

# Structured Models for Vision-and-Language Reasoning

*Ronghang Hu*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2020-50

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-50.html>

May 17, 2020

Copyright © 2020, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Structured Models for Vision-and-Language Reasoning

by

Ronghang Hu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair

Professor Daniel Klein

Professor David Bamman

Spring 2020

# Structured Models for Vision-and-Language Reasoning

Copyright 2020  
by  
Ronghang Hu



Abstract

Structured Models for Vision-and-Language Reasoning

by

Ronghang Hu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Vision-and-language tasks (such as answering a question about an image, grounding a referring expression, or following a natural language instruction to navigate through a visual environment) require jointly modeling and reasoning over the two modalities of image and text. We have witnessed significant progress in joint visual and linguistic reasoning, often through neural approaches trained with the help of larger datasets and more computation resources. However, is solving these vision-and-language tasks as simple as building models with more parameters and training them on more data? If not, how can we build better reasoning models that are data-efficient and generalize well?

This thesis provides an answer to the above question with structured models for vision-and-language reasoning – models with architectures that take into account the patterns and regularities in human language, visual scenes, and agents’ skills. We begin with the task of referring expression grounding, where we show that significantly better accuracy and generalization can be achieved by taking into account the compositional structures in these expressions with our proposed Compositional Modular Networks (CMNs) in Chapter 2. We further address the visual question answering task in Chapter 3 with the End-to-End Module Networks (N2NMNs) based on dynamic compositional modules that align with the reasoning steps in the questions. In Chapter 4, we extend our work on modular reasoning and propose the Stack Neural Module Networks (SNMNs) that automatically induce a proper module layout with interpretable reasoning steps. Beyond modular reasoning, we also propose to construct context-aware representations of the visual scene with Language-Conditioned Graph Networks (LCGNs) in Chapter 5 for relational reasoning, and address the problem of reading text in images for question answering with iterative pointer-augmented multimodal transformers in Chapter 6. Finally, we show that embodied tasks also require structured models, and propose the Speaker-Follower models for the navigational instruction following task in Chapter 7 with the pair of a speaker model and a follower model that complement each other. In all these scenarios, we show that by taking into account the structures in the tasks and the input modalities, our models perform and generalize significantly better than their unstructured counterparts.

To my wife, Yuqing Zhang, Ph.D.

# Contents

<b>Contents</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Background . . . . .	2
1.3 Thesis Goals and Contributions . . . . .	4
<b>2 Compositional Modular Networks for Grounding</b>	<b>7</b>
2.1 Problem Statement . . . . .	7
2.2 Related Work . . . . .	9
2.3 Compositional Modular Networks (CMNs) . . . . .	10
2.4 Experiments . . . . .	14
2.5 Discussion . . . . .	21
<b>3 End-to-End Module Networks for Compositional VQA</b>	<b>22</b>
3.1 Problem Statement . . . . .	22
3.2 Related Work . . . . .	23
3.3 End-to-End Module Networks (N2NMNs) . . . . .	25
3.4 Experiments . . . . .	30
3.5 Discussion . . . . .	35
<b>4 Explainable Neural Computation via Stack Neural Module Networks</b>	<b>37</b>
4.1 Problem Statement . . . . .	37
4.2 Related Work . . . . .	39
4.3 Stack Neural Module Networks (SNMNs) . . . . .	40
4.4 Experiments . . . . .	44
4.5 Discussion . . . . .	49
<b>5 Language-Conditioned Graph Networks</b>	<b>51</b>
5.1 Problem Statement . . . . .	51
5.2 Related Work . . . . .	53
5.3 Language-Conditioned Graph Networks (LCGNs) . . . . .	54

5.4	Experiments . . . . .	58
5.5	Discussion . . . . .	65
<b>6</b>	<b>Iterative Pointer-Augmented Multimodal Transformers for TextVQA</b>	<b>66</b>
6.1	Problem Statement . . . . .	66
6.2	Related Work . . . . .	68
6.3	Multimodal Multi-Copy Mesh (M4C) . . . . .	69
6.4	Experiments . . . . .	72
6.5	Discussion . . . . .	80
<b>7</b>	<b>Speaker-Follower Models for Instruction Following</b>	<b>81</b>
7.1	Problem Statement . . . . .	81
7.2	Related Work . . . . .	82
7.3	Instruction Following with Speaker-Follower Models . . . . .	84
7.4	Experiments . . . . .	88
7.5	Discussion . . . . .	93
<b>8</b>	<b>Conclusion</b>	<b>94</b>
	<b>Bibliography</b>	<b>96</b>

## Acknowledgments

First and foremost, I would like to express my gratitude to my advisor Trevor Darrell. I still remember how excited I was at the moment six years ago when Trevor decided to take me as a summer visitor in his group. That moment was the light during the most difficult time I have ever had in my life. Had he not granting me this visit, my life would have been entirely different, and probably with a much smaller chance to start a research career. Since the 2014 summer, Trevor has been patiently mentoring me and supporting me throughout my visiting and my entire Ph.D. program. Under his advising, I gradually learned how to turn raw ideas into experiments, scientific analyses, and clearly-written research papers. Trevor helped me develop both the skills and the confidence that allowed me to present my work in front of thousands of others – something that had once frightened me in the past. I sincerely thank Trevor for his guidance, mentoring, and support in such a long journey. I am also very grateful to Kate Saenko, who has been co-advising me for more than 6 years since my summer visit in 2014. During these years, Kate helped me so much in developing research ideas, designing and carrying out experimental analyses, writing clear and rigorous research papers, and polishing my presentations. She also arranged my numerous visits to Boston University, which allowed me to exchange my ideas and results with a broad number of researchers in Boston and New England. The success of my research projects should be largely attributed to Kate’s advising and help. I would also like to thank Daniel Klein and David Bamman, who served on my thesis committee and supported my work.

My research accomplishments were the results of many successful collaborations, and I would like to thank my collaborators for their support in both my research and my career development. Marcus Rohrbach has been working together with me since my first research paper at Berkeley. Jacob Andreas helped me a lot in developing various vision-and-language models, and also in writing and presentation. Anna Rohrbach worked with me patiently on many of my research projects. Daniel Fried enlightened me on numerous topics in natural language understanding and helped me out in both academic and logistics issues. Lisa A. Hendricks collaborated with me on grounding visual explanations and provided valuable insights on my projects during my Ph.D. years. Judy Hoffman and Ning Zhang supported me throughout the memorable summer visit that largely led to the beginning of my research career at Berkeley.

I am grateful to my mentors and collaborators at Facebook AI Research (FAIR). I was fortunate to have the chance to do two summer internships at FAIR. Ross Girshick, Kaiming He and Piotr Dollár guided me onto the path of visual perception research, which led to a successful internship project in 2017 summer and a paper at CVPR 2018. I returned to FAIR in 2019 summer for another internship with Marcus Rohrbach and Amanpreet Singh, who helped me in formulating innovative research ideas and writing efficient and scalable codebase that eventually led to a successful CVPR 2020 paper. It was also such a nice experience to work with Xinlei Chen and Oleksii Sidorov on various projects. Also, I would like to thank Shiguang Shan and Ruiping Wang at the Institute of Computing Technology, Chinese Academy of Sciences (ICT CAS), who guided me into the beautiful world of computer vision.

Life at Berkeley would be much harder without many nice and supportive friends. I still remembered how Evan Shelhamer, Jonathan Long, and Jeff Donahue helped me figuring out how

to use Caffe, which enabled my research in deep learning. Also, Evan's cold-brew coffee was the fuel of AI innovation at Berkeley. Yang Gao, Dequan Wang, Huazhe Xu, Yi Wu, and Huijuan Xu had numerous discussions and chats with me, providing me both insights on research and valuable advice on my career path. Sayna Ebrahimi enlightened me with her wisdom, courage, and perseverance. Besides, I also received a lot of help from many other people at Berkeley and elsewhere. Thanks to all my friends and colleagues, who made my life much better and easier.

I would like to thank my parents Haifeng and Zhihong for their providing, care, and support throughout my life. Without your nurturing since kindergarten, I would have little chance to be admitted to the Computer Science Ph.D. program at Berkeley or even to have an opportunity to go to a college. Your continued love enabled me to pursue a research career. Finally, thanks to the love of my life, Yuqing, who has known me and been my soul mate for twelve years. I was so fortunate to start a family and spend my life with you.

# Chapter 1

## Introduction

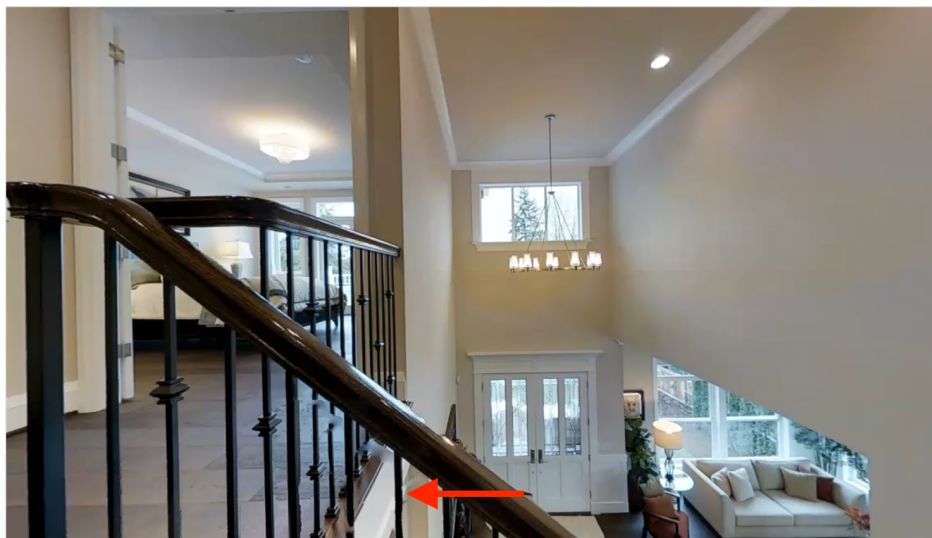
### 1.1 Motivation

Visual perception and language understanding are key to our daily life. As humans, we can look at the visual world around us, and describe what we see to other people in natural language. When told “there is a cat under the table”, we are able to associate the “cat” word in the sentence we hear with the actual cat instance in the surrounding scene. We are also capable of following instructions such as “go to the laboratory on the third floor” or “pick up the apple on the kitchen counter”. Our human intelligence allows us to jointly understand and reason over both the visual world around us and the natural language we hear and speak.

Similarly, it is also crucial for an intelligent machine to jointly model and reason over the two modalities of vision and language. Visual perception systems should be able to not only recognize or detect cars or pedestrians as symbolic class labels, but must also localize things in the visual scene described in natural language queries such as “the man wearing a black hat” and answer human questions such as “are there more apples than oranges in the basket”. An intelligent robot should be able to follow command in human language such as “make left at table clock, and wait at bathroom door threshold” in a physical environment and map “table clock” and “bathroom door threshold” in the instruction to the corresponding visual entities in the scene, as shown in Figure 1.1.

Following the success of neural networks and deep learning, an increasing number of methods have been developed for jointly modeling the two modalities of vision and language based on neural approaches, with a majority of them relying on large amounts of training data (and computation resources). The unreasonable effectiveness of data in the deep learning era has been demonstrated through various successes in vision (*e.g.* [92, 166]) and language (*e.g.* [121, 44]). However, while it has been a successful paradigm to train larger models with more data to get better performance, questions still remain: is solving the problem of joint vision and language understanding as simple as training larger neural networks on more data, and are there smarter approaches that are more data-efficient and generalize better?

This thesis provides an answer with structured models for vision-and-language reasoning – models that incorporate the design choices based on the regularities and patterns in the tasks and



Instruction: *Walk past hall table. Walk into bedroom. Make left at table clock, and wait at bathroom door threshold.*

Figure 1.1: A robot agent following a natural language instruction to navigate through an embodied environment [5] (moving left as shown with the red arrow). To accomplish the task, the agent needs to jointly perceive, understand, and reason over both the visual inputs and the language commands.

input modalities, by exploiting the compositionality of human language, dynamically reusing a set of reasoning operations with neural modules, and decomposing the tasks into complement subtasks. We show that better accuracy and generalization can be achieved by building well-designed structures into the reasoning models under various vision-and-language tasks, including referring expression grounding, visual question answering, and embodied instruction following.

## 1.2 Background

Early studies have started to address the problem of joint vision-and-language reasoning before the deep learning era, but they are often in restricted settings with limited representation power. For example, in [93] image descriptions are generated based on handwritten templates, and restricted logical forms are used for question answering about natural images in [113]. With the renaissance of deep learning [96], these early approaches have largely been replaced by subsequent neural approaches that learn an end-to-end mapping from inputs to outputs.

In particular, convolutional neural networks are shown effective in a wide range of vision tasks such as image recognition (*e.g.* [92]) and object detection (*e.g.* [144]), and recurrent neural networks have been successfully applied to language tasks such as syntactic parsing (*e.g.* [181]) and machine translation (*e.g.* [167]), often through the encoder-decoder approach [36]. Following their success, many neural approaches for vision-and-language reasoning adopt the CNN-RNN paradigm of encoding the visual inputs (images or videos) with convolutional neural networks (CNNs), encoding the language inputs (such as phrases, expressions, and questions) with recurrent



neural networks (RNNs, typically implemented as LSTMs [69] or GRUs [37]), and fusing the encoded representations of the two modalities with a few additional neural network layers to output the results (such as answers in question answering or the agent’s actions in embodied settings).

The CNN-RNN paradigm mentioned above has been widely applied to multiple tasks such as referring expression grounding [117], visual question answering [12], and embodied instruction following [5]. The referring expression grounding task takes as input an image and an expression that describes a particular object in the image (such as “the woman holding a grey umbrella”), and the model is required to ground (localize) the expression into the corresponding image regions. In this task, a widely-used approach is to extract visual features from candidate image regions with a convolutional neural network, extract an embedding vector from the language query with a recurrent neural network (often an LSTM network), and build another matching network on top of the visual features and the language embedding to fuse the two modalities and decide whether each image region can be matched to the query. The visual question answering task requires a machine model to answer a question (in natural language) based on an input image, such as “how many black cats are there in the scene”. A strong baseline of this problem is extracting CNN features from the image, extracting RNN features from the question, and fuse the two modalities to predict an answer [12, 114, 205, 77] (where the fusion step may involve *e.g.* bilinear pooling [51, 200, 29] or other fusion mechanisms). The embodied instruction following task asks an agent to follow a natural language instruction such as “enter the door, make a left turn, walk through the hallway, and stop at the table on your left” and navigate to a goal location in an embodied environment. In this task, a common approach is to extract CNNs features from the agent’s visual sensor inputs, and use a sequence-to-sequence RNN model to translate the sequence of visual features into the output sequence of the agent’s actions [5]. While the CNN-RNN paradigm can approximate any reasoning behavior in principle (given the universal approximation power of neural networks [40]) and will eventually achieve good performance with enough training data, it ignores the structures and patterns in the tasks and the input data and often does not generalize well, especially when it comes to compositionality. For example, when using an RNN-based encoding mechanism, a referring expression grounding model trained on expressions “an apple on the table” and “an orange in the basket” does not necessarily handle “an orange on the table” in a compositional manner. This is primarily because the recurrent neural network as an encoder is lack of the architectural power to encourage such compositionality. This thesis provides a solution to these issues by building models with proper architectures and inductive biases that encourage more compositional behavior and better generalization.

It is worth noting that there is a large amount of prior work on incorporating the structures of the tasks or input data when designing a reasoning model. As a few notable examples, TreeLSTMs [170] take into account the tree structure (usually based on syntactic parsing trees) of a language expression when propagating information, hierarchical co-attention networks [110] align the image and text modalities and extract information at multiple levels of granularity for question answering, and paired listeners and speakers are used in [8] for pragmatic referring expression generation. We leave the detailed discussions of related work on each specific task to the individual chapters. Our work presented in this thesis is aligned with these previous efforts in bringing the right architectures into the modeling aspect for vision-and-language reasoning, and extends previous methods with

dynamic reasoning structures based on neural modules, context-aware representations based on graph networks, pointer networks that allows directly referring to and copying the inputs, and paired models by decomposing the instruction following task into a follower and a speaker.

### 1.3 Thesis Goals and Contributions

This thesis presents a series of structured models for joint visual and linguistic reasoning, including referring expression grounding models based on the compositional structure of natural language expressions, visual question answering models that learn a dynamic and reusable set of reasoning skills based on neural modules, graph networks that build context-aware language-conditioned representations for multiple downstream tasks, text-reading models based on our novel pointer-augmented multimodal transformers, and embodied instruction following models with a follower and speaker pair based on a decomposition of the task into instruction generation and instruction following.

We begin with the referring expression grounding task in Chapter 2. People often refer to entities in an image in terms of their relationships with other entities through referring expressions. For example, “the black cat sitting under the table” refers to both a “black cat” entity and its relationship with another “table” entity. Understanding these relationships is essential for interpreting and grounding such natural language expressions. Most prior work focuses on either grounding entire referring expressions holistically to one region or localizing relationships based on a fixed set of categories. In this chapter, we instead present a modular deep architecture capable of analyzing referring expressions into their components, identifying entities and relationships mentioned in the input expression and grounding them all in the scene. We call this approach Compositional Modular Networks (CMNs): a novel architecture that learns linguistic analysis and visual inference end-to-end. Our approach is built around two types of neural modules that inspect local regions and pairwise interactions between regions. We evaluate CMNs on multiple referring expression datasets, outperforming state-of-the-art approaches on a range of application scenarios.

We further address the visual question answering task in Chapter 3. In the visual question answering task, the natural language questions are inherently compositional, and many are most easily answered by reasoning about their decomposition into modular sub-problems. For example, to answer “is there an equal number of balls and boxes?” we can look for balls, look for boxes, count them, and compare the results. The recently proposed Neural Module Network (NMN) architecture implements this approach to question answering by parsing questions into linguistic substructures and assembling question-specific deep networks from smaller modules that each solves one subtask. However, existing NMN implementations rely on brittle off-the-shelf parsers, and are restricted to the module configurations proposed by these parsers rather than learning them from data. In this chapter, we propose End-to-End Module Networks (N2NMNs), which learn to reason by directly predicting instance-specific network layouts without the aid of a parser. Our model learns to generate network structures (by imitating expert demonstrations) while simultaneously learning network parameters (using the downstream task loss). Experimental results on the new CLEVR dataset targeted at compositional question answering show that N2NMNs achieve an error reduction

of nearly 50% relative to state-of-the-art attentional approaches, while discovering interpretable network architectures specialized for each question.

In Chapter 4, we extended our N2NMN model proposed in Chapter 3 and learn to automatically induce a reasoning structure or a program layout without resorting to a large amount of human annotation. In complex inferential tasks like visual question answering, machine learning models must confront two challenges: the need to implement a compositional reasoning process, and, in many applications, the need for this reasoning process to be interpretable to assist users in both development and prediction. Existing models designed to produce interpretable traces of their decision-making process typically require these traces to be supervised at training time. In this chapter, we present a novel Stack Neural Module Network (SNMN) that performs compositional reasoning by automatically inducing a desired sub-task decomposition without relying on strong supervision. Our SNMN model allows linking different reasoning tasks through shared modules that handle common routines across tasks. Experiments show that the model is more interpretable to human evaluators compared to other state-of-the-art models: users can better understand the model’s underlying reasoning procedure and predict when it will succeed or fail based on observing its intermediate outputs.

In addition to modular reasoning structures for visual question answering and referring expression grounding, in Chapter 5 we build dynamic and context-aware representations of the visual scene conditioned on the reasoning task specified in the language inputs, using graph networks to propagate relational information across visual entities in images. Solving grounded language tasks often requires reasoning about relationships between objects in the context of a given task. For example, to answer the question “what color is the mug on the plate?” we must check the color of the specific mug that satisfies the “on” relationship with respect to the plate. Recent work has proposed various methods capable of complex relational reasoning. However, most of their power is in the inference structure, while the scene is represented with simple local appearance features. In this chapter, we take an alternate approach and build contextualized representations for objects in a visual scene to support relational reasoning. We propose a general framework of Language-Conditioned Graph Networks (LCGNs), where each node represents an object, and is described by a context-aware representation from related objects through iterative message-passing conditioned on the textual input. E.g., conditioning on the “on” relationship to the plate, the object “mug” gathers messages from the object “plate” to update its representation to “mug on the plate”, which can be easily consumed by a simple classifier for answer prediction. We experimentally show that our LCGN approach effectively supports relational reasoning and improves performance across several tasks and datasets.

In Chapter 6, we address a specific vision-and-language reasoning task – visual question answering based on reading comprehension (*i.e.* the TextVQA task) – that requires the right model architecture to be able to read and copy text tokens from images. Many visual scenes contain text that carries crucial information, and it is thus essential to understand text in images for downstream reasoning tasks. For example, a deep water label on a warning sign warns people about the danger in the scene. Recent work has explored the TextVQA task that requires reading and understanding text in images to answer a question. However, existing approaches for TextVQA are mostly based on custom pairwise fusion mechanisms between a pair of two modalities and are restricted to a

single prediction step by casting TextVQA as a classification task. In this work, we propose a novel model for the TextVQA task based on a multimodal transformer architecture accompanied by a rich representation of text in images. Our model naturally fuses different modalities homogeneously by embedding them into a common semantic space where self-attention is applied to model inter- and intra- modality context. Furthermore, it enables iterative answer decoding with a dynamic pointer network, allowing the model to form an answer through multi-step prediction instead of one-step classification. Our model outperforms existing approaches on three benchmark datasets for the TextVQA task by a large margin.

In Chapter 7, we go beyond passive perception and reasoning of static images and further address vision-and-language reasoning in the embodied navigational instruction following task that requires active decision-making and planning. Navigation guided by natural language instructions presents a challenging reasoning problem for instruction followers. Natural language instructions typically identify only a few high-level decisions and landmarks rather than complete low-level motor behaviors; much of the missing information must be inferred based on perceptual context. In machine learning settings, this is doubly challenging: it is difficult to collect enough annotated data to enable learning of this reasoning process from scratch, and also difficult to implement the reasoning process using generic sequence models. Here we describe an approach to vision-and-language navigation that addresses both these issues with an embedded speaker model. We use this speaker model to (1) synthesize new instructions for data augmentation and to (2) implement pragmatic reasoning, which evaluates how well candidate action sequences explain an instruction. Both steps are supported by a panoramic action space that reflects the granularity of human-generated instructions. Experiments show that all three components of this approach – speaker-driven data augmentation, pragmatic reasoning, and panoramic action space – dramatically improve the performance of a baseline instruction follower, more than doubling the success rate over the best existing approach on a standard benchmark.

## Chapter 2

# Compositional Modular Networks for Grounding

### 2.1 Problem Statement

Great progress has been made on object detection, the task of localizing visual entities belonging to a pre-defined set of categories [55, 144, 143, 41, 105]. But the more general and challenging task of localizing entities based on arbitrary natural language expressions remains far from solved. This task, sometimes known as *grounding* or *referring expression comprehension*, has been explored by recent work in both computer vision and natural language processing [117, 74, 146]. Given an image and a natural language expression referring to a visual entity, such as *the young man wearing green shirt and riding a black bicycle*, these approaches localize the image region corresponding to the entity that the expression refers to with a bounding box.

Referring expressions often describe relationships between multiple entities in an image. In Figure 2.1, the expression *the woman holding a grey umbrella* describes a *woman* entity that participates in a *holding* relationship with a *grey umbrella* entity. Because there are multiple women in the image, resolving this referring expression requires both finding a bounding box that contains a person, and ensuring that this bounding box relates in the right way to other objects in the scene. Previous work on grounding referring expressions either (1) treats referring expressions holistically, thus failing to model explicit correspondence between textual components and visual entities in the image [117, 74, 146, 198, 127], or else (2) relies on a fixed set of entity and relationship categories defined *a priori* [108].

In this chapter, we present a joint approach that explicitly models the compositional linguistic structure of referring expressions and their groundings, but which nonetheless supports interpretation of arbitrary language. We focus on referring expressions involving inter-object relationships that can be represented as a subject entity, a relationship and an object entity. We propose Compositional Modular Networks (CMNs), an end-to-end trained model that learns language representation and image region localization jointly as shown in Figure 2.1. Our model differentially parses the referring expression into a subject, relationship and object with three soft attention maps, and aligns

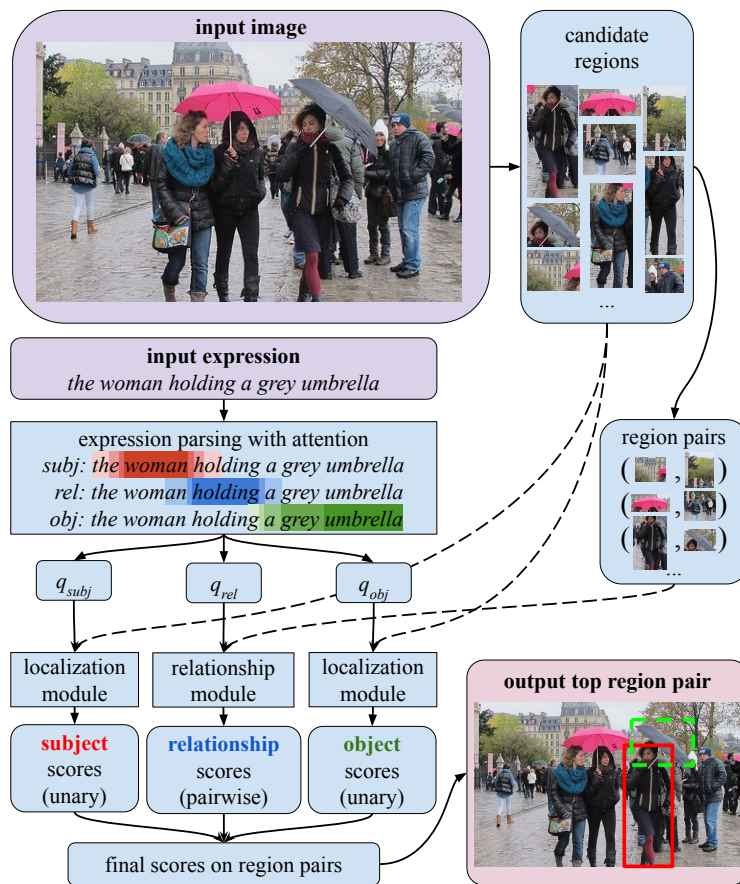


Figure 2.1: Given an image and an expression, we learn to parse the expression into vector representation of subject  $q_{subj}$ , relationship  $q_{rel}$  and object  $q_{obj}$  with attention, and align these textual components to image regions with two types of modules. The localization module outputs scores over each individual region while the relationship module produces scores over region pairs. These outputs are integrated into final scores over region pairs, producing the top region pair as grounding result. (Best viewed in color.)

the extracted textual representations with image regions using a modular neural architecture. There are two types of modules in our model, one used for localizing specific textual components by outputting unary scores over regions for that component, and one for determining the relationship between two pairs of bounding boxes by outputting pairwise scores over region-region pairs. We evaluate our model on multiple datasets, and show that our model outperforms both natural baselines and previous work.

## 2.2 Related Work

**Grounding referring expressions.** The problem of grounding referring expressions can be naturally formulated as a retrieval problem over image regions [117, 74, 146, 51, 198, 127]. First, a set of candidate regions are extracted (*e.g.* via object proposal methods like [174, 13, 89, 211]). Next, each candidate region is scored by a model with respect to the query expression, returning the highest scoring candidate as the grounding result. In [117, 74], each region is scored based on its local visual features and some global contextual features from the whole image. However, local visual features and global contextual from the whole image are often insufficient to determine whether a region matches an expression, as relationships with other regions in the image must also be considered. Two recent methods [198, 127] go beyond local visual features in a single region, and consider multiple regions at the same time. [198] adds contextual feature extracted from other regions in the image, and [127] proposes a model that grounds a referring expression into a pair of regions. All these methods represent language holistically using a recurrent neural network: either generatively, by predicting a distribution over referring expressions [117, 74, 198, 127], or discriminatively, by encoding expressions into a vector representation [146, 51]. This makes it difficult to learn explicit correspondences between the components in the textual expression and entities in the image. In this work, we learn to parse the language expression into textual components in instead of treating it as a whole, and align these components with image regions end-to-end.

**Handling inter-object relationships.** Recently work by [108] trains detectors based on R-CNN [55] and uses a linguistic prior to detect visual relationships. However, this work relies on fixed, predefined categories for subjects, relations, and objects, treating entities like “bicycle” and relationships like and “riding” as discrete classes. Instead of building upon a fixed inventory of classes, our model handles relationships specified by arbitrary natural language phrases, and jointly learns expression parsing and visual entity localization. Although [91] also learns language parsing and perception, it is directly based on logic ( $\lambda$ -calculus) and requires additional classifiers trained for each predicate class. Aside from localizing relationship expressions, [191] generates image descriptions using a recurrent network with attention over image feature grids, and [148, 194] learns to extract visual relation knowledge from images.

**Compositional structure with modules.** Neural Module Networks [11] address visual question answering by decomposing the questions into textual components and dynamically assembling a specific network architecture for the question from a few network modules based on the textual components. However, this method relies on an external language parser for textual analysis instead of end-to-end learned language representation, and is not directly applicable to the task of grounding referring expressions into bounding boxes, since it does not explicitly output bounding boxes as results. Recently, [10] improves over [11] by learning to re-rank parsing outputs from the external parser, but it is still not end-to-end learned since the parser is fixed and not optimized for the task. Inspired by [11], our model also uses a modular structure, but learns the language representation end-to-end from words.

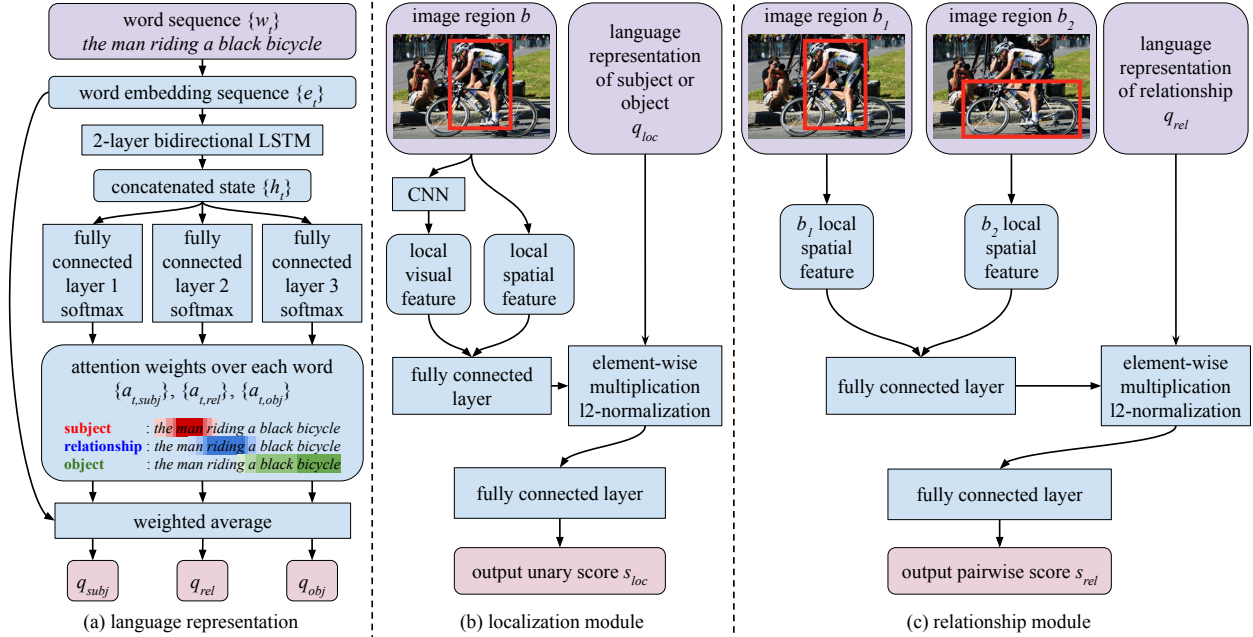


Figure 2.2: Detailed illustration of our model. (a) Our model learns to parse an expression into subject, relationship and object with attention for language representation (Sec. 2.3). (b) The localization module matches subject or object with each image region and returns a unary score (Sec. 2.3). (c) The relationship module matches a relationship with a pair of regions and returns a pairwise score (Sec. 2.3).

## 2.3 Compositional Modular Networks (CMNs)

We propose Compositional Modular Networks (CMNs) to localize visual entities described by a query referring expression. Our model is compositional in the sense that it localizes a referring expression by grounding the components in the expressions and exploiting their interactions, in accordance with the principle of compositionality of natural language – the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them [187]. Our model works in a retrieval setting: given an image  $I$ , a referring expression  $Q$  as query and a set of candidate region bounding boxes  $B = \{b_i\}$  for the image  $I$  (e.g. extracted through object proposal methods), our model outputs a score for each bounding box  $b_i$ , and returns the bounding box with the highest score as grounding (localization) result. Unlike state-of-the-art methods [146, 51], the scores for each region bounding box  $b_i \in B$  are not predicted only from the local feature of  $b_i$ , but also based on other regions in the image. In our model, we focus on the relationships in referring expressions that can be represented as a 3-component triplet (subject, relationship, object), and learn to parse the expressions into these components with attention. For example, *a young man wearing a blue shirt* can be parsed as the triplet (*a young man*, *wearing*, *a blue shirt*). The score of a region is determined by simultaneously looking at whether it matches the description of the subject entity and whether it matches the relationship with another



interacting object entity mentioned in the expression.

Our model handles such inter-object relationships by looking at pairs of regions  $(b_i, b_j)$ . For referring expressions like “the red apple on top of the bookshelf”, we want to find a region pair  $(b_i, b_j)$  such that  $b_i$  matches the subject entity “red apple” and  $b_j$  matches the object entity “bookshelf” and the configuration of  $(b_i, b_j)$  matches the relationship “on top of”. To achieve this goal, our model is based on a compositional modular structure, composed of two modules assembled in a pipeline for different sub-tasks: one localization module  $f_{loc}(\cdot, q_{loc}; \Theta_{loc})$  for deciding whether a region matches the subject or object in the expression, where  $q_{loc}$  is the textual vector representation of the subject component “red apple” or the object component “bookshelf”, and one relationship module  $f_{rel}(\cdot, \cdot, q_{rel}; \Theta_{rel})$  for deciding whether a pair of regions matches the relationship described in the expression represented by  $q_{rel}$ , the textual vector representation of the relationship “on top of”. The representations  $q_{subj}$ ,  $q_{rel}$  and  $q_{obj}$  are learned jointly in our model in Sec. 2.3.

We define the pairwise score  $s_{pair}(b_i, b_j)$  over a pair of image regions  $(b_i, b_j)$  matching an input referring expression  $Q$  as the sum of three components:

$$\begin{aligned} s_{pair}(b_i, b_j) &= f_{loc}(b_i, q_{subj}; \Theta_{loc}) \\ &+ f_{loc}(b_j, q_{obj}; \Theta_{loc}) \\ &+ f_{rel}(b_i, b_j, q_{rel}; \Theta_{rel}), \end{aligned} \quad (2.1)$$

where  $q_{subj}$ ,  $q_{obj}$  and  $q_{rel}$  are vector representations of subject, relationship and object, respectively.

For inference, we define the final subject unary score  $s_{subj}(b_i)$  of a bounding of  $b_i$  corresponding to the subject (e.g. “the red apple” in “the red apple on top of the bookshelf”) as the score of the best possible pair  $(b_i, b_j)$  that matches the entire expression:

$$s_{subj}(b_i) \triangleq \max_{b_j \in B} s_{pair}(b_i, b_j). \quad (2.2)$$

The subject is ultimately grounded (localized) to the highest scoring region as

$$b_{subj}^* = \arg \max_{b_i \in B} (s_{subj}(b_i)). \quad (2.3)$$

## Expression parsing with attention

Given a referring expression  $Q$  like *the tall woman carrying a red bag*, how can we decide which substrings corresponds to the subject, the relationship, and the object, and extract three vector representations  $q_{subj}$ ,  $q_{rel}$  and  $q_{obj}$  corresponding to these three components? One possible approach is to use an external language parser to parse the referring expression into the triplet format (subject, relationship, object) and then process each component with an encoder (e.g. a recurrent neural network) to extract  $q_{subj}$ ,  $q_{rel}$  and  $q_{obj}$ . However, the formal representations of language produced by syntactic parsers do not always correspond to intuitive visual representations. As a simple example, *the apple on top of the bookshelf* is analyzed [209] as having a subject phrase *the apple*, a relationship *on*, and an object phrase *top of the bookshelf*, when in fact the visually salient objects are simply the apple and the bookshelf, while the complete expression *on top of* describes the relationship between them.

Therefore, in this work we learn to decompose the input expression  $Q$  into the above 3 components, and generate vector representations  $q_{subj}$ ,  $q_{rel}$  and  $q_{obj}$  from  $Q$  through a soft attention mechanism over the word sequence, as shown in Figure 2.2 (a). For a referring expression  $Q$  that is a sequence of  $T$  words  $\{w_t\}_{t=1}^T$ , we first embed each word  $w_t$  to a vector  $e_t$  using GloVe [135], and then scan through the word embedding sequence  $\{e_t\}_{t=1}^T$  with a 2-layer bi-directional LSTM network [150]. The first layer takes as input the sequence  $\{e_t\}$  and outputs a forward hidden state  $h_t^{(1, fw)}$  and a backward hidden state  $h_t^{(1, bw)}$  at each time step, which are concatenated into  $h_t^{(1)}$ . The second layer then takes the first layer’s output sequence  $\{h_t^{(1)}\}$  as input and outputs forward and backward hidden states  $h_t^{(2, fw)}$  and  $h_t^{(2, bw)}$  at each time step. All the hidden states in the first layer and second layer are concatenated into a single vector  $h_t = \begin{bmatrix} h_t^{(1, fw)} & h_t^{(1, bw)} & h_t^{(2, fw)} & h_t^{(2, bw)} \end{bmatrix}$ .

The concatenated state  $h_t$  contains information from word  $w_t$  itself and also context from words before and after  $w_t$ . Then the attention weights  $a_{t, subj}$ ,  $a_{t, rel}$  and  $a_{t, obj}$  for subject, relationship, object over each word  $w_t$  are obtained by three linear predictions over  $h_t$  followed by a softmax as

$$a_{t, subj} = \exp(\beta_{subj}^T h_t) / \sum_{\tau=1}^T \exp(\beta_{subj}^T h_\tau) \quad (2.4)$$

$$a_{t, rel} = \exp(\beta_{rel}^T h_t) / \sum_{\tau=1}^T \exp(\beta_{rel}^T h_\tau) \quad (2.5)$$

$$a_{t, obj} = \exp(\beta_{obj}^T h_t) / \sum_{\tau=1}^T \exp(\beta_{obj}^T h_\tau) \quad (2.6)$$

and the language representations of the subject  $q_{subj}$ , relationship  $q_{rel}$  and object  $q_{obj}$  are extracted as weighed average of word embedding vectors  $\{e_t\}$  with attention weights as  $q_{subj} = \sum_{t=1}^T a_{t, subj} e_t$ , and  $q_{rel} = \sum_{t=1}^T a_{t, rel} e_t$  and  $q_{obj} = \sum_{t=1}^T a_{t, obj} e_t$ .

## Localization module

As shown in Figure 2.2 (b), the localization module  $f_{loc}$  outputs a score  $s_{loc} = f_{loc}(b, q_{loc}; \Theta_{loc})$  representing how likely a region bounding box  $b$  matches  $q_{loc}$ , which is either the subject textual vector  $q_{subj}$  or object textual vector  $q_{obj}$ .

This module takes the local visual feature  $x_{vis}$  and spatial feature  $x_{spatial}$  of image region  $b$ . We extract visual feature  $x_v$  from image region  $b$  using a convolutional neural network [156], and extract a 5-dimensional spatial feature  $x_s = \left[ \frac{x_{min}}{W_I}, \frac{y_{min}}{H_I}, \frac{x_{max}}{W_I}, \frac{y_{max}}{H_I}, \frac{S_b}{S_I} \right]$  from  $b$  using the same representation as in [117], where  $[x_{min}, y_{min}, x_{max}, y_{max}]$  and  $S_b$  are bounding box coordinates and area of  $b$ , and  $W_I$ ,  $H_I$  and  $S_I$  are width, height and area of the image  $I$ . Then,  $x_v$  and  $x_s$  are concatenated into a vector  $x_{v,s} = [x_v \ x_s]$  as representation of region  $b$ .

Since element-wise multiplication is shown to be a powerful way to combine representations from different modalities [15], we adopt it here to obtain a joint vision and language representation. In our implementation,  $x_{v,s}$  is first embedded to a new vector  $\tilde{x}_{v,s}$  that has the same dimension as  $q_{loc}$  (which is either  $q_{subj}$  or  $q_{obj}$ ) through a linear transform, and then element-wise multiplied with

$q_{loc}$  to obtain a vector  $z_{loc}$ , which is L2-normalized into  $\hat{z}_{loc}$  to obtain a more robust representation, as follows:

$$\tilde{x}_{v,s} = W_{v,s}x_{v,s} + b_{v,s} \quad (2.7)$$

$$z_{loc} = \tilde{x}_{v,s} \odot q_{loc} \quad (2.8)$$

$$\hat{z}_{loc} = z_{loc} / \|z_{loc}\|_2 \quad (2.9)$$

where  $\odot$  is element-wise multiplication between two vectors. Then the score  $s_{loc}$  is predicted linearly from  $\hat{z}_{loc}$  as

$$s_{loc} = w_{loc}^T \hat{z}_{loc} + b_{loc}. \quad (2.10)$$

The parameters in  $\Theta_{loc}$  are  $(W_{v,s}, b_{v,s}, w_{loc}, b_{loc})$ .

## Relationship module

As shown in Figure 2.2 (c), the relationship module  $f_{rel}$  outputs a score  $s_{rel} = f_{rel}(b_1, b_2, q_{rel}; \Theta_{rel})$  representing how likely a pair of region bounding boxes  $(b_1, b_2)$  matches  $q_{rel}$ , the representation of relationship in the expression.

In our implementation, we use the spatial features  $x_{s1}$  and  $x_{s2}$  of the two regions  $b_1$  and  $b_2$  extracted in the same way as in localization module (we empirically find that adding visual features of  $b_1$  and  $b_2$  leads to no noticeable performance boost while slowing training significantly). Then  $x_{s1}$  and  $x_{s2}$  are concatenated as  $x_{s1,s2} = [x_{s1} \ x_{s2}]$ , and then processed in a similar way as in localization module to obtain  $s_{rel}$ , as shown below:

$$\tilde{x}_{s1,s2} = W_{s1,s2}x_{s1,s2} + b_{s1,s2} \quad (2.11)$$

$$z_{rel} = \tilde{x}_{s1,s2} \odot q_{rel} \quad (2.12)$$

$$\hat{z}_{rel} = z_{rel} / \|z_{rel}\|_2 \quad (2.13)$$

$$s_{rel} = w_{rel}^T \hat{z}_{rel} + b_{rel}. \quad (2.14)$$

The parameters in  $\Theta_{rel}$  are  $(W_{s1,s2}, b_{s1,s2}, w_{rel}, b_{rel})$ .

## End-to-end learning

During training, for an image  $I$ , a referring expression  $Q$  and a set of candidate regions  $B$  extracted from  $I$ , if the ground-truth regions  $b_{subj-gt}$  of the subject entity and  $b_{obj-gt}$  of the object entity are both available, then we can optimize the pairwise score  $s_{pair}$  in Eqn. 2.1 with strong supervision using softmax loss  $Loss_{strong}$ .

$$Loss_{strong} = -\log \left( \frac{\exp(s_{pair}(b_{subj-gt}, b_{obj-gt}))}{\sum_{(b_i, b_j) \in B \times B} \exp(s_{pair}(b_i, b_j))} \right) \quad (2.15)$$

However, it is often hard to obtain ground-truth regions for both subject entity and object entity. For referring expressions like ‘‘a red vase on top of the table’’, often there is only a ground-truth

Method	Accuracy
baseline (loc module)	46.27%
our full model	99.99%

Table 2.1: Accuracy of our model and the baseline on the synthetic shape dataset. See Sec. 2.4 for details.

bounding box annotation  $b_1$  for the subject (vase) in the expression, but no bounding box annotation  $b_2$  for the object (table), so one cannot directly optimize the pairwise score  $s_{pair}(b_1, b_2)$ . To address this issue, we treat the object region  $b_2$  as a latent variable, and optimize the unary score  $s_{subj}(b_1)$  in Eqn. 2.2. Since  $s_{subj}(b_1)$  is obtained by maximizing over all possible region  $b_2 \in B$  in  $s_{pair}(b_1, b_2)$ , this can be regarded as a weakly supervised Multiple Instance Learning (MIL) approach similar to [127]. The unary score  $s_{subj}$  can be optimized with weak supervision using softmax loss  $Loss_{weak}$ .

$$Loss_{weak} = -\log \left( \frac{\exp(s_{subj}(b_{subj\_gt}))}{\sum_{b_i \in B} \exp(s_{subj}(b_i))} \right) \quad (2.16)$$

The whole system is trained end-to-end with backpropagation, and parameters in localization module, relationship module, language representation and visual feature extraction (convolutional neural network) are jointly optimized. Our model is implemented using TensorFlow [1] and our code is available at <http://ronghanghu.com/cmn>.

## 2.4 Experiments

We first evaluate our model on a synthetic dataset to verify its ability to handle inter-object relationships in referring expressions. Next we apply our method to real images and expressions in the Visual Genome dataset [90] and Google-Ref dataset [117]. Since the task of answering pointing questions in visual question answering is similar to grounding referring expressions, we also evaluate our model on the pointing questions in the Visual-7W dataset [210].

### Analysis on a synthetic dataset

Inspired by [11], we first perform a simulation experiment on a synthetic shape dataset. The dataset consists of 30000 images with simple circles, squares and triangles of different sizes and colors on a 5 by 5 grid, and referring expressions constructed using a template of the form [subj] [relationship] [obj], where [subj] and [obj] involve both shape classes and attributes and [relationship] is some spatial relationships such as “above”. The task is to localize the corresponding shape region described by the expression on the 5 by 5 grid. Figure 2.3 (a) shows an example in this dataset with the synthetic expression “the green square right of a red circle”. In the synthesizing procedure, we make sure that the shape region being referred to cannot be inferred

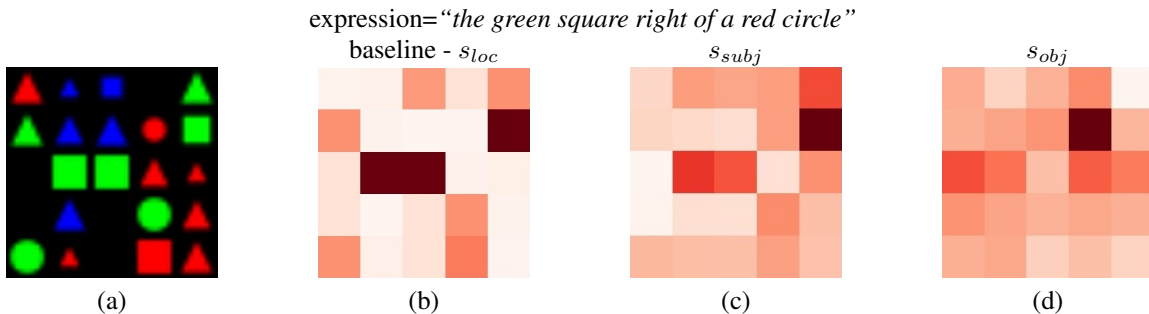


Figure 2.3: For the image in (a) and the expression “the green square right of a red circle”, (b) baseline scores on each location on the 5 by 5 grid using localization module only (darker is higher), and (c, d) scores  $s_{subj}$  and  $s_{obj}$  using our full model.  $s_{subj}$  is highest on the exact green square that is on the right of a red circle, and  $s_{obj}$  is highest on this red circle.

simply from  $[subj]$  as there will be multiple matching regions, and the relationship with another region described by  $[obj]$  has to be taken into consideration.

On this dataset, we train our model with weak supervision by Eqn. 2.16 using the ground-truth subject region  $b_{subj-gt}$  of the subject shape described in the expression. Here the candidate region set  $B$  are the 25 possible locations on the 5 by 5 grid, and visual features are extracted from the corresponding cropped image region with a VGG-16 network [156] pretrained on ImageNET classification. As a comparison, we also train a baseline model using only the localization module, with a softmax loss on its output  $s_{loc}$  in Eqn. 2.10 over all 25 locations on the grid, and language representation  $q_{loc}$  obtained by scanning through the word embedding sequence with a single LSTM network and taking the hidden state at the last time step same as in [146, 73]. This baseline method resembles the supervised version of GroundeR [146], and the main difference between this baseline and our model is that the baseline only looks at a region’s appearance and spatial property but ignores pairwise relationship with other regions.

We evaluate with the accuracy on whether the predicted subject region  $b_{subj}^*$  matches the ground-truth region  $b_{subj-gt}$ . Table 2.1 shows the results on this dataset, where our model trained with weak supervision (the same as the supervision given to baseline) achieves nearly perfect accuracy—significantly outperforming the baseline using a localization module only. Figure 2.3 shows an example, where the baseline can localize green squares but fails to distinguish the exact green square right of a red circle, while our model successfully finds the subject-object pair, although it has never seen the ground-truth location for the object entity during training.

## Localizing relationships in Visual Genome

We also evaluate our method on the Visual Genome dataset [90], which contains relationship expressions annotated over pairs of objects, such as “computer on top of table” and “person wearing shirt”.

On the relationship annotations in Visual Genome, given an image and an expression like “man wearing hat”, we evaluate our method in two test scenarios: retrieving the *subject* region (“man”)

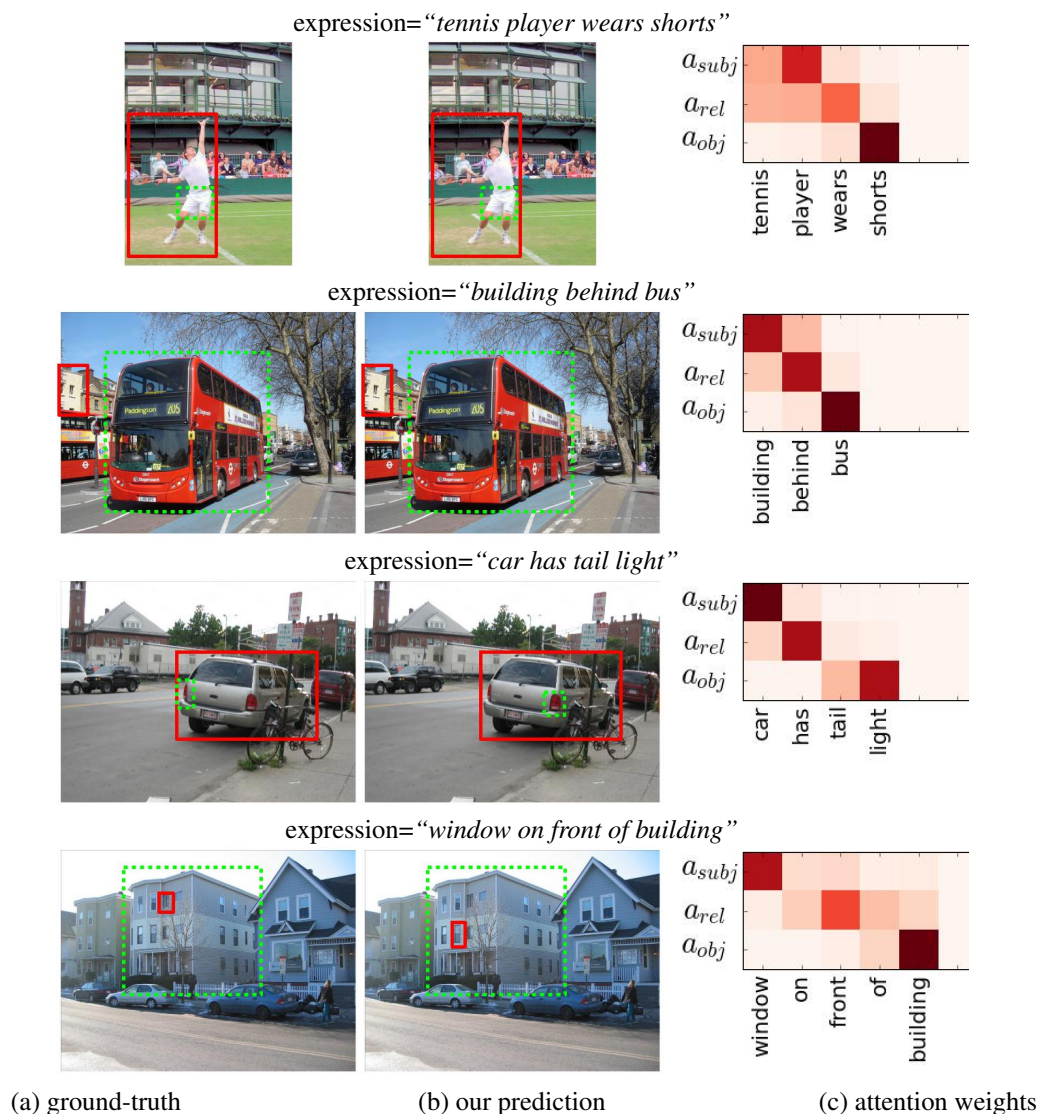


Figure 2.4: Visualization of grounded relationship expressions in the Visual Genome dataset, trained with weak supervision (subject-GT). (a, b) ground-truth region pairs and our predicted region pairs respectively (subject in red solid box and object in green dashed box). (c) attention weights in Eqn. 2.4–2.6 for subject, relationship and object (darker is higher).

and retrieving the *subject-object pair* (both “man” and “hat”). In our experiment, we take the bounding boxes of all the annotated entities in each image (around 35 per image) as candidate region set  $B$  at both training and test time, and extract visual features for each region from fc7 output of a Faster R-CNN VGG-16 network [144] pretrained on MSCOCO detection dataset [101]. We use the same training, validation and test split as in [80].

Since there are ground-truth annotations for both subject region and object region in this dataset, we experiment with two training supervision settings: (1) *weak supervision* by only providing the

Method	training supervision	P@1-subj	P@1-pair
baseline	subject-GT	41.20%	-
baseline	subject-object-GT	-	23.37%
our full model	subject-GT	43.81%	26.56%
our full model	subject-object-GT	<b>44.24%</b>	<b>28.52%</b>

Table 2.2: Performance of our model on relationship expressions in Visual Genome dataset. See Sec. 2.4 for details.

ground-truth region of the subject entity at training time (**subject-GT** in Table 2.2) and optimizing unary subject score  $s_{subj}$  with Eqn. 2.16 and (2) *strong supervision* by providing the ground-truth region pair of both subject and object entities at training time (**subject-object-GT** in Table 2.2) and optimizing pairwise score  $s_{pair}$  with Eqn. 2.15.

Similar to the experiment on the synthetic dataset in Sec. 2.4, we also train a baseline model that only looks at local appearance and spatial properties but ignores pairwise relationships. For the first evaluation scenario of retrieving the subject region, we train a baseline model using a localization module only by optimizing its output  $s_{loc}$  for ground-truth subject region with softmax loss (the same training supervision as subject-GT). For the second scenario of retrieving the subject-object pair, we train two such baseline models optimized with subject ground-truth and object ground-truth respectively, to localize of the subject region and object region separately with each model and at test time combine the predicted subject region and predicted object region from each model be the subject-object pair (same training supervision as subject-object-GT).

We evaluate with top-1 precision (P@1), which is the percentage of test instances where the top scoring prediction matches the ground-truth in each image (P@1-subj for predicted subject regions matching subject ground-truth in the first scenario, and P@1-pair for predicted subject and object regions both matching the ground-truth in the second scenario). The results are summarized in Table 2.2, where it can be seen that our full model outperforms the baseline using only localization modules in both evaluation scenarios. Note that in the second evaluation scenario of retrieving subject-object pairs, our weakly supervised model still outperforms the baseline trained with strong supervision.

Figure 2.4 shows some examples of our model trained with weak supervision (subject-GT) and attention weights in Eqn. 2.4–2.6. It can be seen that even with weak supervision, our model still generates reasonable attention weights over words for subject, relationship and object.

## Grounding referring expressions in images

We apply our model to the Google-Ref dataset [117], a benchmark dataset for grounding referring expressions. As this dataset does not explicitly contain subject-object pair annotation for the referring expressions, we train our model with weak supervision (Eqn. 2.16) by optimizing the subject score  $s_{subj}$  using the expression-level region ground-truth. The candidate bounding

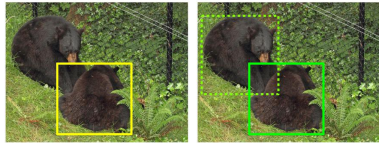

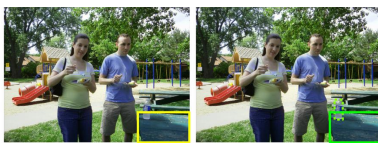



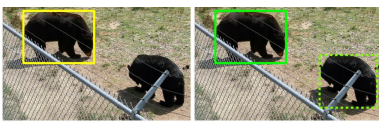

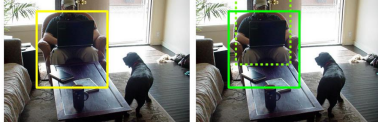
ground-truth	our prediction	ground-truth	our prediction	ground-truth	our prediction
expression= <i>“a bear lying to the right of another bear”</i>		expression= <i>“man in sunglasses walking towards two talking men”</i>		expression= <i>“a picnic table that has a bottle of water sitting on it”</i>	
correct		correct		correct	
expression= <i>“woman in a cream colored wedding dress cutting cake”</i>		expression= <i>“a man going before a lady carrying a cellphone”</i>		expression= <i>“pizza slice not eaten”</i>	
correct		correct		incorrect	
expression= <i>“a full grown brown bear near a young bear”</i>		expression= <i>“black dog standing on all four legs”</i>		expression= <i>“chair being sat in by a man”</i>	
correct		incorrect		correct	

Figure 2.5: Examples of referring expressions in the Google-Ref dataset. The left column shows the ground-truth region and the right column shows the grounded subject region (our prediction) in solid box and the grounded object region in dashed box. A prediction is labeled as correct if the predicted subject region matches the ground-truth region.

box set  $B$  at both training and test time are all the annotated entities in the image (which is the “Ground-Truth” evaluation setting in [117]). As in Sec. 2.4, fc7 output of a MSCOCO-pretrained Faster R-CNN VGG-16 network is used for visual feature extraction. Similar to Sec. 2.4, we also train a GrondeR-like [146] baseline model with localization module which looks only at a region’s local features.

In addition, instead of learning a linguistic analysis end-to-end as in Sec. 2.3, we also experiment with parsing the expression using the Stanford Parser [209, 115]. An expression is parsed into subject, relationship and object component according to the constituency tree, and the components are encoded into vectors  $q_{subj}$ ,  $q_{rel}$  and  $q_{obj}$  using three separate LSTM encoders, similar to the baseline and [146].

Following [117], we evaluate on this dataset using the top-1 precision (P@1) metric, which is the fraction of the highest scoring subject region matching the ground-truth for the expression. Table 2.3 shows the performance of our model, baseline model and previous work. Note that all the methods are trained with the same weak supervision (only a ground-truth subject region). It can be



Method	P@1
Mao <i>et al.</i> [117]	60.7%
Yu <i>et al.</i> [198]	64.0%
Nagaraja <i>et al.</i> [127]	68.4%
baseline (loc module)	66.5%
our model (w/ external parser)	53.5%
our full model	<b>69.3%</b>

Table 2.3: Top-1 precision of our model and previous methods on Google-Ref dataset. See Sec. 2.4 for details.

Method	Accuracy
Zhu <i>et al.</i> [210]	56.10%
baseline (loc module)	71.61%
our model (w/ external parser)	61.66%
our full model	<b>72.53%</b>

Table 2.4: Accuracy of our model and previous methods on the pointing questions in Visual-7W dataset. See Sec. 2.4 for details.

seen that by incorporating inter-object relationships, our full model outperforms the baseline using only localization modules, and works better than previous state-of-the-art methods.

Additionally, replacing the learned expression parsing and language representation in Sec. 2.3 with an external parser (“our model w/ external parser” in Table 2.3) leads to a significant performance drop. We find that this is mainly because existing parsers are not specifically tuned for the referring expression task—as noted in Sec. 2.3, expressions like *chair on the left of the table* are parsed as *(chair, on, the left of the table)* rather than the desired triplet *(chair, on the left of, the table)*. In our full model, the language representation is end-to-end optimized with other parts, while it is hard to jointly optimize an external language parser like [209] for this task.

Figure 2.5 shows some example results on this dataset. It can be seen that although weakly supervised, our model not only grounds the subject region correctly (solid box), but also finds reasonable regions (dashed box) for the object entity.

## Answering pointing questions in Visual-7W

Finally, we evaluate our method on the multiple choice pointing questions (*i.e.* “which” questions) in visual question answering on the Visual-7W dataset [210]. Given an image and a question like “which tomato slice is under the knife”, the task is to select the corresponding region from a few choice regions (4 choices in this dataset) as answer. Since this task is closely related to grounding

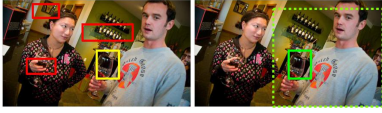


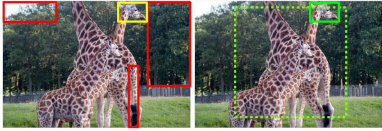
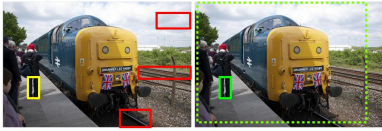
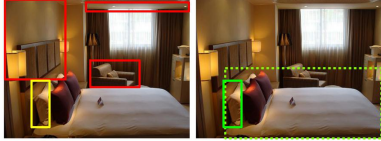


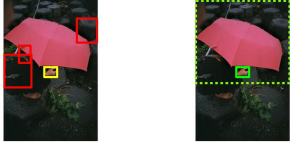
ground-truth	our prediction	ground-truth	our prediction	ground-truth	our prediction
question="Which wine glass is in the man's hand?" 	correct	question="Which person is wearing a helmet?" 	correct	question="Which mouse is on a pad by computer?" 	correct
question="Which head is that of an adult giraffe?" 	correct	question="Which pants belong to the man closest to the train?" 	correct	question="Which white pillow is leftmost on the bed?" 	correct
question="Which red shape is on a large white sign?" 	correct	question="Which is not a pair of a living canine?" 	incorrect	question="Which hand can be seen from under the umbrella?" 	correct

Figure 2.6: Example pointing questions in the Visual-7W dataset. The left column shows the 4 multiple choices (ground-truth answer in yellow) and the right column shows the grounded subject region (predicted answer) in solid box and the grounded object region in dashed box. A prediction is labeled as correct if the predicted subject region matches the ground-truth region.

referring expressions, our model can be trained in the same way as in Sec. 2.4 to score each choice region using subject score  $s_{subj}$  and pick the highest scoring choice as answer.

As before, we train our model with weak supervision through Eqn. 2.16 and use a MSCOCO-pretrained Faster R-CNN VGG-16 network for visual feature extraction. Here we use two different candidate bounding box sets  $B_{subj}$  and  $B_{obj}$  of the subject regions (the choices) and the object regions, where  $B_{subj}$  is the 4 choice bounding boxes, and  $B_{obj}$  is the set of 300 proposal bounding boxes extracted using RPN in Faster R-CNN [144]. Similar to Sec. 2.4, we also train a baseline model using only a localization module to score each choice based only on its local appearance and spatial properties, and a truncated model that uses the Stanford parser [209, 115] for expression parsing and language representation.

The results are shown in Table 2.4. It can be seen that our full model outperforms the baseline and the truncated model with an external parser, and achieves much higher accuracy than previous work [210]. Figure 2.6 shows some question answering examples on this dataset.

## 2.5 Discussion

We have proposed Compositional Modular Networks, a novel end-to-end trainable model for handling relationships in referring expressions. Our model learns to parse input expressions with soft attention, and incorporates two types of modules that consider a region’s local features and pairwise interaction between regions respectively. The model induces intuitive linguistic and visual analyses of referring expressions from only weak supervision, and experimental results demonstrate that our approach outperforms both natural baselines and state-of-the-art methods on multiple datasets.

## Chapter 3

# End-to-End Module Networks for Compositional VQA

### 3.1 Problem Statement

Visual Question Answering (VQA) requires joint comprehension of images and text. This comprehension often depends on compositional reasoning, for example locating multiple objects in a scene and inspecting their properties or comparing them to one another (Figure 3.1). While conventional deep networks have shown promising VQA performance [51], there is limited evidence that they are capable of explicit compositional reasoning [78]. Much of the success of state-of-the-art approaches to VQA instead comes from their ability to discover statistical biases in the data distribution [58]. And to the extent that such approaches are capable of more sophisticated reasoning, their monolithic structure makes these behaviors difficult to understand and explain. Additionally, they rely on the same non-modular network structure for all input questions.

In this chapter, we propose *End-to-End Module Networks (N2NMNs)*: a class of models capable of predicting novel modular network architectures directly from textual input and applying them to images in order to solve question answering tasks. In contrast to previous work, our approach learns to both parse the language into linguistic structures *and* compose them into appropriate layouts.

The present work synthesizes and extends two recent modular architectures for visual problem solving. Standard neural module networks (NMNs) [11] already provide a technique for constructing dynamic network structures from collections of composable modules. However, previous work relies on an external parser to process input text and obtain the module layout. This is a serious limitation, because off-the-shelf language parsers are not designed for language and vision tasks and must therefore be modified using handcrafted rules that often fail to predict valid layouts [78]. Meanwhile, the compositional modular network [72] proposed for grounding referring expressions in images does not need a parser, but is restricted to a fixed (*subject, relationship, object*) structure. None of the existing methods can learn to predict a suitable structure for every input in an end-to-end manner.

Our contributions are 1) a method for learning a layout policy that dynamically predicts a

There is a shiny object that is right of the gray metallic cylinder;  
does it have the same size as the large rubber sphere?

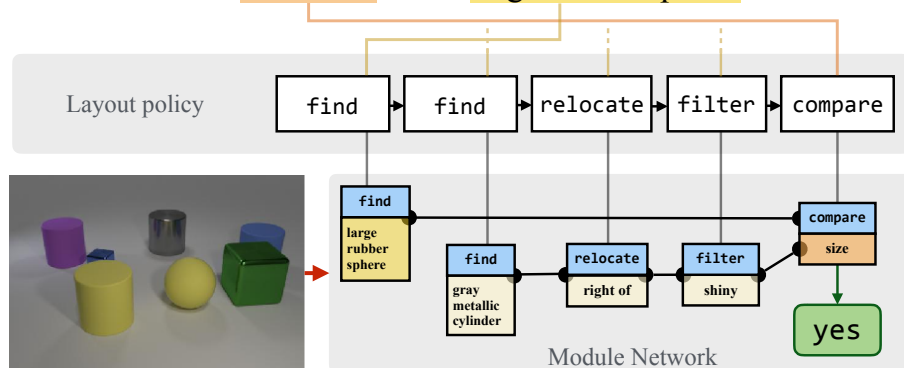


Figure 3.1: For each instance, our model predicts a computational expression and a sequence of attentive module parameterizations. It uses these to assemble a concrete network architecture, and then executes the assembled neural module network to output an answer for visual question answering. (The example shows a real structure predicted by our model, with text attention maps simplified for clarity.)

network structure for each instance, without the aid of external linguistic resources at test time and 2) a new module parameterization that uses a soft attention over question words rather than hard-coded word assignments. Experiments show that our model is capable of directly predicting expert-provided network layouts with near-perfect accuracy, and even improving on expert-designed networks after a period of exploration. We obtain state-of-the-art results on the recently released CLEVR dataset by a wide margin.

## 3.2 Related Work

**Neural module networks.** The recently proposed neural module network (NMN) architecture [11]—a general class of recursive neural networks [160]—provides a framework for constructing deep networks with dynamic computational structure. In an NMN model, every input is associated with a *layout* that provides a template for assembling an instance-specific network from a collection of shallow network fragments called *modules*. These modules can be jointly trained across multiple structures to provide reusable, compositional behaviors. Existing work on NMNs has focused on natural language question answering applications, in which a linguistic analysis of the question is used to generate the layout, and the resulting network applied to some world representation (either an image or knowledge base) to produce an answer. The earliest work on NMNs [11] used fixed rule-based layouts generated from dependency parses [209]. Later work on “dynamic” module networks (D-NMNs) [10] incorporated a limited form of layout prediction by learning to rerank a list of three to ten candidates, again generated by rearranging modules predicted by a dependency

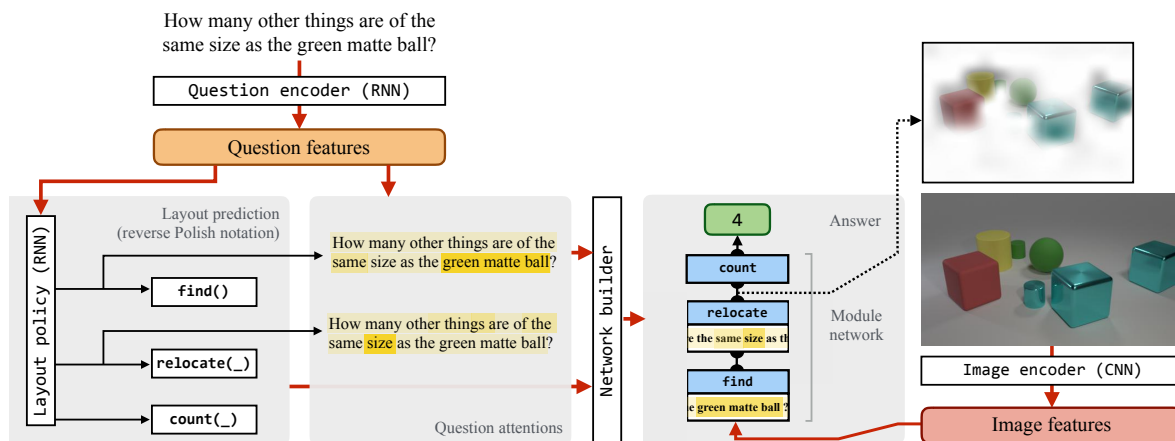


Figure 3.2: Model overview. Our approach first computes a deep representation of the question, and uses this as an input to a layout-prediction policy implemented with a recurrent neural network. This policy emits both a sequence of *structural* actions, specifying a template for a modular neural network in reverse Polish notation, and a sequence of *attentive* actions, extracting parameters for these neural modules from the input sentence. These two sequences are passed to a *network builder*, which dynamically instantiates an appropriate neural network and applies it to the input image to obtain an answer.

parse. Like D-NMNs, the present work attempts to learn an optimal layout predictor jointly with module behaviors themselves. Here, however, we tackle a considerably more challenging prediction problem: our approach learns to optimize over the full space of network layouts rather than acting as a reranker, and requires no parser at evaluation time.

We additionally modify the representation of the assembled module networks themselves: where [11] and [10] parameterized individual modules with a fixed embedding supplied by the parser, here we *predict* these parameters jointly with network structures using a soft attention mechanism. This parameterization resembles the approach used in the “compositional modular network” architecture [72] for grounding referential expressions. However, the model proposed in [72] is restricted to a fixed layout structure of (subject, relationship, object) for every referential expression, and includes no structure search.

**Learning network architectures.** More generally than these dynamic / modular approaches, a long line of research focuses on generic methods for automatically discovering neural network architectures from data. Past work includes techniques for optimizing over the space of architectures using evolutionary algorithms [188, 47], Bayesian methods [21], and reinforcement learning [212]. The last of these is most closely related to our approach in this chapter: both learn a controller RNN to output a network structure, train a neural network with the generated structure, and use the accuracy of the generated network to optimize the controller RNN. A key difference between [212] and the layout policy optimization in our work is that [212] learns a *fixed* layout (network

architecture) that is applied to every instance, while our model learns a layout policy that *dynamically* predicts a specific layout tailored to each individual input example.

**Visual question answering.** The visual question answering task [113] is generally motivated as a test to measure the capacity of deep models to *reason* about linguistic and visual inputs jointly [113]. Recent years have seen a proliferation of datasets [113, 12] and approaches, including models based on differentiable memory [192, 189], dynamic prediction of question-specific computations [128, 10], and core improvements to the implementation of the multi-modal representation and attention mechanism [51, 110]. Together, these approaches have produced substantial gains over the initial baseline results published with the first VQA datasets.

It has been less clear, however, that these improvements correspond to an improvement in the *reasoning* abilities of models. Recent work has found that it is possible to do quite well on many visual QA problems by simply memorizing statistics about question / answer pairs [58] (suggesting that limited visual reasoning is involved), and that models with bag-of-words text representations perform competitively against more sophisticated approaches [77] (suggesting that limited linguistic compositionality is involved). To address this concern, newer visual question answering datasets have focused on exploring specific phenomena in compositionality and generalization; examples include the SHAPES dataset [11], the VQAv2 dataset [58], and the CLEVR dataset [78]. The last of these appears to present the greatest challenges to standard VQA approaches and the hardest reasoning problems in general.

Most previous work on this task other than NMN uses a fixed inference structure to answer every question. However, the optimal reasoning procedure may vary greatly from question to question, so it is desirable to have inference structures that are specific to the input question. Concurrent with our work, [79] proposes a similar model to ours. Our model is different from [79] in that we use a set of specialized modules with soft attention mechanism to provide textual parameters for each module, while [79] uses a generic module implementation with textual parameters hard-coded in module instantiation.

### 3.3 End-to-End Module Networks (N2NMNs)

We propose End-to-End Module Networks (N2NMNs) to address compositionality in visual reasoning tasks. Our model consists of two main components: a set of co-attentive neural modules that provide parameterized functions for solving sub-tasks, and a layout policy to predict a question-specific layout from which a neural network is dynamically assembled. An overview of our model is shown in Figure 3.2.

Given a question, such as *how many other things are there of the same size as the matte ball?*, our layout policy first predicts a coarse functional expression like `count(relocate(find()))` that describes the structure of the desired computation, Next, some subset of function applications within this expression (here `relocate` and `find`) receive parameter vectors predicted from text (here perhaps vector representations of *matte ball* and *size*, respectively). Then a network is assembled with the modules according to this layout expression to output an answer.

Module name	Att-inputs	Features	Output	Implementation details
find	(none)	$x_{vis}, x_{txt}$	att	$a_{out} = \text{conv}_2(\text{conv}_1(x_{vis}) \odot W x_{txt})$
relocate	$a$	$x_{vis}, x_{txt}$	att	$a_{out} = \text{conv}_2(\text{conv}_1(x_{vis}) \odot W_1 \text{sum}(a \odot x_{vis}) \odot W_2 x_{txt})$
and	$a_1, a_2$	(none)	att	$a_{out} = \text{minimum}(a_1, a_2)$
or	$a_1, a_2$	(none)	att	$a_{out} = \text{maximum}(a_1, a_2)$
filter	$a$	$x_{vis}, x_{txt}$	att	$a_{out} = \text{and}(a, \text{find}[x_{vis}, x_{txt}]())$ , i.e. reusing find and and
[exist, count]	$a$	(none)	ans	$y = W^T \text{vec}(a)$
describe	$a$	$x_{vis}, x_{txt}$	ans	$y = W_1^T (W_2 \text{sum}(a \odot x_{vis}) \odot W_3 x_{txt})$
[eq_count, more, less]	$a_1, a_2$	(none)	ans	$y = W_1^T \text{vec}(a_1) + W_2^T \text{vec}(a_2)$
compare	$a_1, a_2$	$x_{vis}, x_{txt}$	ans	$y = W_1^T (W_2 \text{sum}(a_1 \odot x_{vis}) \odot W_3 \text{sum}(a_2 \odot x_{vis}) \odot W_4 x_{txt})$

Table 3.1: The full list of neural modules in our model. Each module takes 0, 1 or 2 attention maps (and also visual and textual features) as input, and outputs either an attention map  $a_{out}$  or a score vector  $y$  for all possible answers. The operator  $\odot$  is element-wise multiplication, and sum is summing the result over spatial dimensions. The vec operation is flattening an attention map into a vector, and adding two extra dimensions: the max and min over attention map.

We describe the implementation details of each neural module  $f_m$  in Sec. 3.3, and our layout policy in Sec. 3.3. In Sec. 3.3, we present a reinforcement learning approach, i.e. jointly optimize the neural modules and the layout policy.

## Attentional neural modules

Our model involves a set of neural modules that can be dynamically assembled into a neural network. A neural module  $m$  is a parameterized function  $y = f_m(a_1, a_2, \dots; x_{vis}, x_{txt}, \theta_m)$  that takes zero, one or multiple tensors  $a_1, a_2, \dots$  as input, using its internal parameter  $\theta_m$  and features  $x_{vis}$  and  $x_{txt}$  from the image and question to perform some computation on the input, and outputs a tensor  $y$ . In our implementation, each input tensor  $a_i$  is an image attention map over the convolutional image feature grid, and the output tensor  $y$  is either an image attention map, or a probability distribution over possible answers.

Table 3.1 shows the set of modules in our N2NMNs model, along with their implementation details. We assign a name to each module according to its input and output type and potential functionality, such as `find` or `describe`. However, we note that each module is in itself merely a function with parameters, and we do not restrict its behavior during training. In addition to the input tensors (that are outputs from other modules), a module  $m$  can also use two additional feature vectors  $x_{vis}$  and  $x_{txt}^{(m)}$ , where  $x_{vis}$  is the spatial feature map extracted from the image with a convolutional neural network, and  $x_{txt}^{(m)}$  is a textual vector for this module  $m$  that contains information extracted from the question  $q$ . In addition, `and` and `or` take two image attention maps as inputs, and return their intersection or union respectively.

In Table 3.1, the `find` module outputs an attention map over the image and can be potentially used to localize some objects or attributes. The `relocate` module transforms the input image attention map and outputs a new attention map, which can be useful for spatial or relationship inference. Also the `filter` module reuses `find` and `and`, and can be used to simplify the layout expression. We use two classes of modules to infer an answer from a single attention map: the first



class has the instances `exist` and `count` (instances share the same structure, but have different parameters). They are used for simple inference by looking only at the attention map. The second class, `describe`, is for more complex inference where visual appearance is needed. Similarly, for pairwise comparison over two attention maps we also have two classes of available modules with (`compare`) or without (`eq_count`, `more`, `less`) access to visual features.

The biggest difference in module implementation between this work and [11] is the textual component. Hard-coded textual components are used in [11], for example, `describe[‘shape’]` and `describe[‘where’]` are two different instantiations that have different parameters. In contrast, our model obtains the textual input using soft attention over question words similar to [72]. For each module  $m$ , we predict an attention map  $\alpha_i^{(m)}$  over the  $T$  question words (in Sec. 3.3), and obtain the textual feature  $x_{txt}$  for each module:

$$x_{txt}^{(m)} = \sum_{i=1}^T \alpha_i^{(m)} w_i \quad (3.1)$$

where  $w_i$  is the word embedding vector for word  $i$  in the question. At runtime, the modules can be assembled into a network according to a layout  $l$ , which is a computation expression consisting of modules, such as  $f_{m2}(f_{m4}(f_{m1}), f_{m3}(f_{m1}, f_{m1}))$ , where each of  $f_{m1}, \dots, f_{m4}$  is one of the modules in Table 3.1.

## Layout policy with sequence-to-sequence RNN

We would like to predict the most suitable reasoning structure tailored to each question. For an input question  $q$  such as *What object is next to the table?*, our layout policy outputs a probability distribution  $p(l|q)$ , and we can sample from  $p(l|q)$  to obtain high probability layout  $l$  such as `describe(relocate(find()))` that are effective for answering the question  $q$ . Then, a neural network is assembled according to the predicted layout  $l$  to output an answer.

Unlike in [10] where the layout search space is restricted to a few parser candidates, in this work, we search over a much larger layout space: in our model, the layout policy  $p(l|q; \theta_{layout})$  predicts a distribution over the space of *all possible layouts*. Every possible layout  $l$  is an expression that consists of neural modules, such as  $f_{m2}(f_{m1}, f_{m3}(f_{m1}, f_{m1}))$ , and can be represented as a syntax tree. So each layout expression can be mapped one-to-one into a linearized sequence  $l = \{m^{(t)}\}$  using Reverse Polish Notation [28] (the post-order traversal over the syntax tree). Figure 3.3 shows an example for an expression and its linearized module token sequence.

After linearizing each layout  $l$  into a sequence of module tokens  $\{m^{(t)}\}$ , the layout prediction problem turns into a sequence-to-sequence learning problem from questions to module tokens. We address this problem using the attentional Recurrent Neural Network [17]. First, we embed every word  $i$  in the question into a vector  $w_i$  (also embedding all module tokens similarly), and use a multi-layer LSTM network as the encoder of the input question. For a question  $q$  with  $T$  words, the encoder LSTM outputs a length- $T$  sequence  $[h_1, h_2, \dots, h_T]$ . The decoder is a LSTM network that has the same structure as the encoder but different parameters. Similar to [17], at each time step in the decoder LSTM, a soft attention map over the input sequence is predicted. At decoder time-step

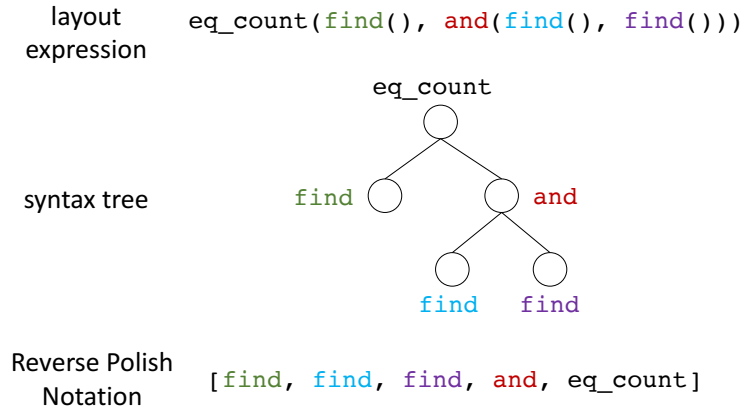


Figure 3.3: An example showing how an arbitrary layout expression can be linearized as a sequence of module tokens.

$t$ , the attention weights  $\alpha_{ti}$  of input word at position  $i \in \{1, \dots, T\}$  are predicted as

$$u_{ti} = v^T \tanh(W_1 h_i + W_2 h_t) \tag{3.2}$$

$$\alpha_{ti} = \frac{\exp(u_{ti})}{\sum_{j=1}^T \exp(u_{tj})} \tag{3.3}$$

where  $h_i$  and  $h_t$  are LSTM outputs at encoder time-step  $i$  and decoder time-step  $t$ , respectively, and  $v$ ,  $W_1$  and  $W_2$  are model parameters to be learned from data. Then a context vector  $c_t$  is obtained as  $\sum_{i=1}^T \alpha_{ti} h_i$ , and the probability for the next module token  $m^{(t)}$  is predicted from  $h_t$  and  $c_t$  as  $p(m^{(t)} | m^{(1)}, \dots, m^{(t-1)}, q) = \text{softmax}(W_3 h_t + W_4 c_t)$ . We sample from  $p(m^{(t)} | m^{(1)}, \dots, m^{(t-1)}, q)$  to discretely get the next token  $m^{(t)}$ , and also construct its textual input  $x_{txt}^{(t)}$  according to Eqn. 3.1 using the attention weights  $\alpha_{ti}$  in Eqn. 3.3. The probability of a layout  $l$  is  $p(l|q) = \prod_{m^{(t)} \in l} p(m^{(t)} | m^{(1)}, \dots, m^{(t-1)}, q)$ . At test time, we deterministically predict a maximum-probability layout  $l$  from  $p(l|q)$  using beam search, and assemble a neural network according to  $l$  to output an answer for the question.

### End-to-end training

During training, we jointly learn the layout policy  $p(l|q)$  and the parameters in each neural module, and minimize the expected loss from the layout policy. Let  $\theta$  be all the parameters in our model. Suppose we obtain a layout  $l$  sampled from  $p(l|q; \theta)$  and receive a final question answering loss  $\tilde{L}(\theta, l; q, I)$  on question  $q$  and image  $I$  after predicting an answer using the network assembled with  $l$ . Our training loss function  $L(\theta)$  is as follows.

$$L(\theta) = E_{l \sim p(l|q; \theta)} [\tilde{L}(\theta, l; q, I)] \tag{3.4}$$

where we use the softmax loss over the output answer scores as  $\tilde{L}(\theta, l; q, I)$  in our implementation.

The loss function in  $L(\theta)$  is not fully differentiable since the layout  $l$  is discrete, so one cannot train it with full back-propagation. We optimize  $L(\theta)$  using back-propagation for differentiable parts, and policy gradient method in reinforcement learning for non-differentiable part. The gradient  $\nabla_{\theta}L$  of the loss  $L(\theta)$  is  $\nabla_{\theta}L = E_{l \sim p(l|q; \theta)} \left[ \tilde{L}(\theta, l) \nabla_{\theta} \log p(l|q; \theta) + \nabla_{\theta} \tilde{L}(\theta, l) \right]$  which can be estimated using Monte-Carlo sampling as

$$\nabla_{\theta}L \approx \frac{1}{M} \sum_{m=1}^M \left( \tilde{L}(\theta, l_m) \nabla_{\theta} \log p(l_m|q; \theta) + \nabla_{\theta} \tilde{L}(\theta, l_m) \right) \quad (3.5)$$

where both  $\log p(l_m|q; \theta)$  and  $\tilde{L}(\theta, l_m)$  are fully differentiable so the above equation can be computed with back-propagation, allowing end-to-end training for the entire model. We use  $M = 1$  in our implementation.

To reduce the variance of the estimated gradient, we introduce a simple baseline  $b$ , by replacing  $\tilde{L}(\theta, l_m)$  with  $\tilde{L}(\theta, l_m) - b$  in Eqn. 3.5, where  $b$  is implemented as an exponential moving average over the recent loss  $\tilde{L}(\theta, l_m)$ . We also use an entropy regularization  $\alpha = 0.005$  over the policy  $p(l|q)$  to encourage exploration through the layout space.

**Behavioral cloning from expert polices.** Optimizing the loss function in Eqn. 3.4 from scratch is a challenging reinforcement learning problem: one needs to simultaneously learn the parameters in the sequence-to-sequence RNN to optimize the layout policy and textual attention weights to construct the textual features  $x_{txt}^{(m)}$  for each module, and also the parameters in the neural modules. This is more challenging than a typical reinforcement learning scenario where one only needs to learn a policy.

On the other hand, the learning would be easier if we have some additional knowledge of module layout. While we do not want to restrict the layout search space to only a few candidates from the parser as in [10], we can treat these candidate layouts as an existing expert policy that can be used to provide additional supervision. More generally, if there is an expert policy  $p_e(l|q)$  that predicts a reasonable layout  $l$  from the question, we can first pre-train our model by behavioral cloning from  $p_e$ . This can be done by minimizing the KL-divergence  $D_{KL}(p_e||p)$  between the expert policy  $p_e$  and our layout policy  $p$ , and simultaneously minimizing the question answering loss  $\tilde{L}(\theta, l; q, I)$  with  $l$  obtained from  $p_e$ . This supervised behavioral cloning from the expert policy can provide a good set of initial parameters in our sequence-to-sequence RNN and each neural module. Note that the above behavioral cloning procedure is only done at training time to obtain a supervised initialization our model, and the expert policy is not used at test time.

The expert policy is not necessarily optimal, so behavioral cloning itself is not sufficient for learning the most suitable layout for each question. After learning a good initialization by cloning the expert policy, our model is further trained end-to-end with gradient  $\nabla_{\theta}L$  computed using Eqn. 3.5, where now the layout  $l$  is sampled from the layout policy  $p(l|q)$  in our model, and the expert policy  $p_e$  can be discarded.

We train our models using the Adam Optimizer [85] in all of our experiments. Our model is implemented using TensorFlow [1] and our code is available at <http://ronghanghu.com/n2nmn/>.

Method	Accuracy
NMN [11]	90.80%
ours - behavioral cloning from expert	100.00%
ours - policy search from scratch	96.19%

Table 3.2: Performance of our model on the SHAPES dataset. “ours - behavioral cloning from expert” corresponds to the supervised behavioral cloning from the expert policy  $p_e$ , and “ours - policy search from scratch” is directly optimizing the layout policy without utilizing any expert policy.

### 3.4 Experiments

We first analyze our model on a relatively small SHAPES dataset [11], and then apply our model to two large-scale datasets: CLEVR [78] and VQA [12].

#### Analysis on the SHAPES dataset

The SHAPES dataset for visual question answering (collected in [11]) consists of 15616 image-question pairs with 244 unique questions. Each image consists of shapes of different colors and sizes aligned on a 3 by 3 grid. Despite its relatively small size, effective reasoning is needed to successfully answer questions like “*is there a red triangle above a blue shape?*”. The dataset also provides a ground-truth parsing result for each question, which is used to train the NMN model in [11].

We analyze our method on the SHAPES dataset under two settings. In the first setting, we train our model using behavioral cloning from an expert layout policy as described in Sec. 3.3. An expert layout policy  $p_e$  is constructed by mapping the the ground-truth parsing for each question to a module layout in the same way as in [11]. Note that unlike [11], in this setting we only need to query the expert policy at training time. At test time, we obtain the layout  $l$  from the learned layout policy  $p(l|q)$  in our model, while NMN [11] still needs to access the ground-truth parsing at test time.

In the second setting, we train our model without using any expert policy, and directly perform policy optimization by minimizing the loss function  $L(\theta)$  in Eqn. 3.4 with gradient  $\nabla_{\theta}L$  in Eqn. 3.5. For both settings, we use a simple randomly initialized two-layer convolutional neural network to extract visual features from the image, trained together with other parts of our model.

The results are summarized in Table 3.2. In the first setting, we find that our model (“ours - behavioral cloning from expert”) already achieves 100% accuracy. While this shows that the expert policy constructed from ground-truth parsing is quite effective on this dataset, the higher performance of our model compared to the previous NMN [11] also suggests that our implementation of modules is more effective than [11], since the NMN is also trained with the same expert module layout obtained from the ground-truth parsing. In the second setting, our model achieves a good performance on this dataset by performing policy search from scratch without resorting to any expert policy. Figure 3.4 shows some examples of predicted layouts and answers on this dataset.

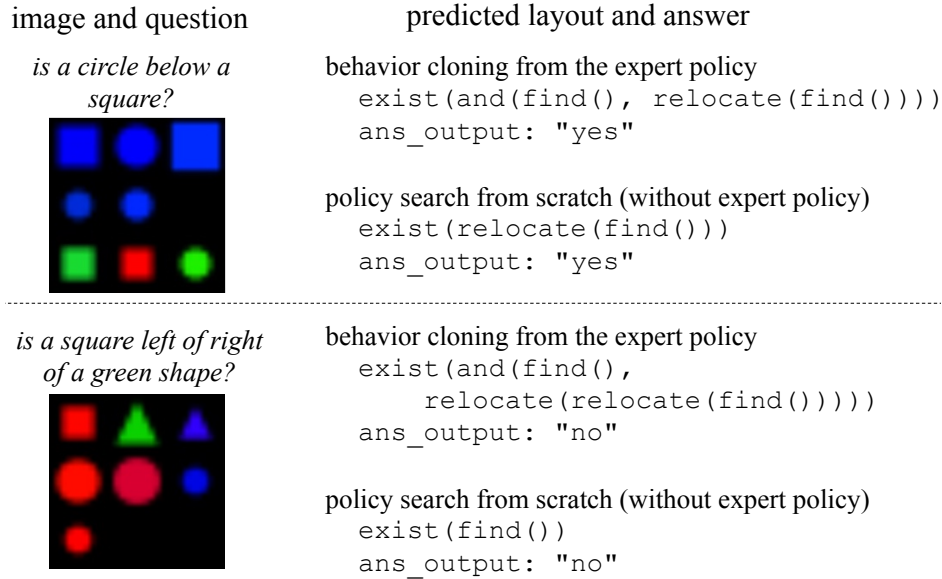


Figure 3.4: Examples of layouts predicted by our model on the SHAPES dataset, under two training settings (Sec. 3.4).

### Evaluation on the CLEVR dataset

We evaluate our End-to-End Module Networks on the recently proposed CLEVR dataset [78] with 100,000 images and 853,554 questions. The images in this dataset are photo-realistic rendered images with objects of different shapes, colors, materials and sizes and possible occlusions, and the questions in this dataset are synthesized with functional programs. Compared to other datasets for visual question answering such as [12], the CLEVR dataset focuses mostly on the *reasoning* ability. The questions in the CLEVR dataset have much longer question length, and require handling long and complex inference chains to get an answer, such as “*what size is the cylinder that is left of the brown metal thing that is left of the big sphere?*” and “*there is an object in front of the blue thing; does it have the same shape as the tiny cyan thing that is to the right of the gray metal ball?*”.

In our experiment on this dataset, we resize each image to  $480 \times 320$ , and extract a  $15 \times 10$  convolutional feature map from each image by forwarding the image through the VGG-16 network [156] trained on ImageNET classification, and take the 512-channel pool5 output. To help reason about spatial properties, we add two extra  $x = \frac{i}{15}$  and  $y = \frac{j}{10}$  dimensions to each location  $(i, j)$  on the feature map similar to [73], so the final visual feature  $x_{vis}$  on each image is a  $15 \times 10 \times 514$  tensor. Each question word is embedded to a 300-dimensional vector initialized from scratch. We use a batch size of 64 during training.

In the first training stage, behavioral cloning is used with an expert layout policy as described in Sec. 3.3. We construct an expert layout policy  $p_e$  that deterministically maps a question  $q$  into a layout  $l_e$  by converting the annotated functional programs in this dataset into a module layout with manually defined rules: first, the program chain is simplified to keep all intermediate computation in the image attention domain, and then each function type is mapped to a module in Table 3.1 that

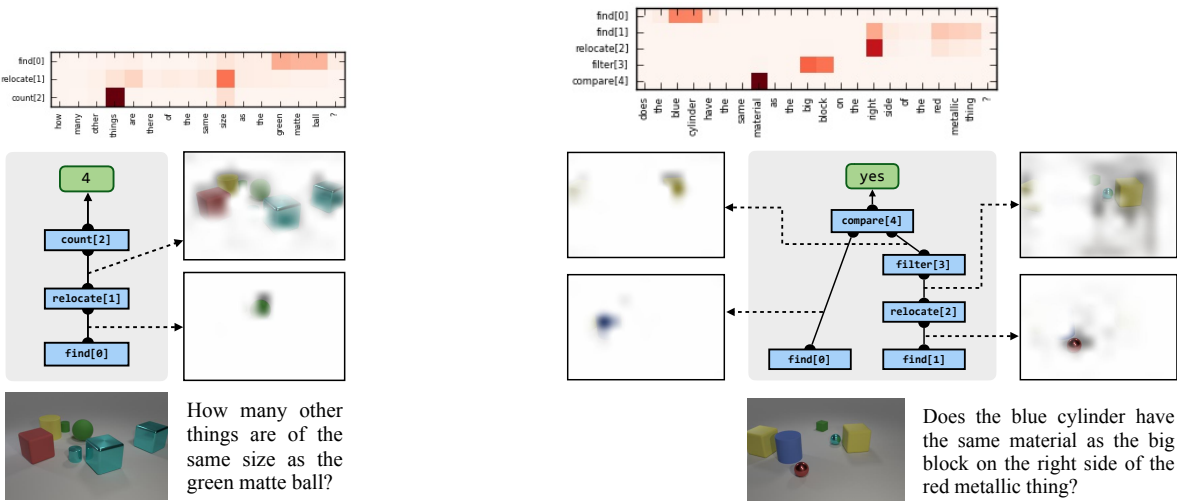


Figure 3.5: Question answering examples on the CLEVR dataset. On the left, it can be seen that the model successfully locates the matte green ball, attends to all the other objects of the same size, and then correctly identifies that there are 4 such objects (excluding the initial ball). On the right, it can be seen the various modules similarly assume intuitive semantics. Of particular interest is the second `find` module, which picks up the word `right` in addition to `metallic red thing`: this suggests that model can use the fact that downstream computation will look to the right of the detected object to focus its initial search in the left half of the image, a behavior supported by our attentive approach but not a conventional linguistic analysis of the question.

has the same number of inputs and closest potential behavior.

While the manually specified expert policy  $p_e$  obtained in this way might not be optimal, it is sufficient to provide supervision to learn good initial model parameters that can be further optimized in the later stage. During behavioral cloning, we train our model with two losses added together: the first loss is the KL-divergence  $D_{KL}(p_e||p) = -\log(p(l = l_e|q))$ , which corresponds to maximizing the probability of the expert layout  $l_e$  in our policy  $p(l|q)$  from the sequence-to-sequence RNN, and the second loss is the question answering loss  $\tilde{L}(\theta, l_e; q, I)$  for question  $q$  and image  $I$ , where the layout  $l_e$  is obtained from the expert. Note that the second loss  $\tilde{L}(\theta, l_e; q, I)$  also affects the parameters in the sequence-to-sequence RNN through the textual attention in Eqn. 3.3.

After the first training stage, we discard the expert policy and continue to train our model for a second stage with end-to-end reinforcement learning, using the gradient in Eqn. 3.5. In this stage, the model is no longer constrained to get close to the expert, but is encouraged to explore the layout space and search for the optimal layout of each question.

As a baseline, we also train our model without using any expert policy, and directly perform policy search from scratch by minimizing the loss function  $L(\theta)$  in Eqn. 3.4.

We evaluate our model on the test set of CLEVR. Table 3.3 shows the detailed performance of our model and previous methods on each question type, where “ours - policy search from scratch” is the baseline using pure reinforcement learning without resorting to the expert, “ours - cloning

Method	O	E	C	Compare Integer			Query Attribute				Compare Attribute			
				eq	le	mr	sz	cl	mt	sp	sz	cl	mt	sp
CNN+BoW [205]	48.4	59.5	38.9	50	54	49	56	32	58	47	52	52	51	52
CNN+LSTM [12]	52.3	65.2	43.7	57	72	69	59	32	58	48	54	54	51	53
CNN+LSTM+MCB [51]	51.4	63.4	42.1	57	71	68	59	32	57	48	51	52	50	51
CNN+LSTM+SA [192]	68.5	71.1	52.2	60	82	74	87	81	88	85	52	55	51	51
NMN (expert layout) [11]	72.1	79.3	52.5	61.2	77.9	75.2	84.2	68.9	82.6	80.2	80.7	74.4	77.6	79.3
ours - policy search from scratch	69.0	72.7	55.1	71.6	85.1	79.0	88.1	74.0	86.6	84.1	50.1	53.9	48.6	51.1
ours - cloning expert	78.9	83.3	63.3	68.2	87.2	85.4	90.5	80.2	88.9	88.3	89.4	52.5	85.4	86.7
ours - policy search after cloning	<b>83.7</b>	<b>85.7</b>	<b>68.5</b>	<b>73.8</b>	<b>89.7</b>	<b>87.7</b>	<b>93.1</b>	<b>84.8</b>	<b>91.5</b>	<b>90.6</b>	<b>92.6</b>	<b>82.8</b>	<b>89.6</b>	<b>90.0</b>

O: Overall; E: Exist; C: Count; eq: equal; le: less; mr: more; sz: size; cl: color; mt: material; sp: shape

Table 3.3: Evaluation of our method and previous work on CLEVR test set. With policy search after cloning, the accuracies are consistently improved on all questions types, with large improvement on some question types like compare color.

expert” is the supervised behavioral cloning from the constructed expert policy in the first stage, and “ours - policy search after cloning” is our model further trained for the second training stage. It can be seen that without using any expert demonstrations, our method with policy optimization from scratch already achieves higher performance than most previous work, and our model trained in the first behavioral cloning stage outperforms the previous approaches by a large margin in overall accuracy. This indicates that our neural modules are capable of reasoning for complex questions in the dataset like “*does the block that is to the right of the big cyan sphere have the same material as the large blue thing?*” Our model also outperforms the NMN baseline [11] trained on the same expert layout as used in our model<sup>1</sup>. This shows that our soft attention module parameterization is better than the hard-coded textual parameters in NMN. Figure 3.5 shows some question answering examples with our model.

By comparing “ours - policy search after cloning” with “ours - cloning expert” in Table 3.3, it can be seen that the performance consistently improves after end-to-end training with policy search using reinforcement learning in the second training stage, with especially large improvement on the *compare color* type of questions, indicating that the original expert policy is not optimal, and we can improve upon it with policy search over the entire layout space. Figure 3.6 shows an example before and after end-to-end optimization.

question: *do the small cylinder that is in front of the small green thing and the object right of the green cylinder have the same material?*  
 ground-truth answer: *no*

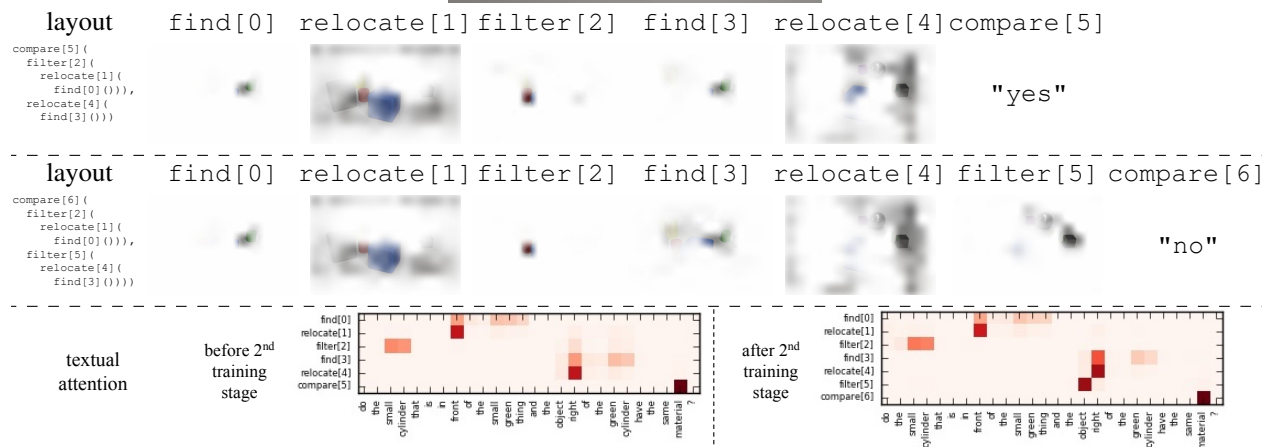
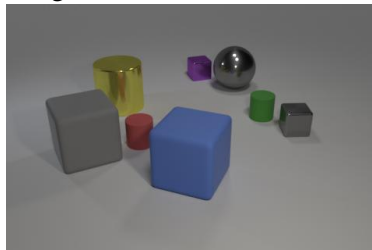


Figure 3.6: An example illustrating the layout change before (top row) and after (middle row) the second stage of end-to-end optimization with reinforcement learning. After end-to-end learning, a new `filter` module is inserted by the layout policy to remove the attention over the non-object area before feeding it into the final `compare` module, correcting the previous error.

### Evaluation on the VQA dataset

We also evaluate our method on the VQA dataset [12] with real images. On the VQA dataset, although there are no underlying functional program annotation for the questions, we can still construct an expert layout policy using a syntactic parse of questions as in [11, 10], and train our model in the same way as in Sec. 3.4. We train our model using different visual features for fair comparison with other methods. Unlike previous work [11, 10], the syntactic parser is only used during the training stage and is not needed at test time.

The results are summarized in Table 3.4 on the VQA dataset, where our method significantly outperforms NMN [11] and D-NMN [10] that also use modular structures, using the same LRCN VGG-16 image features (VGG-16 network fine-tuned for image captioning, as used in [11, 10]). Compared with MCB [51] (the VQA 2016 challenge winner method) trained on the same ResNet-152 image features, our model achieves slightly higher performance while being more interpretable

<sup>1</sup>The question parsing in the original NMN implementation does not work on the CLEVR dataset, as confirmed in [78]. For fair comparison with NMN, we train NMN using the same expert layout as our model.



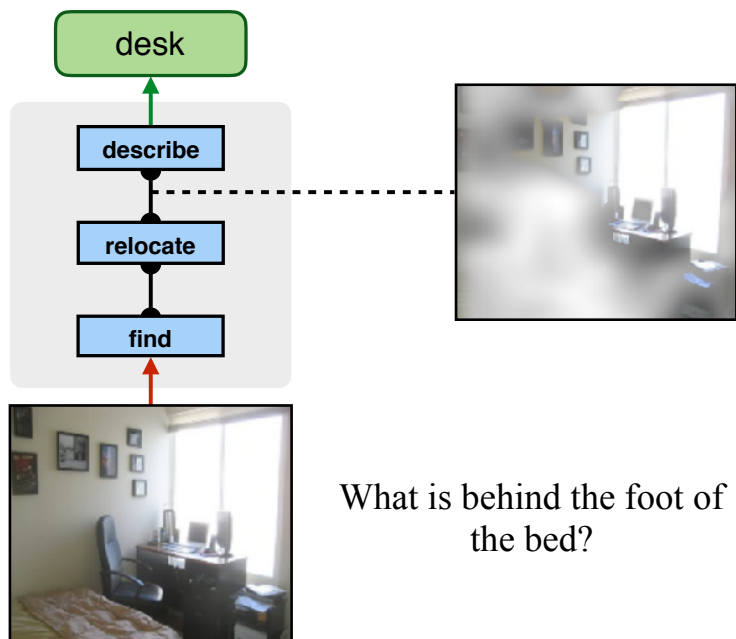


Figure 3.7: An example from our model on the VQA dataset.

Method	Visual feature	Accuracy
NMN [11]	LRCN VGG-16	57.3
D-NMN [10]	LRCN VGG-16	57.9
MCB [51]	ResNet-152	64.7
ours - cloning expert	LRCN VGG-16	61.9
ours - cloning expert	ResNet-152	64.2
ours - policy search after cloning	ResNet-152	<b>64.9</b>

Table 3.4: Evaluation of our method on the VQA test-dev set. Our model outperforms previous work NMN and D-NMN and achieves comparable performance as MCB.

as one can explicitly see the underlying reasoning procedure. Figure 3.7 shows a prediction example on this dataset.

### 3.5 Discussion

In this chapter, we present the End-to-End Module Networks for visual question answering. Our model uses a set of neural modules to break down complex reasoning problems posed in textual questions into a few sub-tasks connected together, and learns to predict a suitable layout expression for each question using a layout policy implemented with a sequence-to-sequence RNN. During training, the model can be first trained with behavioral cloning from an expert layout policy, and

further optimized end-to-end using reinforcement learning. Experimental results demonstrate that our model is capable of handling complicated reasoning problems, and the end-to-end optimization of the neural modules and layout policy can lead to significant further improvement over behavioral cloning from expert layouts.

## Chapter 4

# Explainable Neural Computation via Stack Neural Module Networks

### 4.1 Problem Statement

Deep neural networks have achieved impressive results on many vision and language tasks. Yet the predictive power of generic deep architectures comes at a cost of lost interpretability, as these architectures are essentially black boxes with respect to human understanding of their predictions. This can impair human users’ trust in learning systems and make them harder to refine [45].

These issues have led to recent efforts in explaining neural models, ranging from building in attention layers to post-hoc extraction of implicit model attention, e.g. by gradient propagation [161, 201, 141, 204, 142], post-hoc natural language explanations [66, 6] and network dissection [19]. Such approaches can highlight the image regions that are most important for predicting a particular label or provide a textual interpretation of the network output. However, explainable models of more complex problems involving multiple sub-tasks, such as Visual Question Answering (VQA) [12] and Referential Expression Grounding (REF) [146], are less studied in comparison. Complex problems may require several reasoning steps to solve. For example in Figure 4.1, the question “There is a small gray block; are there any spheres to the left of it?” might require solving the following subtasks: find the “small gray block”, look for “spheres to the left of it” and decide whether such object exists in the image. Therefore, a single heat-map highlighting important spatial regions such as [142] may not tell the full story of how a model performs.

In this chapter, we present a new model that makes use of an explicit, modular reasoning process, but which allows fully differentiable training with back-propagation and without expert supervision of reasoning steps. Existing modular networks first analyze the question and then predict a sequence of pre-defined modules (each implemented as a neural net) that chain together to predict the answer. However, they need an “expert layout”, or supervised module layouts for training the layout policy in order to obtain good accuracy. Our proposed approach, the *Stack Neural Module Network* or *SNMN*, can be trained without layout supervision, and replaces the layout graph of [71] with a stack-based data structure. Instead of making discrete choices on module layout, in this work we

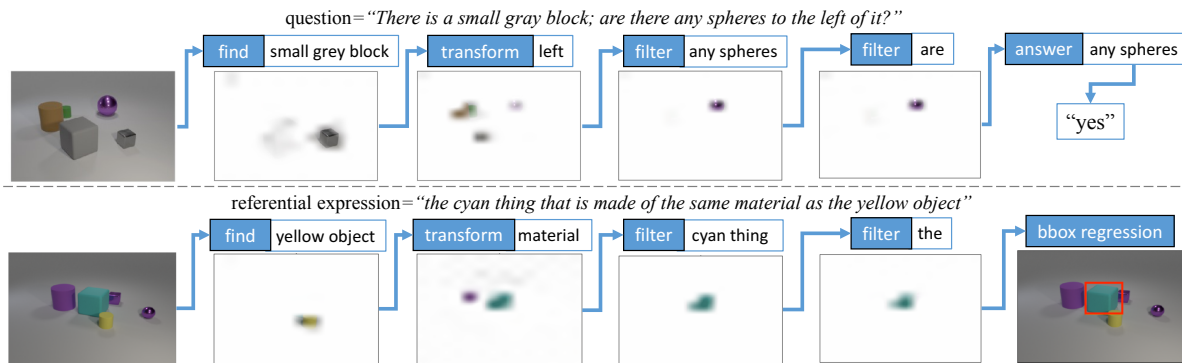


Figure 4.1: Our model reveals interpretable subtask structure by inducing a decomposition of the reasoning procedure into several sub-tasks, each addressed by a neural module. It can simultaneously answer visual questions and ground referential expressions.

make the layout soft and continuous, so that our model can be optimized in a fully differentiable way using gradient descent. We show that this improves both the accuracy and interpretability compared to existing modular approaches. We also show that this model can be extended to handle both Visual Question Answering (VQA) [12] and Referential Expression Grounding (REF) [146] seamlessly in a single model by sharing knowledge across related tasks through common routines as in Figure 4.1.

A variety of different model architectures have been proposed for complex reasoning and question answering. Our evaluation in this chapter focuses on both the accuracy and interpretability of these models. In particular, we ask: *does explicit modular structure make models more interpretable?* We use the CLEVR dataset [78] as a testbed, as it poses a task of high complexity. State-of-the-art models for this task vary in the degree to which they provide “explanations”. Relation Networks [149] and FiLM [136] achieve high performance but do not expose their internal decision process. Other state-of-the-art models on CLEVR use recurrent layers to compute the answer over multiple steps and output different image and/or text attention at each step. These include modular networks [11, 10, 71, 79, 118], and non-modular recurrent attention models [192, 75]. It has been suggested by the authors that the attention and/or module layouts inferred by these methods can be regarded as explanations of the networks’ internal reasoning process. Yet, to the best of our knowledge, their meaningfulness to humans has never been explicitly evaluated; we provide a more rigorous assessment of the interpretability of multi-step attentional VQA models here.

We categorize existing multi-step models in terms of whether they have a discrete library of structured modules for each step (e.g., NMN and related approaches [11, 10, 71, 79, 9, 118]), vs. homogeneous subtask computational elements (e.g., multi-hop attention networks [192, 190], MAC [75], etc.). We assess these models below and identify tradeoffs between accuracy and interpretability of these existing model classes. We find that our proposed SNMN model has comparable performance to existing modular approaches even without expert supervision, while achieving the greatest interpretability among evaluated models with respect to both subjective and objective measures of human understanding.

## 4.2 Related Work

**Visual question answering (VQA).** The task of visual question answering is to infer the answer based on the input question and image. Existing methods on VQA can be mainly categorized into holistic approaches (e.g., [192, 190, 51, 4, 149, 136, 75]), and neural module approaches [11, 10, 71, 79, 118]. The major difference between these two lines of work is that neural module approaches explicitly decompose the reasoning procedure into a sequence of sub-tasks, and have specialized modules to handle the sub-tasks, while holistic approaches do not have explicit sub-task structure, and different kinds of reasoning routines are all handled homogeneously.

Some holistic models perform sequential interactions between the image and the question. For example, SAN [192] uses multi-hop attention to extract information from the image. FiLM [136] uses multiple conditional batch normalization layers to fuse the image representation and question representation. Among these methods, MAC [75] performs multiple steps of reading and writing operations to extract information from the image and update its memory. Although these models have sequential interactions between the input image and the question, they do not explicitly decompose the reasoning procedure into semantically-typed sub-tasks. In our model, we adopt a similar textual attention mechanism as in [75] in Sec. 4.3, while also predicting module weights from the input text.

**Neural module networks (NMNs).** In NMN [11], N2NMN [71], PG+EE [79] and TbD [118], the inference procedure is performed by analyzing the question and decomposing the reasoning procedure into a sequence of sub-tasks. In [71], [79] and [118], a layout policy is used to turn the question into a module layout. Then the module layout is executed with a neural module network. Here, given an input question, the layout policy learns what sub-tasks to perform, and the neural modules learn how to perform each individual sub-tasks.

However, it is shown in these previous work that “expert layouts” (i.e. human annotations of the desired layout) are needed to pretrain or supervise the layout policy in order to get compositional behavior and good accuracy. Without expert guidance, existing models suffer from significant performance drops or fail to converge. This indicates that it is challenging to simultaneously learn “what” and “how” in these models. In this work, we address this problem with soft and continuous module layout, making our model fully differentiable and trainable with using gradient descent without resorting to expert layouts.

**Interpretable reasoning and explainable neural networks.** Recent years have seen increased interest in various aspects of interpretability in learned models [131], particularly in neural networks [130]. This includes work aimed at both explaining the decision rules implemented by learned models, and the mechanisms by which these rules are derived from data [153, 87]. In the present work we are primarily interested in the former. One line of research in this direction attempts to generate post-hoc explanations of decisions from generic model architectures, either by finding interpretable local surrogates in the form of linear models [145], logical rules [46, 202] or natural language descriptions [6, 203], or by visualizing salient features [141, 142].

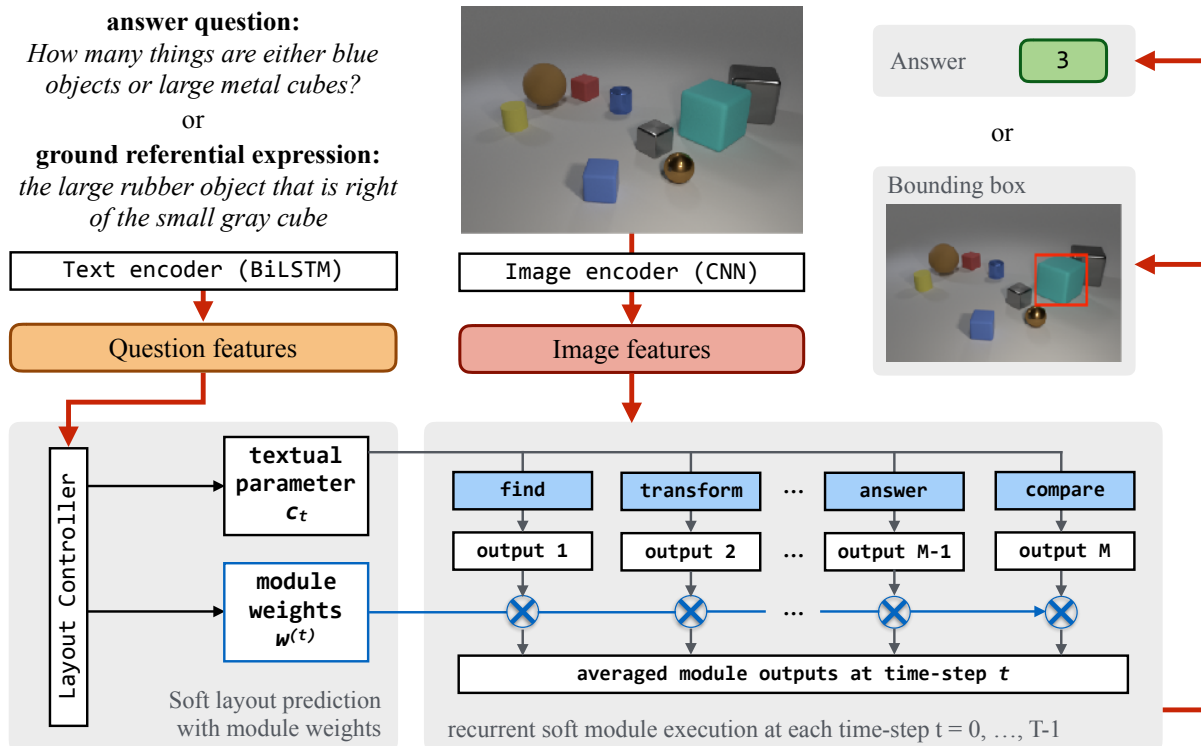


Figure 4.2: Overview of our model. Our model predicts a continuous layout via module weights  $w^{(t)}$  and executes the modules in a soft manner with a memory stack.

An alternative line of work investigates the extent to which models can be explicitly designed from the outset to provide enhanced interpretability, where main focus of study has been visual attention [125, 133]. While the various modular approaches described above are sometimes described as “interpretable” [71], we are not aware of any research evaluating this in practice. In the present work, our goal is to evaluate whether this kind of explicit modular structure, and not just iterated attention, improves interpretability in concrete evaluation scenarios.

**Multi-task learning.** Different from existing multi-task approaches such as sharing common features (e.g., [64]), our model simultaneously handles both Visual Question Answering (VQA) [12] and Referential Expression Grounding (REF) [146] by exploiting the intuition that related tasks should have common sub-routines, and addressing them with a common set of neural modules.

### 4.3 Stack Neural Module Networks (SNMNs)

In this chapter, we analyze and design interpretable neural networks for high-complexity VQA and REF tasks. We evaluate the interpretability of multi-step VQA networks to humans, and in particular compare modular networks to non-modular networks in terms of how well humans can

understand the internal computation process. We describe our proposed evaluation strategy and results in Section 4.4. We also improve modular networks by proposing a new formulation, which we describe in this section. Specifically, we describe Stack Neural Module Networks (SNMNs) with the following components. 1) A layout controller that decomposes the reasoning task into a sequence of sub-tasks, and translates the input question into a *soft layout*, specified via a soft distribution over module weights  $w^{(t)}$  at each timestep  $t$ . The controller also supplies each module with a textual parameter  $c_t$  at every time-step using textual attention. 2) A set of neural modules  $M$  to handle the sub-tasks specified by the controller. Each neural module is a differentiable parameterized function that performs a specific sub-task, and can be executed dynamically on-the-fly according to the soft layout. 3) A differentiable memory stack to store and retrieve intermediate outputs from each module during execution.

Figure 4.2 shows an overview of our model. The overall architecture of our model is conceptually similar to N2NMN [71], where layout controller in our model resembles the previous layout policy. The major difference between our model and this prior work lies in whether the layout selection is continuous or discrete. N2NMN makes discrete choices of module layout in a graph structure and can only be end-to-end optimized with reinforcement learning approaches. On the other hand, our model makes soft layout selection with a differentiable stack structure, by giving each module a continuous-valued weight parameter and averaging the outputs of all modules according to their weights. This makes the execution procedure fully differentiable so that our model is trainable with back-propagation like other neural networks.

## Module layout controller

The layout controller in our model decides what subtask to perform at each execution time step  $t$  by selecting a module  $m_t$  for that time step, and also supplying it with a textual parameter  $c_t$  to give specific instruction to the module  $m_t \in M$ . For example, the controller may decide to look for red things at  $t = 0$ , by running a `FIND` module with a textual parameter  $c_t$  that contains the information for the word “red”.

The structure of our layout controller is similar to the control unit in [75]. Suppose there are  $S$  words in the input question. The layout controller first encodes the input question  $q$  into a  $d$ -dimensional sequence  $[h_1, \dots, h_S]$  of length  $S$  using a bi-directional LSTM as  $[h_1, \dots, h_S] = \text{BiLSTM}(q; \theta_{\text{BiLSTM}})$ , where each  $h_s$  is the concatenation of the forward LSTM output and the backward LSTM output at the  $s$ -th input word. Next, the controller runs in a recurrent manner from time-step  $t = 0$  to time-step  $t = T - 1$ . At each time-step  $t$ , it applies a time-step dependent linear transform to the question  $q$ , and linearly combines it with the previous  $d$ -dimensional textual parameter  $c_{t-1}$  as  $u = W_2 \left[ W_1^{(t)} q + b_1; c_{t-1} \right] + b_2$ , where  $W_1^{(t)}$  and  $W_2$  are  $d \times d$  and  $d \times 2d$  matrices respectively, and  $b_1$  and  $b_2$  are  $d$ -dimensional vectors. Unlike all other parameters in the layout controller,  $W_1^{(t)}$  is not shared across different time steps.

To select the module to execute at the current time-step  $t$ , a small multi-layer perceptron (MLP) is applied to  $u$  to predict a  $|M|$ -dimensional vector  $w^{(t)}$  as  $w^{(t)} = \text{softmax}(\text{MLP}(u; \theta_{\text{MLP}}))$ . The module weight  $w^{(t)}$  contains the weight distribution over each module  $m \in M$  and sums up to one

module name	input attention	output type	implementation details ( $x$ : image feature map, $c$ : textual parameter)
Find	(none)	attention	$a_{out} = \text{conv}_2(\text{conv}_1(x) \odot Wc)$
Transform	$a$	attention	$a_{out} = \text{conv}_2(\text{conv}_1(x) \odot W_1 \sum(a \odot x) \odot W_2c)$
And	$a_1, a_2$	attention	$a_{out} = \text{minimum}(a_1, a_2)$
Or	$a_1, a_2$	attention	$a_{out} = \text{maximum}(a_1, a_2)$
Filter	$a$	attention	$a_{out} = \text{And}(a, \text{Find}())$ , i.e. reusing Find and And
Scene	(none)	attention	$a_{out} = \text{conv}_1(x)$
Answer	$a$	answer	$y = W_1^T (W_2 \sum(a \odot x) \odot W_3c)$
Compare	$a_1, a_2$	answer	$y = W_1^T (W_2 \sum(a_1 \odot x) \odot W_3 \sum(a_2 \odot x) \odot W_4c)$
NoOp	(none)	(none)	(does nothing)

Table 4.1: Neural modules used in our model. The modules take image attention maps as inputs, and output either a new image attention  $a_{out}$  or a score vector  $y$  over all possible answers ( $\odot$  is elementwise multiplication;  $\sum$  is sum over spatial dimensions).

(i.e.  $\sum_{m \in M} w_m^{(t)} = 1$ ), which resembles a probability distribution or soft attention over the modules. It is used to select modules in each time-step  $t$  in a continuous manner.

Finally, the controller predicts a textual parameter  $c_t$  with a textual attention over the encoded question words as  $cv_{t,s} = \text{softmax}(W_3(u \odot h_s))$  and  $c_t = \sum_{s=1}^S cv_{t,s} \cdot h_s$ , where  $\odot$  is element-wise multiplication,  $W_3$  is a  $1 \times d$  matrix,  $cv_{t,s}$  is the word attention score (scalar) on the  $s$ -th question word. Finally,  $c_t$  is the textual parameter supplied to the modules at time-step  $t$ , containing question information needed for a sub-task.

## Neural modules with a memory stack

**Module implementation.** Following the terminology in N2NMN, a neural module is a differentiable function with some internal trainable parameters, and can be used to perform a specific sub-task. For example, the question “how many objects are right of the blue object?” can be answered by the layout `Answer[‘how many’](Transform[‘right’](Find[‘blue’]()))`, where the modules such as `Transform` are selected with module weight  $w^{(t)}$  and the textual information such as ‘blue’ is contained in the textual parameter  $c_t$ .

The module implementation basically follows [71]. We also simplify the implementation in [71] by merging unary answering modules (`Count`, `Exist`, `Describe`) into a single `Answer` module, and pairwise comparison (`More`, `Less`, `Equal`, `Compare`) into a single `Compare` module. Finally, we introduce a `NoOp` module that does nothing, which can be used to pad arbitrary module layouts to a maximum length  $T$ . Our module implementation is summarized in Table 4.1.

**Differentiable memory stack.** In our model, different modules may take different numbers of input, and the model sometimes needs to take what it currently sees and compare it with what it has previously seen before. This is typical in tree-structured layouts, such as `Compare(Find(),`



`Transform(Find())`). To handle tree-structured layouts, the model needs to have a memory to remember the outputs from the previous reasoning time-steps. Similar to Memory Networks [165], we provide a differentiable memory pool to store and retrieve the intermediate outputs. However, to encourage compositional behavior, we restrict our memory pool to be a Last-In-First-Out (LIFO) stack similar to [59]. The LIFO behavior encourages the neural modules to work like functions in a computer program, allowing only arguments and returned values to be passed between the modules, without arbitrary memory modification.

Our memory stack can be used to store vectors with fixed dimensions. It consists of a length- $L$  memory array  $A = \{A_i\}_{i=1}^L$  (where  $L$  is the stack length) and a stack-top pointer  $p$ , implemented as a  $L$ -dimensional one-hot vector. The stack  $(A, p)$  implements differentiable push and pop operations as follows. Pushing a new vector  $z$  into stack  $(A, p)$  is done via pointer increment as  $p := \text{1d\_conv}(p, [0, 0, 1])$  followed by value writing as  $A_i := A_i \cdot (1 - p_i) + z \cdot p_i$ , for each  $i = 1, \dots, L$ . Similarly, popping the current stack-top vector  $z$  from stack  $(A, p)$  is done via value reading as  $z := \sum_{i=1}^L A_i \cdot p_i$  followed by pointer decrement as  $p := \text{1d\_conv}(p, [1, 0, 0])$ . Here  $A_i$  is the vector at stack depth  $i$  in  $A$ . In both push and pop operations, the one-hot stack pointer  $p$  is incremented or decremented using 1-d convolution.

In our model, we use the above memory stack to store the  $H \times W$  dimensional image attention maps, where  $H$  and  $W$  are the height and the width of the image feature map. Using the memory stack, each module first pops from the stack to obtain input image attentions, and then pushes its result back to the stack. For example, in tree-like layouts such as `Compare(Find(), Transform(Find()))`, the `Find` module pushes its localization result into the stack, the `Transform` module pops one image attention map from the stack and pushes back the transformed attention, and the `Compare` module pops two image attention maps and uses them to predict the answer.

## Soft program execution

Our model performs continuous selection of module layout through the soft module weights  $w^{(t)}$ . At each time step  $t$ , we execute all the modules in our module list  $M$  (shown in Table 4.1), and perform a weighted average of their results with respect to the weights  $w^{(t)}$  predicted by the layout controller. Specifically, the resulting memory stacks from the execution of each module are weighted-averaged with respect to  $w_m^{(t)}$  to produce a single updated memory stack.

At time step  $t = 0$ , we initialize the memory stack  $(A, p)$  with uniform image attention and set stack the pointer  $p$  to point at the bottom of the stack (one-hot vector with 1 in the 1st dimension). Then, at each time step  $t$ , for every module  $m \in M$  we execute it on the current memory stack  $(A^{(t)}, p^{(t)})$ . During execution, each module  $m$  may pop from the stack and push back its results, producing an updated stack  $(A_m^{(t)}, p_m^{(t)})$  as  $(A_m^{(t)}, p_m^{(t)}) = \text{run\_module}(m, A^{(t)}, p^{(t)})$ , for each  $m \in M$ . We average the resulting new stack from each module according to its weight  $w_m^{(t)}$  as  $A^{(t+1)} = \sum_{m \in M} A_m^{(t)} \cdot w_m^{(t)}$ , and then sharpen the stack pointer with a softmax operation to keep it as a (nearly) one-hot vector as  $p^{(t+1)} = \text{softmax}\left(\sum_{m \in M} p_m^{(t)} \cdot w_m^{(t)}\right)$ .

**Final output.** We apply this model to both the Visual Question Answering (VQA) task and the Referential Expressions Grounding (REF) task. To obtain the answer in the VQA task, we collect the output answer logits (i.e. scores) in all time-steps from those modules that have answer outputs (`Answer` and `Compare` in Table 4.1), and accumulate them with respect to their module weights as  $y = \sum_{t=0}^{T-1} \sum_{m \in M_{\text{ans}}} y_m^{(t)} w_m^{(t)}$  where  $M_{\text{ans}}$  contains `Answer` and `Compare`.

To output grounding results in the REF task, we take the image-attention map at the top of the final stack at  $t = T$ , and extract attended image features from this attention map. Then, a linear layer is applied on the attended image feature to predict the bounding box offsets from the feature grid location.

## Training

Unlike previous modular approaches N2NMN [71], PG+EE [79] and TbD [118], our model does not require expert layouts to achieve good performance. When such expert layout supervision is available, our model can also utilize it by supervising the soft module weights  $w_{(t)}$  with a cross-entropy loss to match the expert’s module choice. But as the entire network is fully differentiable, it can be trained effectively without reinforcement learning, from task supervision alone, in the absence of expert layout supervision.

For VQA, we train with softmax cross entropy loss on the final answer scores  $y$ . For REF, we map the center of the ground-truth bounding box to a location on the feature grid. Then we train with a softmax cross entropy loss on the final image attention map to put all the attention on the ground-truth feature grid, and a bounding box regression loss on the bounding box offsets to match the ground-truth box. We train with the Adam optimizer with  $10^{-4}$  learning rate. Our code and data are available at <http://ronghanghu.com/snmn/>.

## 4.4 Experiments

We evaluate our model on the Visual Question Answering (VQA) task on the large-scale CLEVR dataset [78]. The dataset consists of 70000, 15000 and 15000 images for training, validation and test, and each image is associated with 10 questions. The images in the dataset are rendered from a graphics engine, and the questions are synthesized with complex reasoning procedure.

To evaluate our model on the Referential Expression Grounding (REF) task [146], we build a new *CLEVR-Ref* dataset with images and referential expressions in CLEVR style using the code base of [78]. Our new CLEVR-Ref dataset has the same scale as the original CLEVR dataset for VQA, but contains referential expressions instead of questions. Each referential expression refers to a unique object in the image, and the model is required to ground (i.e. localize) the corresponding object with a bounding box. The grounded bounding box is considered correct if it overlaps with the ground-truth bounding box by at least 0.5 intersection-over-union (IoU). Similar to question answering in the CLEVR dataset, the referential expressions also involve complex reasoning and relationship handling. See Figure 4.3 for an example of the CLEVR-Ref dataset.

trained on	expert layout	VQA accuracy	REF accuracy
VQA	yes	<b>96.6</b>	n/a
REF	yes	n/a	96.0
VQA+REF	yes	96.5	<b>96.2</b>
VQA	no	93.0	n/a
REF	no	n/a	93.4
VQA+REF	no	<b>93.9</b>	<b>95.4</b>

Table 4.2: Validation accuracy on the CLEVR dataset (VQA) and the CLEVR-Ref dataset (REF). Our model simultaneously handles both tasks with high accuracy.

## Model performance

Our model aims to simultaneously handle both VQA and REF tasks, and to decompose the reasoning procedure into sub-tasks by inducing a suitable module layout on each question or referential expression.

We train our model on the CLEVR dataset for the VQA task, and the CLEVR-Ref dataset for the REF task. We experiment with training only on the VQA task, training only on the REF task, and joint training on both tasks (VQA+REF) using the loss from both tasks. To test whether our model can induce a reasonable sub-task decomposition and module layout, we experiment with both using expert layout supervision (same as in [71]) and training from scratch without expert layout. We use a ResNet-101 convnet [65] pretrained on ImageNet classification to extract visual features from the image.

The results are summarized in Table 4.2. It can be seen that when training on each individual task, our model achieves over 90% accuracy on both tasks (which is reasonably good performance), whether using expert layout supervision or not. Furthermore, joint training can lead to even higher accuracy on these two tasks (especially when not using expert layout). Our model can simultaneously handle these two tasks by exploiting the common sub-tasks in them, such as finding object and handling relationships.

**Sub-task decomposition and layout induction.** By comparing the bottom 3 rows (trained without using expert layout) and the top 3 rows (trained with expert layout supervision), it can be seen that although the models trained with expert layout still outperforms training from scratch, the gap between the two scenarios is relatively small. This indicates that our model can still work well without layout supervision, which is something previous modular approaches such as N2NMN [71], PG+EE [79] and TbD [118] could not handle.

We visualize the reasoning procedure our multi-task model on both VQA and REF task, for both with expert layout and without expert layout supervision. Figure 4.3 shows the module layout, the intermediate reasoning outputs and the most attended words from textual attention ( $cv_{t,s}$  in Sec. 4.3). It can be seen that our model can induce a reasonable decomposition of the inference procedure into sub-tasks without expert layout supervision, and it learns to share common sub-tasks such as

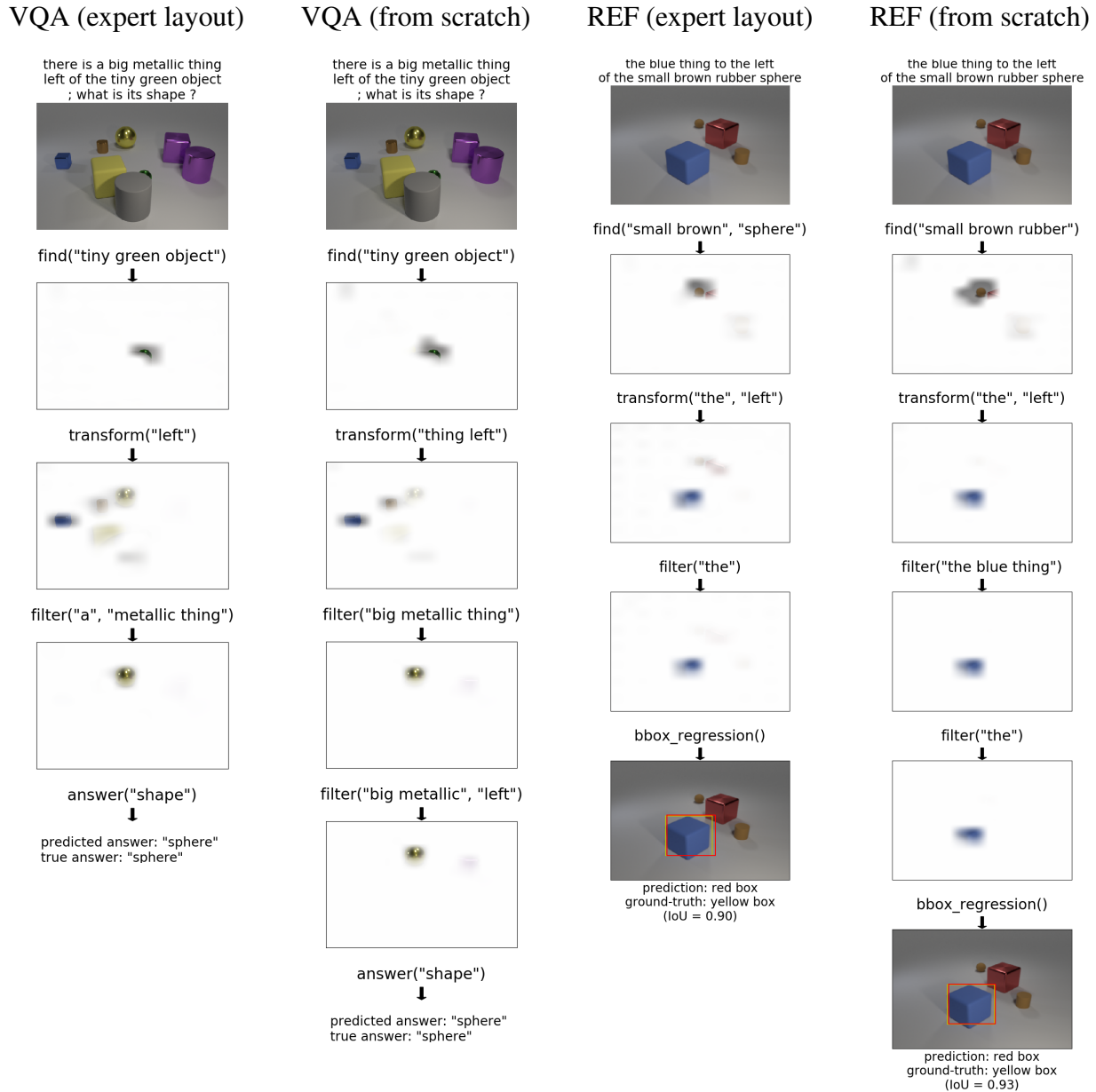


Figure 4.3: Examples of our model on VQA (left) and REF (right). At each step, we visualize the module with the highest weight, the words receiving most textual attention ( $cv_{t,s}$  in Sec. 4.3) and the module output.

find (localization) and transform in across the two tasks.

We note that our model learns peaky module weights after convergence. The average entropy of the learned soft module weights (which can be seen as a probability distribution) is 0.082 when trained without layout supervision (corresponds to putting over 98% weights on one module), and  $7.5 \times 10^{-5}$  when trained with layout supervision (corresponds to putting over 99.99% weights on

one module). This shows that even without any strong supervision on module layout, our model learns to almost discretely select one module through the soft module weights at test time. Hence, our proposed framework can be regarded as a novel end-to-end differentiable training approach for modular networks.

We further experiment with test-time layout discretization by replacing the soft module weights with a one-hot argmax vector. This results in slightly lower performance on the CLEVR validation set (90.0% when trained without layout supervision and 94.8% with layout supervision). Considering the discrepancy between training and test time, the relatively small accuracy drop ( $< 4\%$ ) from test-time layout discretization indicates that our model works similar to previous modular networks at test time, rather than acting as a mixture of experts.

**Evaluation of accuracy.** We first compare the accuracy of our model on the CLEVR VQA dataset with the previous modular approaches N2NMN [71], PG+EE [79] and TbD [118]. N2NMN uses a layout policy to predict discrete layouts and a neural module network to answer the question. PG+EE and TbD are also modular approaches similar to N2NMN, where the program generator is similar to the layout policy, and the execution engine is essentially a neural module network. For fair comparison with previous work, we train our model on the CLEVR VQA dataset only (without using CLEVR-Ref for joint training).

The results are shown in Table 4.3. It can be seen from the top 4 rows that among all the modular approaches (N2NMN, PG+EE, TbD and Ours), when layout supervision is available, our model outperforms N2NMN by a large margin, and achieves comparable performance with PG+EE while underperforms TbD by a small margin. We note that even when using expert layout, our model still uses less supervision than PG+EE or TbD as they both require fine-grained module specification (e.g. finding shape and finding color are different modules in [79, 118] while the same module with different textual attention in our model).

The bottom 4 rows show the results without using expert layout supervision, where our model significantly outperform N2NMN. In this case, N2NMN has large performance drop while PG+EE and TbD fails to converge or cannot not be trained without layout supervision. This can be attributed to the fact that N2NMN, PG+EE and TbD all use discrete non-differentiable layout, while our model is fully differentiable and can be trained with back-propagation.

We note that the best non-modular architectures [75] achieve higher performance without using expert layout supervision, and compare those against modular performance on both accuracy and interpretability in Sec. 4.4.

**Results on real-image VQA datasets.** We also evaluate our method on real-image visual question answering datasets and compare with N2NMN [71]. We run our approach on both VQAv1 and VQAv2 datasets [12, 58] following the same settings (e.g. using ResNet-152 image features and single model at test time without ensemble) as in [71], where the results are in Table 4.4. Although the question answering task in these datasets focuses more on visual recognition than on compositional reasoning, our method still outperforms [71] even without expert layout supervision (the expert layouts are obtained by a syntactic parser).

Method	expert layout	accuracy on CLEVR
N2NMN [71]	yes	83.7
PG+EE [79]	yes	96.9
TbD [118]	yes	<b>99.1</b>
Ours	yes	96.5
N2NMN [71]	no	69.0
PG+EE [79]	no	(does not converge)
TbD [118]	no	(not supported)
Ours	no	<b>93.0</b>

Table 4.3: Comparison of our model and other modular approaches on the CLEVR dataset for VQA. Our model achieves the best accuracy when not relying on expert layout, while N2NMN has significant performance drop in this case. The best non-modular architectures (e.g., [75]) do achieve higher performance; we compare those against modular performance on both accuracy and interpretability in Sec. 4.4.

Method	expert layout	accuracy on VQAv1	accuracy on VQAv2
N2NMN [71]	yes	64.9	63.3
ours	no	65.5	<b>64.1</b>
ours	yes	<b>66.0</b>	64.0

Table 4.4: Single-model accuracy of our method and N2NMN [71] on both VQAv1 [12] and VQAv2 [58] datasets, using the same experimental settings (e.g. visual features).

## Model interpretability

**Evaluation of interpretability.** It is often suggested in existing works [71, 79, 118] that modular networks can be more interpretable to humans compared to holistic models. However, there is a lack of human studies in these works to support this claim. In this section, we evaluate how well the user can understand the internal reasoning process within our model, and compare it with MAC [75].<sup>1</sup> We compare to MAC because it is a state-of-the-art holistic model that also performs multi-step sequential reasoning and has image and textual attention at each time-step, while other models (e.g., FiLM [136] and Relation Net [149]) have lower performance and do not have any image or textual attention to visualize. MAC is a multi-step recurrent structure with a control unit and a reading-writing unit. Similar to our model, it also attend to text and image in each reasoning step. But unlike our model, there is not explicit modular structure in MAC.

Here, we investigate two distinct, but related questions: does modular structure improve humans’ *subjective perceptions* of model interpretability, and does this structure allow users to form *truthful beliefs* about model behavior? To this end, we present two different sets of experiments (subjective

<sup>1</sup>In our comparison, we use a fixed number of 12 steps for MAC (following the default setting in [75]), which is longer than the average number of steps in our SNMN model.

understanding and forward prediction) with human evaluators. With respect to the taxonomy of interpretability evaluations presented in [45], these are both “human-grounded” metrics aimed at testing “general notions of the quality of an explanation”.

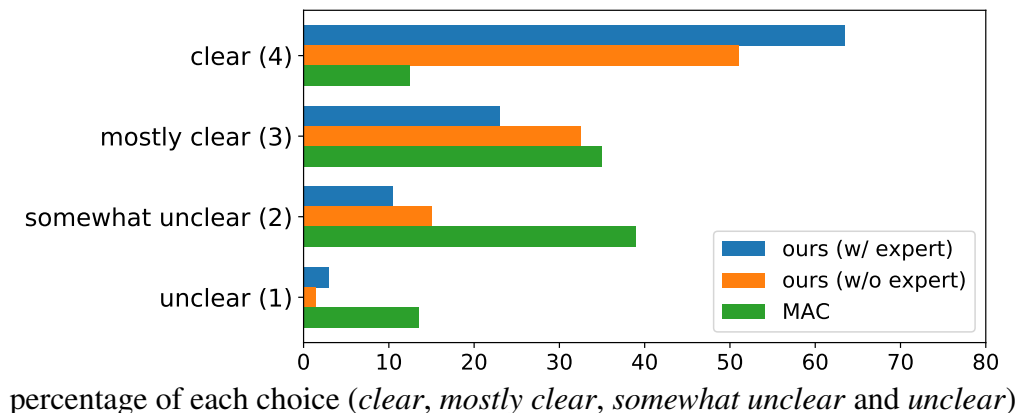
In the **subjective understanding** evaluation, we visualize model’s intermediate outputs such as the image attention and textual attention at each step, and we also show the model’s final prediction. The visualizations can be seen in Figure 4.3. Then the human evaluators are asked to judge how well they can understand the internal reasoning process, or whether it clear to the user what the model is doing at each step. Each example is rated on a 4-point Likert scale (*clear*, *mostly clear*, *somewhat unclear* and *unclear*) corresponding to numerical scores of 4, 3, 2 and 1. The averaged scores and the percentage of each choice are shown in Table 4.5, where it can be seen that our model has higher subjective understanding scores than MAC [75] and is much more often rated as “clear” in both cases (using or not using expert layout supervision). This shows that the users can more clearly understand the reasoning procedure in our model.

In the **forward prediction** evaluation, we investigate whether humans can predict the model’s answer and detect its failure based on these visualizations. We split the test set into half correct and half incorrect model predictions, and the final answer output is not shown, so that human baseline performance should be chance or 50%. Our hypothesis is that if humans can predict whether the model succeed or fail better than chance, they understand something about the model’s decision process. In Table 4.5, we show the human accuracy on this task along with 95% confidence interval. It can be seen that our model allows them to predict whether the model will get the correct answer or fail consistently higher than chance when trained without expert layout supervision. We also notice that when using supervision from expert layout, our model does worse at human prediction of model’s failure. We suspect it is because predicting the answer requires human to understand how the model works. When supervising the layout, the model may overfit to the expert layout, at the expense of predictability. It may output an “intuitive” layout by mimicking the training data, but that layout may not actually be how it is solving the problem. On the other hand, the unsupervised model is not being forced to predict any particular layouts to minimize loss, so its layouts may be more directed at minimizing the answer loss.

Finally, we compare our model with MAC on VQA accuracy in Table 4.5. Our model underperforms the state-of-the-art MAC in terms of VQA accuracy. However, our model is more interpretable to a human user. This is in line with the intuition that there may be an accuracy-explainability tradeoff, e.g., linear models are less accurate but more interpretable than non-linear models. However, our model greatly reduces the accuracy gap with the top performing models, without requiring expert layout supervision at training time.

## 4.5 Discussion

In this chapter, we have proposed a novel model for visual question answering and referential expression grounding. We demonstrate that our model simultaneously addresses both tasks by exploiting the intuition that related tasks should share common sub-tasks, and sharing a common set of neural modules between tasks. Compared with previous modular approaches, our model



Method	expert layout	subjective understanding	forward prediction (failure detection) accuracy $\pm$ 95% confidence interval	VQA accuracy
Ours	yes	<b>3.47</b>	$0.545 \pm 0.069$	96.5
Ours	no	3.33	<b><math>0.625 \pm 0.067</math></b>	93.0
MAC [75]	n/a	2.46	$0.565 \pm 0.069$	<b>98.9</b>

Table 4.5: Human evaluation of our model and the state-of-the-art non-modular MAC model [75]. Based on the models’ intermediate outputs, the evaluators are asked to (a) judge how clearly they can understand the reasoning steps performed by these models on a 4-point scale (i.e. subjective understanding) and (b) do forward prediction (failure detection) and decide whether the model fails without seeing the final output answer. The results show that our model is more interpretable to human users. However, our model underperforms the non-modular MAC approach in VQA accuracy, which is in line with the intuition that there may be an accuracy-explainability tradeoff.

induces a decomposition of the inference procedure into sub-tasks while not requiring expert layout supervision. The proposed model can explain its reasoning steps with a sequence of soft module choices, image attentions, and textual attentions. Experimental evaluation found that these explanations produced better understanding in human users with respect to both subjective and objective evaluations, even in the absence of human-provided explanations at training time.



# Chapter 5

## Language-Conditioned Graph Networks

### 5.1 Problem Statement

Grounded language comprehension tasks, such as visual question answering (VQA) or referring expression comprehension (REF), require finding the relevant objects in the scene and reasoning about certain relationships between them. For example in Figure 5.1, to answer the question *is there a person to the left of the woman holding a blue umbrella*, we must locate the relevant objects – *person*, *woman* and *blue umbrella* – and model the specified relationships – *to the left of* and *holding*.

How should we build a model to perform reasoning in grounded language comprehension tasks? Prior works have explored various approaches from learning joint visual-textual representations (*e.g.* [51, 136]) to pooling over pairwise relationships (*e.g.* [149, 197]) or constructing explicit reasoning steps with modular or symbolic representations (*e.g.* [11, 196]). Although these models are capable of performing complex relational inference, their scene representations are built upon local visual appearance features that do not contain much contextual information. Instead, they tend to rely heavily on manually designed inference structures or modules to perform reasoning about relationships, and are often specific to a particular task.

In this work, we propose an alternative way to facilitate reasoning with a context-aware scene representation, suitable for multiple tasks. Our proposed Language-Conditioned Graph Network (LCGN) model augments the local appearance feature of each entity in the scene with a relational contextualized feature. Our model is a graph network built upon visual entities in the scene, which collects relational information through multiple iterations of message passing between the entities. It dynamically determines which objects to collect information from on each round, by weighting the edges in the graph, and sends messages through the graph to propagate just the right amount of relational information. The key idea is to condition the message passing on the specific contextual relationships described in the input text. Figure 5.1 illustrates this process, where the *person* would be represented not only by her local appearance, but also by contextualized features indicating her relationship to other relevant objects in the scene, *e.g.*, *left of a woman*. Our contextualized representation can be easily plugged into task-specific models to replace standard local appearance

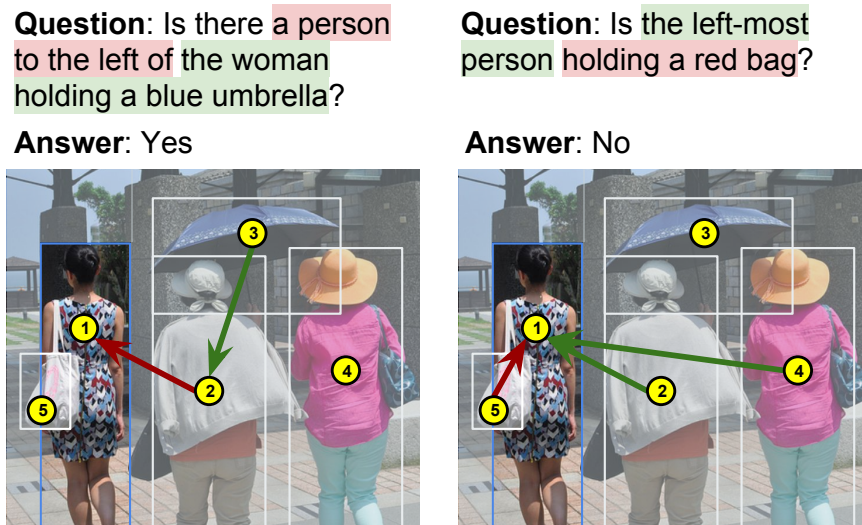


Figure 5.1: In this work, we create context-aware representations for objects by sending messages between relevant objects in a dynamic way that depends on the input language. In the left example, the first round of message passing updates object 2 with features of object 3 based on *the woman holding a blue umbrella* (green arrow), and the second round updates object 1 with object 2’s features based on *person to the left* (red arrow). The final answer prediction can be made by a single attention hop over the most relevant object (blue box).

features, facilitating reasoning with rich relational information. E.g. for the question answering task, it is sufficient to perform a single attention hop over the relevant object, whose representation is contextualized (e.g. blue box in Figure 5.1).

Importantly, our scene representation is constructed with respect to the given reasoning task. An object in the scene may be involved in multiple relations in different contexts: in Figure 5.1, the person can be simultaneously *left of a woman holding a blue umbrella*, *holding a white bag*, and *standing on a sidewalk*. Rather than building a complete representation of all the first- and higher-order relational information for each object (which can be enormous and unnecessary), we focus the contextual representation on relational information that is helpful to the reasoning task by conditioning on the input text (Figure 5.1 left vs. right).

We apply our Language-Conditioned Graph Networks to two reasoning tasks with language inputs—Visual Question Answering (VQA) and Referring Expression Comprehension (REF). In these tasks, we replace the local appearance-based visual representations with the context-aware representations from our LCGN model, and demonstrate that our context-aware scene representations can be used as inputs to perform complex reasoning via simple task-specific approaches, with a consistent improvement over the local appearance features across different tasks and datasets. We obtain state-of-the-art results on the GQA dataset [76] for VQA and the CLEVR-Ref+ dataset [104] for REF. Our code is available at <http://ronghanghu.com/lcgn>.

## 5.2 Related Work

We first provide an overview of the reasoning tasks addressed in this chapter. Then we review related work on graph networks and other contextualized representations. Finally, we discuss alternative approaches to reasoning problems.

**Visual question answering (VQA) and referring expression comprehension (REF).** VQA and REF are two popular tasks that require reasoning about image content. While in VQA the goal is to answer a question about an image [12], in REF one has to localize an image region that corresponds to a referring expression [117]. While the real-world VQA dataset [12, 58] focuses more on perception than complex reasoning, the more recent synthetic CLEVR [78] dataset is a standard benchmark for relational reasoning. An even more recent GQA dataset [76] brings together the best of both worlds: real images and relational questions. It is built upon the Visual Genome dataset [90] and construct the balanced question-answer pairs from scene graphs.

For REF, there are a number of standard benchmarks such as RefCOCO [198] and RefCOCOg [117], with natural language referring expressions and images from the COCO dataset [101]. However, many of the expressions in these datasets do not require resolving relations. Recently, a new CLEVR-Ref+ dataset [104] has been proposed for REF. It is built using the CLEVR environment and involves very complex queries, aiming to assess the reasoning capabilities of existing models and find their limitations.

In this work we tackle both VQA and REF tasks on three datasets in total. Notably, in all cases, we use the same approach, Language-Conditioned Graph Network (LCGN), to build contextualized representations of objects/image regions. This shows the generality and effectiveness of our approach for various visual reasoning tasks.

**Graph networks and contextualized representations.** Graph networks are powerful models that can perform relational inference through message passing [18, 54, 86, 100, 179, 206]. The core idea is to enable communication between image regions to build contextualized representations of these regions. Graph networks have been successfully applied to various tasks, from object detection [106] and region classification [34] to human-object interaction [138] and activity recognition [68]. Besides, self-attention models [176] and non-local networks [184] can also be cast as graph networks in a general sense. Below we review some of the recent works that rely on graph networks and other contextualized representations for VQA and REF.

A prominent work that introduced relational reasoning in VQA is [149], which proposes Relation Networks (RNs) for modeling relations between all pairs of objects, conditioned on a question. [31] extends RNs with the Broadcasting Convolutional Network module, which globally broadcasts objects' visuo-spatial features. The first work to use graph networks in VQA is [173], which combines dependency parses of questions and scene graph representations of abstract scenes. [208] proposes modeling structured visual attention over a Conditional Random Field on image regions. A recent work, [129], conditions on a question to learn a graph representation of an image, capturing object interactions with the relevant neighbours via spatial graph convolutions. Later, [29] extends this idea to modeling spatial-semantic pairwise relations between all pairs of regions.

For the REF task, [183] proposes Language-guided Graph Attention Networks, where attention over nodes and edges is guided by a referring expression, which is decomposed into subject, intra-class and inter-class relationships.

Our work is related to, yet distinct from, the approaches above. While [129] predicts a sparsely connected graph (conditioned on the question) that remains fixed for each step of graph convolution, our LCGN model predicts dynamic edge weights to focus on different connections in each message passing iteration. Besides, [129] is tailored to VQA and is non-trivial to adapt to REF (since it includes max-pooling over node representations). Compared to [29], instead of max-pooling over explicitly constructed pairwise vectors, our model predicts normalized edge weights that both improve computation efficiency in message passing and make it easier to visualize and inspect connections. Finally, [183] is tailored to REF by modeling specific subject attention and inter- and intra-class relations, and does not gather higher-order relational information in an iterative manner. We propose a more general approach for scene representation that is applicable to both VQA and REF.

**Reasoning models.** A multitude of approaches have been recently proposed to tackle visual reasoning tasks, such as VQA and REF. Neural Module Networks (NMNs) [11, 71] are multi-step models that build question-specific layouts and execute them against an image. NMNs have also been applied to REF, *e.g.* CMN [72] and Stack-NMN [70]. MAC [75] performs multi-step reasoning while recording information in its memory. FiLM [136] is an approach which modulates image representation with the given question via conditional batch normalization, and is extended in [193] with a multi-step reasoning procedure where both modalities can modulate each other. QGHC [53] predicts question-dependent convolution kernels to modulate visual features. DFAF [52] introduces self-attention and co-attention mechanisms between visual features and question words, allowing information to flow across modalities. The Neural-Symbolic approach [196] disentangles reasoning from image and language understanding, by first extracting symbolic representations from images and text, and then executing symbolic programs over them. MAttNet [197], a state-of-the-art approach to REF, uses attention to parse an expression and ground it through several modules.

Our approach is not meant to substitute the aforementioned reasoning models, but to complement them. Our contextualized visual representation can be combined with other reasoning models to replace the local feature representation. A prominent reasoning model capable of addressing both VQA and REF is Stack-NMN [70], and we empirically compare to it in Section 5.4.

### 5.3 Language-Conditioned Graph Networks (LCGNs)

Given a visual scene and a textual input for a reasoning task such as VQA or REF, we propose to construct a contextualized representation for each entity in the scene that contains the relational information needed for the reasoning procedure specified in the language input.

This contextualized representation is obtained in our novel Language-Conditioned Graph Networks (LCGN) model, through iterative message passing conditioned on the language input. It can be then used as input to a task-specific output module such as a single-hop VQA classifier.

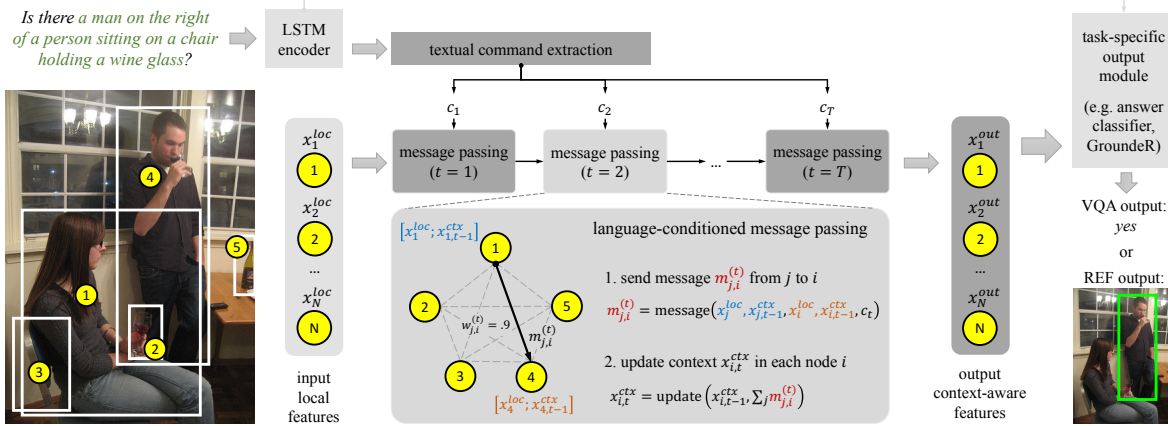


Figure 5.2: We propose Language-Conditioned Graph Networks (LCGN) to address reasoning tasks such as VQA and REF. Our model constructs a context-aware representation  $x_i^{out}$  for each object  $i$  through iterative message passing conditioned on the input text. During message passing, each object  $i$  is represented by a local feature  $x_i^{loc}$  and a context feature  $x_i^{ctx}$ . In every iteration, each object  $j$  sends a message vector  $m_{j,i}^{(t)}$  to each object  $i$ , which is collected by  $i$  to update its context feature  $x_{i,t}^{ctx}$ . The local feature  $x_i^{loc}$  and the final context feature  $x_{i,T}^{ctx}$  are combined into a joint context-aware feature  $x_i^{out}$ , which is used in simple task-specific output modules for VQA or REF.

## Context-aware scene representation

For an image  $I$  and a textual input  $Q$  that represents a reasoning task, let  $N$  be the number of entities in the scene, where each entity can be a detected object or a spatial location on the convolutional feature map of the image. Let  $x_i^{loc}$  (where  $i = 1, \dots, N$ ) be the local feature representation of the  $i$ -th entity, *i.e.* the  $i$ -th detected object’s visual feature or the convolutional feature at the  $i$ -th location on the feature grid. We would like to output a context-aware representation  $x_i^{out}$  for each entity  $i$  conditioned on the textual input  $Q$  that contains the relational context associated with entity  $i$ . This is obtained through iterative message passing over  $T$  iterations with our Language-Conditioned Graph Networks, as shown in Figure 5.2.

We use a fully-connected graph over the scene, where each node corresponds to an entity  $i$  as defined above, and there is a directed edge  $i \rightarrow j$  between every pair of entities  $i$  and  $j$ . Each node  $i$  is represented by a local feature  $x_i^{loc}$  that is fixed during message passing, and a context feature  $x_{i,t}^{ctx}$  that is updated during each iteration  $t$ . A learned parameter is used as the initial context representation  $x_{i,0}^{ctx}$  at  $t = 0$  for all nodes, before the message passing starts.

**Textual command extraction.** To incorporate the textual input in the iterative message passing, we build a textual command vector for each iteration  $t$  (where  $t = 1, \dots, T$ ). Given a textual input  $Q$  for the reasoning task, such as a question in VQA or a query in REF, we extract a set of vectors  $\{c_t\}$  from the text  $Q$ , using the same multi-step textual attention mechanism as in Stack-NMN [70]

and MAC [75]. Specifically,  $Q$  is encoded into a sequence  $\{h_s\}_{s=1}^S$  and a summary vector  $q$  with a bi-directional LSTM as:

$$[h_1, h_2, \dots, h_S] = \text{BiLSTM}(Q) \quad \text{and} \quad q = [h_1; h_S] \quad (5.1)$$

where  $S$  is the number of words in  $Q$ , and  $h_s = [\vec{h}_s; \overleftarrow{h}_s]$  is the concatenation of the forward and backward hidden states for word  $s$  from the bi-directional LSTM output. At each iteration  $t$ , a textual attention  $\alpha_{t,s}$  is computed over the words, and the textual command  $c_t$  is obtained from the textual attention as follows:

$$\alpha_{t,s} = \text{Softmax}_s \left( W_1 \left( h_s \odot \left( W_2^{(t)} \text{ReLU}(W_3 q) \right) \right) \right) \quad (5.2)$$

$$c_t = \sum_{s=1}^S \alpha_{t,s} \cdot h_s \quad (5.3)$$

where  $\odot$  is element-wise multiplication. Each  $c_t$  can be seen as a textual command supplied during the  $t$ -th iteration. Unlike all other parameters that are shared across iterations, here  $W_2^{(t)}$  is learned separately for each iteration  $t$ .

**Language-conditioned message passing.** At the  $t$ -th iteration where  $t = 1, \dots, T$ , we first build a joint representation of each entity. Then, we compute the (directed) connection weights  $w_{j,i}^{(t)}$  from every entity  $j$  (the sender,  $j = 1, \dots, N$ ) to every entity  $i$  (the receiver,  $i = 1, \dots, N$ ). Finally, each entity  $j$  sends a message vector  $m_{j,i}^{(t)}$  to each entity  $i$ , and each entity  $i$  sums up all of its incoming messages  $m_{j,i}^{(t)}$  to update its contextual representation from  $x_{i,t-1}^{ctx}$  to  $x_{i,t}^{ctx}$  as described below.

Step 1. We build a joint representation  $\tilde{x}_{i,t}$  for each node, by concatenating  $x_i^{loc}$  and  $x_{i,t-1}^{ctx}$  and their element-wise product (after linear mapping) as

$$\tilde{x}_{i,t} = [x_i^{loc}; x_{i,t-1}^{ctx}; (W_4 x_i^{loc}) \odot (W_5 x_{i,t-1}^{ctx})] \quad (5.4)$$

Step 2. We compute the directed connection weights  $w_{j,i}^{(t)}$  from node  $j$  (the sender) to node  $i$  (the receiver), conditioning on the textual command  $c_t$  at iteration  $t$ . Here, the connection weights are normalized with a softmax function over  $j$ , so that the sender weights sum up to 1 for each receiver, *i.e.*  $\sum_{j=1}^N w_{j,i}^{(t)} = 1$  for all  $i = 1, \dots, N$  as follows:

$$w_{j,i}^{(t)} = \text{Softmax}_j \left( (W_6 \tilde{x}_{i,t})^T ((W_7 \tilde{x}_{j,t}) \odot (W_8 c_t)) \right) \quad (5.5)$$

Step 3. Each node  $j$  sends a message  $m_{j,i}^{(t)}$  to each node  $i$  conditioning on the textual input  $c_t$  and weighted by the connection weight  $w_{j,i}^{(t)}$ . Then, each node  $i$  sums up the incoming messages and updates its context representation:

$$m_{j,i}^{(t)} = w_{j,i}^{(t)} \cdot ((W_9 \tilde{x}_{j,t}) \odot (W_{10} c_t)) \quad (5.6)$$

$$x_{i,t}^{ctx} = W_{11} \left[ x_{i,t-1}^{ctx}; \sum_{j=1}^N m_{j,i}^{(t)} \right] \quad (5.7)$$

A naive implementation would involve  $N^2$  pairwise vectors  $m_{j,i}^{(t)}$ , which is inefficient for large  $N$ . We implement it more efficiently by building an  $N$ -row matrix  $M$  containing  $N$  unweighted messages  $\tilde{m}_j^{(t)} = (W_9 \tilde{x}_{j,t}) \odot (W_{10} c_t)$  in Eqn. 5.6, which is left multiplied by the edge weight matrix  $E$  (where  $E_{ij} = w_{j,i}^{(t)}$ ) to obtain the sums  $\sum_{j=1}^N m_{j,i}^{(t)}$  in Eqn. 5.7 for all nodes in a single matrix multiplication. With this implementation, we can train our LCGN model efficiently with  $N$  as large as 196 in our experiments.

**Final representation.** We combine each entity’s local feature  $x_i^{loc}$  and context feature  $x_{i,T}^{ctx}$  (after  $T$  iterations) as its final representation  $x_i^{out}$ :

$$x_i^{out} = W_{12} [x_i^{loc}, x_{i,T}^{ctx}] \quad (5.8)$$

The  $x_i^{out}$  can be used as input to subsequent task-specific modules such as VQA or REF models, instead of the original local representation  $x_i^{loc}$ .

## Application to VQA and REF

To apply our LCGN model to language-based reasoning tasks such as Visual Question Answering (VQA) and Referring Expression Comprehension (REF), we build simple task-specific output modules based on the language input and the contextualized representation of each entity. Our LCGN model and the subsequent task-specific modules are jointly trained end-to-end.

**A single-hop answer classifier for VQA.** The VQA task requires outputting an answer for an input image  $I$  and a question  $Q$ . We adopt the commonly used classification approach and build a single-hop attention model as a classifier to select one of the possible answers from the training set.

First, the question  $Q$  is encoded into a vector  $q$  with the Bi-LSTM in Eqn. 5.1. Then a single-hop attention  $\beta_i$  is used over the objects to aggregate visual information, which is fused with  $q$  to predict the score vector  $y$  for each answer.

$$\beta_i = \text{Softmax}_i (W_{13} (x_i^{out} \odot (W_{14} q))) \quad (5.9)$$

$$y = W_{15} \text{ReLU} \left( W_{16} \left[ \sum_{i=1}^N \beta_i x_i^{out}, q \right] \right) \quad (5.10)$$

During training, a softmax or sigmoid classification loss is applied on the output scores  $y$  for answer classification.

**GroundER [146] for REF.** The REF task requires outputting a target bounding box as the grounding result for an input referring expression  $Q$ . Here, we use a retrieval approach as in previous works and select one target entity from the  $N$  candidate entities in the scene (either object detection results or spatial locations on a convolutional feature map). To select the target object  $p$  from the  $N$  candidates, we encode expression  $Q$  to vector  $q$  as in Eqn 5.1 and build a model similar to the

fully-supervised version of GroundeR [146] to output a matching score  $r_i$  for each entity  $i$ . In the case of using spatial locations on a convolutional feature map, we further output a 4-dimensional vector  $u$  to predict the bounding box offset from the feature grid location.

$$r_i = W_{17} (x_i^{out} \odot (W_{18}q)) \quad (5.11)$$

$$p = \arg \max_i r_i \quad (5.12)$$

$$u = W_{19}x_p^{out} \quad (5.13)$$

During training, we use a softmax loss over the scores  $r_i$  among the  $N$  candidates to select the target entity  $p$ , and an L2 loss over the box offset  $u$  to refine the box location.

## 5.4 Experiments

We apply our LCGN model to two tasks – VQA and REF – for language-conditioned reasoning. For the VQA task, we evaluate on the GQA dataset [76] and the CLEVR dataset [78], which both require resolving relations between objects. For the REF task, we evaluate on the CLEVR-Ref+ dataset [104]. In particular, the CLEVR and CLEVR-Ref+ datasets contain many complicated questions or expressions with higher-order relations, such as *the ball on the left of the object behind a blue cylinder*.

### Visual Question Answering (VQA)

**Evaluation on the GQA dataset.** We first evaluate our LCGN model on the GQA dataset [76] for visual question answering. The GQA dataset is a large-scale visual question answering dataset with real images from the Visual Genome dataset [90] and balanced question-answer pairs. Each training and validation image is also associated with scene graph annotations describing the classes and attributes of those objects in the scene, and their pairwise relations. Along with the images and question-answer pairs, the GQA dataset provides two types of pre-extracted visual features for each image – convolutional grid features of size  $7 \times 7 \times 2048$  extracted from a ResNet-101 network [65] trained on ImageNet, and object detection features of size  $N_{det} \times 2048$  (where  $N_{det}$  is the number of detected objects in each image with a maximum of 100 per image) from a Faster R-CNN detector [144].

We apply our LCGN model together with the single-hop classifier (“**single-hop + LCGN**”) in Sec. 5.3 for answer prediction. We use  $T = 4$  rounds of message passing in our LCGN model, which takes approximately 20 hours to train using a single Titan Xp GPU. As a comparison to our LCGN model, we also train the single-hop classifier with only the local features  $x^{loc}$  in Eqn. 5.9 (“**single-hop**”).

We first experiment with using the released object detection features in the GQA dataset as our local features  $x^{loc}$ , which is shown in [76] to perform better than the convolutional grid features, and



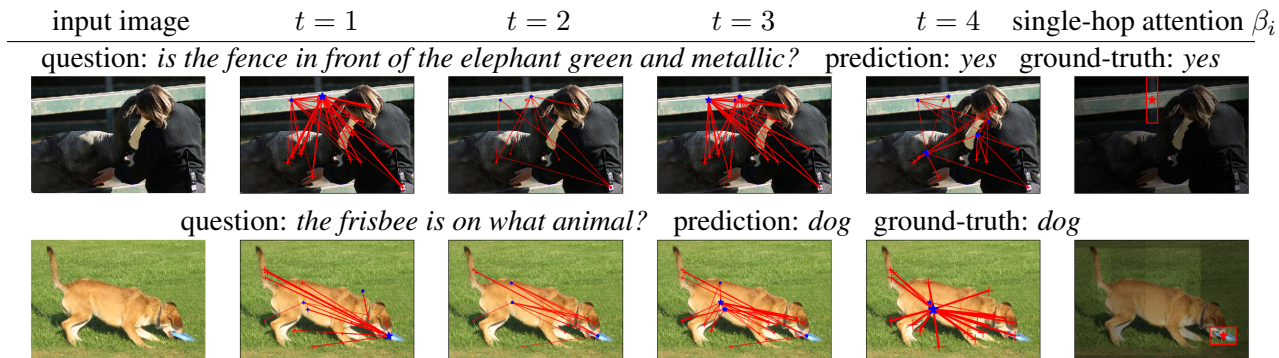


Figure 5.3: Examples from our LCGN model on the validation split of the GQA dataset for VQA. In the middle 4 columns, each red line shows an edge  $j \rightarrow i$  along the message passing paths (among the  $N$  detected objects) where the connection edge weight  $w_{j,i}^{(t)}$  exceeds a threshold. The blue star on each line is the sender node  $j$ , and the line width corresponds to its connection weight. In the upper example, the person, the elephant and the fence propagate messages with each other, and fence receives messages from the elephant in  $t = 4$ . In the lower example, the frisbee collect messages from the dog as contextual information in multiple rounds, and is picked up by the single-hop classifier. The red star (along with the box) in the last column shows the object with the highest single-hop attention  $\beta_i$  in Eqn. 5.9.

Method	Accuracy		
	val	test-dev	test
CNN+LSTM [76]	49.2%	–	46.6%
Bottom-Up [4]	52.2%	–	49.7%
MAC [75]	57.5%	–	54.1%
single-hop	62.0%	53.8%	54.4%
single-hop + LCGN (ours)	<b>63.9%</b>	<b>55.8%</b>	<b>56.1%</b>

Table 5.1: VQA performance on the GQA dataset.

compare with previous works.<sup>1</sup> Similar to MAC [75], we initialize question word embedding from GloVe [135] and maintain an exponential moving average of model parameters during training. To facilitate spatial reasoning, we concatenate the Faster R-CNN object detection features with their corresponding box coordinates. The results are shown in Table 5.1. By comparing “single-hop + LCGN” with “single-hop” in the last two rows, it can be seen that our LCGN model brings around 2% (absolute) improvement in accuracy, indicating that our LCGN model facilitates reasoning

<sup>1</sup>We learned from the GQA dataset authors that its *test-dev* and *test* splits were collected differently from its *train* and *val* splits, with a noticeable domain shift from *val* to *test-dev* and *test*. We train on the *train* split and report results on three GQA splits (*val*, *test-dev* and *test*). The performance of previous work on *val* was obtained from the dataset authors.

Method	Local features	Accuracy	
		val	test-dev
single-hop	convolutional	55.0%	48.6%
single-hop + LCGN	grid features	<b>55.3%</b>	<b>49.5%</b>
single-hop	object features	62.0%	53.8%
single-hop + LCGN	from detection	<b>63.9%</b>	<b>55.8%</b>
single-hop	GT objects	87.0%	n/a
single-hop + LCGN	and attributes <sup>2</sup>	<b>90.2%</b>	n/a

Table 5.2: Ablation on different local features on the GQA dataset.

by replacing the local features  $x^{loc}$  with the contextualized features  $x^{out}$  containing rich relational information for the reasoning task. Figure 5.3 shows question answering examples from our model on this dataset.

We compare with three previous approaches in Table 5.1. CNN+LSTM [76] and Bottom-Up [4] are simple fusion approaches between the text and the image, using the released GQA convolutional grid features or object detection features respectively. The MAC model [75] is a multi-step attention and memory model with specially designed control, reading and writing cells, and is trained on the same object detection features as our model. Our approach outperforms the MAC model that performs multi-step inference, obtaining the state-of-the-art results on the GQA dataset.

We further apply our LCGN model to other types of local features, and experiment with using either the same  $7 \times 7 \times 2048$ -dimensional convolutional grid features (where each  $x_i^{loc}$  is a feature map location and  $N = 49$ ) as used in CNN+LSTM in Table 5.1 or an “oracle” symbolic local representation at both training and test time, based on a set of ground-truth objects along with their class and attribute annotations (“GT objects and attributes”) in the scene graph data of the GQA dataset. In the latter setting with symbolic representation, we construct two one-hot vectors to represent each object’s class and attributes, and concatenate them as each object’s  $x_i^{loc}$ .<sup>2</sup> The results are shown in Table 5.2, where our LCGN model delivers consistent improvements over all three types of local feature representations.

**Evaluation on the CLEVR dataset.** We also evaluate our LCGN model on the CLEVR dataset [78], a dataset for VQA with complicated relational questions, such as *what number of other objects are there of the same size as the brown shiny object*. Following previous works, we use the  $14 \times 14 \times 1024$  convolutional grid features extracted from the C4 block of an ImageNet-pretrained

<sup>2</sup>In this setting, we can only evaluate on the *val* split with public scene graph annotations. We note that this is the only setting where we use the scene graphs in the GQA dataset. In all other settings, we only use the images and question-answer pairs to train our models. Also, our model does not rely on the GQA question semantic step annotations in any settings.

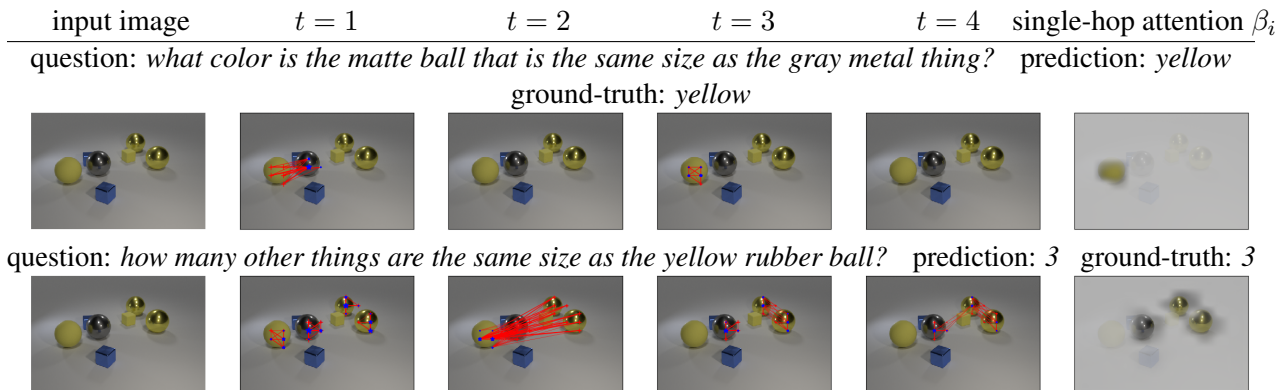


Figure 5.4: Examples from our LCGN model on the validation split of the CLEVR dataset for VQA. The middle 4 columns show the connection edge weights  $w_{j,i}^{(t)}$  similar to Figure 5.3, where the blue stars are the sender nodes. The last column shows the single-hop attention  $\beta_i$  in Eqn. 5.9 over the  $N = 14 \times 14$  feature grid. In the upper example, in  $t = 1$  the matte ball (leftmost) collects messages from the gray metal ball (of the same size), and then in  $t = 3$  messages are propagated within the convolutional grids on the matte ball, possibly to refine the collected context from the gray ball. In the lower example, in  $t = 1$  all four balls try to propagate messages within the convolutional grids of each ball region, and in  $t = 2$  the three other balls (of the same size) receive messages from the rubber ball (leftmost) and are picked up by the single-hop classifier.

ResNet-101 network [65] as the local features  $x^{loc}$  on the CLEVR dataset (*i.e.* each  $x_i^{loc}$  is a feature map location and  $N = 196$ ).

Similar to our experiments on the GQA dataset, we apply our LCGN model together with the single-hop answer classifier and compare it with using only the local features in the answer classifier. We also compare with previous works that use only question-answer pairs as supervision (*i.e.* without relying on the functional program annotations in [78]).

The results are shown in Table 5.3. It can be seen that the single-hop classifier only achieves 72.6% accuracy when using the local convolutional grid features  $x^{loc}$  (“**single-hop**”), which is unsurprising since the CLEVR dataset often involves resolving multiple and higher-order relations beyond the capacity of the single-hop classifier alone. However, when trained together with the context-aware representation  $x^{out}$  from our LCGN model, this same single-hop classifier (“**single-hop + LCGN**”) achieves a significantly higher accuracy of 97.9% comparable to several state-of-the-art approaches on this dataset, showing that our LCGN model is able to embed relational context information in its output scene representation  $x^{out}$ . Among previous works, Stack-NMN [70] and MAC [75] rely on multi-step inference procedures to predict an answer. RN [149] pools over all  $N^2$  pairwise object-object vectors to collect relational information in a single step. FiLM [136] modulates the batch normalization parameters of a convolutional network with the input question. NS-CL [116] learns symbolic representations of the scene and uses quasi-logical reasoning. Except for Stack-NMN [70], most previous works are tailored to the VQA task, and it is non-trivial to apply them to other tasks such as REF, while our LCGN model provides a generic scene representation

Method	Accuracy
Stack-NMN [70]	93.0%
RN [149]	95.5%
FiLM [136]	97.6%
MAC [75]	98.9%
NS-CL [116]	<b>99.2%</b>
single-hop	72.6%
single-hop + LCGN (ours)	97.9%

Table 5.3: VQA performance on the test split of the CLEVR dataset. We use  $T = 4$  rounds of message passing in our LCGN model.

applicable to multiple tasks. Figure 5.4 shows question answering examples of our model.

We further experiment with varying the number  $T$  of message passing iterations in our LCGN model. In addition, to isolate the effect of conditioning on textual inputs during message passing, we also train and evaluate a restricted version of LCGN without text conditioning (“**single-hop + LCGN w/o txt**”), by replacing the  $c_t$ ’s from Eqn 5.3 with a vector of all ones. The results are shown in Table 5.4, where it can be seen that using multiple rounds of iterations ( $T > 1$ ) leads to a significant performance increase, and it is crucial to incorporate the textual information  $c_t$  into the message passing procedure. This is likely because the CLEVR dataset involves complicated questions that need multi-step context propagation. In addition, it is more efficient to collect the specific relational context relevant to the input question, instead of building a scene representation with a complete and unconditional knowledge base of all relational information that any input questions can query from.

Given that multi-round message passing ( $T > 1$ ) works better than using only a single round ( $T = 1$ ), we further study whether it is beneficial to have dynamic connection weights  $w_{j,i}^{(t)}$  in Eqn. 5.5 that can be different in each iteration  $t$  to allow an object  $i$  to focus on different context objects  $j$  in different rounds. As a comparison, we train a restricted version of LCGN with static connection weights  $w_{j,i}$  (“**single-hop + LCGN w/ static  $w_{j,i}$** ”), where we only predict the weights  $w_{j,i}^{(1)}$  in Eqn. 5.5 for the first round  $t = 1$ , and reuse it in all subsequent rounds (*i.e.* setting  $w_{j,i}^{(t)} = w_{j,i}^{(1)}$  for all  $t > 1$ ). From the last row of Table 5.4 it can be seen that there is a performance drop when restricting to static connection weights  $w_{j,i}$  predicted only in the first round, and we also observe a similar (but larger) drop for the REF task in Sec. 5.4 and Table 5.5. This suggests that it is better to have dynamic connections during each iteration, instead of first predicting a fixed connection structure on which iterative message passing is performed (*e.g.* [129]).

Method	Steps $T$	Accuracy
single-hop	n/a	72.6%
single-hop + LCGN	$T = 1$	94.0%
single-hop + LCGN	$T = 2$	94.5%
single-hop + LCGN	$T = 3$	96.4%
single-hop + LCGN	$T = 4$	<b>97.9%</b>
single-hop + LCGN	$T = 5$	96.9%
single-hop + LCGN w/o txt	$T = 4$	78.6%
single-hop + LCGN w/ static $w_{j,i}$	$T = 4$	96.5%

Table 5.4: Ablation on iteration steps  $T$  and whether to condition on the text or have dynamic weights, on the CLEVR validation split.

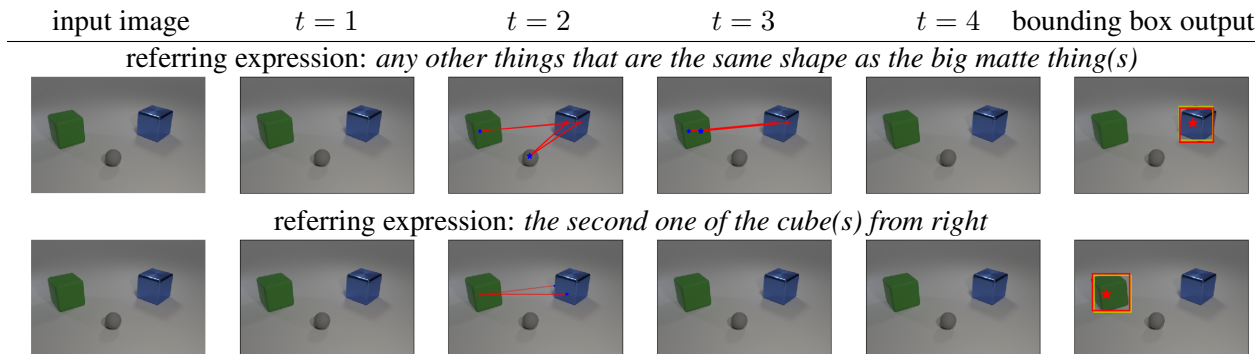


Figure 5.5: Examples from our LCGN model on the validation split of the CLEVR-Ref+ dataset for REF. The middle 4 columns show the connection edge weights  $w_{j,i}^{(t)}$  similar to Figure 5.3, where the blue stars are the sender nodes. The last column shows the selected target grid location  $p$  on the  $N = 14 \times 14$  spatial grid (the red star) in Eqn. 5.12, along with the ground-truth (yellow) box and the predicted box (red box from bounding box regression  $u$  in Eqn. 5.13). In the upper example, the blue cube (the target object) collects messages from the two other objects in  $t = 2$ , and then the blue cube further collects messages from the big matte green cube on the left (which has the same shape) in  $t = 3$ . In the lower example, the green cube checks for other cubes by collecting messages from things on its right in  $t = 2$ .

## Referring Expression Comprehension (REF)

Our LCGN model provides a generic approach to building context-aware scene representations and is not restricted to a specific task such as VQA. We also apply our LCGN model to the referring expression comprehension (REF) task, where given a referring expression that describes an object in the scene, the model is asked to localize the target object with a bounding box.

We experiment with the CLEVR-Ref+ dataset [104], which contains similar images as in the

Method	Accuracy
Stack-NMN [70]	56.5%
SLR [199]	57.7%
MAttNet [197]	60.9%
GroundeR [146]	61.7%
GroundeR + LCGN w/o txt	65.0%
GroundeR + LCGN w/ static $w_{j,i}$	71.4%
GroundeR + LCGN (ours)	<b>74.8%</b>

Table 5.5: Performance on the CLEVR-Ref+ dataset for REF.

CLEVR dataset [78] and complicated referring expressions requiring relation resolution. On the CLEVR-Ref+ dataset, we evaluate with the bounding box detection task in [104], where the output is a bounding box of the target object and there is only one single target object described by the expression. A localization is considered correct if it overlaps with the ground-truth box with at least 50% IoU. Same as in our VQA experiments on the CLEVR dataset in Sec. 5.4, here we also use the  $14 \times 14 \times 1024$  convolutional grid features from ResNet-101 C4 block as our local features  $x^{loc}$  (*i.e.* each  $x_i^{loc}$  is a feature map location and  $N = 196$ ), with  $T = 4$  rounds of message passing. The final target bounding box is predicted with a 4-dimensional bounding box offset vector  $u$  in Eqn. 5.13 from the selected grid location  $p$  in Eqn. 5.12.

We apply our LCGN model to build a context-aware representation  $x^{out}$  conditioned on the input referring expression, which is used as input to our implementation of the GroundeR approach [146] (Sec. 5.3) for bounding box prediction (“**GroundeR + LCGN**”). As a comparison, we train and evaluate the GroundeR model without our context-aware representation (“**GroundeR**”), using local features  $x^{loc}$  as inputs in Eqn. 5.11. Similar to our experiments on the CLEVR dataset for VQA in Sec. 5.4, we also ablate our LCGN model with not conditioning on the input expression in message passing (“**GroundeR + LCGN w/o txt**”) or using static connection weights  $w_{j,i}$  predicted from the first round (“**GroundeR + LCGN w/ static  $w_{j,i}$** ”).

The results are shown in Table 5.5, where our context-aware scene representation from LCGN leads to approximately 13% (absolute) improvement in REF accuracy. Consistent with our observation on the VQA task, for the REF task we find it important for the message passing procedure to depend on the input expression, and allowing the model to have dynamic connection weights  $w_{j,i}^{(t)}$  that can differ for each round  $t$ . Our model outperforms previous work by a large margin, achieving the state-of-the-art performance for REF on the CLEVR-Ref+ dataset. Figure 5.5 shows example predictions of our model on the CLEVR-Ref+ dataset.

In previous works, SLR [199] and MAttNet [197] are specifically designed for the REF task. SLR jointly trains an expression generation model (speaker) and an expression comprehension model (listener), and MAttNet relies on modular structure for subject, location and relation comprehension. While Stack-NMN [70] is also a generic approach that is applicable to both the VQA task and

the REF task, the major contribution of Stack-NMN is to construct an explicit step-wise inference procedure with compositional modules, and it relies on hand-designed module structures and local appearance-based scene representations. On the other hand, our work augments the scene representation with rich relational context. We show that our approach outperforms Stack-NMN on both the VQA and the REF tasks.

## 5.5 Discussion

In this work, we propose Language-Conditioned Graph Networks (LCGN), a generic approach to language-based reasoning tasks such VQA and REF. Instead of building task-specific inference procedures, our LCGN model constructs rich context-aware *representations* of the scene through iterative message passing. Experimentally, we show that the context-aware representations from our LCGN model can improve over the local appearance-based representations across various types of local features and multiple datasets, and it is crucial for the message passing procedure to depend on the language inputs.

## Chapter 6

# Iterative Pointer-Augmented Multimodal Transformers for TextVQA

### 6.1 Problem Statement

As a prominent task for visual reasoning, the Visual Question Answering (VQA) task [12] has received wide attention in terms of both datasets (*e.g.* [12, 58, 81, 78, 76]) and methods (*e.g.* [51, 4, 20, 84, 109]). However, these datasets and methods mostly focus on the visual components in the scene. On the other hand, they tend to ignore a crucial modality – text in the images – that carries essential information for scene understanding and reasoning. For example, in Figure 6.1, *deep water* on the sign warns people about the danger in the scene. To address this drawback, new VQA datasets [158, 23, 123] have been recently proposed with questions that explicitly require understanding and reasoning about text in the image, which is referred to as the TextVQA task.

The TextVQA task distinctively requires models to see, read and reason over three modalities: the input question, the visual contents in the image such as visual objects, and the text in the image. Several approaches [158, 23, 123, 22] have been proposed for the TextVQA task, based on OCR results of the image. In particular, LoRRA [158] extends previous VQA models [157] with an OCR attention branch and adds OCR tokens as a dynamic vocabulary to the answer classifier, allowing copying a single OCR token from the image as the answer. Similarly in [123], OCR tokens are grouped into blocks and added to the output space of a VQA model.

While these approaches enable reading text in images to some extent, they typically rely on custom pairwise multimodal fusion mechanisms between two modalities (such as single-hop attention over image regions and text tokens, conditioned on the input question), which limit the types of possible interactions between modalities. Furthermore, they treat answer prediction as a single-step classification problem – either selecting an answer from the training set answers or copying a text token from the image – making it difficult to generate complex answers such as book titles or signboard names with multiple words, or answers with both common words and specific image text tokens, such as *McDonald’s burger* where *McDonald’s* is from text in the image and *burger* is from the model’s own vocabulary. In addition, the word embedding based image text



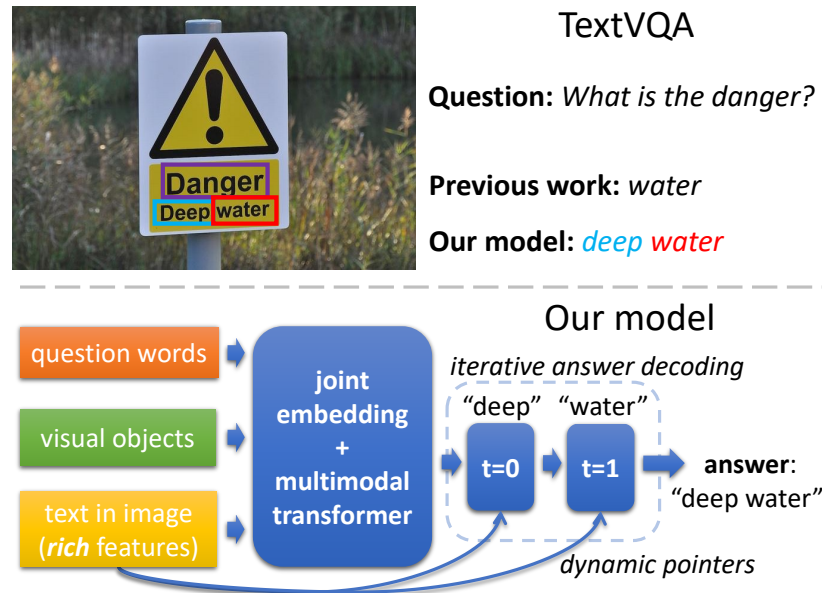


Figure 6.1: Compared to previous work (e.g. [158]) on the TextVQA task, our model, accompanied by rich features for image text, handles all modalities with a multimodal transformer over a joint embedding space instead of pairwise fusion mechanisms between modalities. Furthermore, answers are predicted through iterative decoding with pointers instead of one-step classification over a fixed vocabulary or copying single text token from the image.

features in previous work have limited representation power and miss important cues such as the appearance (e.g. font and color) and the location of text tokens in images. For example, tokens that have different fonts and are spatially apart from each other usually do not belong to the same street sign.

In this chapter, we address the above limitations with our novel Multimodal Multi-Copy Mesh (M4C) model for the TextVQA task, based on the transformer [176] architecture accompanied by iterative answer decoding through dynamic pointers, as shown in Figure 6.1. Our model naturally fuses the three input modalities and captures intra- and inter- modality interactions homogeneously within a multimodal transformer, which projects all entities from each modality into a common semantic embedding space, and applies the self-attention mechanism [132, 176] to collect relational representations for each entity. Instead of casting answer prediction as a classification task, we perform iterative answer decoding in multiple steps and augment our answer decoder with a dynamic pointer network that allows selecting text in the image in a permutation-invariant way without relying on any ad-hoc position indices in previous work such as LoRRA [158]. Furthermore, our model is capable of combining its own vocabulary with text in the image in a generated answer, as shown in examples in Figure 6.4 and 6.5. Finally, we introduce a rich representation for text tokens in the images based on multiple cues, including its word embedding, appearance, location, and character-level information.

Our contributions in this chapter are as follows: 1) We show that multiple (more than two)

input modalities can be naturally fused and jointly modeled through our multimodal transformer architecture. 2) Unlike previous work on TextVQA, our model reasons about the answer beyond a single classification step and predicts it through our pointer-augmented multi-step decoder. 3) We adopt a rich feature representation for text tokens in images and show that it is better than features based only on word embedding in previous work. 4) Our model significantly outperforms previous work on three challenging datasets for the TextVQA task: TextVQA [158] (+25% relative), ST-VQA [23] (+65% relative), and OCR-VQA [123] (+32% relative). Our code is available at <http://ronghanghu.com/m4c>.

## 6.2 Related Work

**VQA based on reading and understanding image text.** Recently, a few datasets and methods [158, 23, 123, 22] have been proposed for visual question answering based on text in images (referred to as the TextVQA task). LoRRA [158], a prominent prior work on this task, extends the Pythia [157] framework for VQA and allows it to copy a single OCR token from the image as the answer, by applying a single attention hop (conditioned on the question) over the OCR tokens and including the OCR token indices in the answer classifier’s output space. A conceptually similar model is proposed in [123], where OCR tokens are grouped into blocks and added to both the input features and the output answer space of a VQA model. In addition, a few other approaches [23, 22] enable text reading by augmenting existing VQA models with OCR inputs. However, these existing methods are limited by their simple feature representation of image text, multimodal learning approaches, and one-step classification for answer outputs. In this work, we address these limitations with our M4C model.

**Multimodal learning in vision-and-language tasks.** Early approaches on vision-and-language tasks often combined the image and text through attention over one modality conditioned on the other modality, such as image attention based on text (*e.g.* [192, 110]). Some approaches have explored multimodal fusion mechanisms such as bilinear models (*e.g.* [51, 84]), self-attention (*e.g.* [52]), and graph networks (*e.g.* [98]). Inspired by the success of Transformer [176] and BERT [44] architectures in natural language tasks, several recent works [109, 2, 171, 99, 97, 162, 207, 35] have also applied transformer-based fusion between image and text with self-supervision on large-scale datasets. However, most existing works treat each modality with a specific set of parameters, which makes them hard to scale to more input modalities. On the other hand, in our work we project all entities from each modality into a joint embedding space and treat them homogeneously with a transformer architecture over the list of all things. Our results suggest that joint embedding and self-attention are efficient when modeling multiple (more than two) input modalities.

**Dynamic copying with pointers.** Many answers in the TextVQA task come from text tokens in the image such as book titles or street signs. As it is intractable to have every possible text token in the answer vocabulary, copying text from the image would often be an easier option for answer prediction. Prior work has explored dynamically copying the inputs in different tasks such as text

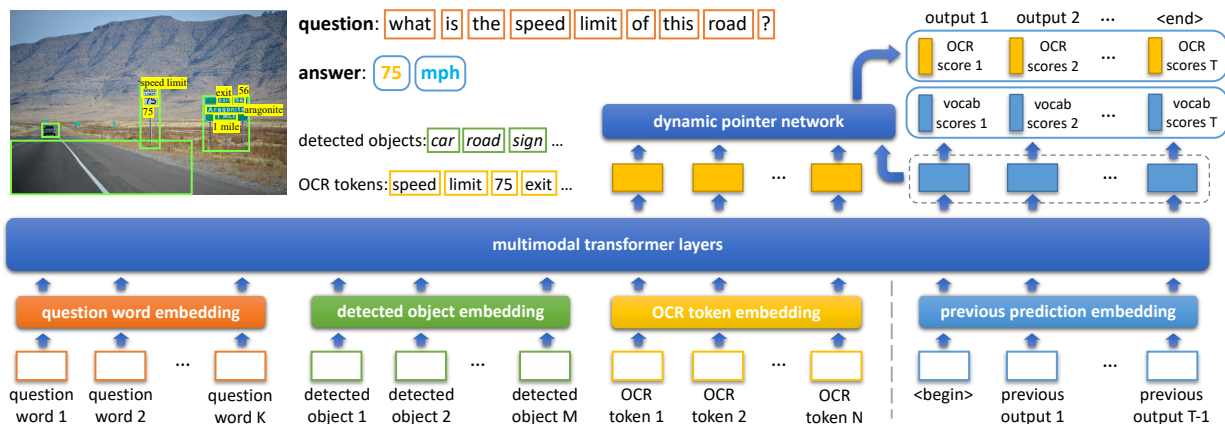


Figure 6.2: An overview of our M4C model. We project all entities (question words, detected visual objects, and detected OCR tokens) into a common  $d$ -dimensional semantic space through domain-specific embedding approaches and apply multiple transformer layers over the list of projected things. Based on the transformer outputs, we predict the answer through iterative auto-regressive decoding, where at each step our model either selects an OCR token through our dynamic pointer network, or a word from its fixed answer vocabulary.

summarization [152], knowledge retrieval [195], and image captioning [111] based on Pointer Networks [180] and its variants. For the TextVQA task, recent works [158, 123] have proposed to copy OCR tokens by adding their indices to classifier outputs. However, apart from their limitation of copying only a single token (or block), one drawback of these approaches is that they require a pre-defined number of OCR tokens (since the classifier has a fixed output dimension) and their output is dependent on the ordering of the tokens. In this work, we overcome this drawback using a permutation-invariant pointer network together with our multimodal transformer.

### 6.3 Multimodal Multi-Copy Mesh (M4C)

In this work, we present Multimodal Multi-Copy Mesh (M4C), a novel approach for the TextVQA task based on a pointer-augmented multimodal transformer architecture with iterative answer prediction. Given a question and an image as inputs, we extract feature representations from three modalities – the question, the visual objects in the image, and the text present in the image. These three modalities are represented respectively as a list of question words features, a list of visual object features from an off-the-shelf object detector, and a list of OCR token features based on an external OCR system.

Our model projects the feature representations of entities (in our case, question words, detected objects, and detected OCR tokens) from the three modalities as vectors in a learned common embedding space. Then, a multi-layer transformer [176] is applied on the list of all projected features, enriching their representations with intra- and inter- modality context. Our model learns to predict the answer through iterative decoding accompanied by a dynamic pointer network. During

decoding, it feeds in the previous output to predict the next answer component in an auto-regressive manner. At each step, it either copies an OCR token from the image, or selects a word from its fixed answer vocabulary. Figure 6.2 shows an overview of our model.

## A common embedding space for all modalities

Our model receives inputs from three modalities – question words, visual objects, and OCR tokens. We extract feature representations for each modality and project them into a common  $d$ -dimensional semantic space through domain-specific embedding approaches as follows.

**Embedding of question words.** Given a question as a sequence of  $K$  words, we embed these words into the corresponding sequence of  $d$ -dimensional feature vectors  $\{x_k^{\text{ques}}\}$  (where  $k = 1, \dots, K$ ) using a pretrained BERT model [44].<sup>1</sup> During training, the BERT parameters are fine-tuned using the question answering loss.

**Embedding of detected objects.** Given an image, we obtain a set of  $M$  visual objects through a pretrained detector (Faster R-CNN [144] in our case). Following prior work [4, 157, 158], we extract appearance feature  $x_m^{\text{fr}}$  using the detector’s output from the  $m$ -th object (where  $m = 1, \dots, M$ ). To capture its location in the image, we introduce a 4-dimensional location feature  $x_m^{\text{b}}$  from  $m$ -th object’s relative bounding box coordinates  $[x_{\min}/W_{\text{im}}, y_{\min}/H_{\text{im}}, x_{\max}/W_{\text{im}}, y_{\max}/H_{\text{im}}]$ , where  $W_{\text{im}}$  and  $H_{\text{im}}$  are image width and height respectively. Then, the appearance feature and the location feature are projected into the  $d$ -dimensional space with two learned linear transforms (where  $d$  is the same as in the question word embedding above), and are summed up as the final object embedding  $\{x_m^{\text{obj}}\}$  as

$$x_m^{\text{obj}} = \text{LN}(W_1 x_m^{\text{fr}}) + \text{LN}(W_2 x_m^{\text{b}}) \quad (6.1)$$

where  $W_1$  and  $W_2$  are learned projection matrices.  $\text{LN}(\cdot)$  is layer normalization [16], added on the output of the linear transforms to ensure that the object embedding has the same scale as the question word embedding. We fine-tune the last layer of the Faster R-CNN detector during training.

**Embedding of OCR tokens with rich representations.** Intuitively, to represent text in images, one needs to encode not only its characters, but also its appearance (*e.g.* color, font, and background) and spatial location in the image (*e.g.* words appearing on the top of a book cover are more likely to be book titles). We follow this intuition in our model and use a rich OCR representation consisting of four types of features, which is shown in our experiments to be significantly better than word embedding (such as FastText) alone in prior work [158]. After obtaining a set of  $N$  OCR tokens in an image through external OCR systems, from the  $n$ -th token (where  $n = 1, \dots, N$ ) we extract 1) a 300-dimensional FastText [25] vector  $x_n^{\text{ft}}$ , which is a word embedding with sub-word information, 2) an appearance feature  $x_n^{\text{fr}}$  from the same Faster R-CNN detector in the object detection above,

<sup>1</sup>In our implementation, we extract question word features from the first 3 layers of BERT-BASE. We find it sufficient to use its first few layers instead of using all its 12 layers, which saves computation.

extracted via RoI-Pooling on the OCR token’s bounding box, 3) a 604-dimensional Pyramidal Histogram of Characters (PHOC) [3] vector  $x_n^p$ , capturing what characters are present in the token – this is more robust to OCR errors and can be seen as a coarse character model, and 4) a 4-dimensional location feature  $x_n^b$  based on the OCR token’s relative bounding box coordinates  $[x_{\min}/W_{\text{im}}, y_{\min}/H_{\text{im}}, x_{\max}/W_{\text{im}}, y_{\max}/H_{\text{im}}]$ . We linearly project each feature into  $d$ -dimensional space, and sum them up (after layer normalization) as the final OCR token embedding  $\{x_n^{\text{ocr}}\}$  as below

$$x_n^{\text{ocr}} = \text{LN}(W_3x_n^{\text{ft}} + W_4x_n^{\text{fr}} + W_5x_n^p) + \text{LN}(W_6x_n^b) \quad (6.2)$$

where  $W_3, W_4, W_5$  and  $W_6$  are learned projection matrices and  $\text{LN}(\cdot)$  is layer normalization.

## Multimodal fusion and iterative answer prediction with pointer-augmented transformers

After embedding all entities (question words, visual objects, and OCR tokens) from each modality as vectors in the  $d$ -dimensional joint embedding space as described in Sec. 6.3, we apply a stack of  $L$  transformer layers [176] with a hidden dimension of  $d$  over the list of all  $K + M + N$  entities from  $\{x_k^{\text{ques}}\}$ ,  $\{x_m^{\text{obj}}\}$ , and  $\{x_n^{\text{ocr}}\}$ . Through the multi-head self-attention mechanism in transformers, each entity is allowed to freely attend to all other entities, regardless of whether they are from the same modality or not. For example, an OCR token is allowed to attend to another OCR token, a detected object, or a question word. This enables modeling both inter- and intra- modality relations in a homogeneous way through the same set of transformer parameters. The output from our multimodal transformer is a list of  $d$ -dimensional feature vectors for entities in each modality, which can be seen as their enriched embedding in multimodal context.

We predict an answer to the question through iterative decoding, using exactly the same transformer layers as a decoder. We decode the answer word by word in an auto-regressive manner for a total of  $T$  steps, where each decoded word may be either an OCR token in the image or a word from our fixed vocabulary of frequent answer words. As illustrated in Figure 6.2, at each step during decoding, we feed in an embedding of the previously predicted word, and predict the next answer word based on the transformer output with a dynamic pointer network.

Let  $\{z_1^{\text{ocr}}, \dots, z_N^{\text{ocr}}\}$  be the  $d$ -dimensional transformer outputs of the  $N$  OCR tokens in the image. Assume we have a vocabulary of  $V$  words that frequently appear in the training set answers. At the  $t$ -th decoding step, the transformer model outputs a  $d$ -dimensional vector  $z_t^{\text{dec}}$  corresponding to the input  $x_t^{\text{dec}}$  at step  $t$  (explained later in this section). From  $z_t^{\text{dec}}$ , we predict both the  $V$ -dimensional scores  $y_t^{\text{voc}}$  of choosing a word from fixed answer vocabulary and the  $N$ -dimensional scores  $y_t^{\text{ocr}}$  of selecting an OCR token from the image at decoding step  $t$ . In our implementation, the fixed answer vocabulary score  $y_{t,i}^{\text{voc}}$  for the  $i$ -th word (where  $i = 1, \dots, V$ ) is predicted as a simple linear layer as

$$y_{t,i}^{\text{voc}} = (w_i^{\text{voc}})^T z_t^{\text{dec}} + b_i^{\text{voc}} \quad (6.3)$$

where  $w_i^{\text{voc}}$  is a  $d$ -dimensional parameter for the  $i$ -th word in the answer vocabulary, and  $b_i^{\text{voc}}$  is a scalar parameter.

To select a token from the  $N$  OCR tokens in the image, we augment the transformer model with a dynamic pointer network, predicting a copying score  $y_{t,n}^{\text{ocr}}$  (where  $n = 1, \dots, N$ ) for each token via bilinear interaction between the decoding output  $z_t^{\text{dec}}$  and each OCR token’s output representation  $z_n^{\text{ocr}}$  as

$$y_{t,n}^{\text{ocr}} = (W^{\text{ocr}} z_n^{\text{ocr}} + b^{\text{ocr}})^T (W^{\text{dec}} z_t^{\text{dec}} + b^{\text{dec}}) \quad (6.4)$$

where  $W^{\text{ocr}}$  and  $W^{\text{dec}}$  are  $d \times d$  matrices, and  $b^{\text{ocr}}$  and  $b^{\text{dec}}$  are  $d$ -dimensional vectors.

During prediction, we take the argmax on the concatenation  $y_t^{\text{all}} = [y_t^{\text{voc}}; y_t^{\text{ocr}}]$  of fixed answer vocabulary scores and dynamic OCR-copying scores, selecting the top scoring element (either a vocabulary word or an OCR token) from all  $V + N$  candidates.

In our iterative auto-regressive decoding procedure, if the prediction at decoding time-step  $t$  is an OCR token, we feed in its OCR representation  $x_n^{\text{ocr}}$  as the transformer input  $x_{t+1}^{\text{dec}}$  to the next prediction step  $t + 1$ . Otherwise (the previous prediction is a word from the fixed answer vocabulary), we feed in its corresponding weight vector  $w_i^{\text{voc}}$  in Eqn. 6.3 as the next step’s input  $x_{t+1}^{\text{dec}}$ . In addition, we add two extra  $d$ -dimensional vectors as inputs – a positional embedding vector corresponding to step  $t$ , and a type embedding vector corresponding to whether the previous prediction is a fixed vocabulary word or an OCR token. Similar to machine translation, we augment our answer vocabulary with two special tokens, `<begin>` and `<end>`. Here `<begin>` is used as the input to the first decoding step, and we stop the decoding process after `<end>` is predicted.

To ensure causality in answer decoding, we mask the attention weights in the self-attention layers of the transformer architecture [176] such that question words, detected objects and OCR tokens cannot attend to any decoding steps, and all decoding steps can only attend to previous decoding steps in addition to question words, detected objects and OCR tokens. This is similar to prefix LM in [140].

## Training

During training, we supervise our multimodal transformer at each decoding step. Similar to sequence prediction tasks such as machine translation, we use teacher-forcing [95] (*i.e.* using ground-truth inputs to the decoder) to train our multi-step answer decoder, where each ground-truth answer is tokenized into a sequence of words. Given that an answer word can appear in both fixed answer vocabulary and OCR tokens, we apply multi-label sigmoid loss (instead of softmax loss) over the concatenated scores  $y_t^{\text{all}}$ .

## 6.4 Experiments

We evaluate our model on three challenging datasets for the TextVQA task, including TextVQA [158], ST-VQA [23], and OCR-VQA [123] (we use these datasets for research purposes only). Our model outperforms previous work by a significant margin on all the three datasets.

#	Method	Question enc. pretraining	OCR system	OCR token representation	Output module	Accu. on val	Accu. on test
1	LoRRA [158]	GloVe	Rosetta-ml	FastText	classifier	26.56	27.63
2	M4C w/o dec.	GloVe	Rosetta-ml	FastText	classifier	29.36	–
3	M4C w/o dec.	(none)	Rosetta-ml	FastText	classifier	29.55	–
4	M4C w/o dec.	BERT	Rosetta-ml	FastText	classifier	30.15	–
5	M4C w/o dec.	BERT	Rosetta-en	FastText	classifier	31.28	–
6	M4C w/o dec.	BERT	Rosetta-en	FastText + bbox	classifier	33.32	–
7	M4C w/o dec.	BERT	Rosetta-en	FastText + bbox + FRCN	classifier	34.38	–
8	M4C w/o dec.	BERT	Rosetta-en	FastText + bbox + FRCN + PHOC	classifier	<b>35.70</b>	–
9	M4C (ours - ablation)	(none)	Rosetta-ml	FastText + bbox + FRCN + PHOC	decoder	36.06	–
10	M4C (ours - ablation)	BERT	Rosetta-ml	FastText + bbox + FRCN + PHOC	decoder	37.06	–
11	M4C (ours)	BERT	Rosetta-en	FastText + bbox + FRCN + PHOC	decoder	<b>39.40</b>	39.01
12	DCD.ZJU (ensemble) [102]	–	–	–	–	31.48	31.44
13	MSFT.VTI [163]	–	–	–	–	32.92	32.46
14	M4C (ours; w/ ST-VQA)	BERT	Rosetta-en	FastText + bbox + FRCN + PHOC	decoder	<b>40.55</b>	<b>40.46</b>

Table 6.1: On the TextVQA dataset, we ablate our M4C model and show a detailed comparison with prior work LoRRA [158]. Our multimodal transformer (line 3 vs 1), our rich OCR representation (line 8 vs 5) and our iterative answer prediction (line 11 vs 8) all improve the accuracy significantly. Notably, our model still outperforms LoRRA by 9.5% (absolute) even when using fewer pretrained parameters (line 9 vs 1). Our final model achieves 39.01% (line 11) and 40.46% (line 14) test accuracy without and with the ST-VQA dataset as additional training data respectively, outperforming the challenge-winning DCD.ZJU method by 9% (absolute). See Sec. 6.4 for details.

## Evaluation on the TextVQA dataset

The TextVQA dataset [158] contains 28,408 images from the Open Images dataset [94], with human-written questions asking to reason about text in the image. Similar to VQAv2 [58], each question in the TextVQA dataset has 10 human annotated answers, and the final accuracy is measured via soft voting of the 10 answers.<sup>2</sup>

We use  $d = 768$  as the dimensionality of the joint embedding space and extract question word features with BERT-BASE using the 768-dimensional outputs from its first three layers, which are fine-tuned during training.

For visual objects, following Pythia [157] and LoRRA [158], we detect objects with a Faster R-CNN detector [144] pretrained on the Visual Genome dataset [90], and keeps 100 top-scoring objects per image. Then, the fc6 feature vector is extracted from each detected object. We apply the Faster R-CNN fc7 weights on the extracted fc6 features to output 2048-dimensional fc7 appearance features and fine-tune fc7 weights during training. However, we do not use the ResNet-152

<sup>2</sup>See <https://visualqa.org/evaluation> for details.

convolutional features [65] as in LoRRA.

Finally, we extract text tokens on each image using the Rosetta OCR system [26]. Unlike the prior work LoRRA [158] that uses a multilingual Rosetta version, in our model we use an English-only version of Rosetta that we find has higher recall. We refer to these two versions as **Rosetta-ml** and **Rosetta-en**, respectively. As mentioned in Sec. 6.3, from each OCR token we extract **FastText** [25] feature, appearance feature from Faster R-CNN (**FRCN**), **PHOC** [3] feature, and bounding box (**bbox**) feature.

In our multimodal transformer, we use  $L = 4$  layers of multimodal transformer with 12 attention heads. Other hyper-parameters (such as dropout ratio) follow BERT-BASE [44]. However, we note that the multimodal transformer parameters are initialized from scratch rather than from a pretrained BERT model. We use  $T = 12$  maximum decoding step in answer prediction unless otherwise specified, which is sufficient to cover almost all answers.

We collect the top 5000 frequent words from the answers in the training set as our answer vocabulary. During training, we use a batch size of 128, and train for a maximum of 24,000 iterations. Our model is trained using the Adam optimizer, with a learning rate of  $1e-4$  and a staircase learning rate schedule, where we multiply the learning rate by 0.1 at 14000 and at 19000 iterations. The best snapshot is selected using the validation set accuracy. The entire training takes approximately 10 hours on 4 Nvidia Tesla V100 GPUs.

As a notable prior work on this dataset, we show a step-by-step comparison with the LoRRA model [158]. LoRRA uses two single-hop attention layers over image visual features and OCR features. The attended visual and OCR features are then fused with a vector encoding of the question and fed into a single-step classifier to select either a frequent answer from the training set or a single OCR token from the image. Unlike our rich OCR representation in Sec. 6.3, in the LoRRA model each OCR token is only represented as a 300-dimensional FastText vector.

**Ablations on pretrained question encoding and OCR systems.** We first experiment with a restricted version of our model using the multimodal transformer architecture but without iterative decoding in answer prediction, *i.e.* **M4C (w/o dec.)** in Table 6.1. In this setting, we only decode for one step, and either select a frequent answer<sup>3</sup> from the training set or copy a single OCR token in the image as the answer. As a step-by-step comparison with LoRRA, we start with extracting OCR tokens from Rosetta-ml, representing OCR tokens only with FastText vectors, and initializing question encoding parameters in Sec. 6.3 from scratch (rather than from a pretrained BERT-BASE model). The result is shown in line 3 of Table 6.1. Compared with LoRRA in line 1, this restricted version of our model already outperforms LoRRA by around 3% (absolute) on TextVQA validation set. This result shows that our multimodal transformer architecture is more efficient for jointly modeling the three input modalities. We also experiment with initializing the word embedding from GloVe [135] as in LoRRA and the remaining parameters from scratch, shown in line 2. However, we find that this setting slightly under-performs initializing everything from scratch, which we suspect is due to different question tokenization between LoRRA and the BERT tokenizer used in

<sup>3</sup>In this case, we predict the entire (multi-word) answer, instead of a single word from our answer word vocabulary as in our full model.



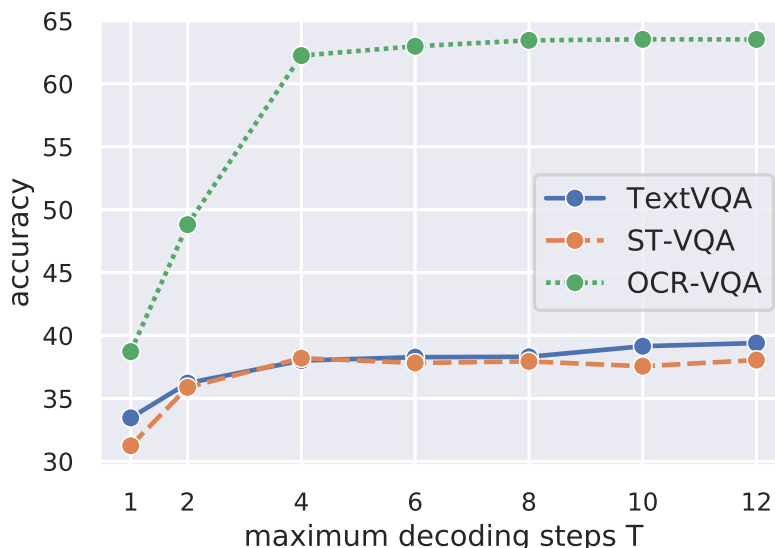


Figure 6.3: Accuracy under different maximum decoding steps  $T$  on the validation set of TextVQA, ST-VQA, and OCR-VQA. There is a major gap between single-step ( $T = 1$ ) and multi-step ( $T > 1$ ) answer prediction. We use 12 steps by default in our experiments.

our model. We then switch to a pretrained BERT for question encoding in line 4, and Rosetta-en for OCR extraction in line 5. Comparing line 3 to 5, we see that a pretrained BERT leads to around 0.6% higher accuracy, and Rosetta-en gives another 1% improvement.

**Ablations on OCR feature representation.** We analyze the impact of our rich OCR representation in Sec. 6.3 through ablations in Table 6.1 line 5 to 8. We see that OCR location (bbox) features and the RoI-pooled appearance features (FRCN) both improve the performance by a noticeable margin. In addition, we find that PHOC is also helpful as a character-level representation of the OCR token. Our rich OCR representation gives around 4% (absolute) accuracy improvement compare with using only FastText features as in LoRRA (line 8 vs 5). We note that our extra OCR features do not require more pretrained models, as we apply exactly the same Faster R-CNN model use in object detection for OCR appearance features, and PHOC is a manually-designed feature that does not need pretraining.

**Iterative answer decoding.** We then apply our full M4C model with iterative answer decoding to the TextVQA dataset. The results are shown in Table 6.1 line 11, which is around 4% (absolute) higher than its counterpart in line 8 using a single-step classifier and 13% (absolute) higher than LoRRA in line 1. In addition, we ablate our model using Rosetta-ml and randomly initialized question encoding parameters in line 9 and 10. Here, we see that our model in line 9 still outperforms LoRRA (line 1) by as much as 9.5% (absolute) when using the same OCR system as LoRRA and even fewer pretrained components. We also analyze the performance of our model with respect to the maximum decoding steps, shown in Figure 6.3, where decoding for multiple steps greatly

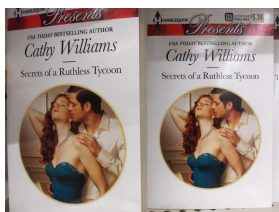


What does the light sign read on the farthest right window?

LoRRA: **exit**

M4C (ours): **bud light**

human: **bud light; all 2 liters**

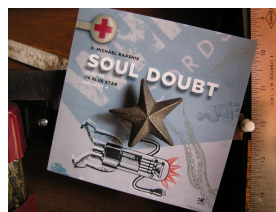


Who is usa today's best-selling author?

LoRRA: **roger zelazny**

M4C (ours): **cathy williams**

human: **cathy williams**



What is the name of the band?

LoRRA: **7**

M4C (ours): **soul doubt**

human: **soul doubt; h. michael karshis; unanswerable**



what is the time?

LoRRA: **1:45**

M4C (ours): **3:44**

human: **5:40; 5:41; 5:42; 8:00**

Figure 6.4: Qualitative examples from our M4C model on the TextVQA validation set (orange words are from OCR tokens and blue words are from fixed answer vocabulary). Compared to the previous work LoRRA [158] which selects one answer from training set or copies only a single OCR token, our model can copy multiple OCR tokens and combine them with its fixed vocabulary through iterative decoding.

improves the performance compared with a single step. Figure 6.4 shows qualitative examples (more examples in appendix) of our M4C model on the TextVQA dataset in comparison to LoRRA [158], where our model is capable of selecting multiple OCR tokens and combining them with its fixed vocabulary in predicted answers.

**Qualitative insights.** When inspecting the errors, we find that a major source of errors is OCR failure (*e.g.* in the last example in Figure 6.4, we find that the digits on the watch are not detected). This suggests that the accuracy of our model could be improved with better OCR systems, as supported by the comparison between line 10 and 11 in Table 6.1. Another possible future direction is to dynamically recognize text in the image based on the question (*e.g.* if the question asks about the price of a product brand, one may want to directly localize the brand name in the image). Some other errors of our model include resolving relations between objects and text or understanding large chunks of text in images (such as book pages). However, our model is able to correct a large number of mistakes in previous work where copying multiple text tokens is required to form an answer.

**TextVQA Challenge 2019.** We also compare to the winning entries in the TextVQA Challenge 2019.<sup>4</sup> We compare our method to DCD [102] (the challenge winner, based on ensemble) and MSFT\_VTI [163] (the top entry after the challenge), both relying on one-step prediction. We show that our single model (line 11) significantly outperforms these challenge winning entries on the

<sup>4</sup><https://textvqa.org/challenge>



What is the name of the street on which the Stop sign appears?

prediction: **45th parallel dr**

GT: **45th parallel dr**



What does the white sign say?

prediction: **tokyo station**

GT: **tokyo station**



How many cents per pound are the bananas?

prediction: **99**

GT: **99**



What kind of stop sign is in the image?

prediction: **stop all way**

GT: **all way**

Figure 6.5: Qualitative examples from our M4C model on the ST-VQA validation set (**orange** words from OCR tokens and **blue** words from fixed answer vocabulary). Our model can select multiple OCR tokens and combine them with its fixed vocabulary to predict an answer.

TextVQA test set by a large margin. We also experiment with using the ST-VQA dataset [23] as additional training data (a practice used by some of the previous challenge participants), which gives another 1% improvement and 40.46% final test accuracy – a new state-of-the-art on the TextVQA dataset.

## Evaluation on the ST-VQA dataset

The ST-VQA dataset [23] contains natural images from multiple sources including ICDAR 2013 [83], ICDAR 2015 [82], ImageNet [43], VizWiz [62], IIIT STR [122], Visual Genome [90], and COCO-Text [178].<sup>5</sup> The format of the ST-VQA dataset is similar to the TextVQA dataset in Sec. 6.4. However, each question is accompanied by only one or two ground-truth answers provided by the question writer. The dataset involves three tasks, and its Task 3 - Open Dictionary (containing 18,921 training-validation images and test 2,971 images) corresponds to our general TextVQA setting where no answer candidates are provided at test time.

The ST-VQA dataset adopts Average Normalized Levenshtein Similarity (ANLS)<sup>6</sup> as its official evaluation metric, defined as scores  $1 - d_L(a_{\text{pred}}, a_{\text{gt}}) / \max(|a_{\text{pred}}|, |a_{\text{gt}}|)$  (where  $a_{\text{pred}}$  and  $a_{\text{gt}}$  are prediction and ground-truth answers and  $d_L$  is edit distance) averaged over all questions. Also, all scores below the threshold 0.5 are truncated to 0 before averaging. To facilitate comparison, we report both accuracy and ANLS in our experiments.

As the ST-VQA dataset does not have an official split for training and validation, we randomly select 17,028 images as our training set and use the remaining 1,893 images as our validation set.

<sup>5</sup>We notice that many images from COCO-Text [178] in the downloaded ST-VQA data (around 1/3 of all images) are resized to  $256 \times 256$  for unknown reasons, which degrades the image quality and distorts their aspect ratios. In our experiments, we replace these images with their original versions from COCO-Text as inputs to object detection and OCR systems.

<sup>6</sup><https://rrc.cvc.uab.es/?ch=11&com=tasks>

#	Method	Output module	Accu. on val	ANLS on val	ANLS on test
1	SAN+STR [23]	–	–	–	0.135
2	VTA [22]	–	–	–	0.282
3	M4C w/o dec.	classifier	33.52	0.397	–
4	M4C (ours)	decoder	<b>38.05</b>	<b>0.472</b>	<b>0.462</b>

Table 6.2: On the ST-VQA dataset, our restricted model without decoder (M4C w/o dec.) already outperforms previous work by a large margin. Our final model achieves +0.18 (absolute) ANLS boost over the challenge winner, VTA [22]. See Sec. 6.4 for details.

We train our model on the ST-VQA dataset following exactly the same setting (line 11 in Table 6.1) as in our TextVQA experiments in Sec. 6.4, where we extract image text tokens using Rosetta-en, use FastText + bbox + FRCN + PHOC as our OCR representation, and initialize question encoding parameters from a pretrained BERT-BASE model. The results are shown in Table 6.2.

**Ablations of our model.** We train two versions of our model, one restricted version (M4C w/o dec. in Table 6.2) with a fixed one-step classifier as output module (similar to line 8 in Table 6.1) and one full version (M4C) with iterative answer decoding. Comparing the results of these two models, it can be seen that there is a large improvement from our iterative answer prediction mechanism.

**Comparison to previous work.** We compare with two previous methods on this dataset: 1) SAN+STR [23], which combines SAN for VQA [192] and Scene Text Retrieval [56] for answer vocabulary retrieval, and 2) VTA [22], the ICDAR 2019 ST-VQA Challenge<sup>6</sup> winner, based on BERT [44] for question encoding and BUTD [4] for VQA. From Table 6.2, it can be seen that our restricted model (M4C w/o dec.) already achieves higher ANLS than these two models, and our full model achieves as much as +0.18 (absolute) ANLS boost over the best previous work.

We also ablate the maximum copying number in our model in Figure 6.3, showing that it is beneficial to decode for multiple (as opposed to one) steps. Figure 6.5 shows qualitative examples of our model on the ST-VQA dataset.

## Evaluation on the OCR-VQA dataset

The OCR-VQA dataset [123] contains 207,572 images of book covers, with template-based questions asking about the title, author, edition, genre, year or other information about the book. Each question has a single ground-truth answer, and the dataset assumes that the answers to these questions can be inferred from the book cover images.

We train our model using the same hyper-parameters as in Sec. 6.4 and 6.4, but use  $2\times$  the total iterations and adapted learning rate schedule since the OCR-VQA dataset contains more images.

#	Method	Output module	Accu. on val	Accu. on test
1	BLOCK [123]	–	–	42.0
2	CNN [123]	–	–	14.3
3	BLOCK+CNN [123]	–	–	41.5
4	BLOCK+CNN+W2V [123]	–	–	48.3
5	M4C w/o dec.	classifier	46.3	–
6	M4C (ours)	decoder	<b>63.5</b>	<b>63.9</b>

Table 6.3: On the OCR-VQA dataset, we experiment with using either an iterative decoder (our full model) or a single-step classifier (M4C w/o dec.) as the output module, where our iterative decoder greatly improves the accuracy and largely outperforms the baseline methods. See Sec. 6.4 for details.



Figure 6.6: Qualitative examples from our M4C model on the OCR-VQA validation set (**orange** words from OCR tokens and **blue** words from fixed answer vocabulary).

The results are shown in Table 6.3. Compared to using a one-step classifier (M4C w/o dec.), our full model with iterative decoding achieves significantly better accuracy, which coincides with Figure 6.3 that having multiple decoding steps is greatly beneficial on this dataset. This is likely because the OCR-VQA dataset often contains multi-word answers such as book titles and author names.

We compare to four baseline approaches from [123], which are VQA systems based on 1) visual features from a convolutional network (CNN), 2) grouping OCR tokens into text blocks (BLOCK) with manually defined rules, 3) an averaged word2vec (W2V) feature over all the OCR tokens in the image, and 4) their combinations. Note that while the BLOCK baseline can also select multiple

OCR tokens, it relies on manually defined rules to merge tokens into groups and can only select one group as answer, while our method learns from data how to copy OCR tokens to compose answers. Compare to these baselines, our M4C has over 15% (absolute) higher test accuracy. Figure 6.6 shows qualitative examples of our model on this dataset.

## 6.5 Discussion

In this chapter, we present Multimodal Multi-Copy Mesh (M4C) for visual question answering based on understanding and reasoning about text in images. M4C adopts rich representations for text in the images, jointly models all modalities through a pointer-augmented multimodal transformer architecture over a joint embedding space, and predicts the answer through iterative decoding, outperforming previous work by a large margin on three challenging datasets for the TextVQA task. Our results suggest that it is efficient to handle multiple modalities through domain-specific embedding followed by homogeneous self-attention and to generate complex answers as multi-step decoding instead of one-step classification.

# Chapter 7

## Speaker-Follower Models for Instruction Following

### 7.1 Problem Statement

In the vision-and-language navigation task [5], an agent is placed in a realistic environment, and provided a natural language instruction such as “*Go down the stairs, go slight left at the bottom and go through door, take an immediate left and enter the bathroom, stop just inside in front of the sink*”. The agent must follow this instruction to navigate from its starting location to a goal location, as shown in Figure 7.1 (left). To accomplish this task the agent must learn to relate the language instructions to the visual environment. Moreover, it should be able to carry out new instructions in unseen environments.

Even simple navigation tasks require nontrivial *reasoning*: the agent must resolve ambiguous references to landmarks, perform a counterfactual evaluation of alternative routes, and identify incompletely specified destinations. While a number of approaches [120, 124, 185] have been proposed for the various navigation benchmarks, they generally employ a single model that learns to map directly from instructions to actions from a limited corpus of annotated trajectories.

In this chapter we treat the vision-and-language navigation task as a trajectory search problem, where the agent needs to find (based on the instruction) the best trajectory in the environment to navigate from the start location to the goal location. Our model involves an instruction interpretation (*follower*) module, mapping instructions to action sequences; and an instruction generation (*speaker*) module, mapping action sequences to instructions (Figure 7.1), both implemented with standard sequence-to-sequence architectures. The speaker learns to give textual instructions for visual routes, while the follower learns to follow routes (predict navigation actions) for provided textual instructions. Though explicit probabilistic reasoning combining speaker and follower agents is a staple of the literature on computational pragmatics [48], application of these models has largely been limited to extremely simple decision-making tasks like single forced choices.

We incorporate the speaker both at training time and at test time, where it works together with the learned instruction follower model to solve the navigation task (see Figure 7.2 for an overview



of our approach). At training time, we perform speaker-driven data augmentation where the speaker helps the follower by synthesizing additional route-instruction pairs to expand the limited training data. At test time, the follower improves its chances of success by looking ahead at possible future routes and pragmatically choosing the best route by scoring them according to the probability that the speaker would generate the correct instruction for each route. This procedure, using the external speaker model, improves upon planning using only the follower model. We construct both the speaker and the follower on top of a panoramic action space that efficiently encodes high-level behavior, moving directly between adjacent locations rather than making low-level visuomotor decisions like the number of degrees to rotate (see Figure 7.3).

To summarize our contributions: We propose a novel approach to vision-and-language navigation incorporating a visually grounded speaker–follower model, and introduce a panoramic representation to efficiently represent high-level actions. We evaluate this speaker–follower model on the Room-to-Room (R2R) dataset [5], and show that each component in our model improves performance at the instruction following task. Our model obtains a final success rate of 53.5% on the unseen test environment, an absolute improvement of 30% over existing approaches. Our code and data are available at [http://ronghanghu.com/speaker\\_follower](http://ronghanghu.com/speaker_follower).

## 7.2 Related Work

**Natural language instruction following.** Systems that learn to carry out natural language instructions in an interactive environment include approaches based on intermediate structured and executable representations of language [172, 32, 14, 107, 63] and approaches that map directly from language and world state observations to actions [27, 7, 120, 124]. The embodied vision-and-language navigation task studied in this chapter [5] differs from past situated instruction following tasks by introducing rich visual contexts. Recent work [185] has applied techniques from model-based and model-free reinforcement learning [186] to the vision-and-language navigation problem. Specifically, an environment model is used to predict a representation of the state resulting from an action, and planning is performed with respect to this environment model. Our work differs from this prior work by reasoning not just about state transitions, but also about the relationship between states and the language that describes them—specifically, using an external speaker model to predict how well a given sequence of states explains an instruction.

**Pragmatic language understanding.** A long line of work in linguistics, natural language processing, and cognitive science has studied *pragmatics*: how linguistic meaning is affected by context and communicative goals [60]. Our work here makes use of the Rational Speech Acts framework [48, 57], which models the interaction between speakers and listeners as a process where each agent reasons probabilistically about the other to maximize the chances of successful communicative outcomes. This framework has been applied to model human use of language [49], and to improve the performance of systems that generate [8, 117, 177, 39] and interpret [199, 112, 175] referential language. Similar modeling tools have recently been applied to generation and interpretation of language about sequential decision-making [50]. The present work makes use of a pragmatic





Figure 7.1: The task of vision-and-language navigation is to perform a sequence of actions (navigate through the environment) according to human natural language instructions. Our approach consists of an instruction *follower* model (left) and a *speaker* model (right).

instruction follower in the same spirit. Here, however, we integrate this with a more complex visual pipeline and use it not only at inference time but also at *training* time to improve the quality of a base listener model.

**Semi- and self-supervision.** The semi-supervised approach we use is related to model bootstrapping techniques such as self-training [151, 119] and co-training [24] at a high-level. Recent work has used monolingual corpora to improve the performance of neural machine translation models structurally similar to the sequence-to-sequence models we use [61, 65, 154]. In a grounded navigation context, [67] use a word-prediction task as training time supervision for a reinforcement learning agent. The approach most relevant to our work is the SEQ4 model [88], which applies semi-supervision to a navigation task by sampling new environments and maps (in synthetic domains without vision), and training an autoencoder to reconstruct routes, using language as a latent variable. The approach used here is much simpler, as it does not require constructing a differentiable surrogate to the decoding objective.

Semi-supervised data augmentation has also been widely used in computer vision tasks. In Data Distillation [139], additional annotation for object and key-point detection is obtained by ensembling and refining a pretrained model’s prediction on unannotated images. Self-play in adversarial groups of agents is common in multi-agent reinforcement learning [155, 164]. In actor-critic approaches [168, 169] in reinforcement learning, a critic learns the value of a state and is used to provide supervision to the actor’s policy during training. In this work, we use a speaker to synthesize additional navigation instructions on unlabeled new routes, and use this synthetic data from the speaker to train the follower.

**Grounding language in vision.** Existing work in visual grounding [137, 117, 74, 146, 127] has addressed the problem of *passively* perceiving a static image and mapping a referential expression to a bounding box [137, 117, 74] or a segmentation mask [73, 103, 197], exploring various techniques including proposal generation [33] and relationship handling [182, 127, 72, 38]. In our work, the vision-and-language navigation task requires the agent to *actively* interact with the environment to find a path to the goal following the natural language instruction. This can be seen as a grounding

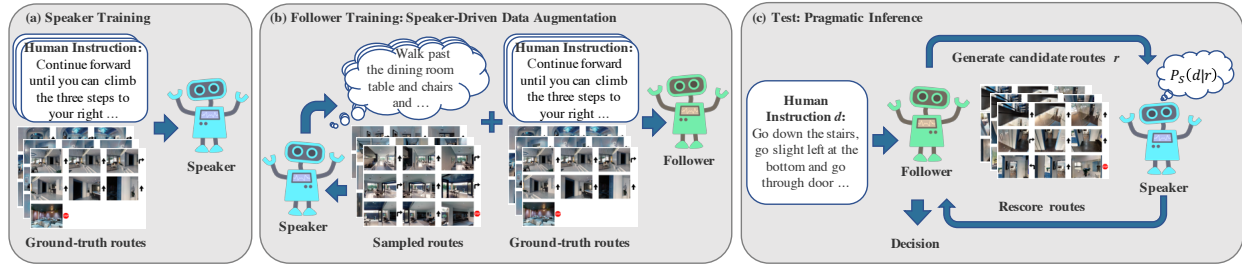


Figure 7.2: Our approach combines an instruction *follower* model and a *speaker* model. (a) The speaker model is trained on the ground-truth routes with human-generated descriptions; (b) it provides the follower with additional synthetic instruction data to bootstrap training; (c) it also helps the follower interpret ambiguous instructions and choose the best route during inference. See Sec. 7.3 for details.

problem in linguistics where the language instruction is grounded into a trajectory in the environment but requires more reasoning and planning skills than referential expression grounding.

### 7.3 Instruction Following with Speaker-Follower Models

To address the task of following natural language instructions, we rely on two models: an instruction-*follower* model of the kind considered in previous work, and a *speaker* model—a learned instruction generator that models how humans describe routes in navigation tasks.

Specifically, we base our follower model on the sequence-to-sequence model [5], computing a distribution  $P_F(r \mid d)$  over routes  $r$  (state and action sequences) given route descriptions  $d$ . The follower encodes the sequence of words in the route description with an LSTM [69], and outputs route actions sequentially, using an attention mechanism [17] over the description. Our speaker model is symmetric, producing a distribution  $P_S(d \mid r)$  by encoding the sequence of visual observations and actions in the route using an LSTM, and then outputting an instruction word-by-word with an LSTM decoder using attention over the encoded input route (Figure 7.1).

We combine these two base models into a speaker-follower system, where the speaker supports the follower both at training time and at test time. An overview of our approach is presented in Figure 7.2. First, we train a speaker model on the available ground-truth navigation routes and instructions. (Figure 7.2 (a)). Before training the follower, the speaker produces synthetic navigation instructions for novel sampled routes in the training environments, which are then used as additional supervision for the follower, as described in Sec. 7.3 (Figure 7.2 (b)). At follower test time, the follower generates possible routes as interpretations of a given instruction and starting context, and the speaker pragmatically ranks these, choosing one that provides a good explanation of the instruction in context (Sec. 7.3 and Figure 7.2 (c)). Both follower and speaker are supported by the panoramic action space in Sec. 7.3 that reflects the high-level granularity of the navigational instructions (Figure 7.3).

## Speaker-Driven Data Augmentation

The training data only covers a limited number of navigation instruction and route pairs, which we refer to as  $\mathcal{D} = (d_1, r_1) \dots (d_N, r_N)$ . To allow the agent to generalize better to new routes, we use the speaker to generate synthetic instructions on sampled new routes in the training environments. To create a synthetic training set, we sample a collection of  $M$  routes  $\hat{r}_1, \dots, \hat{r}_M$  through the training environments, using the same shortest-path approach used to generate the routes in the original training set [5]. We then generate a human-like textual instruction  $\hat{d}_k$  for each instruction  $\hat{r}_k$  by performing greedy prediction in the speaker model to approximate  $\hat{d}_k = \arg \max_d P_S(d | \hat{r}_k)$ .

These  $M$  synthetic navigation routes and instructions  $\mathcal{S} = (\hat{d}_1, \hat{r}_1), \dots, (\hat{d}_M, \hat{r}_M)$  are combined with the original training data  $\mathcal{D}$  into an augmented training set  $\mathcal{S} \cup \mathcal{D}$  (Figure 7.2(b)). During training, the follower is first trained on this augmented training set, and then further fine-tuned on the original training set  $\mathcal{D}$ . This speaker-driven data augmentation aims to overcome data scarcity issue, allowing the follower to learn how to navigate on new routes following the synthetic instructions.

## Speaker-Driven Route Selection

We use the base speaker ( $P_S$ ) and follower ( $P_F$ ) models described above to define a *pragmatic follower* model. Drawing on the Rational Speech Acts framework [48, 57], a pragmatic follower model should choose a route  $r$  through the environment that has high probability of having caused the speaker model to produce the given description  $d$ :  $\arg \max_r P_S(d | r)$  (corresponding to a rational Bayesian follower with a uniform prior over routes). Such a follower chooses a route that provides a good explanation of the observed description, allowing counterfactual reasoning about instructions, or using global context to correct errors in the follower’s path, which we call *pragmatic inference*.

Given the sequence-to-sequence models that we use, exactly solving the maximization problem above is infeasible; and may not even be desirable, as these models are trained discriminatively and may be poorly calibrated for inputs dissimilar to those seen during training. Following previous work on pragmatic language generation and interpretation [159, 8, 126, 50], we use a rescoring procedure: produce candidate route interpretations for a given instruction using the base follower model, and then rescore these routes using the base speaker model (Figure 7.2(c)).

Our pragmatic follower produces a route for a given instruction by obtaining  $K$  candidate paths from the base follower using a search procedure described below, then chooses the highest scoring path under a combination of the follower and speaker model probabilities:

$$\arg \max_{r \in R(d)} P_S(d | r)^\lambda \cdot P_F(r | d)^{(1-\lambda)} \quad (7.1)$$

where  $\lambda$  is a hyper-parameter in the range  $[0, 1]$  which we tune on validation data to maximize the accuracy of the follower.<sup>1</sup>

<sup>1</sup>In practice, we found best performance with values of  $\lambda$  close to 1, relying mostly on the score of the speaker to select routes. Using only the speaker score (which corresponds to the standard RSA pragmatic follower) did not substantially reduce performance compared to using a combination with the follower score, and both improved substantially upon using only the follower score (corresponding to the base follower).

**Candidate route generation.** To generate candidate routes from the base follower model, we perform a search procedure where candidate routes are produced incrementally, action-by-action, and scored using the probabilities given by  $P_F$ . Standard beam search in sequence-to-sequence models (e.g. [167]) forces partial routes to compete based on the number of actions taken. We obtain better performance by instead using a *state-factored* search procedure, where partial output sequences compete at the level of states in the environment, where each state consists of the agent’s location and discretized heading, keeping only the highest-scoring path found so far to each state. At a high-level, this search procedure resembles graph search with a closed list, but since action probabilities are non-stationary (potentially depend on the entire sequence of actions taken in the route), it is only approximate, and so we allow re-expanding states if a higher-scoring route to that state is found.

At each point in our state-factored search for searching and generating candidates in the follower model, we store the highest-probability route (as scored by the follower model) found so far to each state. States contain the follower’s discrete location and heading (direction it is facing) in the environment, and whether the route has been completed (had the STOP action predicted). The highest-scoring route, which has not yet been expanded (had successors produced), is selected and expanded using each possible action from the state, producing routes to the neighboring states. For each of these routes  $r$  with final state  $s$ , if  $s$  has not yet been reached by the search, or if  $r$  is higher-scoring under the model than the current best path to  $s$ ,  $r$  is stored as the best route to  $s$ . We continue the search procedure until  $K$  routes ending in distinct states have predicted the STOP action, or there are no remaining unexpanded routes.

Since route scores are products of conditional probabilities, route scores are non-increasing, and so this search procedure generates routes that do not pass through the same state twice—which we found to improve accuracy both for the base follower model and the pragmatic rescoring procedure, since instructions typically describe acyclic routes.

We generate up to  $K = 40$  candidate routes for each instruction using this procedure, and rescore using Eq. 7.1. In addition to enabling pragmatic inference, this state-factored search procedure improves the performance of the follower model on its own (taking the candidate route with highest score under the follower model), when compared to standard greedy search.

## Panoramic Action Space

The sequence-to-sequence agent in [5] uses low-level visuomotor control (such as turning left or right by 30 degrees), and only perceives frontal visual sensory input. Such fine-grained visuomotor control and restricted visual signal introduce challenges for instruction following. For example in Figure 7.3, to “turn left and go towards the sofa”, the agent needs to perform a series of turning actions until it sees a sofa in the center of its view, and then perform a “go forward” action. This requires strong skills of planning and memorization of visual inputs. While a possible way to address this challenge is to learn a hierarchical policy such as in [42], in our work we directly allow the agent to reason about high-level actions, using a panoramic action space with panoramic representation, converted with built-in mapping from low-level visuomotor control.

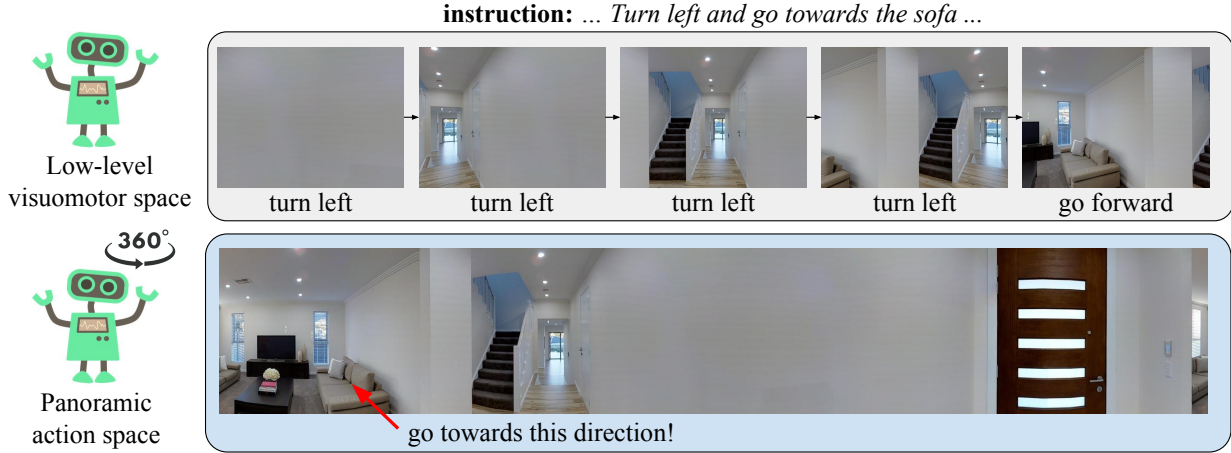


Figure 7.3: Compared with low-level visuomotor space, our panoramic action space (Sec. 7.3) allows the agents to have a complete perception of the scene, and to directly perform high-level actions.

As shown in Figure 7.3, in our panoramic representation, the agent first “looks around” and perceives a 360-degree panoramic view of its surrounding scene from its current location, which is discretized into 36 view angles (12 headings  $\times$  3 elevations with 30 degree intervals – in our implementation). Each view angle  $i$  is represented by an encoding vector  $v_i$ . At each location, the agent can only move towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction  $j$  represented by an encoding vector  $u_j$ . The encoding vectors  $v_i$  and  $u_j$  of each view angle and navigable direction are obtained by concatenating an appearance feature (ConvNet feature extracted from the local image patch around that view angle or direction) and a 4-dimensional orientation feature  $[\sin \psi; \cos \psi; \sin \theta; \cos \theta]$ , where  $\psi$  and  $\theta$  are the heading and elevation angles respectively. Also, we introduce a STOP action encoded by  $u_0 = \vec{0}$ . The agent can take this STOP action when it decides it has reached the goal location (to end the episode).

To make a decision on which direction to go, the agent first performs one-hop visual attention to look at all of the surrounding view angles, based on its memory vector  $h_{t-1}$ . The attention weight  $\alpha_{t,i}$  of each view angle  $i$  is computed as  $a_{t,i} = (W_1 h_{t-1})^T W_2 v_{t,i}$  and  $\alpha_{t,i} = \exp(a_{t,i}) / \sum_i \exp(a_{t,i})$ .

The attended feature representation  $v_{t,att} = \sum_i \alpha_{t,i} v_{t,i}$  from the panoramic scene is then used as visual-sensory input to the sequence-to-sequence model (replacing the 60-degree frontal appearance vector in [5]) to update the agent’s memory. Then, a bilinear dot product is used to obtain the probability  $p_j$  of each navigable direction  $j$ :  $y_j = (W_3 h_t)^T W_4 u_j$  and  $p_j = \exp(y_j) / \sum_j \exp(y_j)$ .

The agent then chooses a navigable direction  $u_j$  (with probability  $p_j$ ) to go to the adjacent location along that direction (or  $u_0$  to stop and end the episode). We use a built-in mapping that seamlessly translates our panoramic perception and action into visuomotor control such as turning and moving.

## 7.4 Experiments

### Experimental Setup

**Dataset.** We use the Room-to-Room (R2R) vision-and-language navigation dataset [5] for our experimental evaluation. In this task, the agent starts at a certain location in an environment and is provided with a human-generated navigation instruction, that describes a path to a goal location. The agent needs to follow the instruction by taking multiple discrete actions (e.g. turning, moving) to navigate to the goal location, and executing a “stop” action to end the episode. Note that differently from some robotic navigation settings [134], here the agent is not provided with a goal image, but must identify from the textual description and environment whether it has reached the goal.

The dataset consists of 7,189 paths sampled from the Matterport3D [30] navigation graphs, where each path consists of 5 to 7 discrete viewpoints and the average physical path length is 10m. Each path has three instructions written by humans, giving 21.5k instructions in total, with an average of 29 words per instruction. The dataset is split into training, validation, and test sets. The validation set is split into two parts: *seen*, where routes are sampled from environments seen during training, and *unseen* with environments that are not seen during training. All the test set routes belong to new environments unseen in the training and validation sets.

**Evaluation metrics.** Following previous work on the R2R task, our primary evaluation metrics are navigation error (NE), measuring the average distance between the end-location predicted by the follower agent and the true route’s end-location, and success rate (SR), the percentage of predicted end-locations within 3m of the true location. As in previous work, we also report the oracle success rate (OSR), measuring success rate at the closest point to the goal that the follower has visited along the route, allowing the agent to overshoot the goal without being penalized.

**Implementation details.** Following [5] and [185], we produce visual feature vectors  $v$  using the output from the final convolutional layer of a ResNet [65] trained on the ImageNet [147] classification dataset. These visual features are fixed, and the ResNet is not updated during training. To better generalize to novel words in the vocabulary, we also experiment with using GloVe embeddings [135], to initialize the word-embedding vectors in the speaker and follower.

In the baseline without using synthetic instructions, we train follower and speaker models using the human-generated instructions for routes present in the training set. The training procedure for the follower model follows [5] by training with *student-forcing* (sampling actions from the model during training, and supervising using a shortest-path action to reach the goal state). We use the training split in the R2R dataset to train our speaker model, using standard maximum likelihood training with a cross-entropy loss.

In speaker-driven data augmentation (Sec. 7.3), we augment the data used to train the follower model by sampling 178,000 routes from the training environments. Instructions for these routes are generated using greedy inference in the speaker model (which is trained only on human-produced instructions). The follower model is trained using student-forcing on this augmented data for 50,000 iterations, and then fine-tuned on the the original human-produced data for 20,000 iterations. For

Method	Validation-Seen			Validation-Unseen			Test (unseen)			
	NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑	TL ↓
Random	9.45	15.9	21.4	9.23	16.3	22.0	9.77	13.2	18.3	9.89
Student-forcing [5]	6.01	38.6	52.9	7.81	21.8	28.4	7.85	20.4	26.6	8.13
RPA [185]	5.56	42.9	52.6	7.65	24.6	31.8	7.53	25.3	32.5	9.15
ours	<b>3.08</b>	<b>70.1</b>	<b>78.3</b>	<b>4.83</b>	<b>54.6</b>	<b>65.2</b>	<b>4.87</b>	<b>53.5</b>	63.9	11.63
ours (challenge participation)*	–	–	–	–	–	–	<b>4.87</b>	<b>53.5</b>	<b>96.0</b>	1257.38
Human	–	–	–	–	–	–	1.61	86.4	90.2	11.90

Table 7.1: Performance comparison of our method to previous work. NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%) respectively (higher is better). Trajectory length (TL) on the test set is reported for completeness. \*: *When submitting to the Vision-and-Language Navigation Challenge, we modified our search procedure to maintain physical plausibility and to comply with the challenge guidelines. The resulting trajectory has higher oracle success rate while being very long.*

all experiments using pragmatic inference, we use a speaker weight of  $\lambda = 0.95$ , which we found to produce the best results on both the seen and unseen validation environments.

## Results and Analysis

We first examine the contribution from each of our model’s components on the validation splits. Then, we compare the performance of our model with previous work on the unseen test split.

### Component Contributions

We begin with a baseline (Row 1 of Table 7.2), which uses only a follower model with a non-panoramic action space at both training and test time, which is equivalent to the student-forcing model in [5].<sup>2</sup>

**Speaker-driven data augmentation.** We first introduce the speaker at training time for data augmentation (Sec. 7.3). Comparing Row 1 (the baseline follower model trained only with the original training data) against Row 2 (training this model on augmented data) in Table 7.2, we see that adding the augmented data improves success rate (SR) from 40.3% to 46.8% on validation seen and from 19.9% to 24.6% on validation unseen, respectively. This higher relative gain on unseen environments shows that the follower can learn from the speaker-annotated routes to better generalize to new scenes.

Note that given the noise in our augmented data, we fine-tune our model on *the original training data* at the end of training as mentioned in Sec. 7.3. We find that increasing the amount

<sup>2</sup>Note that our results for this baseline are slightly higher on val-seen and slightly lower on val-unseen than those reported by [5], due to differences in implementation details and hyper-parameter choices.

#	Data	Pragmatic	Panoramic	Validation-Seen			Validation-Unseen		
	Augmentation	Inference	Space	NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑
1				6.08	40.3	51.6	7.90	19.9	26.1
2	✓			5.05	46.8	59.9	7.30	24.6	33.2
3		✓		5.23	51.5	60.8	6.62	34.5	43.1
4			✓	4.86	52.1	63.3	7.07	31.2	41.3
5	✓	✓		4.28	57.2	63.9	5.75	39.3	47.0
6	✓		✓	3.36	66.4	73.8	6.62	35.5	45.0
7		✓	✓	3.88	63.3	71.0	5.24	49.5	63.4
8	✓	✓	✓	<b>3.08</b>	<b>70.1</b>	<b>78.3</b>	<b>4.83</b>	<b>54.6</b>	<b>65.2</b>

Table 7.2: Ablations showing the effect of each component in our model. Rows 2-4 show the effects of adding a single component to the baseline system (Row 1); Rows 5-7 show the effects of removing a single component from the full system (Row 8). NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%); higher is better. See Sec. 7.4 for details.

of augmented data is helpful in general. For example, when using 25% of the augmented data, the success rate improves to 22.8% on validation unseen, while with all the augmented data the success rate reaches 24.6% on validation unseen, which is a good balance between performance and computation overhead.

**Pragmatic inference.** We then incorporate the speaker at test time for pragmatic inference (Sec. 7.3), using the speaker to rescore the route candidates produced by the follower. Adding this technique brings a further improvement in success rate on both environments (compare Row 2, the data-augmented follower without pragmatic inference, to Row 5, adding pragmatic inference). This shows that when reasoning about navigational directions, large improvements in accuracy can be obtained by scoring how well the route explains the direction using a speaker model. Importantly, when using only the follower model to score candidates produced in search, the success rate is 49.0% on val-seen and 30.5% on val-unseen, showing the importance of using the speaker model to choose among candidates (which increases success rates to 57.2% and 39.3%, respectively).

**Panoramic action space.** Finally, we replace the visuomotor control space with the panoramic representation (Sec. 7.3). Adding this to the previous system (compare Row 5 and Row 8) shows that the new representation leads to a substantially higher success rate, achieving 70.1% and 54.6% success rate on validation seen and validation unseen, respectively. This suggests that directly acting in a higher-level representation space makes it easier to accurately carry out instructions. Our final model (Row 8) has over twice the success rate of the baseline follower in the unseen environments.



**Importance of all components.** Above we have shown the gain from each component, after being added incrementally. Moreover, comparing Rows 2-4 (adding each component independently to the base model) to the baseline (Row 1) shows that each component in isolation provides large improvements in success rates, and decreases the navigation error. Ablating each component (Rows 5-7) from the full model (Row 8) shows that each of them is important for the final performance.

**Qualitative results.** Here we provide qualitative examples further explaining how our model improves over the baseline. The intuition behind the speaker model is that it should help the agent more accurately interpret instructions specifically in ambiguous situations. Figure 7.4 shows how the introduction of a speaker model helps the follower with pragmatic inference.

### Comparison to Prior Work

We compare the performance of our final model to previous approaches on the R2R held-out splits, including the test split which contains 18 new environments that do not overlap with any training or validation splits, and are only seen once at test time.

The results are shown in Table 7.1. In the table, “Random” is randomly picking a direction and going towards that direction for 5 steps. “Student-forcing” is the best performing method in [5], using exploration during training of the sequence-to-sequence follower model. “RPA” [185] is a combination of model-based and model-free reinforcement learning (see also Sec. 7.2 for details). “ours” shows our performance using the route selected by our pragmatic inference procedure, while “ours (challenge participation)” uses a modified inference procedure for submission to the Vision-and-Language Navigation Challenge. Prior work has reported higher performance on the seen rather than unseen environments [5, 185], illustrating the issue of generalizing to new environments. Our method more than doubles the success rate of the state-of-the-art RPA approach, and on the test set achieves a final success rate of 53.5%. This represents a large reduction in the gap between machine and human performance on this task.



Figure 7.4: Navigation examples on unseen environments with and without pragmatic inference from the speaker model (*best visualized in color*). (a) The follower without pragmatic inference misinterpreted the instruction and went through a wrong door into a room with no bed. It then stopped at a table (which resembles a bed). (b) With the help of a speaker for pragmatic inference, the follower selected the correct route that enters the right door and stopped at the bed.

## 7.5 Discussion

The language-and-vision navigation task presents a pair of challenging reasoning problems: in language, because agents must interpret instructions in a changing environmental context; and in vision, because of the tight coupling between local perception and long-term decision-making. The comparatively poor performance of the baseline sequence-to-sequence model for instruction following suggests that more powerful modeling tools are needed to meet these challenges. In this work, we have introduced such a tool, showing that a follower model for vision-and-language navigation is substantially improved by carefully structuring the action space and integrating an explicit model of a *speaker* that predicts how navigation routes are described. We believe that these results point toward further opportunities for improvements in instruction following by modeling the global structure of navigation behaviors and the pragmatic contexts in which they occur.

## Chapter 8

# Conclusion

An intelligent machine must be able to perceive, understand, and reason jointly over both vision and language modalities. This thesis proposes a series of structured models for multiple vision-and-language reasoning tasks, with the goal of incorporating the proper architectural designs and structural inductive biases. In Chapter 2, we introduce the Compositional Modular Networks (CMNs) for the referring expression grounding task by decomposing the referring expressions into the subject-relationship-object triplet structure. In Chapter 3, we address the visual question answering task with our End-to-End Module Networks (N2NMNs) based on dynamic module layouts that reflect the reasoning structure of each question. We further propose the Stack Neural Module Networks (SNMNs) in Chapter 4 that automatically induce module layouts in a differentiable manner without resorting to human-annotated reasoning steps. In Chapter 5, we propose the Language-Conditioned Graph Networks (LCGNs) for relational reasoning based on context-aware representations of the visual scene with input-conditioned message propagation on graphs. In Chapter 6, we propose the iterative pointer-augmented multimodal transformers, using a pointer network to allow reading and copying text in images for reading comprehension and question answering. Finally, in Chapter 7 we propose the Speaker-Follower models for the navigational instruction following task, by decomposing the task into instruction following (with a follower) and instruction generation (with a speaker). In these scenarios above, we incorporate the corresponding reasoning structures into the model architectures, which largely outperform their unstructured counterparts as shown in our experiments.

While it is perhaps unsurprising that models with well-designed structures for each vision-and-language reasoning task achieve better performance and generalization compared to their generic, unstructured counterparts (such as a convolutional neural network on visual inputs, a recurrent neural network on language inputs, and a fusion network for final prediction), one question remains: how do we get the proper model structures and architectures for reasoning? In this thesis, we have obtained these structures through multiple sources. In CMN, N2NMN, and SNMN (Chapter 2, 3, and 4), the compositionality of language inputs and reasoning steps is the primary motivation for the modularity in these three models. LCGN (Chapter 5) is largely motivated by dynamically and efficiently propagating only those relations needed for reasoning, using the language inputs to drive the message propagation. The iterative pointer-augmented multimodal

transformers (Chapter 6) for TextVQA is designed based on the property and the requirement of the reading comprehension task – that it is necessary to output a text token from the image and it would be a lot easier if the model can directly copy it rather than predicting it from the model’s fixed vocabulary. The Speaker-Follower models (Chapter 7) for instruction following is largely inspired by pragmatic language understanding, where the task is decomposed into a listener-speaker pair. We anticipate that the above principles that have motivated the models presented in this thesis, including compositionality, modularity, dynamic message-passing conditioned on inputs, the reference to inputs based on pointers, pragmatic language understanding, and task decomposition, will also be crucial in designing the right architectures and model structures for other vision-and-language reasoning tasks, such as visual dialog, embodied question answering, and language-based multi-agent collaboration.

## Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv:1603.04467*, 2016.
- [2] Chris Alberti, Jeffrey Ling, Michael Collins, and David Reitter. Fusion of detected objects in text for visual question answering. *arXiv preprint arXiv:1908.05054*, 2019.
- [3] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566, 2014.
- [4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.
- [5] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [6] Jacob Andreas, Anca Dragan, and Dan Klein. Translating neuralese. In *ACL*, 2017.
- [7] Jacob Andreas and Dan Klein. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.
- [8] Jacob Andreas and Dan Klein. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, 2016.

- [9] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [10] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2016.
- [11] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [12] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.
- [13] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [14] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.
- [15] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [19] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3319–3327. IEEE, 2017.

- [20] Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2612–2620, 2017.
- [21] James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123, 2013.
- [22] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluís Gomez, Marçal Rusiñol, Minesh Mathew, CV Jawahar, Ernest Valveny, and Dimosthenis Karatzas. Icdar 2019 competition on scene text visual question answering. *arXiv preprint arXiv:1907.00490*, 2016.
- [23] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluís Gomez, Marçal Rusiñol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. Scene text visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [24] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.
- [25] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [26] Fedor Borisjuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. ACM, 2018.
- [27] S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 82–90. Association for Computational Linguistics, 2009.
- [28] Arthur W Burks, Don W Warren, and Jesse B Wright. An analysis of a logical machine using parenthesis-free notation. *Mathematical tables and other aids to computation*, 8(46):53–57, 1954.
- [29] Remi Cadene, Hedi Ben-younes, Matthieu Cord, and Nicolas Thome. Murel: Multimodal relational reasoning for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [30] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.



- [31] Simyung Chang, John Yang, SeongUk Park, and Nojun Kwak. Broadcasting convolutional network for visual relational reasoning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 754–769, 2018.
- [32] David L. Chen. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 430–439, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [33] Kan Chen, Rama Kovvuri, and Ram Nevatia. Query-guided regression network with context policy for phrase grounding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [34] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018.
- [35] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.
- [36] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [37] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [38] Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Phillippe Morency. Using syntax to ground referring expressions in natural images. In *32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.
- [39] Reuben Cohn-Gordon, Noah Goodman, and Chris Potts. Pragmatically informative image captioning with character-level reference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.
- [40] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24(48):7, 2001.
- [41] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

- [42] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [45] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017.
- [46] Włodzisław Duch, Rafał Adamczak, and Krzysztof Grabczewski. Extraction of logical rules from neural networks. *Neural Processing Letters*, 7(3):211–219, 1998.
- [47] Dario Floreano, Peter Dürri, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
- [48] Michael C Frank and Noah D Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998, 2012.
- [49] Michael C Frank, Noah D Goodman, Peter Lai, and Joshua B Tenenbaum. Informative communication in word production and word learning. In *Proceedings of the Annual Conference of the Cognitive Science Society*, 2009.
- [50] Daniel Fried, Jacob Andreas, and Dan Klein. Unified pragmatic models for generating and following instructions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.
- [51] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
- [52] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6639–6648, 2019.
- [53] Peng Gao, Hongsheng Li, Shuang Li, Pan Lu, Yikang Li, Steven CH Hoi, and Xiaogang Wang. Question-guided hybrid convolution for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 469–485, 2018.

- [54] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.
- [55] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [56] Lluís Gómez, Andrés Mafla, Marçal Rusinol, and Dimosthenis Karatzas. Single shot scene text retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 700–715, 2018.
- [57] Noah D Goodman and Andreas Stuhlmüller. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science*, 5(1):173–184, 2013.
- [58] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.
- [59] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.
- [60] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, San Diego, CA, 1975.
- [61] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.
- [62] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018.
- [63] Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.
- [64] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.
- [65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [66] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.
- [67] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. *CoRR*, abs/1706.06551, 2017.
- [68] Roei Herzig, Elad Levi, Huijuan Xu, Eli Brosh, Amir Globerson, and Trevor Darrell. Classifying collisions with spatio-temporal action graph networks. *arXiv preprint arXiv:1812.01233*, 2018.
- [69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [70] Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69, 2018.
- [71] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [72] Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [73] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [74] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [75] Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2018.
- [76] Drew A Hudson and Christopher D Manning. Gqa: a new dataset for compositional question answering over real-world images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [77] Allan Jabri, Armand Joulin, and Laurens van der Maaten. Revisiting visual question answering baselines. In *European Conference on Computer Vision*, pages 727–739. Springer, 2016.

- [78] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [79] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [80] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [81] Kushal Kafle and Christopher Kanan. An analysis of visual question answering algorithms. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1965–1973, 2017.
- [82] Dimosthenis Karatzas, Lluís Gomez-Bigorda, Angelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.
- [83] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluís Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluís Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013.
- [84] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574, 2018.
- [85] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [86] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [87] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.
- [88] Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. Semantic parsing with semi-supervised sequential autoencoders.

- In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas, November 2016. Association for Computational Linguistics.
- [89] Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.
- [90] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.
- [91] Jayant Krishnamurthy and Thomas Kollar. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206, 2013.
- [92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [93] Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Baby talk: Understanding and generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [94] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.
- [95] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.
- [96] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [97] Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*, 2019.
- [98] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering. *arXiv preprint arXiv:1903.12314*, 2019.
- [99] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

- [100] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.
- [101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.
- [102] Yuetan Lin, Hongrui Zhao, Yanan Li, and Donghui Wang. DCD\_ZJU, TextVQA Challenge 2019 winner. <https://visualqa.org/workshop.html>.
- [103] Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan Yuille. Recurrent multimodal interaction for referring image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [104] Runtao Liu, Chenxi Liu, Yutong Bai, and Alan Yuille. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [105] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott Reed. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [106] Yong Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Structure inference net: object detection using scene-level context and instance-level relationships. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6985–6994, 2018.
- [107] Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.
- [108] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [109] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, 2019.
- [110] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical co-attention for visual question answering. *Advances in Neural Information Processing Systems (NIPS)*, 2, 2016.
- [111] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Neural baby talk. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7219–7228, 2018.
- [112] Ruotian Luo and Gregory Shakhnarovich. Comprehension-guided referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [113] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems*, pages 1682–1690, 2014.
- [114] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask Your Neurons: A Deep Learning Approach to Visual Question Answering. *arXiv: 1605.02697*, 2016.
- [115] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.
- [116] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019.
- [117] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [118] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4942–4950, 2018.
- [119] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics, 2006.
- [120] Hongyuan Mei, Mohit Bansal, and Matthew Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2016.
- [121] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [122] Anand Mishra, Karteek Alahari, and CV Jawahar. Image retrieval using textual cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3040–3047, 2013.
- [123] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2019.



- [124] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.
- [125] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.
- [126] Will Monroe, Robert Hawkins, Noah Goodman, and Christopher Potts. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338, 2017.
- [127] Varun K Nagaraja, Vlad I Morariu, and Larry S Davis. Modeling context between objects for referring expression understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [128] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [129] Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pages 8344–8353, 2018.
- [130] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.
- [131] Clemens Otte. Safe and interpretable machine learning: a methodological review. In *Computational intelligence in intelligent data analysis*, pages 111–122. Springer, 2013.
- [132] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.
- [133] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*, 2016.
- [134] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. *arXiv preprint arXiv:1804.08606*, 2018.
- [135] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- [136] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

- [137] Bryan Plummer, Liwei Wang, Chris Cervantes, Juan Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [138] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–417, 2018.
- [139] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. *arXiv preprint arXiv:1712.04440*, 2017.
- [140] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.
- [141] Vasili Ramanishka, Abir Das, Jianming Zhang, and Kate Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.
- [142] RS Ramprasaath, D Abhishek, V Ramakrishna, C Michael, P Devi, and B Dhruv. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CVPR 2016*, 2016.
- [143] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [144] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.
- [145] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.
- [146] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [147] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

- [148] Fereshteh Sadeghi, Santosh K Kumar Divvala, and Ali Farhadi. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [149] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983, 2017.
- [150] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [151] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.
- [152] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.
- [153] Andrew D Selbst and Solon Barocas. The intuitive appeal of explainable machines. *SSRN*, 2018.
- [154] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96, 2016.
- [155] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- [156] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [157] Amanpreet Singh, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia-a platform for vision & language research. In *SysML Workshop, NeurIPS*, volume 2018, 2018.
- [158] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.
- [159] Nathaniel J Smith, Noah Goodman, and Michael Frank. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in neural information processing systems*, pages 3039–3047, 2013.

- [160] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.
- [161] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [162] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vi-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.
- [163] Anonymous submission. MSFT\_VTI, TextVQA Challenge 2019 top entry (post-challenge). <https://evalai.cloudcv.org/web/challenges/challenge-page/244/>.
- [164] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.
- [165] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.
- [166] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.
- [167] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [168] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [169] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.
- [170] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
- [171] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.
- [172] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2, 2011.

- [173] Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2017.
- [174] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- [175] Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. Object referring in visual scene with spoken language. In *Proc. IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2018.
- [176] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [177] Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. Context-aware captions from context-agnostic supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017.
- [178] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.
- [179] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [180] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.
- [181] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.
- [182] Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. Structured matching for phrase localization. In *European Conference on Computer Vision*, pages 696–711. Springer, 2016.
- [183] Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

- [184] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.
- [185] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. *arXiv:1803.07729*, 2018.
- [186] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.
- [187] Markus Werning, Wolfram Hinzen, and Edouard Machery. *The Oxford handbook of compositionality*. Oxford University Press, 2012.
- [188] Daan Wierstra, Faustino J Gomez, and Jürgen Schmidhuber. Modeling systems with internal state using evolino. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1795–1802. ACM, 2005.
- [189] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.
- [190] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer, 2016.
- [191] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.
- [192] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [193] Yiqun Yao, Jiaming Xu, Feng Wang, and Bo Xu. Cascaded mutual modulation for visual reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- [194] Mark Yatskar, Vicente Ordonez, and Ali Farhadi. Stating the obvious: Extracting visual common sense knowledge. In *Proceedings of NAACL-HLT*, 2016.

- [195] Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, and Dilek Hakkani-Tur. Deepcopy: Grounded response generation with hierarchical pointer networks. *arXiv preprint arXiv:1908.10731*, 2019.
- [196] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1039–1050, 2018.
- [197] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018.
- [198] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [199] Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. A joint speaker-listener-reinforcer model for referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7282–7290, 2017.
- [200] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 1821–1830, 2017.
- [201] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016.
- [202] Quanshi Zhang, Yu Yang, Ying Nian Wu, and Song-Chun Zhu. Interpreting cnns via decision trees. *arXiv preprint arXiv:1802.00121*, 2018.
- [203] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.
- [204] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929. IEEE, 2016.
- [205] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

- [206] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.
- [207] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. *arXiv preprint arXiv:1909.11059*, 2019.
- [208] Chen Zhu, Yanpeng Zhao, Shuaiyi Huang, Kewei Tu, and Yi Ma. Structured attentions for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1291–1300, 2017.
- [209] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*, pages 434–443, 2013.
- [210] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. *arXiv preprint arXiv:1511.03416*, 2015.
- [211] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.
- [212] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.