit the relational context information of the input image, and combines the GGNN with an attention mechanism. Given the current graph, and a current object, D-GGNN selects two actions: i) a new neighbor and its type ii) the type of the new edge from the current object to the new object. The reward for each action is a function of how well the predicted types measure the ground-truth labels. At the test time, the D-GGNN builds a graph for a given input by sequentially selecting the best actions with the greatest expected accuracies (Q-values).

Closely related to D-GGNN, Liang et al. (2017) proposed a model for visual relationship and attribute detection based on reinforcement learning. In their method, as a state feature representation, they used the embedding of the last two relationships and attributes that were searched during the graph construction. This fixed-length representation will lead to limited representational power, since the resulting representation depends on the order of the actions, that is, the order that the algorithm selects node and edges. For example, this method may generate different representations using different order of actions for the same graph. In contrast, our state feature representation is based on a Gated Graph Neural Network (GGNN) which is tailored towards graph-structured input data and extracts features from the entire graph.

In summary, our contributions are as follows: i) We propose a new deep generative architecture that uses reinforcement learning to generate from unstructured inputs a heterogeneous graph, which represents the input information with multiple types of nodes

and edges. Unlike the recently proposed Gated Graph Neural Networks (GGNN), the D-GGNN can be applied to an input without requiring that the structure of the scene graph is known in advance. ii) We apply the D-GGNN to the scene graph generation task. The D-GGNN can exploit domain priors and the relational context of objects in an input image to generate more accurate scene graphs than previous work. iii) Our model scales to predict thousands of predicates and object classes. The content of this section has been published in the proceedings of the 2018 Asian Conference on Computer Vision (Khademi and Schulte, 2018a).

5.2.1 New Reinforcement Learning Architecture for Graph Structure Generation

The state feature vectors in our RL framework encode an entire candidate graph. The first step in building the graph encoding is to encode node information by feature vectors (embedding, representations) such that each node feature vector takes into account contextual information from the neighbor nodes. For example, in scene graph prediction task, if we know that a node has an incoming edge from another node which represents a *man* and the type of the edge is *riding*, then the type of the node is likely *horse* or *bicycle*. There are various approaches for finding node embeddings which can be applied within our RL framework. We describe an approach that utilizes the Gated Graph Neural Network (GGNN) (Li et al., 2015), which is a recent state-of-the-art approach.

As we discussed in Chapter 2, a GGNN takes as input a heterogeneous directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each directed edge $e = (v, u) \in \mathcal{E}$ has an edge-type $\texttt{type}(e) \in \{1, \ldots, K\}$, where K is the number of edge-types (classes, labels). The given graph may also contain node-types $\texttt{type}(v) \in \{1, \ldots, M\}$ for each node v, where M is the number of node-types (classes, labels). Given the graph structure and the edge types, a GGNN iteratively computes a new node embedding $\mathbf{h}_v^{(t)} \in \mathbb{R}^d$, $t = 1, \ldots, T$, for each node v via a propagation model, where d is the embedding size. For each node the state vector from the last time step is used as the node representation (see Chapter 2 for the recurrence of the propagation and update equations of the GGNN).

Given the graph structure and the edge types, the GGNN model can be applied to each node to get a node representation which takes into account contextual information from the neighbor nodes . However, the graph structure and edge-types are not often given in tasks such as scene graph generation. This prohibits the use of this model for many of the real-world tasks. In this work, we extend GGNN to infer edges and edge-types.

Given an input image and a set of candidate bounding-boxes $\mathcal{P}$, our goal is to find a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \subseteq \mathcal{P}$, and assign the edge-types and node-types to it jointly such that $\mathcal{G}$ represents contextual information of the given input. The algorithm that we describe here can be applied for extracting relational information for a variety of

input data sources. However, our presentation focuses on scene graph generation, the target application of this work.

Initial Information Extraction

We extract initial information to constrain the reinforcement learning graph construction.

Global Type Information. We extract the following information from the training set to model type constraints in the target domain.

1. A set of M node types.
2. For each ordered pair of node-types i and j, a set of possible edge-types $\texttt{e-types}(i, j)$. For example, we may find that $\texttt{e-types}(man, horse) = \{riding, nextto, on, has\}$.

Image Node and Type Candidates. We extract the following information from each training image.

1. A set $\mathcal{P}$ of *candidate nodes*.
2. For each candidate node $v \in \mathcal{P}$:
   (a) A confidence score $s(v)$ that measures how likely v is to be a node in the scene graph.
   (b) A set of candidate node-types $\texttt{n-types}(v) \subseteq \{1, \ldots, M\}$.
   (c) A feature vector $\hat{\mathbf{x}}_v$.
   (d) A *vicinity* set $\texttt{vic}(v)$ is given such that $u \notin \texttt{vic}(v)$ implies that $e = (v, u)$ is not an edge in the scene graph.

This information is extracted as follows. Objects (nodes) are represented by bounding-boxes. Each bounding-box v is specified by a confidence score $s(v)$ and its coordinates $(v_x, v_y, v_{x'}, v_{y'})$, where (v_x, v_y) and $(v_{x'}, v_{y'})$ are the top left and bottom right corners of the bounding-box, respectively. Given an image dataset with ground truth annotations, we train an *object detector*, using the Tensorflow Object Detection API from `https://github.com/tensorflow/models/tree/master/research/object_detection`. We use `faster_rcnn_nas` trained on MS-COCO dataset as the pretrained model. Also, we use default values for all hyper-parameters of the object detector API.

For each input image, the object detector outputs a set of object bounding-boxes with their objectness scores, classification scores, and bounding-box coordinates. We used 100 bounding-box per image with the highest objectness scores as the candidate nodes $\mathcal{P}$, and their objectness scores as the confidence scores. For each bounding box, the classification scores rank the possible node-types. The set of candidate object categories $\texttt{n-types}(v)$ comprises the highest-scoring types. For example, $\texttt{n-types}(v)$ can be $\{man, boy, skier\}$.

To extract a feature vector $\hat{\mathbf{x}}_v$ for each bounding-box $v \in \mathcal{P}$, we first feed the image to the 152-layer ResNet (He et al., 2015), pretrained on ImageNet (Deng et al., 2009), and obtain 1024 feature maps of size 14×14 from layer `res4b35x`. Then, we apply a Region of Interest pooling layer (Girshick, 2015), based on the coordinates of the bounding-box, to get 1024 feature maps of size 7×7. Next, we use two fully-connected layers with ReLU activation function to get a 512-dimensional feature vector. Finally the set $\texttt{vic}(v)$ is a subset of bounding-boxes in $\mathcal{P}$ which are spatially close to bounding-box v ($u_w = u'_x - u_x$ and $u_h = u'_y - u_y$):

$$\texttt{vic}(v) = \{u : u \neq v, |u_x - v_x| < 0.5(u_w + v_w), |u_y - v_y| < 0.5(u_h + v_h)\}$$

Dynamic Gated Graph Neural Networks

Figure 5.6 shows the architecture of a D-GGNN for scene graph generation task. The algorithm simultaneously builds the graph and assigns node-types and edge-types using Deep Q-Learning (Mnih et al., 2013, 2015). The full algorithm is presented in Algorithm 3. The Q-function maps a state-action pair to an expected cumulative reward value. Actions and states are defined as follows.

Actions. The node with the highest confidence score is the starting node v, and the type with the highest classification score its type l. At each time step t, given a current subject node $v \in \mathcal{P}$ with node type l, we expand the scene graph by adding a new object node and a new edge, and selecting the type of the new node and the type of the new edge. These choices represent two actions.

1. Select a node u_t and a node-type a_t from the pairs

$$\mathcal{A} = \{(a, u) : u \in \texttt{vic}(v) - \texttt{prv}(v), a \in \texttt{n-types}(u)\} \cup \{\texttt{STOP}\} \tag{5.24}$$

   where $\texttt{prv}(v)$ denotes the set of previously selected nodes for neighbors of v, and `STOP` is a special action that indicates the end of searching for neighbors of node v. We first select the node type. If there are multiple nodes in $\mathcal{A}$ with the same type as the selected type, we randomly select one of them.

2. Select an edge-type b_t from

$$\mathcal{B} = \texttt{e-types}(l_t, a_t). \tag{5.25}$$

States. The current RL *state* is a (partial) scene graph denoted by s_t. The new state (graph) is obtained as $s_{t+1} = s_t + u_t + e_t$, where $s_t + u_t$ is obtained by adding node u_t with type a_t to graph s_t, and $s_t + u_t + e_t$ is obtained by adding edge $e_t = (v, u_t)$ with type b_t to graph $s_t + u_t$. A GGNN computes a state representation for state s_t. For each node v' of s_t, we initialize the node representation as $\mathbf{h}^{(0)}_{v'} = W(\mathbf{x}_{v'}, \hat{\mathbf{x}}_{v'})$, where $W \in \mathbb{R}^{d \times (512+M)}$

is a trainable matrix; so instead of padding the one-hot node type vector $\mathbf{x}_{v'}$ with 0, we concatenate it with the ResNet feature vector $\hat{\mathbf{x}}_{v'}$. Then, the state is represented by a vector $\texttt{GGNN}(s_t)$ computed as follows

$$\texttt{GGNN}(s_t) = \tanh\Big(\sum_{v'} \sigma\big(f(\mathbf{x}_{v'}, \mathbf{h}_{v'}^{(T)})\big) \odot \tanh\big(g(\mathbf{x}_{v'}, \mathbf{h}_{v'}^{(T)})\big)\Big) \tag{5.26}$$

where f and g are two neural networks, σ is the sigmoid function. The vector $\sigma\big(f(\mathbf{x}_{v'}, \mathbf{h}_{v'}^{(T)})\big)$ is a soft attention mechanism that decides which nodes are relevant to generate the vector representation for state s_t. Intuitively, $\sigma\big(f(\mathbf{x}_{v'}, \mathbf{h}_{v'}^{(T)})\big)$ represents the degree of importance of node v' to represent the current state, i.e. the contribution of node v' for selecting the next node-types and edge-types.

Rewards. Given the current object bounding-box v and the new object bounding-box u, the reward functions for taking actions a (non-`STOP`) and b at state s are defined as follows (`IoU` stands for Intersection over Union):

1. $r(a, u, s) = +1$ if there exists a ground truth bounding-box h with object category a such that $\texttt{IoU}(u, h) \geq 0.5$, otherwise $r(a, u, s) = -1$

2. $r'(b, v, u, s) = +1$ if there exists two ground truth bounding-boxes h and h' such that $\texttt{IoU}(v, h) \geq 0.5$, $\texttt{IoU}(u, h') \geq 0.5$ and the type of edge $e = (v, u)$ is b, otherwise $r'(b, v, u, s) = -1$.

Temporal-Difference Training of Q-networks. We use two separate Q-networks to select the actions: node-type Q-network and edge-type Q-network. We use two fully-connected layers with ReLU activation function to implement each Q-network. The input to each Q-network is $\phi_t = \phi(s_t) = \big(\texttt{GGNN}(s_t), \mathbf{x}\big)$, where $\mathbf{x}$ is a *global image feature vector* extracted from the last layer of the 152-layer ResNet (He et al., 2015). The global feature vector adds contextual information of the image to the current state information.

After each 5000 steps, node-type Q-network trainable parameters $\theta^{(t)}$ and edge-type Q-network trainable parameters $\theta'^{(t)}$ are copied to $\hat{\theta}^{(t)}$ and $\hat{\theta}'^{(t)}$ which are used to compute the target Q-values for the node-type and edge-type Q-networks, respectively. This helps stabilize the optimization.

To reduce correlation between samples and keep experiences from the past episodes, we use an experience replay technique (Mnih et al., 2013, 2015). To update the parameters of the Q-networks, we choose a random minibatch from the replay memory. Let $(\phi_j, s_j, a_j, b_j, r_j, r'_j, \phi_{j+1}, s_{j+1})$ be a random transition sample, and v_{j+1} the subject node in state s_{j+1}. The target Q-values for both networks are obtained as follows:

$$y_j = r_j + \gamma \max_{(a,u)\in\mathcal{A}_{j+1}} Q\big(\phi(s_{j+1} + u); \hat{\theta}^{(t)}\big) \tag{5.27}$$

$$y'_j = r'_j + \gamma \max_{b\in\mathcal{B}_{j+1}} Q\big(\phi(s_{j+1} + u^{\dagger}_{j+1} + e^{\dagger}_{j+1}); \hat{\theta}'^{(t)}\big) \tag{5.28}$$

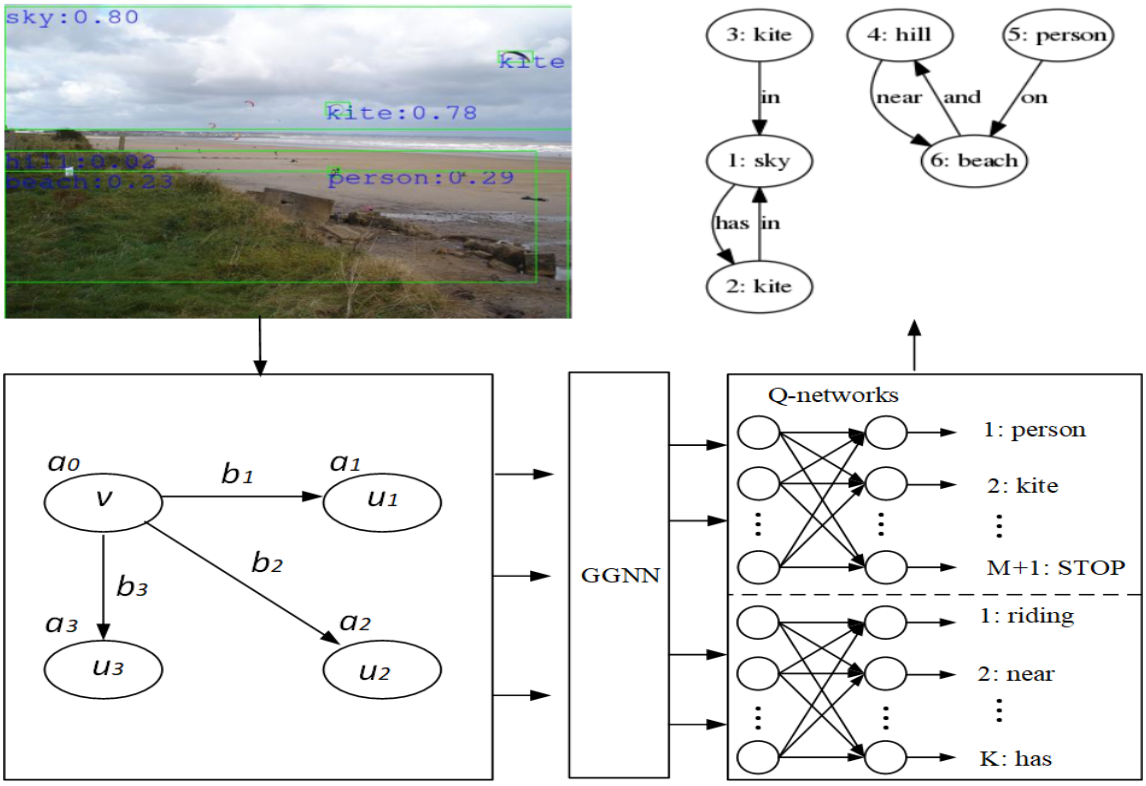


Figure 5.6: Dynamic Gated Graph Neural Networks for scene graph generation task. Given an image and its candidate object bounding-boxes, we simultaneously build the graph and assign node-types and edge-types to the nodes and edges in a Deep Q-Learning framework. We use two separate Q-networks to select the actions, i.e. node-types and edge-types. We use two fully-connected layers with ReLU activation function to implement each Q-network. The input to the Q-networks is the concatenation of the GGNN representation of the current graph and a global feature vector from the image. The search for the scene graph continues with the next node in a breadth-first search order.

where $\mathcal{A}_{j+1}$ and $\mathcal{B}_{j+1}$ are actions that can be taken in state s_{j+1}, $u^{\dagger}_{j+1}$ is the node that maximizes 5.27, and $e^{\dagger}_{j+1} = (v_{j+1}, u^{\dagger}_{j+1})$.

Finally, the parameters of the model are updated as follows:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\Big(y_j - Q(\phi(s_j + u^{\dagger}_j); \theta^{(t)})\Big)\nabla_{\theta} Q(\phi(s_j + u^{\dagger}_j); \theta^{(t)}) \tag{5.29}$$

$$\theta'^{(t+1)} = \theta'^{(t)} + \alpha\Big(y'_j - Q(\phi(s_j + u^{\dagger}_j + e^{\dagger}_j); \theta'^{(t)})\Big)\nabla_{\theta'} Q(\phi(s_j + u^{\dagger}_j + e^{\dagger}_j); \theta'^{(t)}) \tag{5.30}$$

Search Strategy. The search for the scene graph continues with the next node in a breadth-first search order. The algorithm continues until all nodes that are reachable from the starting node have been visited. If some unvisited nodes are remained, we repeat the search for the next component, until the highest confidence score of the unvisited nodes is less than a threshold β, or a maximum number of steps n is reached. This allows the search to generate disconnected graphs. To train the RL model, we use ϵ-*greedy learning*, that is, with probability ϵ a random action is selected, and with probability $1 - \epsilon$ an optimal action is selected, as indicated by the Q-networks. For test images, we construct the graph by sequentially selecting the optimal actions.

Optimization and Implementation Details

We used RMSProp with minibatch size of 100. The learning rate α was 0.00001. The number of iterations for the GGNN is set to $T = 2$. The maximum number of steps to construct a graph for an image is set to $n = 300$. The discount factor γ is set to 0.85 and ϵ is annealed from 1 to 0.05 during the first 50 epochs, and is fixed after epoch 50. The embedding size d is set to 512.

To avoid overfitting, we used a low-rank bilinear method (Pirsiavash et al., 2009) to reduce the rank of the weight matrix for each edge type. This technique effectively reduces the number of trainable parameters. We train our model for 100 epochs. Our model takes around two weeks to train on two NVIDIA Titan X GPUs.

Algorithm 3: Dynamic Gated Graph Neural Networks: Deep Q-learning

Hyperparameters are described in the text
Input : A set of bounding-boxes $\mathcal{P}$ of an image with their feature vectors
Input : A global feature vector $\mathbf{x}$ extracted from the image
Result: Updates the parameters of the model (the weight matrices of the GGNN for each edge-type and parameters of both Q-networks)

$\mathcal{C} \leftarrow \mathcal{P}$ ▷ *Mark all nodes as unvisited ($\mathcal{C}$ is the set of all unvisited nodes)*
Let s_0 be an empty graph, $t \leftarrow 0$ ▷ *Initialize the state (graph)*
while $\mathcal{C}$ is not empty **do**
  From $\mathcal{C}$ select node v with the highest confidence score and its node-type l
  if $s(v) < \beta$ or $t > n$ **then** break
  Add node v to s_t, and compute $\phi(s_t) = \Big(\texttt{GGNN}(s_t), \mathbf{x}\Big)$ ▷ *Equation 5.26*
  $q = \text{Queue}(), q.\text{enqueue}((v, l))$ ▷ *Make an empty queue and add (v, l) to it*
  while q is not empty **do**
    $(v, l) = q.\text{dequeue}()$, $\texttt{prv}(v) \leftarrow \emptyset$ ▷ *Remove an element from the queue*
    repeat
      Generate a random number $z \in (0, 1)$ with uniform distribution
      if $z < \epsilon$ **then**
        $\mathcal{A} \leftarrow \{(a, u) : u \in \texttt{vic}(v) - \texttt{prv}(v), a \in \texttt{n-types}(u)\} \cup \{\texttt{STOP}\}$
        Select a random node-type a_t and its node u_t from $\mathcal{A}$
        if a_t is not `STOP` **then**
          Select a random edge-type b_t from $\mathcal{B} = \texttt{e-type}(l, a_t)$
        end
      else
        Select $(a_t, u_t) = \arg\max_{(a,u)\in\mathcal{A}} Q\Big(\phi(s_t + u); \theta\Big)$
        if a_t is not `STOP` **then**
          Select $b_t = \arg\max_{b\in\mathcal{B}} Q\Big(\phi(s_t + u_t + e_t); \theta'\Big)$
        end
      end
      if a_t is not `STOP` **then**
        Compute rewards r_t and r'_t
        $s_{t+1} \leftarrow s_t + u_t + e_t$
      else
        $s_{t+1} \leftarrow s_t$
      end
      $\phi_{t+1} \leftarrow \phi(s_{t+1}) = \Big(\texttt{GGNN}(s_{t+1}), \mathbf{x}\Big)$ ▷ *Equation 5.26*
      Store transition $(\phi_t, s_t, a_t, b_t, r_t, r'_t, \phi_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
      Sample minibatch of transitions $(\phi_j, s_j, a_j, b_j, r_j, r'_j, \phi_{j+1}, s_{j+1})$ from replay memory $\mathcal{D}$
      Compute target Q-values y_j and y'_j ▷ *Equations 5.7 and 5.28*
      Update the parameters of the model ▷ *Equations 5.29 and 5.30*
      if u_t is in $\mathcal{C}$ **then**
        $q.\text{enqueue}((u_t, a_t))$ ▷ *Add (u_t, a_t) to the queue*
      end
      $\texttt{prv}(v) = \texttt{prv}(v) \cup \{u_t\}$, $t \leftarrow t + 1$ ▷ *Add node u_t to the set prv(v)*
    until a_t is `STOP`
    $\mathcal{C} \leftarrow \mathcal{C} - \Big(\{v\} \cup \texttt{vic}(v)\Big)$ ▷ *Mark node v and its vicinity as visited*
  end
end

5.2.2 Experiments

We introduce the datasets, baseline models and evaluation metrics that we use in our experiments. Then, the experimental results are presented and discussed.

Data, Metrics, and Baseline Models

Datasets. We used Visual Genome version 1.4 release. Following Xu et al. (2017), we use the most frequent 150 object categories and 50 predicates for scene graph prediction task. We call this variation VG1.4-a which is the same as the dataset which we used to evaluate the DG-PGNN algorithm. This results in a scene graph of about 11.5 objects and 6.2 relationships per image. The training and test splits contains 70% and 30% of the images, respectively.

Following Liang et al. (2017), we use the most frequent 1750 object categories and 347 predicates for scene graph prediction task. We call this variation VG1.4-b. Following Liang et al. (2017), we used 5000 images for validation, and 5000 for testing. This variation of the data allows large scale evaluation of our model.

Metrics. Top-K recall (`Rec@K`) is used as the metric, which is the fraction of the ground truth relationship triplets (subject-predicate-object) hit in the top-K predictions in an image. Predictions are ranked by the product of the objectness confidence scores and the Q-values of the selected predicates. Following Xu et al. (2017), we evaluate our model on VG1.4-a based on three tasks: Predicate classification (PRED-CLS) task, Scene graph classification (SG-CLS) task, and Scene graph generation (SG-GEN) task.

Following Liang et al. (2017), we evaluate our model on VG1.4-b based on two tasks as follows:

1. Relationship phrase detection (REL-PHRASE-DET): to predict a phrase (subject-predicate-object), such that the detected box of the entire relationship has at list 0.5 IoU overlap with a bounding-box annotation.

2. Relationship detection (REL-DET): to predict a phrase (subject-predicate-object), such that detected boxes of the subject and object have at least 0.5 IoU overlap with the corresponding bounding-box annotations.

Baseline Models. We compare our model with several baseline models including the state-of-the-art models (Liang et al., 2017; Xu et al., 2017). Faster R-CNN uses R-CNN to detect object proposals, while CNN+TRPN trains a separate region proposal network (RPN) on VG1.4-b.

Lu et al. (2016a) uses language priors from semantic word embeddings, while Xu et al. (2017) uses an RNN and learns to improves its predictions iteratively through message passing across the scene graph. After a few steps the learned features are classified.

VRL (Liang et al., 2017) detects visual relationship and attributes based on a reinforcement learning framework. To extract a state feature, they used the embedding of the last two relationships and attributes that were searched during the graph construction.

Results and Discussion

Tables 5.1 and 5.4 report our experimental results on VG1.4-a and VG1.4-b datasets, respectively. CNN+RPN, Faster R-CNN, and CNN+TRPN train independent detectors for object and predicate classes. As a result, they cannot exploit relational information in the given image. D-GGNN outperforms the state-of-the art models (VRL (Liang et al., 2017) and Xu et al. (2017)) for scene graph generation.

Both D-GGNN and the message passing approach (Xu et al., 2017) leverage the power of RNNs to learn a feature vector for each bounding-box. However, the message passing suffers from imbalanced classification problem (often there is no edge between many pairs of objects).

Both D-GGNN and VRL use deep Q-learning to generate the scene graph. However, the representational power of VRL is limited, since it represents a state by the embedding of the last two relationships and attributes that were searched during the graph construction. In contrast, our state feature representation is based on a GGNN which exploits contextual clues from the entire graph to more effectively represent objects and their relationships.

Figure 5.7 and 5.8 illustrate some scene graphs generated by our model. The D-GGNN predicts a rich semantic representation of the given image by recognizing objects, their locations, and relationships between them. For example, D-GGNN can correctly detect spatial relationships (*trees behind fence*, *horse near water*, *building beside bus*), parts of objects (*bus has tire*, *woman has leg*), and interactions (*man riding motorcycles*, *man wearing hat*).

	REL-PHRASE-DET		REL-DET	
Model	R@100	R@50	R@100	R@50
CNN+RPN (Simonyan and Zisserman, 2014)	01.39	01.34	01.22	01.18
Faster R-CNN (Ren et al., 2015)	02.25	02.19	-	-
CNN+TRPN (Ren et al., 2015)	02.52	02.44	02.37	02.23
(Lu et al., 2016a)	10.23	09.55	07.96	06.01
VRL (Liang et al., 2017)	16.09	14.36	13.34	12.57
D-GGNN	**18.21**	**15.78**	**14.85**	**14.22**

Table 5.4: Experimental Results of the relationship phrase detection and relationship detection tasks on the VG1.4-b dataset.

5.3 Discussion: Deep Generative Probabilistic Graph Neural Networks versus Dynamic Gated Graph Neural Networks

The DG-PGNN is a generative probabilistic extension to the Graph Network (GN) that we discussed in Chapter 2. The D-GGNN is a generative extension to the Gated Graph Neural networks (GGNN). Both algorithms use deep Q-learning to construct a graph for the given image. However, the actions in the Q-networks are different. In DG-PGNN, the actions

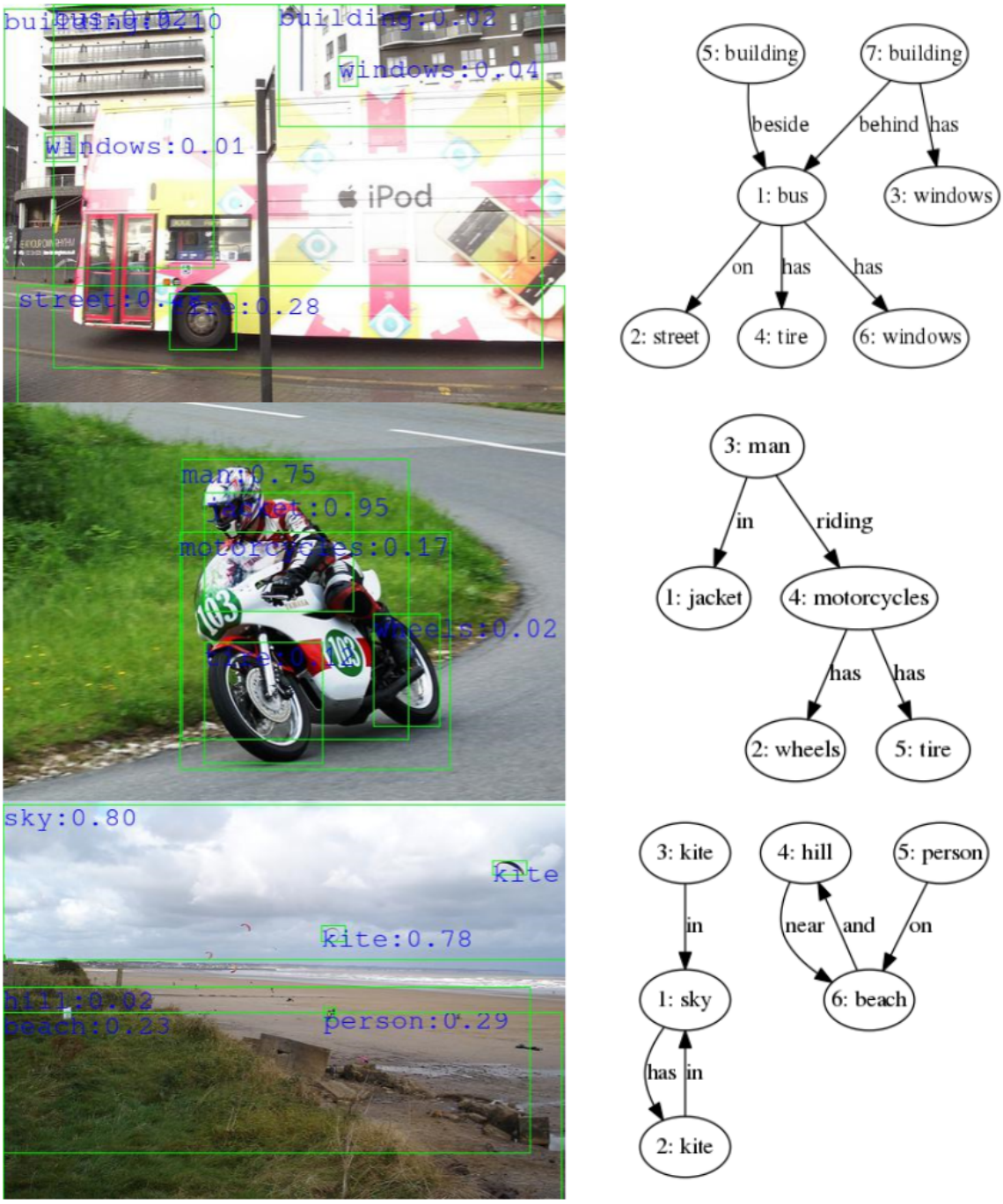


Figure 5.7: Some scene graphs generated by our model.

Figure 5.8: Some scene graphs generated by our model.

choose the candidate bounding-boxes, while in D-GGNN the actions choose node-types (object categories) and edge-types (predicates). Hence the D-GGNN uses two Q-networks: a Q-network for node-types and another Q-network for edge-types.

Another major difference is that the DG-PGNN learns a feature vector and a probability distribution over the possible edge-types for each edge, while the D-GGNN explicitly choose an edge-type based on the trained Q-network. Also, the graph construction in these two algorithms is different. The DG-PGNN assumes an edge between each pair of nodes in the graph that is constructed so far. But, the D-GGNN constructs the graph base on a breadth-first search schema.

Table 5.1 shows that the DG-PGNN significantly outperforms the D-GGNN which is likely due to the fact that DG-PGNN exploits the region-grounded captions to extract the scene graph, while the D-GGNN does not use textual information of the input image. Also, comparing the training time of these two algorithms shows that DG-PGNN is faster than D-GGNN. However, for the tasks which have thousands of candidate nodes and a few node-types/edge-types, the D-GGNN may perform better than DG-PGNN, since for such tasks the number of trainable parameters in the Q-network of the DG-PGNN will significantly increases which may lead to overfitting.

Chapter 6

Discussion and Conclusion

In this thesis, we addressed three multimodal learning tasks: visual question answering (VQA), scene graph generation, and image caption generation. We showed that graph neural networks are effective tools to achieve better performance on these tasks. Our new algorithms for scene graph generation, Deep Generative Probabilistic Graph Neural Networks (DG-PGNN) and Dynamic Gated Graph Neural Networks (D-GGNN), can be seen as a generic generative graph neural network algorithm. These algorithms can be applied to graph-structured data, or to infer graph structures, where the structure of the data is not given. A limitation of the DG-PGNN and D-GGNN is that the reinforcement learning framework which is used in these works is slow. Applying Monte Carlo tree search (MCTS) may help to decrease the computational cost for training the Q-network.

An interesting research direction is to use Variational Graph Auto-Encoders (Kipf and Welling, 2016b) or Graph Convolutional Networks, discussed in Chapter 2, as the backbone graph neural network in our DG-PGNN and D-GGNN. Variational Graph Auto-Encoders (Kipf and Welling, 2016b) is a powerful generative model. However, this model generates a graph using a fixed set of nodes. Our DG-PGNN may be integrated into the Variational Graph Auto-Encoders to obtain a generative graph neural network model which can generate arbitrary graphs from a large set of candidate nodes.

In Chapter 3, we have presented a novel, attention-based, contextual, deep architecture for image caption generation. Experimental results on the MS-COCO dataset showed the robustness of our model in terms of quantitative evaluations and qualitative results. The visualization results showed that our model can attend to the right concept during the image caption generation. We believe that, by leveraging the power of BiGrid LSTM, our architecture can generate attention maps that are more compatible with the human attention maps than other state-of-the-art models. However, a limitation of this work is that our architecture uses a grid of fixed image regions rather that a set of object bounding-box proposals. The supervision from object bounding-box proposals may help to improve the results since captions often describe objects in the input image. A future research direction is to apply the Graph Network model to this task (similar to the VQA task in Chapter 4).

This allows to learn a contextualized representation and an attention weight for each object bounding-box of the input image, instead of each fixed region of a grid.

In chapter 4, we proposed a new architecture, called Multi-modal Neural Graph Memory Networks, for the VQA task. The MN-GMN represents bi-modal local features as node attributes in a graph. It leverages a graph neural network model, Graph Network, to reason about objects and their interactions in the scene. In experiments on three datasets MN-GMN showed superior quantitative and qualitative performance, compared to both state-of-the-art comparison and lesion approaches. A future research direction is to combine RGCs with distant supervision by an external knowledge base to answer visual questions which need external knowledge; for example *Which animal in this photo can climb a tree?*

In Chapter 5, we proposed two new algorithms for scene graph generation based on graph neural networks: Deep Generative Probabilistic Graph Neural Networks (DG-PGNN) and Dynamic Gated Graph Neural Networks (D-GGNN). Our experiments show that the DG-PGNN algorithm significantly outperforms the state-of-the-art results on Visual Genome dataset for scene graph generation task. The scene graphs constructed by DG-PGNN improve performance on VQA task on Visual7W and CLEVR datasets. A future research direction is to train an *end-to-end* model which learns to answer visual question about an image by constructing a scene graph for the image.

Our D-GGNN model builds on the recently proposed GGNN model for computing latent node representations that combines relational and feature information. Unlike the recently proposed GGNN, the D-GGNN can be applied to an input when the structure of the input digraph is not known in advance. The target application in this work was the scene graph generation task.

Finally, although the focus of this thesis was multimodal learning tasks, as we discussed in Chapter 1, an interesting research direction is to apply graph neural networks to other graph-structured data. For example, we applied the Gated Graph Neural Networks to two tasks related to understanding source codes. We proposed to represent both the syntactic and semantic structure of a source code using graphs and applied Gated Graph Neural Networks to learn to reason over the source code structure. This work is published in the proceedings of International Conference on Learning Representations 2018 (Allamanis et al., 2017). The details can be found from `https://openreview.net/pdf?id=BJOFETxR-`.

Bibliography

Agrawal, A., Batra, D., and Parikh, D. (2016). Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Allamanis, M., Brockschmidt, M., and Khademi, M. (2017). Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*.

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

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

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

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Cadene, R., Ben-Younes, H., Cord, M., and Thome, N. (2019). Murel: Multimodal relational reasoning for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1989–1998.

Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., and Chua, T.-S. (2017). Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6298–6306. IEEE.

Chen, X. and Lawrence Zitnick, C. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2422–2431.

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

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

De Cao, N. and Kipf, T. (2018). Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

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

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *In Proceedings of the Ninth Workshop on Statistical Machine Translation*. Citeseer.

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

Fang, H., Gupta, S., Iandola, F., Srivastava, R. K., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J. C., et al. (2015). From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482.

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

Geman, D., Geman, S., Hallonquist, N., and Younes, L. (2015). Visual turing test for computer vision systems. *Proceedings of the National Academy of Sciences*, 112(12):3618–3623.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272.

Girshick, R. (2015). Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

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

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. (2017). Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813.

Jabri, A., Joulin, A., and Van Der Maaten, L. (2016). Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer.

Johnson, D. D. (2017). Learning graphical state transitions. In *International Conference on Learning Representations (ICLR)*.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017a). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017b). Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998.

Johnson, J., Karpathy, A., and Fei-Fei, L. (2015a). Densecap: Fully convolutional localization networks for dense captioning. *arXiv preprint arXiv:1511.07571*.

Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. (2015b). Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678.

Kafle, K. and Kanan, C. (2016). Visual question answering: Datasets, algorithms, and future challenges. *arXiv preprint arXiv:1610.01465*.

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

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

Khademi, M. and Schulte, O. (2018a). Dynamic gated graph neural networks for scene graph generation. In *Asian Conference on Computer Vision*, pages 669–685. Springer.

Khademi, M. and Schulte, O. (2018b). Image caption generation with hierarchical contextual visual spatial attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1943–1951.

Kim, J.-H., Jun, J., and Zhang, B.-T. (2018). Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N. and Welling, M. (2016b). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014). Multimodal neural language models. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603. JMLR Workshop and Conference Proceedings.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Lake, B. M. and Baroni, M. (2017). Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*.

Lebret, R., Pinheiro, P. O., and Collobert, R. (2015). Phrase-based image captioning. *arXiv preprint arXiv:1502.03671*.

Li, L., Gan, Z., Cheng, Y., and Liu, J. (2019). Relation-aware graph attention network for visual question answering. *arXiv preprint arXiv:1903.12314*.

Li, Y., Ouyang, W., Zhou, B., Wang, K., and Wang, X. (2017). Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1261–1270.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Liang, X., Lee, L., and Xing, E. P. (2017). Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4408–4417. IEEE.

Lu, C., Krishna, R., Bernstein, M., and Fei-Fei, L. (2016a). Visual relationship detection with language priors. In *European Conference on Computer Vision*, pages 852–869. Springer.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 6.

Lu, J., Yang, J., Batra, D., and Parikh, D. (2016b). Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Ma, C., Shen, C., Dick, A., Wu, Q., Wang, P., van den Hengel, A., and Reid, I. (2018). Visual question answering with memory-augmented networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6975–6984.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30(1), page 3.

Malinowski, M. and Fritz, M. (2014a). A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690.

Malinowski, M. and Fritz, M. (2014b). Towards a visual turing challenge. *arXiv preprint arXiv:1410.8027*.

Malinowski, M. and Fritz, M. (2015). Hard to cheat: A turing test based on answering questions about images. *arXiv preprint arXiv:1501.03302*.

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

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. L. (2015). Learning like a child: Fast novel visual concept learning from sentence descriptions of images. *CoRR*, abs/1504.06692.

Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. L. (2014b). Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*.

Marcus, G. (2018a). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marcus, G. (2018b). Innateness, alphazero, and artificial intelligence. *arXiv preprint arXiv:1801.05667*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Nam, H., Ha, J.-W., and Kim, J. (2016). Dual attention networks for multimodal reasoning and matching. *arXiv preprint arXiv:1611.00471*.

Newell, A. and Deng, J. (2017). Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, pages 2171–2180.

Norcliffe-Brown, W., Vafeias, S., and Parisot, S. (2018). Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pages 8334–8343.

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

Pearl, J. (2018). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

Pirsiavash, H., Ramanan, D., and Fowlkes, C. C. (2009). Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pages 1482–1490.

Posner, M. I. (1980). Orienting of attention. *Quarterly journal of experimental psychology*, 32(1):3–25.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schuster, S., Krishna, R., Chang, A., Fei-Fei, L., and Manning, C. D. (2015). Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

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

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

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

Tang, K., Zhang, H., Wu, B., Luo, W., and Liu, W. (2019). Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6619–6628.

Teney, D., Anderson, P., He, X., and van den Hengel, A. (2018). Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4223–4232.

Teney, D., Liu, L., and van den Hengel, A. (2016). Graph-structured representations for visual question answering. *CoRR, abs/1609.05600*, 3.

Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. (2014). Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2014). Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, Q., Shen, C., Liu, L., Dick, A., and van den Hengel, A. (2016). What value do explicit high level concepts have in vision to language problems? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–212.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2019). A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596.

Xiong, C., Merity, S., and Socher, R. (2016). Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*.

Xu, D., Zhu, Y., Choy, C. B., and Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Xu, H. and Saenko, K. (2015). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Xu, K., Ba, J., Kiros, R., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Yang, J., Lu, J., Lee, S., Batra, D., and Parikh, D. (2018a). Graph r-cnn for scene graph generation. *arXiv preprint arXiv:1808.00191*.

Yang, Z., Yu, J., Yang, C., Qin, Z., and Hu, Y. (2018b). Multi-modal learning with prior visual relation reasoning. *arXiv preprint arXiv:1812.09681*.

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

Yao, T., Pan, Y., Li, Y., Qiu, Z., and Mei, T. (2016). Boosting image captioning with attributes. *arXiv preprint arXiv:1611.01646*.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. *CoRR*, abs/1603.03925.

Yu, Z., Yu, J., Xiang, C., Fan, J., and Tao, D. (2018). Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE transactions on neural networks and learning systems*, 29(12):5947–5959.

Yuille, A. L. and Liu, C. (2018). Deep nets: What have they ever done for vision? *arXiv preprint arXiv:1805.04025*.

Zellers, R., Yatskar, M., Thomson, S., and Choi, Y. (2018). Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5831–5840.

Zhang, Y., Hare, J., and Prügel-Bennett, A. (2018a). Learning to count objects in natural images for visual question answering. *arXiv preprint arXiv:1802.05766*.

Zhang, Z., Cui, P., and Zhu, W. (2018b). Deep learning on graphs: A survey. *CoRR*, abs/1812.04202.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., and Sun, M. (2018). Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434.

Zhu, Y., Groth, O., Bernstein, M., and Fei-Fei, L. (2015). Visual7w: Grounded question answering in images. *arXiv preprint arXiv:1511.03416*.