

博士學位論文

論文題目 Machine Intelligence that Understands
Visual and Linguistic Information and
Interacts with Humans and Environments

提出者 東北大学大学院情報科学研究科

システム情報科学専攻

氏名 Nguyen Van Quang

TOHOKU UNIVERSITY
Graduate School of Information Sciences

Machine Intelligence that Understands Visual and Linguistic
Information and Interacts with Humans and Environments
(視覚と言語情報を理解し人間や環境と作用し合う機械知能)

A dissertation submitted for the degree of Doctor of Philosophy (Information Sciences)
Department of System Information Sciences

by

Nguyen Van Quang

July 8, 2022

Machine Intelligence that Understands Visual and Linguistic Information and Interacts with Humans and Environments

Nguyen Van Quang

Abstract

Over the past years, Artificial Intelligence has witnessed significant progress in computer vision and natural language processing thanks to deep learning advancements. Inspired by remarkable success in these two independent fields, there has been a growing interest in the problems at the intersection of visual and linguistic understanding. It is believed that advances in solving those mentioned above and related problems would open the door to many real-world applications, bringing fundamental change to society. Take virtual assistants that aid the visually impaired, automatic surveillance systems for querying over visual databases, and in-home robots that perform household tasks as examples. Thus, the integration of vision and language is a viable approach to achieving one of AI's visionary goals: building machines that can understand both the visual and linguistic worlds, communicate with humans in natural language, and further interact with environments.

In this dissertation, we aim to build and improve agents endowed with such intelligence as a continuation of collective efforts by research communities. Specifically, we focus our attention on three representative vision language tasks, namely *image captioning*, *visual dialog*, and *interactive instruction following tasks*.

In the first part of the work, we revisit how to extract and utilize visual representations, aiming to build a better and faster model for image captioning. In image captioning, understanding visual information is crucial to correctly describing its content in words. Therefore, extracting good visual representations from the input image is necessary. Current state-of-the-art methods employ region-based features extracted by high-performance object detectors, e.g., Faster R-CNN. However, they have several issues, for example, the lack of contextual information, the risk of incorrect detection, and the high computational cost. The first two could be addressed by additionally using grid-based features. However, how to extract and integrate these two types of features was uncharted. We propose a transformer-only neural architecture, dubbed GRIT (Grid and Region-based Image captioning Transformer), that can effectively extract and integrate the two visual features to generate better captions for input images. Specifically, GRIT replaces the CNN-based detector employed in previous

methods with a DETR-based one, making it computationally faster and end-to-end trainable. We find that the proposed method brings about significant performance improvement, outperforming previous methods in inference accuracy and speed.

In the second part of this work, we tackle the visual dialog task, which requires agents to maintain a meaningful conversation with humans about the content of input images by answering questions. Unlike image captioning, the agent must handle multiple inputs, i.e., an image, a question, a dialog history, or even its individual dialog components. Thus, the key to success lies in how to model all the interactions between these inputs effectively and efficiently. We introduce a neural architecture, LTMI (dubbed Light-weight Transformer for Many Inputs), that can efficiently deal with all the interactions between multiple inputs in the visual dialog. It has a block structure similar to the Transformer and employs the same design for attention computation. With a similar setting on visual dialog, a layer built upon the proposed attention block has less than one-tenth of the parameters compared with its counterpart, a natural Transformer extension. It has only a small number of parameters yet has sufficient representational power for the purpose. The experimental results on the VisDial dataset validate the effectiveness of our proposed method.

In the last part of this work, we study interactive instruction-following tasks. An embodied AI agent is required to perform a sequence of actions to accomplish a complicated task in the interactive environment by following natural language directives. Recent studies have tackled the problem using ALFRED, a well-designed dataset for the task, but have obtained only very low accuracy. To this end, we propose a novel method based on a combination of several new ideas, which surpasses the existing methods by a large margin. One is a two-stage interpretation of the given instructions. The method first chooses and decodes an instruction without visual information, yielding a tentative sequence of object and action predictions. It then integrates this prediction with the visual information to generate the final prediction of an action and an object. It can localize the object of interest accurately from the input image. Furthermore, the proposed method utilizes multiple egocentric views of the environment and extracts crucial information by applying hierarchical attention conditioned on the selected instruction. It leads to better accuracy in predicting navigation actions. Our proposed method attains an unseen success rate of 8.37%.

Contents

Abstract	I
Table of Contents	i
List of Figures	v
List of Tables	viii
1 Introduction	1
1.1 Artificial Intelligence Overview	1
1.1.1 Artificial Intelligence Progress	1
1.1.2 Integration of Vision and Language	1
1.2 Our Research Problems	2
1.2.1 Image Captioning	3
1.2.2 Visual Dialog	4
1.2.3 Interactive Instruction Following	5
1.2.4 Research Questions	7
1.3 Dissertation Outline and Contributions	8
2 Preliminaries	11
2.1 Preliminary Background	11
2.1.1 Supervised Learning	11
2.1.2 Optimization	12
2.1.3 Deep Learning Workflow	14
2.2 Deep Neural Networks	15
2.2.1 FeedForward Neural Networks	16
2.2.2 Convolutional Neural Networks	16
2.2.3 Recurrent Neural Networks	19
2.2.4 Transformer Neural Networks	21
3 GRIT: Integrating Dual Visual Features for Image Captioning	23
3.1 Introduction	23
3.2 Related Work	25
3.2.1 Visual Representations for Image Captioning	25
3.2.2 Application of Transformer in Vision/Language Tasks	26
3.3 Proposed Method	27
3.3.1 Extracting Visual Features from Images	27

3.3.2	Caption Generation Using Dual Visual Features	30
3.4	Experiments	33
3.4.1	Datasets	33
3.4.2	Implementation Details	34
3.4.3	Training Details	35
3.4.4	Object Detection Results	36
3.4.5	Performance of Different Configurations	36
3.4.6	Results on the COCO Dataset	38
3.4.7	Results on the ArtEmis and nocaps Datasets	41
3.4.8	Computational Efficiency	42
3.4.9	Qualitative Results	44
3.5	Summary and Conclusion	44
4	LTMI: Lightweight Transformer for Many Inputs in Visual Dialog	49
4.1	Introduction	49
4.2	Related Work	51
4.2.1	Attention Mechanisms for Vision-Language Tasks	51
4.2.2	Visual Dialog	52
4.3	Lightweight Transformer for Many Utilities	53
4.3.1	Attention Mechanism of Transformer	53
4.3.2	Application to Bi-Modal Tasks	54
4.3.3	Lightweight Transformer for Many Inputs	55
4.3.4	Interactions between All Utilities	57
4.4	Implementation Details for Visual Dialog	57
4.4.1	Problem Definition	57
4.4.2	Representation of Utilities	57
4.4.3	Overall Network Design	61
4.4.4	Design of Decoders	62
4.4.5	Multi-Task Learning	64
4.5	Experiments on Visual Dialog	64
4.5.1	Experimental Setup	64
4.5.2	Comparison with State-of-the-art Methods	66
4.5.3	Ablation Study	69
4.5.4	Qualitative Results	71
4.6	Experiments on Audio Visual Scene-aware Dialog	73
4.6.1	Network Design	74
4.6.2	Experimental Setup	75
4.6.3	Experimental Results	75
4.7	Summary and Conclusion	75
5	LWIT: Improving Performance on Instruction Following Tasks	81
5.1	Introduction	81
5.2	Related Work	83
5.2.1	Embodied Vision-Language Tasks	83

5.2.2	Existing Methods for ALFRED	83
5.3	Proposed Method	84
5.3.1	Summary of ALFRED	84
5.3.2	Feature Representations	85
5.3.3	Instruction Decoder	86
5.3.4	Action Decoder	89
5.3.5	Mask Decoder	91
5.4	Experiments	92
5.4.1	Experimental Configuration	92
5.4.2	Experimental Results	94
5.4.3	Ablation Study	95
5.4.4	Qualitative Results	97
5.5	Analyses of Failure Cases	99
5.5.1	Navigation Failures	99
5.5.2	Manipulation Failures	101
5.6	Summary and Conclusion	101
6	Conclusion	103
	Bibliography	107
	Acknowledgments	127

List of Figures

1.1	From Image Classification to Image Captioning. Left) Predict an category for an image; Right) Generate a description in a sentence for the image.	3
1.2	From Visual Question Answering to Visual Dialog. Left) Answer a single question; Right) Answer multiple questions in dialog.	5
1.3	An example of the ALFRED task with highlighted frames corresponding to a portion of accompanying instructions. Source: [1].	6
1.4	Illustrations of the research problems with our corresponding proposed agents in this dissertation. Each proposed agent/model is encapsulated with a black bounding box, outputting the predictions highlighted in green. Top) We propose GRIT, a Grid- and Region-based Transformer for Image captioning. Middle) We propose LTMI, a lightweight transformer that handles multiple inputs in Visual Dialog. Bottom) We propose LWIT, a neural agent that performs household tasks following humans' instructions.	8
2.1	Examples of cat images. They are of different cat breed, position, size, color intensity, etc. Source: Kaggle dataset.	17
2.2	The architecture of LeNet-5, the first convolutional neural network introduced by Lecun et al. [2] for character recognition. The architecture of LeNet-5 consists of two convolutional layers, two subsampling (pooling) layers, and two fully connected layers. Source: [2].	17
2.3	Illustration of a Long Short-Term Memory cell.	20
2.4	(left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Source: [3]. . .	21
3.1	Comparison of GRIT and other region-based methods for image captioning. Left: Running time per image of performing inference with beam size of five and the maximum length of 20 on a V100 GPU. Right: Their architectures	24
3.2	Overview of the architecture of GRIT	27
3.3	Three designs of cross-attention mechanism to use dual visual features	31

3.4	Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.	45
3.5	Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.	46
3.6	Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.	47
3.7	Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.	48
4.1	(a) Source-to-target attention for bi-modal problems implemented by the standard Transformer block; the source Y is attended by weights computed from the similarity between the target X and Y . (b) The proposed block that can deal with many utilities; the source features $\{Y_1, \dots, Y_{U-1}\}$ are attended by weights computed between them and the target X . Shaded boxes have learnable weights	54
4.2	(a) Simplified symbol of the proposed block shown in Fig. 4.1(b). (b) Its application to Visual Dialog	56
4.3	The entire network built upon the proposed LTMI for Visual Dialog .	61
4.4	Examples of visualization for the attention weights generated in our model at two Q&A rounds on two images. See Sec. 4.5.4 for details.	73
4.5	Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.	76
4.6	Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.	77
4.7	Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.	78
4.8	Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.	79
5.1	Architecture overview of the proposed model. It consists of the modules encoding the visual inputs and the language directives (Sec. 5.3.2), the instruction decoder with an instruction selector (Sec. 5.3.3), the action decoder (Sec. 5.3.4), and the mask decoder (Sec. 5.3.5).	85
5.2	An example illustrates how we reinitialize the hidden states of the two LSTMs in the instruction encoder by s_{m_t} when $m_t = m_{t-1} + 1$ ($m_t = 4$).	87
5.3	Our agent completes an Examine task “ <i>Examine an empty box by the light of a floor lamp</i> ” in an unseen environment.	98

5.4	Our agent completes a Pick & Place task “ <i>Place the green bottle on the toilet basin</i> ” in an unseen environment.	98
5.5	Our agent completes a Pick Two & Place task “ <i>To move two bars of soap to the cabinet</i> ” in an unseen environment.	99
5.6	Our agent completes a Cool & Place task “ <i>Put chilled lettuce on the counter</i> ” in an unseen environment.	99
5.7	Our agent completes a Heat & Place task “ <i>Put a heated apple next to the lettuce on the middle shelf in the refrigerator</i> ” in an unseen environment.	100
5.8	The prediction masks generated by Shridhar <i>et al.</i> and our method where the agents are moved to the same location to accomplish Slice sub-goal.	100

List of Tables

3.1	Statistics of the pretraining datasets for object detection.	33
3.2	Performance of object detection on the COCO and Visual Genome datasets. ‘4DS’ denotes the four object detection datasets.	36
3.3	Results of ablation tests on the COCO test split. All the models are trained with the XE loss and finetuned by the CIDEr optimization	37
3.4	Offline results evaluated on the COCO Karpathy test split. ‘V. E. type’ indicates the type of visual features; ‘# VL Data’ is the number of image-text pairs used for vision-language pretraining.	39
3.5	Online evaluation results on the COCO image captioning dataset.	40
3.6	Performance on the ArtEmis and nocaps datasets.	41
3.7	The inference time on feature extraction of different methods.	43
3.8	The inference time on caption generation of different methods.	43
4.1	Hyper-paramters used in the training procedure.	66
4.2	Comparison of the performances of different methods on the validation set of VisDial v1.0 with discriminative and generative decoders.	66
4.3	Comparison in terms of single-model performance on the blind test-standard v1.0 split of the VisDial v1.0 dataset. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger	68
4.4	Comparison in terms of ensemble-model performance on the blind test-standard v1.0 split of the VisDial v1.0 dataset. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger	69
4.5	Comparison in terms of the number of parameters of the attention mechanism. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger	70
4.6	Ablation study on the components of our method on the val v1.0 split of VisDial dataset. \uparrow indicates the higher the better.	71
4.7	Ablation study on the components of our method on the val v1.0 split of VisDial dataset. \uparrow indicates the higher the better.	72

4.8	Comparison of response generation evaluation results with objective measures.	73
5.1	Task and Goal-Condition Success Rate. For each metric, the corresponding path weighted metrics are given in (parentheses). The highest values per fold and metric are shown in bold	93
5.2	Sub-goal success rate. All values are in percentage. The agent is evaluated on the Validation set. Highest values per fold are indicated in bold	94
5.3	Success rate across 7 task types. All values are in percentages. The agent is evaluated on the validation set. Highest values per split are indicated in bold	95
5.4	Ablation study for the components of the proposed model. We report the success rate (Task score) on the validation seen and unseen splits. The X mark denotes that a corresponding component is removed from the proposed model.	96
5.5	Results of an ablation test for examining the effectiveness of each component of the proposed model. The path weighted scores are reported in the parentheses.	97
5.6	Results of experiments comparing activation functions in the module for aggregating and encoding multi-view visual inputs. The path weighted scores are reported in the parentheses.	97

Chapter 1

Introduction

1.1 Artificial Intelligence Overview

1.1.1 Artificial Intelligence Progress

Artificial Intelligence (AI), or Machine Intelligence, has advanced rapidly in recent years. This achievement is arguably attributed to tremendous progress in AI sub-fields, including computer vision (CV) and natural language processing (NLP), thanks to deep learning advancements. Computer vision has seen many remarkable achievements in many tasks, such as image recognition [4–7], object detection [8], semantic segmentation [9], using large labeled datasets [10], or employing self-supervision [11] on large-scale unlabeled data. Similarly, NLP has experienced unprecedented studies using deep learning, achieving remarkable performance on many downstream tasks powered by neural networks pre-trained on large-scale text corpora [12–14]. Consequently, there is also a growing interest in the problems at the intersection between these two independent fields that require visual and linguistic understanding.

1.1.2 Integration of Vision and Language

Integrating the two fields of computer vision and natural language processing is a viable approach toward one long-standing goal of AI: building machines that can perceive the worlds of vision and language, communicate with humans in natural language, and further interact with the physical environments around us. Inspired

by the tremendous success in CV and NLP, an increasing amount of attention has been paid to the problems lying at the intersection between vision and language domains, taking further steps towards this visionary goal. Many pilot tasks in this intersecting region have been designed and introduced to the research community, together with datasets. It remains challenging as the tasks require future machine intelligence not only to (1) acquire a comprehensive understanding of visual and/or linguistic information but also to (2) generate descriptions or stories about the visual content [15, 16], (3) specify salient regions and objects and their relationships in the image to reason about, or answer arbitrary questions about its content [17, 18], (4) navigate through and interact with physical environments by leveraging natural language instructions [1, 19–21], etc. Methods that can deal with and translate between various modalities (for example, visual and linguistic inputs) are classified as a sub-category of multi-modal models, which were originally described [22].

Practical Applications Advances in tackling the aforementioned and other related challenges are expected to open the door to a broad range of practical applications, bringing about a radical change in society. For example, the visually impaired can be helped better by a future generation of virtual assistants. They can obtain helpful information about a scene from generated descriptions and by being able to ask questions about it. It can be used in automatic surveillance for querying from enormous image and video databases using natural language and in personal navigation systems that can process or even generate navigation instructions in natural language [18, 19]. It also includes in-home robots that perform household tasks [1]. Lastly, these problems play a critical role in evaluating the performance of AI systems that contribute to the progress of designing better machines with more collectively comprehensive intelligence than independent ones in CV and NLP.

1.2 Our Research Problems

In this dissertation, we aim to build and improve agents endowed with such the intelligence as a continuation of collective efforts by research communities. In particular, we study three representative vision language tasks, namely **image caption-**

ing [15], **visual dialog** [18], and **interactive instruction following** [1].

Challenges When tackling tasks that require visual and linguistic understanding with the translation between the two modalities, humans can do it with ease. For example, a human can point out and convey an enormous quantity of information about a visual scene with just a cursory look at it. On the other hand, computers find it difficult when they deal with unstructured data, e.g., images and text. Next, we will describe the three tasks, the challenges we confront when designing agents, and the shortcomings of previous studies, and summarize our proposed approaches.

1.2.1 Image Captioning

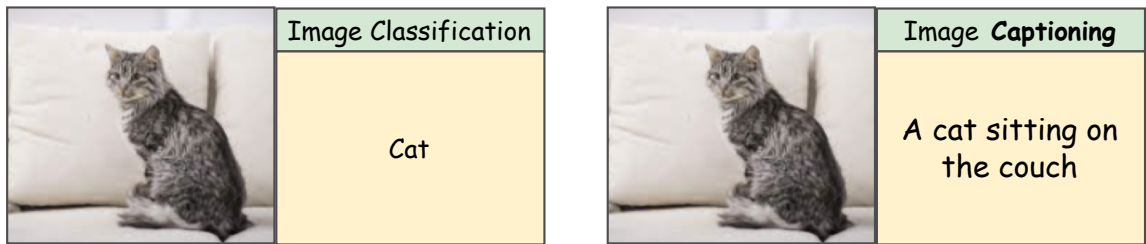


Figure 1.1: From Image Classification to Image Captioning. Left) Predict an category for an image; Right) Generate a description in a sentence for the image.

The task of image captioning requires generating a sentence describing the content of a scene for a given image. We usually represent an image by an array of pixels with intensity values in the color channel(s) (e.g., RGB images have three color channels while black-white images have a single channel); a typical image might have millions of pixels. To recognize the objects in the image (e.g., a cat as in Fig.1.1), the agent must convert these raw intensity values into high-level concepts of objects. On the other hand, many objects share similar low-level patterns (e.g., from cats and dogs to carpets and coats, they all have fur). It makes the recognition of objects in the image by manually programming unattainable.

Generating a sentence in natural language is also a challenging task. Unlike the object recognition task that assigns one or a few labels to the image, computers must output a sequence of words in a large vocabulary to reflect their understanding of the visual content (see Fig. 1.1). Therefore, image captioning requires a complex pattern

recognition process of identifying salient objects and regions and annotating them with a sequence of integers to represent words in the caption.

It is believed in the community that extracting visual representations from an input image plays a crucial role in generating better captions. Identifying existing objects and their relationships in the image is especially beneficial for precisely describing its visual content. The state-of-the-art methods utilize the region-based features obtained from CNN-based detectors, such as Faster R-CNN [8], since they encode detected objects directly. However, the region-based features have several issues, such as a lack of contextual information, risk of false detection, and expensive computation. The grid-based features are the high-level feature maps extracted from the entire image. They thus represent contextual information while being free from the risk of incorrect object detection.

In this dissertation, we revisit how to extract these dual visual features from input images and how to combine such region and grid features in an integrated manner, aiming to build a better and faster model for image captioning. The underlying idea is that appropriate integration of the two visual features will provide a better representation of the input image since they are complementary, as explained above. In particular, in Chapter 3, we propose a Transformer-only neural architecture, dubbed GRIT (Grid and Region-based Image captioning Transformer), that effectively utilizes the two visual features to generate better captions. GRIT replaces the CNN-based detector employed in previous methods with a DETR-based one, making it computationally faster. Moreover, the monolithic design consisting only of Transformers makes it end-to-end trainable. We find that the proposed method obtains considerable performance gain, surpassing previous methods in both inference accuracy and speed.

1.2.2 Visual Dialog

Unlike the image captioning task, where agents perform only one-way communication by returning only the image description to humans, visual dialog agents communicate with humans in a two-way manner (see Fig. 1.4). Specifically, the task of visual dialog demands an agent to maintain a meaningful conversation with humans in natural language by answering questions about the visual content. Visual dialog

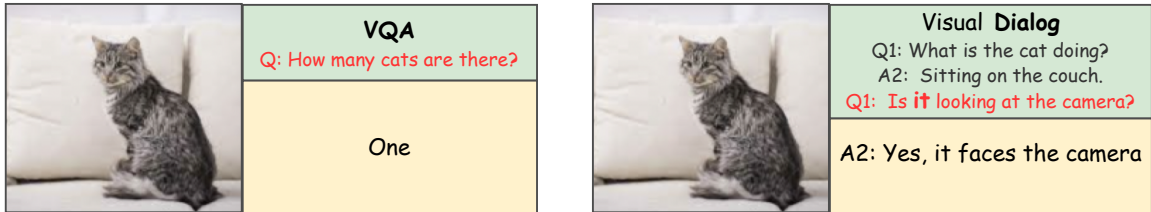


Figure 1.2: From Visual Question Answering to Visual Dialog. Left) Answer a single question; Right) Answer multiple questions in dialog.

has been developed aiming at a higher level of vision-language interactions [18], as compared with VQA (visual question answering) [17] and VCR (visual commonsense reasoning); see Figure 1.2. It extends VQA to multiple rounds; given an image and a history of question-answer pairs about the image, an agent is asked to answer a new question. This task requires multiple sub-problems ranging from visual understanding to natural language tasks such as language generation, co-reference resolution, etc. For example, to answer the question ‘*What color are they?*’, the agent needs to understand the context from a dialog history to know what ‘*they*’ refers to and look at the relevant image region to find out the color.

The key to success in visual dialog lies in how to model the interactions between multiple inputs, i.e., the input image, the dialog history, and a question. To this end, we propose in Chapter 4 a neural architecture named Light-weight Transformer for Many Inputs (LTMI) that can efficiently deal with all the interactions between multiple such inputs in visual dialog. It has a block structure similar to the Transformer and employs the same design for attention computation. In contrast, it has only a small number of parameters yet has sufficient representational power for the purpose. With a similar setting on visual dialog, a layer built upon the proposed attention block has less than one-tenth of the parameters compared with its counterpart, a natural Transformer extension. The experimental results on the VisDial dataset validate the effectiveness of the proposed method.

1.2.3 Interactive Instruction Following

The image captioning and visual dialog problems are beneficial for stimulating and evaluating progress on the problems involving both vision and language domains.



Figure 1.3: An example of the ALFRED task with highlighted frames corresponding to a portion of accompanying instructions. Source: [1].

They do, however, share a flaw: they are both passive. In these tasks, the agent is not allowed to move or manipulate the camera or interact with the environment. Therefore, the visual inputs are static, i.e., fixed images. It underestimates one of the most important aspects of several practical applications stated in Section 1.1.2, as each of these examples (e.g., in-home robots) requires an embodied agent.

In the last part of our work, we focus on the interactive instruction following tasks. We examine a more complex problem by addressing a recently constructed benchmark known as ALFRED [1]. An agent must do a household task in an interactive environment by following verbal directions. In comparison with a close problem called vision-language navigation (VLN) [19], ALFRED is more difficult because the agent must (1) reason over a larger number of instructions and (2) predict actions from a wider action space in order to complete a task across longer time horizons. The agent must also (3) predict the pixel-wise masks to localize the objects of interest. Previous studies (e.g., [1]) employ a Seq2Seq model, which performs well on the VLN tasks [23]. However, it does not work well on ALFRED. Consequently, previous approaches have poor performance and a large gap compared with humans.

In Chapter 5, we present a new method, which surpasses the prior methods by a significant margin. We propose an embodied agent based on several new ideas.

One is a two-stage interpretation of the given instructions. The method first selects and decodes an instruction without visual information, yielding a tentative sequence prediction of objects and actions. It then combines this prediction with the visual information, obtaining the final prediction of an action and an object with better accuracy. Furthermore, our method utilizes multiple ego-centric views and extracts crucial information using hierarchical attention conditioned on the selected instruction.

1.2.4 Research Questions

As a result, we will address three main research questions (**RQs**) while studying the three research problems:

- RQ₁**: In vision-language tasks, understanding visual information is of importance. It is reasonably valid for image captioning, in which agents must grasp visual information before describing an input image in words. To this end, we need a form of representation for visual inputs so that neural networks can learn helpful information to caption an image effectively. We raise a question: **how do we extract good visual representations from input images?**
- RQ₂**: In **RQ₁**, neural networks process only the visual inputs. Meanwhile, visual dialog tasks require neural networks to gain understanding from both visual and linguistic inputs in order to answer questions. We deal with multiple inputs in these tasks, e.g., an image, a dialog history, and a question. It is thus essential to address a question: **how do we model the interactions between multiple inputs efficiently?**
- RQ₃**: Agents process information passively in the above problems, i.e., agents only receive and process the inputs given by humans. We ask a question: **how do we build embodied agents that interact with physical environments and collect visual inputs actively itself?** We attempt to answer this question by tackling ALFRED, a well-defined embodied problem.

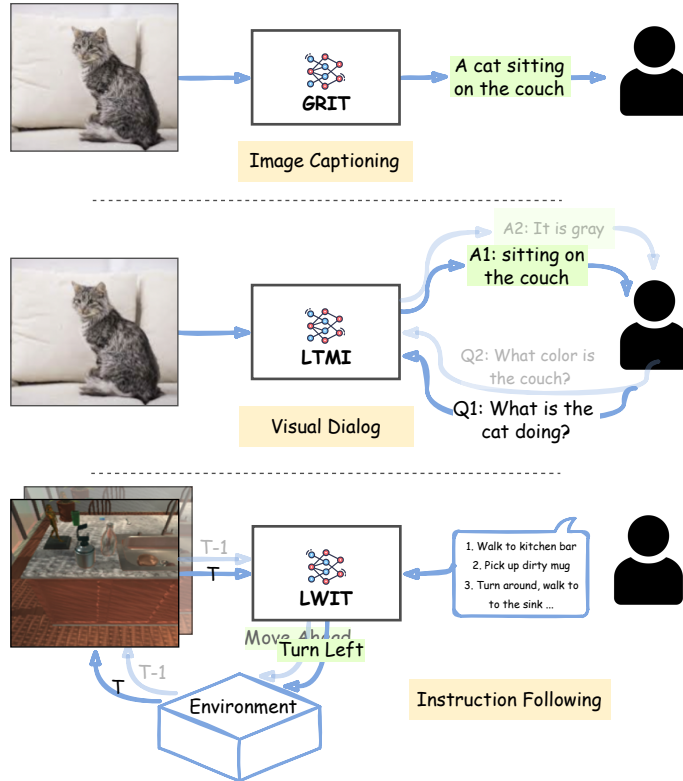


Figure 1.4: Illustrations of the research problems with our corresponding proposed agents in this dissertation. Each proposed agent/model is encapsulated with a black bounding box, outputting the predictions highlighted in green. **Top)** We propose GRIT, a Grid- and Region-based Transformer for Image captioning. **Middle)** We propose LTMI, a lightweight transformer that handles multiple inputs in Visual Dialog. **Bottom)** We propose LWIT, a neural agent that performs household tasks following humans’ instructions.

1.3 Dissertation Outline and Contributions

In this dissertation, we develop new models for machine intelligence that understand visual and linguistic information and interact with humans and environments, focusing on the three representative vision-language problems; See the overview in Figure 1.4. We present the dissertation’s outline and summarize each chapter’s contributions (if any) as follows.

Chapter 2–Background This chapter provides some fundamental background that is applied throughout the rest of the dissertation.

Chapter 3–GRIT: Integrating Dual Visual Features for Image Captioning

In this chapter, we revisit the representation of visual features, aiming to build a better and faster image captioning model. We then propose GRIT, an **Grid- and Region-based Image-captioning Transformer** that effectively utilizes the two visual features to generate better captions. GRIT replaces the CNN-based detector with a DETR-based one, making it computationally faster and end-to-end trainable. The experimental results show that GRIT outperforms the previous methods by a large margin in inference speed and accuracy.

Chapter 4–LTMI: Light-weight Transformer for Many Inputs in Visual Dialog

In this chapter, we tackle the visual dialog task, which requires the agent to handle multiple inputs, i.e., an image, a question, a dialog history, or its components. We introduce a neural architecture named **Light-weight Transformer for Many Inputs (LTMI)** that can efficiently deal with all the interactions between multiple such inputs in visual dialog. LTMI possesses sufficient representational power with much fewer parameters and a similar block structure to Transformer. The experimental results on the VisDial dataset validate the effectiveness of our proposed method.

Chapter 5–LWIT: Improving Performance on Instruction Following Tasks

In this chapter, we tackle ALFRED, the interactive instruction following tasks. Previous methods only show limited performance on the task; there is a huge gap with human performance. We propose LWIT (**Look Wide Interpret Twice**), a new method which outperforms the previous methods by a large margin. It is based on a combination of several new ideas, such as a two-stage interpretation of the provided instructions, using multiple egocentric views of the environment and extracting essential information by applying hierarchical attention conditioned on the current instruction, etc.

Chapter 6–Conclusion and Future Directions We conclude the dissertation by summarizing the main contributions and discussing future research toward improving this machine intelligence.

Chapter 2

Preliminaries

This chapter summarizes the introductory knowledge on recent deep learning that we will apply throughout the dissertation. To acquire a broader and deeper understanding, we highly recommend the Deep Learning textbook [24] which gives a more comprehensive introduction.

2.1 Preliminary Background

The proposed methods in this dissertation are mainly built upon modern deep learning techniques. Recent deep learning advances provide a powerful framework for supervised learning. Throughout this dissertation, we tackle several supervised learning problems that require learning an input-to-output mapping from training examples. We will present a generic supervised learning problem, optimization techniques to learn the mapping, and a typical workflow for building deep learning algorithms.

2.1.1 Supervised Learning

A supervised learning problem requires learning a mapping $f : X \rightarrow Y$, where X and Y represent the space for input and output, respectively. Supervised learning algorithms are those that are able to learn from a labeled dataset. Each sample in the dataset is represented by a vector x and associated with a label y , forming a datapoint (x, y) . A classical example of supervised learning problems is Iris classification, in which each datapoint is associated with a plant. We represent a plant by a single

vector x which is composed of four measurements: the petal length, the sepal length, the petal width, and the sepal width. The species form the output space Y . Take image captioning as a more complicated supervised learning example in this dissertation. The input x is an image, while the output y is a sentence describing the content of the image. It can often be challenging to manually define the mapping f that can be explicitly programmed using standard techniques. As an instance, take writing a hand-coded program that can describe an image in natural language. On the other hand, supervised learning algorithms provide an alternate approach by learning from many training examples $(x, y) \in X \times Y$ that can be collected easily in practice.

Formally, a supervised learning model can be defined as $y = f(x; \theta)$ where θ are the parameters learned from the training examples that result in the best approximation of f^* . Given n training examples $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, the best approximation f^* is defined as

$$f^* = \operatorname{argmin}_{\theta} \sum_i^n \mathcal{L}(f(x_i; \theta), y_i), \quad (2.1)$$

where \mathcal{L} is the per-sample loss (e.g., the mean square error (MSE), cross entropy). The loss function $J(\theta)$ measuring the model performance over the entire training samples is defined by

$$J(\theta) = \sum_i^n \mathcal{L}(f(x_i; \theta), y_i). \quad (2.2)$$

2.1.2 Optimization

As of now, it becomes an optimization problem by minimizing the loss function, e.g., $J(\theta)$. Maximization can be performed via a minimization algorithm by minimizing its negative function, e.g., $-J(\theta)$. In deep learning, the loss function can be the cost function, objective function, or training error.

Gradient Descent Suppose a function $y = f(x)$, where x and y are real numbers. The slope of $f(x)$ at the point x is given by its derivative, denoted as $f'(x)$ or $\frac{dy}{dx}$. It reflects how $f(x)$ changes in relation to the change in x . Using this knowledge, we can minimize $f(x)$ by moving x in small steps in the opposite direction of the

derivative’s sign. This is known as the **gradient descent** algorithm. It is natural to apply this algorithm to the optimization problem in 2.1 in order to find a function f^* by adjusting θ values. It is required computing the following gradient

$$\nabla_{\theta} = \sum_i^n \nabla_{\theta} \mathcal{L}(f(x_i; \theta), y_i). \quad (2.3)$$

Stochastic Gradient Descent In many practical problems, a sufficiently large dataset is required to seek a better model with sufficient generalizability. As a result, applying gradient descent directly to Eq.2.1 is computationally expensive, slowing down the optimization. In practice, to train a deep learning model on a large dataset, we usually use its extension, Stochastic Gradient Descent (SGD). The loss function is computed over a small number of training examples, which are sampled randomly during the training phase. Specifically, at one optimization step, we sample a **mini-batch** of m samples $\{(x_i, y_i)\}_{i=1}^m$ from the training set. Thus, rather than computing the gradient in 2.3 across all training examples, we only need to compute the gradient shown below.

$$\nabla_{\theta} = \sum_i^m \nabla_{\theta} \mathcal{L}(f(x_i; \theta), y_i), \quad (2.4)$$

where m is a relatively smaller number that is independent of the number of training examples n . As a result, the model’s parameters θ are updated as follows:

$$\theta \longleftarrow \theta - \epsilon \nabla_{\theta}, \quad (2.5)$$

where *epsilon* represents the learning rate. It is critical to set the learning rate to a suitable value that allows the optimization to properly converge. Inappropriate learning rates will slow convergence or cause the optimization to diverge.

Advanced Optimization Algorithms Several advanced gradient-based optimization algorithms have been developed that result in faster convergence. The **momentum** methods [25, 26] can be used to accelerate training with SGD. It accumulates an exponentially decaying moving average of previous gradients and keeps moving in their direction. Several approaches, including Adam [27], AdaGrad [28], and RMSProp [29], use adaptive learning rates to speed up optimization. It should be noted

that this dissertation mostly employs Adam as the primary optimizer choice.

2.1.3 Deep Learning Workflow

We already know how to solve a generic supervised learning problem using optimization. To solve this optimization problem, we can use the SGD algorithm to find the best f^* possible. In general, any deep learning algorithm can be thought of as an instance of a simple recipe that combines a dataset, a cost function, a model, and an optimization function. The following is a typical workflow for applying any deep learning model:

Data preparation The first step in most deep learning workflows is to obtain and prepare the dataset. The dataset is divided into training, development, and testing splits. In this dissertation, we focus on well-defined tasks with datasets that are primarily divided into training and development/validation splits. Each dataset also includes a testing split without any annotations; the result of this split can only be obtained by submitting it to the dataset’s online server. Following that, it is critical to inspect a dataset to gain an initial understanding of the task at hand, such as its distribution, how samples are collected and labeled, existing issues, and so on.

Data preprocessing Preprocessing data usually speeds up the training procedure, allowing the optimization to converge more quickly. Images are preprocessed, for example, by normalizing pixel values in each dimension with a given mean and standard deviation. This step is necessary in two situations: (1) when we extract features or finetune a pretrained backbone on a larger dataset, such as ImageNet, and (2) when we preprocess the samples during deployment using the same estimate statistics as training samples.

Architecture design All the deep learning algorithms require specifying the function space or the architecture family the optimization will seek. Based on the understanding of the dataset, one can make a heuristic on which kind of architecture can be employed. For example, CNNs are commonly used for grid-like data (e.g., images), whereas RNNs can handle sequential data (e.g., text), and so on. A heuristic like

this would help us quickly build viable architectures and establish initial baselines. Section 2.2 will present several common neural architectures upon which we build our proposed methods. Designing architecture is not limited by prior experience; thus, we can experiment with various architectures through trials and errors.

Training and Validation During the training steps, the designed model learns patterns from the training samples by utilizing optimization algorithms. While there are many popular optimization algorithms, researchers recommend using Adam with the default learning rate and first and second moment coefficients at the very first training. It is also common to use learning rate schedulers, which adjust the learning rate throughout training. We can also validate the model’s prediction ability on the validation split during the training steps.

Hyper-parameter optimization Deep learning algorithms involve numerous hyper-parameter decisions. Because it is difficult to find the optimal combination for all hyper-parameters, it is common to begin with the default settings in previous studies. Grid search and random search are the two standard techniques for performing hyper-parameter search efficiently. By evaluating the model on the validation set, we perform a hyper-parameter search. The final best model with the best validation performance is then chosen to perform on the test split in order to measure its performance.

2.2 Deep Neural Networks

Neural networks are well-known in the deep learning regime for approximating f . In the previous section, we defined the arbitrary function f that uses an optimization algorithm to learn the mapping $X \rightarrow Y$ for supervised learning problems. This section will explore several deep neural networks commonly used for visual and linguistic understanding.

2.2.1 FeedForward Neural Networks

The development of Feedforward Neural Networks dates back to 1958 when Frank Rosenblatt introduced the first Artificial Neural Network [30], named **perceptron**. The concept of the perceptron was inspired by the operation of a biological brain. Perceptrons were not learning efficiently until 1986 when David et al. [31] proposed the back-propagation algorithm, which can compute the gradient efficiently.

Feedforward neural networks are constructed by combining multiple functions. The most common method of building feedforward neural networks is by stacking multiple functions in a chain structure. For example, a two-layer network can be formed as $f(x) = W_2\sigma(W_1x + b_1) + b_2$, where W_1 and W_2 are learnable matrices, b_1 and b_2 are learnable biases, and σ is a non-linear function (e.g., tanh, sigmoid). A function transforming input x into $Wx + b$ with learnable parameters W and b forms a **Fully-Connected** (FC) layer. It is noted that an **activation function**, denoted by σ , is added to introduce non-linearity and enhance the representational power of the neural network (its learning capacity). The number of layers in these structures indicates the depth of neural networks, giving birth to the terminology **Deep Neural Networks** (DNN).

Feedforward neural networks are the foundation of many architecture designs. The specialized feedforward neural network types include convolutional neural networks, recurrent neural networks, and transformers. In the sections that follow, we will introduce briefly the neural architecture of these networks.

2.2.2 Convolutional Neural Networks

Simple neural networks composed of multiple FC layers make no assumptions about the input data. It becomes inefficient when processing high-dimensional input data, such as images with several hundred pixels per dimension. Many statistical properties of natural images are invariant to translation.

For instance, we classify the cat images as “cat” regardless of their breed, scale, and location in the image; see Figure 2.1. Convolutional neural networks (CNN) [2] are proposed to deal with these grid-like data (e.g., images, videos, etc). CNNs take this invariance into account by performing **convolution** across different locations in



Figure 2.1: Examples of cat images. They are of different cat breed, position, size, color intensity, etc. Source: Kaggle dataset.

the input representations with the kernels using shared parameters.

CNNs possess a number of characteristics that make them ideal for grid-like data [24]. First, because convolutions are translation invariant, objects and other characteristics can be recognized regardless of where they are in the image. Second, CNNs use the same kernels at every input location. Therefore, convolutional architecture is computationally more efficient than architectures with fully connected layers. Lastly, interactions in a convolutional layer are sparse, as only a few neighboring tensors are convolved with the kernels.

A typical convolutional neural network is formed by stacking **convolutional layers** and **pooling layers**. LeNet-5 [2] has two convolutional layers, two subsampling (pooling) layers, two fully connected layers, and two fully connected layers. All the layers are stacked in a chain structure as shown in Figure 2.2.

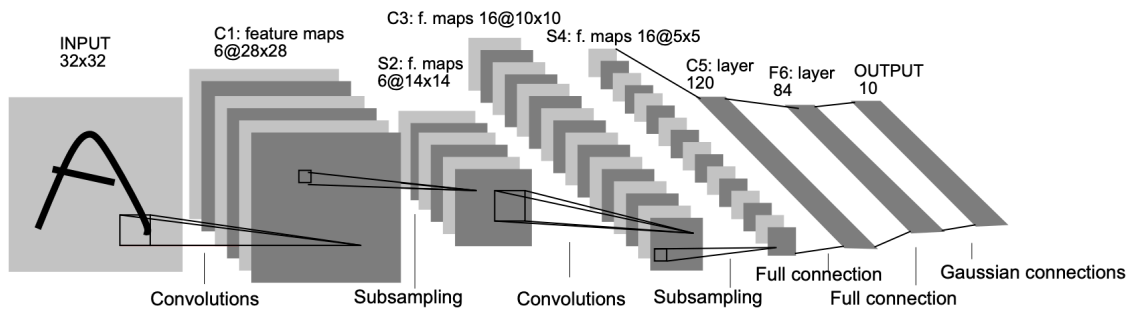


Figure 2.2: The architecture of LeNet-5, the first convolutional neural network introduced by Lecun et al. [2] for character recognition. The architecture of LeNet-5 consists of two convolutional layers, two subsampling (pooling) layers, and two fully connected layers. Source: [2].

Convolutional layers The input to the convolutional layer is a tensor. It then produces an output tensor by convolving the input with a set of filters. We can

convolve the filter by sliding it across all spatial positions of the input tensor and computing a dot product between it and a small region of the input tensor at each spatial position. This will result in an activation map. As a specific example, consider a $256 \times 256 \times 3$ input that is processed with a layer having $32 \ 5 \times 5 \times 3$ filters with padding of 2 and stride of 1. In this instance, the output would be $256 \times 256 \times 32$, representing the firing of all filters at all spatial locations.

Following are characteristics of a typical convolutional layer.

- It takes a tensor of size $W_1 \times H_1 \times D_1$ (e.g., 3D tensors).
- It has four hyper-parameters: K is the number of filters, F is their spatial extent, S is the stride, and P is zero padding on the input's borders.
- It produces an output having a size of $W_2 \times H_2 \times D_2$, in which $W_2 = (W_1 - F + 2P)/S + 1$; $H_2 = (H_1 - F + 2P)/S + 1$, and $D_2 = K$.
- Each filter is $(F * F * D_1)$ parameters, for a total of $(F * F * D_1 * K)$ weights and K biases. It is noted that the receptive field of the filters is small with an area of $F \times F$, but that it always traverses the entire depth of the input tensor (D_1).
- The size of the d -th of the output tensor is $(W_2 \times H_2)$. It is the result of convolving the d -th filter over the input tensor with a stride of S and then offsetting by the d -th bias.

Pooling layers In addition to convolutional layers, pooling layers are typically used to downsample feature maps. Pooling layers, unlike convolutional layers, transform input tensors without employing any learnable parameters. In particular, the pooling layers perform independently on each channel (activation map) and spatially downsample them. *Max pooling layer* and *average pooling layer* are frequently used as the building blocks in designing CNN architectures. Consider a max pooling layer with 2×2 filters and a stride of 2, where each filter performs the max operation over four numbers. Therefore, an input tensor is precisely downsampled by a factor of two in both width and height, and the representation size is reduced by a factor of four at the cost of some local spatial information loss.

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) [32] is a type of neural network designed to process sequential data. Unlike other feedforward neural networks, RNNs contain one or more feedback loops and a hidden state (memory) that is updated as each element in an input or output sequence is processed. This structure enables the network to process and remember signals, thereby enabling the model to learn sequential data dependencies. Given an input vector x_t at time step t , the hidden state h_t is computed by using a recurrent formulation:

$$h_t = f_\theta(h_{t-1}, x_t), \quad (2.6)$$

where f represents the computation of RNNs by using the same learnable parameters θ for all the time steps. Thus, it can process any sequence of arbitrary length. The hidden state h_{t-1} can be interpreted as a running memory from all previous time steps. Specifically, h_t in a **Vanilla Recurrent Neural Network** is computed as follows:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1}), \quad (2.7)$$

where W_{xh} and W_{hh} are the learnable parameters, and $\tanh(\cdot)$ is a hyperbolic activation function. Equation 2.6 and 2.7 omits bias vectors for the sake of brevity.

Long Short-Term Memory Networks The undesirable nature of vanilla RNNs is that the gradients tend to either vanish or explode over long time steps. Long Short-Term Memory (LSTM) networks [33] are a particular RNN implementation which is designed to handle the RNN limitations. In addition to the hidden state h_t , it also maintains the memory cell state c_t as follows:

$$h_t, c_t = \text{LSTM}(x_t, h_{t-1}, c_{t-1}). \quad (2.8)$$

Figure 2.3 illustrates all the feed-forward computations and internal updates of LSTM. At each time step, the LSTM can choose to read from, write to, or reset the cell using explicit gating mechanisms. Specifically, given x_t as input to a LSTM layer of N hidden units, the N -dimensional input gate i_t , forget gate f_t , output gate o_t , and

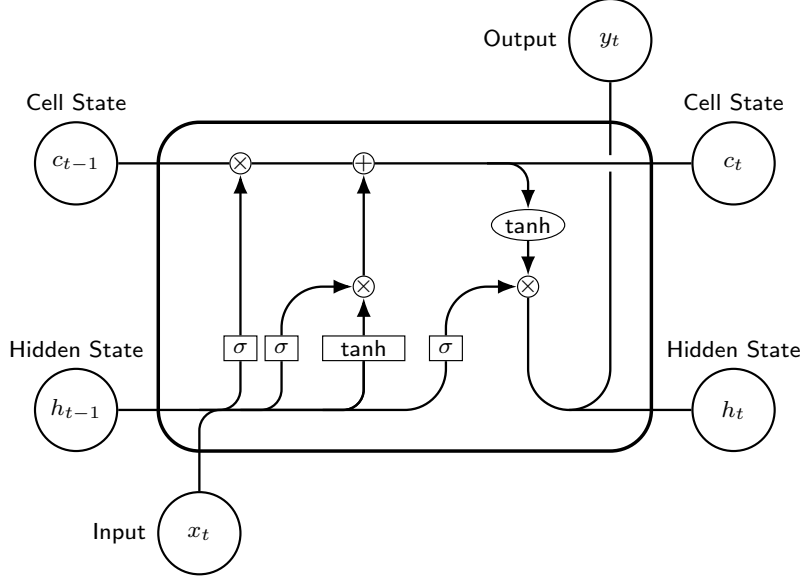


Figure 2.3: Illustration of a Long Short-Term Memory cell.

input modulation gate g_t at time step t are updated as

$$i_t = \text{sigm}(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \quad (2.9)$$

$$f_t = \text{sigm}(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \quad (2.10)$$

$$o_t = \text{sigm}(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \quad (2.11)$$

$$g_t = \text{tanh}(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (2.12)$$

where $h_{t-1} \in \mathbb{R}^N$ is the hidden state from the previous time step, W and b are learned weights and biases, and $\text{sigm}(\cdot)$ and $\text{tanh}(\cdot)$ are sigmoid and tanh functions, respectively. The above gates control the memory cell activation vector $c_t \in \mathbb{R}^N$ and output $h_t \in \mathbb{R}^N$ of the LSTM as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (2.13)$$

$$h_t = o_t \odot \text{tanh}(c_t), \quad (2.14)$$

where \odot represents element-wise multiplication.

2.2.4 Transformer Neural Networks

Transformers [3] are powerful neural networks with remarkable success in numerous areas across different data modalities, from language [12, 13] to images [7, 34], etc. The self-attention mechanism of transformers is their most important success factor. This mechanism learns the self-alignment between the tokens by calculating the similarity of a given token to all other tokens. Each token is then updated with a weighted representation of all tokens (including itself). It is observed that the weight value is proportional to each token pair’s affinity score. It will be described in detail in the text that follows.

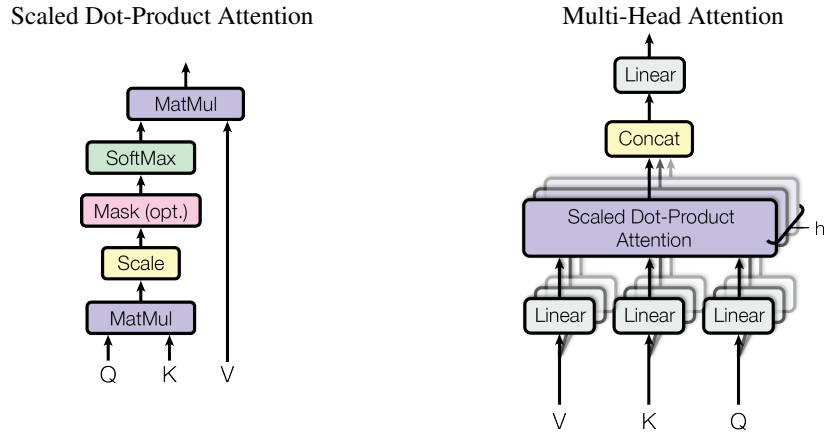


Figure 2.4: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Source: [3].

Self-Attention A self-attention function requires a query Q of dimension d_k and a set of key-value pairs (i.e., K, V) of dimension d_v for the sequence as inputs. It outputs a weighted sum of the values, in which we associate each value with a weight by computing the similarity between its respective query and key. Specifically, the function computes the dot products of the query with all the keys, divides each by $\sqrt{d_k}$, then applies a softmax function to obtain the normalized weights on the values. The attention function is given by

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (2.15)$$

Transformers utilize a multi-head attention mechanism, as shown in Figure 2.4,

to learn multiple attended features from different sub-spaces at different positions.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \quad (2.16)$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_m \times d_k}$, $W_i^K \in \mathbb{R}^{d_m \times d_k}$, $W_i^V \in \mathbb{R}^{d_m \times d_v}$, and $W_i^O \in \mathbb{R}^{d_m \times d_v}$.

Position-wise Feed-Forward Networks. In addition to the Self-Attention sub-layer, a typical transformer layer also includes a Feedforward Network (FFN) that is applied separately and identically to each position. The network consists of two linear transformations separated by a ReLU activation. Its computation is provided by

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2. \quad (2.17)$$

Chapter 3

GRIT: Integrating Dual Visual Features for Image Captioning

3.1 Introduction

Image captioning is the task of generating a semantic description of a scene in natural language, given its image. It requires a comprehensive understanding of the scene and its description reflecting the understanding. Therefore, most existing methods solve the task in two corresponding steps; they first extract visual features from the input image and then use them to generate a scene’s description. The key to success lies in the problem of how we can extract good features.

Researchers have considered several approaches to the problem. There are two primary methods, referred to as grid features [35–37] and region features [38]. Grid features are local image features extracted at the regular grid points, often obtained directly from a higher layer feature map(s) of CNNs/ViTs. Region features are a set of local image features of the regions (i.e., bounding boxes) detected by an object detector.

The current state-of-the-art methods employ the region features since they encode detected object regions directly. Identifying objects and their relations in an image will be useful to correctly describing the image. However, the region features have several issues. First, they do not convey contextual information such as objects’ relation since the regions do not cover the areas between objects. Second, there is

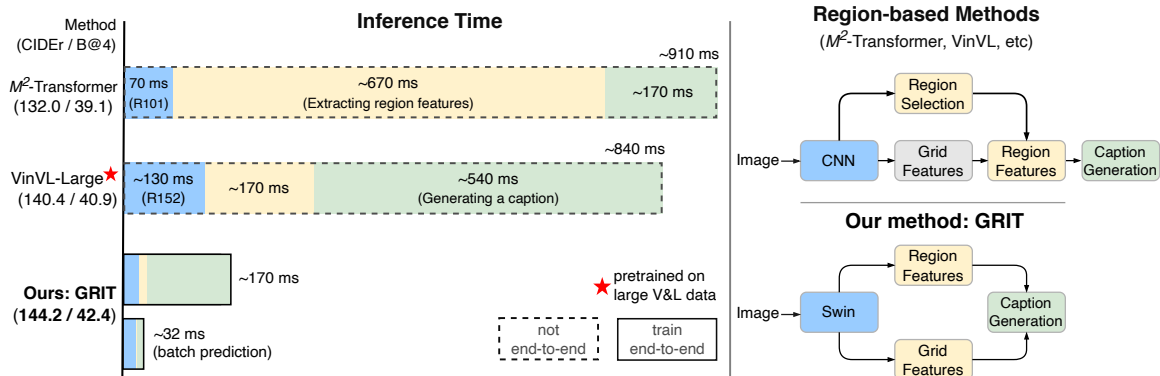


Figure 3.1: Comparison of GRIT and other region-based methods for image captioning. Left: Running time per image of performing inference with beam size of five and the maximum length of 20 on a V100 GPU. Right: Their architectures

a risk of erroneous detection of objects; important objects could be overlooked, etc. Third, computing the region feature is computationally costly, which is especially true when using a high-performance CNN-based detector, such as Faster R-CNN [8].

The grid features are extracted from the entire image, typically a high-layer feature map of a backbone network. While they do not convey object-level information, they are free from the first two issues with the region features. They may represent contextual information such as objects’ relations in images, and they are free from the risk of erroneous object detection.

In this study, we consider using such region and grid features in an integrated manner, aiming to build a better model for image captioning. The underlying idea is that properly integrating the two types of features will provide a better representation of input images since they are complementary, as explained above. While a few recent studies consider their integration [39, 40], it is still unclear what the best way is. In this study, we reconsider how to extract each from input images and then consider how to integrate them.

There is yet another issue with the region features, usually obtained by a CNN-based detector. At the last stage of its computation, CNN-based detectors employ non-maximum suppression (NMS) to eliminate redundant bounding boxes. This makes the end-to-end training of the entire model hard, i.e., jointly training the decoder part of the image-captioning model and the detector by minimizing a single loss. Recent studies detach the two parts in training; they first train a detector on

the object detection task and then train only the decoder part on image captioning. This could be a drag on achieving optimal performance of image captioning.

To overcome this limitation of CNN-based detectors and also cope with their high-computational cost, we employ the framework of DETR [41], which does not need NMS. We choose Deformable DETR [42], an improved variant, for its high performance, and also replace a CNN backbone used in the original design with Swin Transformer [34] to extract initial features from the input image. We also obtain the grid features from the same Swin Transformer. We input its last layer features into a simple self-attention Transformer and update them to obtain our grid features. This aims to model spatial interaction between the grid features, retrieving contextual information absent in our region features.

The extracted two types of features are fed into the second half of the model, the caption generator. We design it as a lightweight Transformer generating a caption sentence in an autoregressive manner. It is equipped with a unique cross-attention mechanism that computes and applies attention from the two types of visual features to caption sentence words.

These components form a Transformer-only neural architecture, dubbed GRIT (**Grid- and Region-based Image-captioning Transformer**). Our experimental results show that GRIT has established a new state-of-the-art on the standard image captioning benchmark of COCO [43]. Specifically, in the offline evaluation using the Karpathy test split, GRIT outperforms all the existing methods without vision and language (V&L) pretraining. It also performs at least on a par with SimVLM_{huge} [44] leveraging V&L pretraining on 1.8B image-text pairs.

3.2 Related Work

3.2.1 Visual Representations for Image Captioning

Recent image captioning methods typically employ an encoder-decoder architecture. Specifically, given an image, the encoder extracts visual features; the decoder receives the visual features as inputs and generates a sequence of words. Early methods use a CNN to extract a global feature as a holistic representation of the input

image [16, 45]. Although it is simple and compact, this holistic representation suffers from information loss and insufficient granularity. To cope with this, several studies [35–37] employed more fine-grained grid-based features to represent input images and also used attention mechanisms to utilize the granularity for better caption generation. Later, Anderson et al. [38] introduced the method of using an object detector, such as Faster R-CNN, to extract object-oriented features, called region features, showing that this leads to performance improvement in many V&L tasks, including image captioning and visual question answering. Since then, region features have become the de facto choice of visual representation for image captioning. Pointing out the high computational cost of the region features, Jiang et al. [46] showed that the grid features extracted by an object detector perform well on the VQA task. RSTNet [47] has recently applied these grid features to image captioning.

3.2.2 Application of Transformer in Vision/Language Tasks

Transformer has long been a standard neural architecture in natural language processing [3, 12, 13], and started to be extended to computer vision tasks. Besides ViT [7] for image classification, it was also applied to object detection, leading to DETR [41], followed by several variants [42, 48, 49]. A recent study [50] applied the framework of DETR to pretraining for various V&L tasks, where they did not use it to obtain the region features.

Transformer has been applied to image captioning, where it is used as an encoder for extracting and encoding visual features and a decoder for generating captions. Specifically, Yang et al. [51] proposed to use the self-attention mechanism to encode visual features. Li et al. [52] used Transformer for obtaining the region features in combination with a semantic encoder that exploits knowledge from an external tagger. Several following studies proposed several variants of Transformer tailored to image captioning, such as Attention on Attention [53], X-Linear Attention [54], Memory-augmented Attention [55], etc. Transformer is naturally employed also as a caption decoder [39, 44, 56, 57].

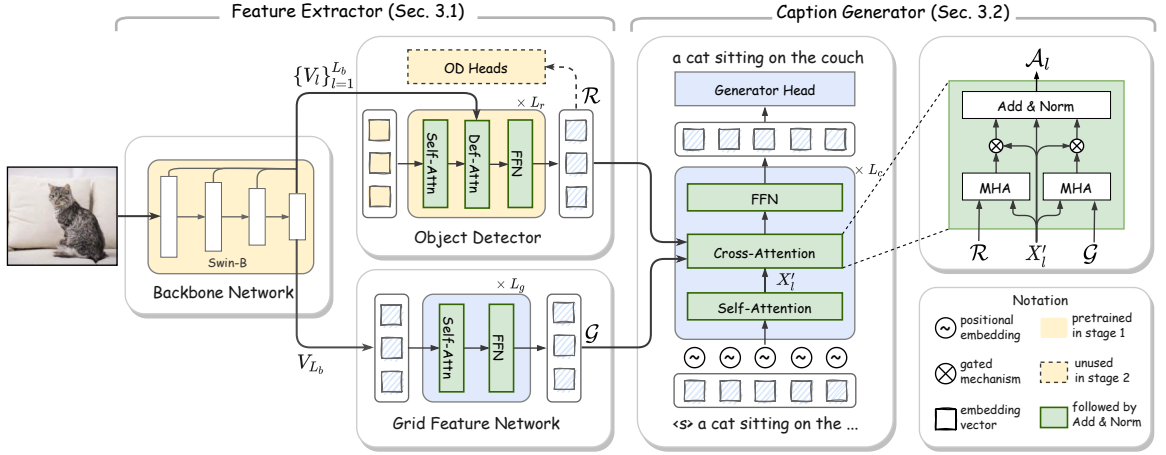


Figure 3.2: Overview of the architecture of GRIT

3.3 Proposed Method

This section describes the architecture of GRIT (Grid- and Region-based Image-captioning Transformer). It consists of two parts, one for extracting the dual visual features from an input image (Sec. 3.3.1) and the other for generating a caption sentence from the extracted features (Sec. 3.3.2).

3.3.1 Extracting Visual Features from Images

Backbone Network for Extracting Initial Features

A lot of efforts have been made to apply the Transformer architecture to various computer vision tasks since ViT [7] applied it to image classification. ViT divides an input image into small patches and computes global attention over them. This is not suitable for tasks requiring spatially dense prediction, e.g., object detection since the computational complexity increases quadratically with the image resolution.

Swin Transformer [34] mitigates this issue to a great extent by incorporating operations such as patch reduction and shifted windows that support local attention. It is currently a de facto standard as a backbone network for various computer vision tasks. We employ it to extract initial visual features from the input image in our model.

We briefly summarize its structure, explaining how we extract features from the

input image and send them to the components following the backbone. Given an input image of resolution $H \times W$, Swin Transformer computes and updates feature maps through multiple stages; it uses the patch merging layer after every stage (but the last stage) to downsample feature maps in their spatial dimension by the factor of 2. We apply another patch merging layer to downsample the last layer’s feature map. We then collect the feature maps from all the stages, obtaining four multi-scale feature maps, i.e., $\{V_l\}_{l=1}^{L_b}$ where $L_b = 4$, which have the resolution from $H/8 \times W/8$ to $H/64 \times W/64$. These are inputted to the subsequent modules, i.e., the object detector and the network for generating grid features.

Generating Region Features

As in previous image captioning methods, ours also rely on an object detector to create region features. However, we employ a Transformer-based decoder framework, i.e., DETR [41] instead of CNN-based detectors, such as Faster-RCNN, which is widely employed by the SOTA image captioning models [38]. DETR formulates object detection as a direct set prediction problem, which makes the model free of the unideal computation for us, i.e., NMS and RoI alignment. This enables the end-to-end training of the entire model from the input image to the final output, i.e., a generated caption, and also leads to a significant reduction in computational time while maintaining the model’s performance on image captioning compared with the SOTA models.

Specifically, we employ Deformable DETR [42], a variant of DETR. Deformable DETR extracts multi-scale features from an input image with its encoder part, which are fed to the decoder part. We use only the decoder part, to which we input the multi-scale features from the Swin Transformer backbone. This leads to further reduction in computational time. We will refer this decoder part as “object detector” in what follows; see Fig. 3.2.

The object detector receives two inputs: the multi-scale feature maps generated by the backbone, and N learnable object queries $R_0 = \{r_i\}_{i=1}^N$, in which $r_i \in \mathbb{R}^d$. Before forwarding them into the object detector, we apply linear transformation to the multi-scale feature maps, mapping them into d -dimensional vectors as $V_l \leftarrow W_l^r V_l$, where $\{W_l^r\}_{l=1}^{L_b}$ is a learnable projection matrix.

Receiving these two inputs, the object detector updates the object queries through a stack of L_r deformable layers, yielding $R_{L_r} \in \mathbb{R}^{N \times d}$ from the last layer; see [42] for details. We use $R_{L_r} \in \mathbb{R}^{N \times d}$ as our region features \mathcal{R} . We forward this to the caption generator.

Although we train it as a part of our entire model, we pretrain our ‘‘object detector’’ including the vision backbone on object detection before the training of image-captioning. For the pretraining, we follow the procedure of Deformable DETR; placing a three-layer MLP and a linear layer on its top to predict box coordinates and class category, respectively. We then minimize a set-based global loss that forces unique predictions via bipartite matching.

Following [38, 58], we pretrain the model (i.e., our object detector including the vision backbone) in two steps. We first train it on object detection following the training method of Deformable DETR. We then fine-tune it on a joint task of object detection and object attribute prediction, aiming to make it learn fine-grained visual semantics with the following loss:

$$\mathcal{L}_v(y, \hat{y}) = \sum_{i=1}^N \left[\underbrace{-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{c_i \neq \emptyset} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})}_{\text{object detection}} \underbrace{-\log \hat{p}_{\hat{\sigma}(i)}(a_i)}_{\text{attribute prediction}} \right], \quad (3.1)$$

where $\hat{p}_{\hat{\sigma}(i)}(a_i)$ and $\hat{p}_{\hat{\sigma}(i)}(c_i)$ are the attribute and class probabilities, $\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})$ is the loss for normalized bounding box regression for object i [42].

Grid Feature Network

This network receives the last one of the multi-scale feature maps from the Swin Transformer backbone, i.e., $V_{L_b} \in \mathbb{R}^{M \times d_{L_b}}$, where $M = H/64 \times W/64$. As with the input to the object detector, we apply a linear transformation with a learnable matrix $W^g \in \mathbb{R}^{d \times d_{L_b}}$ to V_{L_b} , obtaining $G_0 = W^g V_{L_b}$. We employ the standard self-attention Transformer having L_g layers. This network updates V_{L_b} through these layers, yielding our grid features \mathcal{G} represented as a $M \times d$ matrix. We intend to extract contextual information hidden in the input image by modeling the spatial interaction between the grid features.

3.3.2 Caption Generation Using Dual Visual Features

Overall Design of Caption Generator

The caption generator receives the two types of visual features, the region features $\mathcal{R} \in \mathbb{R}^{N \times d}$ and the grid features $\mathcal{G} \in \mathbb{R}^{M \times d}$, as inputs. Apart from this, we employ the basic design employed in previous studies [3, 56] that is based on the Transformer architecture. It generates a caption sentence in an autoregressive manner; receiving the sequence of predicted words (rigorously their embeddings) at time $t-1$, it predicts the next word at time t . We employ the sinusoidal positional embedding of time step t [3]; we add it to the word embedding to obtain the input $x_0^t \in \mathbb{R}^d$ at t .

The caption generator consists of a stack of L_c identical layers. The initial layer receives the sequence of predicted words and the output from the last layer is input to a linear layer whose output dimension equals the vocabulary size to predict the next word.

Each transformer layer has a sub-layer of masked self-attention over the sentence words and a sub-layer(s) of cross-attention between them and the visual features in this order, followed by a feedforward network (FFN) sub-layer. The masked self-attention sub-layer at the l -th layer receives an input sequence $\{x_{l-1}^i\}_{i=0}^t$ at time step t , and computes and applies self-attention over the sequence to update the tokens with the attention mask to prevent the interaction from the future words during training.

The cross-attention sub-layer in the layer l , located after the self-attention sub-layer, fuses its output with the dual visual features by cross-attention between them, yielding \mathcal{A}_l . We consider the three design choices shown in Fig. 3.3 and described below. We examine their performance through experiments.

Cross-attention between Caption Word and Dual Visual Features

We show three designs of cross-attention between the caption word features and the dual visual features (i.e., the region features \mathcal{R} and the grid features \mathcal{G}) as below.

Concatenated Cross-Attention The simplest approach is to concatenate the two visual features and use the resultant features as keys and values in the standard multi-head attention sub-layer, where the sentence words serve as queries; see Fig. 3.3(a).

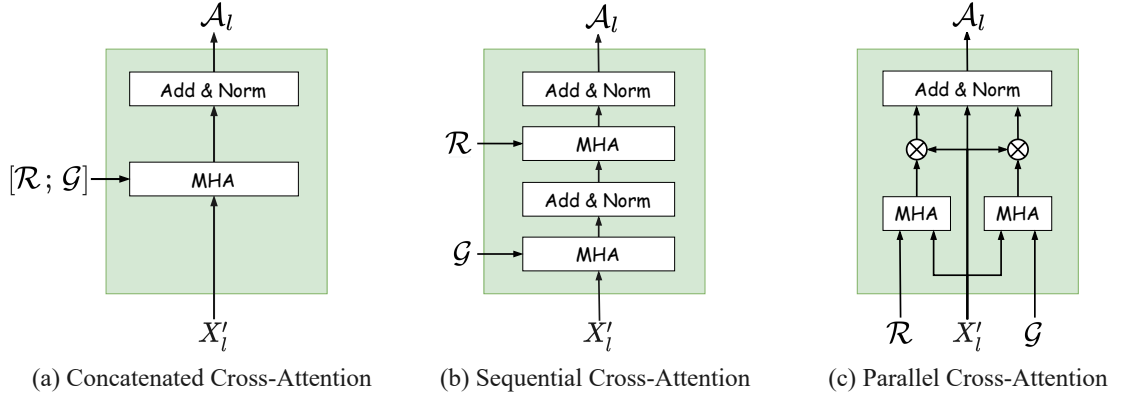


Figure 3.3: Three designs of cross-attention mechanism to use dual visual features

Sequential Cross-Attention Another approach is to perform cross-attention computation separately for the two visual features. The corresponding design is to place two independent multi-head attention sub-layers in a sequential fashion, and uses one for the grid features and the other for the region features (or the opposite combination); see Fig. 3.3(b). Note that their order could affect the performance.

Parallel Cross-Attention The third approach is to perform multi-head attention computation on the two visual features in parallel. To do so, we use two multi-head attention mechanisms with independent learnable parameters. The detailed design is as follows. Let $X_{l-1} = \{x_i^{l-1}\}$ be the word features inputted to the meta-layer l containing this cross attention sub-layer. As shown in Fig. 3.2, they are first input to the self-attention sub-layer, converted into $X'_l = \{x'_i\}$ (layer index l omitted for brevity) and then input to this cross attention sub-layer. In this sub-layer, multi-head attention (MHA) is computed with $\{x'_i\}$ as queries and the region features \mathcal{R} as keys and values, yielding attended features $\{a_i^r\}$. The same computation is performed in parallel with the grid features \mathcal{G} as keys and values, yielding $\{a_i^g\}$. Next, we concatenate them with x'_i as $[a_i^r; x'_i]$ and $[a_i^g; x'_i]$, projecting them back to d -dimensional vector using learnable affine projections. Normalizing them with sigmoid

into probabilities $\{c_i^r\}$ and $\{c_i^g\}$, respectively, we have

$$c_i^g = \text{sigmoid}(W^g[a_i^g; x_i'] + b^g), \quad (3.2)$$

$$c_i^r = \text{sigmoid}(W^r[a_i^r; x_i'] + b^r). \quad (3.3)$$

We then multiply them with $\{a_i^r\}$ and $\{a_i^g\}$, add the resultant vectors to $\{x_i'\}$, and finally feed to layer normalization, obtaining $\mathcal{A}_l = \{a_i^{(l)}\}$ as follows:

$$a_i^{(l)} = \text{LN}(c_i^g \otimes a_i^g + c_i^r \otimes a_i^r + x_i'). \quad (3.4)$$

Caption Generator Losses

Following a standard practice of image captioning studies, we pre-train our model with a cross-entropy loss (XE) and finetune it using the CIDEr-D optimization with self-critical sequence training strategy [36]. Specifically, the model is first trained to predict the next word x_t^* at $t = 1..T$, given the ground-truth sentence $x_{1:T}^*$. This is equal to minimize the following XE loss with respect to the model’s parameter θ :

$$L_{XE}(\theta) = - \sum_{t=1}^T \log(p_\theta(x_t^* | x_{0:t-1}^*)). \quad (3.5)$$

We then finetune the model with the CIDEr-D optimization, where we use the CIDEr score as the reward and the mean of the rewards as the reward baseline, following [55].

The loss for self-critical sequence training is given by

$$\mathcal{L}_{RL}(\theta) = -\frac{1}{k} \sum_{i=1}^k (r(\mathbf{w}^i) - b) \log p(\mathbf{w}^i), \quad (3.6)$$

where \mathbf{w}^i is the i -th sentence in the beam; $r(\cdot)$ is the reward function; and b is the reward baseline; and k is the number of samples in the batch.

3.4 Experiments

3.4.1 Datasets

Object Detection

As mentioned earlier, we train our object detector (including the backbone) in two steps. In the first step, we train it on object detection using either Visual Genome [59] or a combination [58] of four datasets: COCO [43], Visual Genome, Open Images [60], and Object365 [61], depending on what previous methods we experimentally compare. In the second step, we train the model on object detection plus attribute prediction using Visual Genome. Note that following the standard practice, we exclude the duplicated samples appearing in the testing and validation splits of the COCO and nocaps [62] datasets to remove data contamination. See the supplementary material for more details.

When pretraining our model on the four datasets (i.e., Visual Genome (VG), COCO, OpenImages, and Objects365), we follow [58] to build a unified training corpus with the statistics shown in Table 3.1 except that we do not use the annotations from COCO stuff [63]. The resultant corpus has 2.49M unique images with 1848 categories.

Table 3.1: Statistics of the pretraining datasets for object detection.

Source	VG	COCO	Objects365	OpenImages
Images	97k	111k	609k	1.67M
Categories	1594	80	365	500
Sampling	×8	×8	×2	×1

Image Captioning

We conduct our experiments on the COCO dataset, the standard for the research of image captioning [43]. The dataset contains 123,287 images, each annotated with five different captions. For offline evaluation, we follow the widely adopted Karpathy

split [64], where 113,287, 5,000, and 5,000 images are used for training, validation, and testing respectively.

To test our method’s effectiveness on other image captioning datasets, we also report the performances on the nocaps dataset and the Artemis dataset [65]. See the supplementary material for more details.

3.4.2 Implementation Details

Evaluation Metrics

We employ the standard evaluation protocol for the evaluation of methods. Specifically, we use the full set of captioning metrics: BLEU@N [66], METEOR [67], ROUGE-L [68], CIDEr [69], and SPICE [70]. We will use the abbreviations, B@N, M, R, C, and S, to denote BLEU@N, METEOR, ROUGE-L, CIDEr, and SPICE, respectively.

Hyperparameters Settings

In our model, we set the dimension d of each layer to 512, the number of heads to eight. We employ dropout with the dropout rate of 0.2 on the output of each MHA and FFN sub-layer following [3].

For the object detector, we set the number of queries $N = 150$ and the hidden dimension $d = 512$. The backbone network weights are initialized by the weights of Swin-Base (384×384) pretrained on ImageNet21K [34]. Following [42], the loss for normalized bounding box regression for object i , $\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})$ is computed as the weighted summation of a box distance \mathcal{L}_{l_1} and a GIoU loss \mathcal{L}_{iou} :

$$\mathcal{L}_{l_1}(b_i, \hat{b}_{\hat{\sigma}(i)}) = \|b_i - \hat{b}_{\hat{\sigma}(i)}\|_1, \quad (3.7)$$

$$\mathcal{L}_{iou}(b_i, \hat{b}_{\hat{\sigma}(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\hat{\sigma}(i)}|}{|b_i \cup \hat{b}_{\hat{\sigma}(i)}|} - \frac{|\mathbf{B}(b_i, \hat{b}_{\hat{\sigma}(i)}) \setminus b_i \cup \hat{b}_{\hat{\sigma}(i)}|}{|\mathbf{B}(b_i, \hat{b}_{\hat{\sigma}(i)})|} \right), \quad (3.8)$$

$$\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) = \alpha_{l_1} \mathcal{L}_{l_1}(b_i, \hat{b}_{\hat{\sigma}(i)}) + \alpha_{iou} \mathcal{L}_{iou}(b_i, \hat{b}_{\hat{\sigma}(i)}), \quad (3.9)$$

where $\alpha_{l_1} = 5$, $\alpha_{iou} = 2$, and \mathbf{B} outputs the largest box covering b_i and $\hat{b}_{\hat{\sigma}(i)}$. We also employ two training strategies, i.e., iterative bounding box refinement and auxiliary

losses; see [42] for details.

We set the number of layers as $L_r = 6$ for the object detector, as $L_g = 3$ for the grid feature network, and as $L_c = 3$ for the caption generator. Following previous studies, we convert all the captions to lower-case, remove punctuation characters, and perform tokenization with the SpaCy toolkit [71]. We build the vocabularies, excluding the words which appear less than five times in the training and validation splits.

3.4.3 Training Details

First Stage In the first stage, we pretrain the object detector with the backbone. We consider several existing region-based methods for comparison, which employ similar pretraining of an object detector but use different datasets. For a fair comparison, we consider two settings. One uses Visual Genome for training, following most previous methods. We train our detector for 150,000 iterations with a batch size of 32. The other (results indicated with † in what follows) uses the four datasets mentioned above, following [58]. We train the detector for 125,000 iterations with a batch size of 256. In both settings, the input image is resized so that the maximum for the shorter side is 800 and for the longer side is 1333. We use Adam optimizer [72] with a learning rate of 10^{-4} , decreased by 10 at iteration 120,000 and 100,000 in the first and second settings, respectively. We follow [42] for other training procedures. After this, we finetune the models on object detection plus attribute prediction using Visual Genome for additional five epochs with a learning rate of 10^{-5} , following [38, 58]. The supplementary material presents the details of implementation and experimental results on object detection.

Second Stage We train the entire model for the image-captioning task in the second stage. We employ the standard method for word representation, i.e., linear projections of one-hot vectors to vectors of dimension $d = 512$. In this stage, we resize all the input images so that the maximum dimensions for the shorter side and longer side are 384 and 640, respectively. We train models, as explained earlier. Specifically, we train models with the cross-entropy loss \mathcal{L}_{XE} for ten epochs, in which we warm up the learning rates for the grid feature network and the caption generator from 10^{-5}

to 10^{-4} in the first epoch, while we fix those for the backbone network and the object detector at 10^{-5} . Then, we finetune the model based on the CIDEr-D optimization for ten epochs, where we set the fixed learning rate to 5×10^{-6} for the entire model. We use the Adam optimizer [72] with a batch size of 128. For the inference inside the CIDEr-D optimization, we use beam search with a beam size of five and a maximum length of 20.

3.4.4 Object Detection Results

Table 3.2: Performance of object detection on the COCO and Visual Genome datasets. ‘4DS’ denotes the four object detection datasets.

Model	Training Data	mAP (COCO)	mAP ⁵⁰ (VG)
BUTD [38]	VG	-	10.2
VinVL [58]	4DS	50.5	13.8
GRIT	VG	33.6	14.2
GRIT [†]	4DS	50.8	15.1

Table 3.2 shows the performance on the COCO validation split and the Visual Genome test split of our object detector compared with VinVL and BUTD [38]. It is seen that the object detector of GRIT attains comparable or higher performance on the two datasets as compared with BUTD and VinVL when pretrained on the similar datasets.

3.4.5 Performance of Different Configurations

Our method has several design choices. We conduct experiments to examine which configuration is the best. The results are shown in Table 3.3. We used an identical configuration unless otherwise noted. Specifically, we use the feature extractor pretrained on the four datasets and parallel cross-attention for fusing the region and grid features.

Effects of Object Detection Datasets The first block of Table 3.3(a) shows the effects of different (pre)training strategies of the visual backbone on image-captioning

Table 3.3: Results of ablation tests on the COCO test split. All the models are trained with the XE loss and finetuned by the CIDEr optimization

(a)			(b)		
Factor	Choice	CIDEr B@4	Cross Attention	Choice	CIDEr B@4
(1) Backbone Network - Training data	ImageNet	135.5 41.5	(1) Concatenated - Visual features	\mathcal{G}	142.1 41.7
	VG	142.3 41.9		\mathcal{R}	142.9 41.9
	4DS	144.2 42.4		$[\mathcal{G} ; \mathcal{R}]$	143.1 41.9
(2) Region features - Number of vectors (trained on VG)	50	141.4 41.9	(2) Sequential - Sequential order		
	100	141.8 41.5		$\mathcal{G} \rightarrow \mathcal{R}$	144.0 42.1
	150	142.3 41.9		$\mathcal{R} \rightarrow \mathcal{G}$	143.6 42.1
(3) Training strategy - End-to-end training	Yes	144.2 42.4	(3) Parallel - Gated activation	Sigmoid	144.2 42.4
	No	139.6 42.7		Identity	143.9 41.6

performance. The ‘ImageNet’ column shows the result of the model using a Swin Transformer backbone pretrained on ImageNet21K and the grid features alone; ‘VG’ and ‘4DS’ indicate the models with a detector pretrained on Visual Genome and the four datasets, respectively. They show that using more datasets leads to better performance.

Effects of Number of Region Features The second block of Table 3.3(a) shows the effects of the number of object queries, or equivalently region features. The performance increases as they vary as 50, 100, and 150. We also confirmed that the performance is saturated for more region features, while the computational cost and false detection increase.

Impact of End-to-end Training The third block shows the effect of the end-to-end training of the entire model. ‘Yes’ indicates the end-to-end training of the entire model and ‘No’ indicates training the model but the vision backbone. The results show that the end-to-end training considerably improves CIDEr score (from 139.6 to 144.3) with little sacrifice of B@4. This validates our expectation about the effectiveness of the end-to-end training; it arguably helps reduce the domain gap

between object detection and image captioning.

Effects of Fusion of Dual Visual Features The first block of Table 3.3(b) shows the performances of the model employing the concatenated cross-attention and its two variants using the grid features alone or the region features alone. They show that the region features alone work better than the grid features alone, and their fusion achieves the highest performance.

Effects of Cross-attention Architecture The three blocks of Table 3.3(b) show the performances of the three cross-attention architectures explained in Sec. 3.3.2. The second block shows the two variants of the sequential cross-attention, and the third block shows the two variants of the parallel cross-attention with different gated activation functions, i.e., sigmoid and identity. By identity activation, we mean setting all the values of c_i^g and c_i^r in Eq.(3.4) to one. These results show that the parallel cross-attention with sigmoid activation function performs the best; the sequential cross-attention in the order $\mathcal{G} \rightarrow \mathcal{R}$ attains the second best result.

3.4.6 Results on the COCO Dataset

We next show complete results on the COCO dataset by the offline and online evaluations. We present example results in the supplementary material.

Offline Evaluation Table 3.4 shows the performances of our method and the current state-of-the-art methods on the offline Karpathy test split. The compared methods are as follows: grid-based methods [16, 36, 47, 75], region-based methods [38, 52, 53, 53–57, 76–81], the methods employing both grid and region features [39, 40], and also the methods relying on large-scale pretraining on vision and language (V&L) tasks using a large image-text corpus [58, 73, 74], including SimVLM_{huge}, a model pre-trained on an extremely large dataset (i.e., 1.8 billion image-caption pairs) [44].

For fair comparison with the region-based methods, we report the results of two variants of our model, one with the object detector pretrained on Visual Genome alone and the other (marked with †) with the object detector pretrained on the four datasets, as explained earlier. It is seen from Table 3.4 that our models, regardless

Table 3.4: Offline results evaluated on the COCO Karpathy test split. ‘V. E. type’ indicates the type of visual features; ‘# VL Data’ is the number of image-text pairs used for vision-language pretraining.

Method	V. E.	# VL	Performance Metrics					
	Type	Data	B@1	B@4	M	R	C	S
w/ VL pretraining								
UVLP [73]	\mathcal{R}	3.0M	-	39.5	29.3	-	129.3	23.2
Oscar _{base} [74]	\mathcal{R}	6.5M	-	40.5	29.7	-	137.6	22.8
VinVL _{large} [†] [58]	\mathcal{R}	8.9M	-	41.0	31.1	-	140.9	25.2
SimVLM _{huge} [44]	\mathcal{G}	1.8B	-	40.6	33.7	-	143.3	25.4
w/o VL pretraining								
SAT [16]	\mathcal{G}	-	-	31.9	25.5	54.3	106.3	-
SCST [36]	\mathcal{G}	-	-	34.2	26.7	55.7	114.0	-
LSTM-A [75]	\mathcal{G}	-	78.6	35.5	27.3	56.8	118.3	22.0
RSTNet [47]	\mathcal{G}	-	81.8	40.1	29.8	59.5	135.6	23.0
Up-Down [38]	\mathcal{R}	-	79.8	36.3	27.7	56.9	120.1	21.4
RFNet [76]	\mathcal{R}	-	79.1	36.5	27.7	57.3	121.9	21.2
GCN-LSTM [77]	\mathcal{R}	-	80.5	38.2	28.5	58.3	127.6	22.0
LBPF [78]	\mathcal{R}	-	80.5	38.3	28.5	58.4	127.6	22.0
SGAE [79]	\mathcal{R}	-	80.8	38.4	28.4	58.6	127.8	22.1
AoA [53]	\mathcal{R}	-	80.2	38.9	29.2	58.8	129.8	22.4
GET [80]	\mathcal{R}	-	81.5	39.5	29.3	58.9	131.6	22.8
ORT [56]	\mathcal{R}	-	80.5	38.6	28.7	58.4	128.3	22.6
ETA [52]	\mathcal{R}	-	81.5	39.3	28.8	58.9	126.6	22.6
\mathcal{M}^2 Transformer [55]	\mathcal{R}	-	80.8	39.1	29.2	58.6	131.2	22.6
X-LAN [54]	\mathcal{R}	-	80.8	39.5	29.5	59.2	132.0	23.4
TCIC [81]	\mathcal{R}	-	81.8	40.8	29.5	59.2	135.4	22.5
Dual Global [40]	$\mathcal{R}+\mathcal{G}$	-	81.3	40.3	29.2	59.4	132.4	23.3
DLCT [39]	$\mathcal{R}+\mathcal{G}$	-	81.4	39.8	29.5	59.1	133.8	23.0
GRIT	$\mathcal{R}+\mathcal{G}$	-	83.5	41.9	30.5	60.5	142.2	24.2
GRIT [†]	$\mathcal{R}+\mathcal{G}$	-	84.2	42.4	30.6	60.7	144.2	24.3

Table 3.5: Online evaluation results on the COCO image captioning dataset.

Method	Ensemble	B-1		B-2		B-3		B-4		M		R		C	
		c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
w/ VL pretraining															
VinVL _{large} [58]	✗	81.9	96.9	66.9	92.4	52.6	84.7	40.4	74.9	30.6	40.8	60.4	76.8	134.7	138.7
w/o VL pretraining															
SCST [36]	✓	78.1	93.7	61.9	86.0	47.0	75.9	35.2	64.5	27.0	35.5	56.3	70.7	114.7	116.7
Up-Down [38]	✓	80.2	95.2	64.1	88.8	49.1	79.4	36.9	68.5	27.6	36.7	57.1	72.4	117.9	120.5
HAN [82]	✓	80.4	94.5	63.8	87.7	48.8	78.0	36.5	66.8	27.4	36.1	57.3	71.9	115.2	118.2
GCN-LSTM [77]	✓	80.8	95.2	65.5	89.3	50.8	80.3	38.7	69.7	28.5	37.6	58.5	73.4	125.3	126.5
SGAE [79]	✓	81.0	95.3	65.6	89.5	50.7	80.4	38.5	69.7	28.2	37.2	58.6	73.6	123.8	126.5
AoA [53]	✓	81.0	95.0	65.8	89.6	51.4	81.3	39.4	71.2	29.1	38.5	58.9	74.5	126.9	129.6
\mathcal{M}^2 Trans. [55]	✓	81.6	96.0	66.4	90.8	51.8	82.7	39.7	72.8	29.4	39.0	59.2	74.8	129.3	132.1
X-LAN [54]	✓	81.9	95.7	66.9	90.5	52.4	82.5	40.3	72.4	29.6	39.2	59.5	75.0	131.1	133.5
DLCT [39]	✓	82.4	96.6	67.4	91.7	52.8	83.8	40.6	74.0	29.8	39.6	59.8	75.3	133.3	135.4
GRIT [†]	✗	83.7	97.4	68.5	92.8	53.9	85.3	41.5	75.6	30.3	40.2	60.2	75.9	138.3	141.8
GRIT [†]	✓	84.1	97.6	69.4	93.5	54.9	86.3	42.5	76.8	30.9	41.0	61.2	77.1	141.3	143.8

of the datasets used for the detector’s pretraining, outperform all the methods that do not use large-scale pretraining of vision and language tasks (i.e., the methods in the second block entitled ‘w/o VL pretraining’). Moreover, our model with the detector pretrained solely on Visual Genome (i.e., ‘GRIT’) performs better than those relying on large-scale V&L pretraining but SimVLM_{huge}. Finally, our model with the pretrained detector on multiple datasets (i.e., ‘GRIT[†]’) outperforms SimVLM_{huge} leveraging large-scale V&L pretraining in CIDEr score (i.e., 144.2 vs 143.3).

Online Evaluation We also evaluate our models (i.e., a single model and an ensemble of six models) on the 40K testing images by submitting their results on the official evaluation server. Table 3.5 shows the results and those of all the published methods on the leaderboard. Table 3.5 presents the metric scores based on five (c5) and 40 reference captions (c40) per image. We can see that our method achieves the best scores for all the metrics. Note that even our single model outperforms all the published methods that use ensembles.

Table 3.6: Performance on the ArtEmis and nocaps datasets.

a) Performance on the ArtEmis test split								b) Performance on the nocaps validation split							
Method	V. E.	Performance Metrics						Method	V.E	In-Domain		Out-Domain		Overall	
		Type	B@1	B@2	B@3	B@4	M			R	Type	C	S	C	S
NN [65]	\mathcal{H}	36.4	13.9	5.4	2.2	10.2	21.0	NBT [62]	\mathcal{R}	62.7	10.1	54.0	8.6	53.9	9.2
ANP [65]	\mathcal{G}	39.6	13.4	4.2	1.4	8.8	20.2	Up-down [62]	\mathcal{R}	78.1	11.6	31.3	8.3	55.3	10.1
SAT [65]	\mathcal{G}	53.6	29.0	15.5	8.7	14.2	29.7	Trans. [55]	\mathcal{R}	78.0	11.0	29.7	7.8	54.7	9.8
\mathcal{M}^2 Trans. [65]	\mathcal{R}	50.7	28.2	15.9	9.5	13.7	28.0	\mathcal{M}^2 Trans. [55]	\mathcal{R}	85.7	12.1	38.9	8.9	64.5	11.1
GRIT [†]	$\mathcal{R}+\mathcal{G}$	70.1	40.1	20.9	11.3	16.8	33.3	GRIT [†]	$\mathcal{R}+\mathcal{G}$	105.9	13.6	72.6	11.1	90.2	12.8

3.4.7 Results on the ArtEmis and nocaps Datasets

As explained above, we evaluate our method on the ArtEmis and nocaps datasets. For ArtEmis, we train the model in the same way as COCO except for the number of training epochs, precisely, five epochs each for the training with the XE loss and that with the CIDEr-D optimization. For nocaps, we evaluate zero-shot inference performance, i.e., the performance of the model trained on COCO.

Table 3.6(a) shows the results of our method on the test split of ArtEmis [65]. It also show the results of existing methods reported in [65], which are grid-based [16,83], region-based [55], and a nearest neighbor method using a holistic vector to encode images (denoted as \mathcal{H}). Our method outperforms all these methods by a large margin.

Table 3.6(b) shows the results on the nocaps dataset, including the baseline methods reported in [55,62]. All the models are trained on the training split of the COCO datasets and tested on the validation split of nocaps, which consists of images with novel objects and captions with unseen vocabularies. Our method surpasses all the other methods including region-based methods [38, 55, 84] in both in-domain and out-of-domain images.

In brief, our model achieves the best performance across all the metrics in the both datasets among the compared methods. These results confirm the effectiveness of our method on datasets in different domains. See the supplementary material for the full results.

3.4.8 Computational Efficiency

Technical Measurement Details We measured the inference time of GRIT and two representative region-based methods, VinVL [58] and \mathcal{M}^2 Transformer [55]. It is the computational time per image from image input to caption generation. Specifically, we measured the time to generate a caption of length 20 with a beam size of five on a V100 GPU and averaged it over 1K times. The input image resolution was set to 800×1333 for VinVL and \mathcal{M}^2 Transformer as reported in [38, 58]. We set it to 384×640 for GRIT since it already achieves higher accuracy.

We measured the inference time of GRIT and two representative region-based methods, VinVL [58] and \mathcal{M}^2 Transformer [55], on the same machine having a Tesla V100-SXM2 of 16GB memory with CUDA version 10.0 and Driver version 410.104. It has Intel(R) Xeon(R) Gold 6148 CPU. The comparison was conducted following [46, 85]. Specifically, we excluded the time of preprocessing the image and loading it to the GPU device. Also, the images are rescaled to the resolutions such that all the compared methods achieve its highest performance for image captioning. For the compared methods, we used the official implementations of \mathcal{M}^2 Transformer¹ and VinVL². Regarding feature extraction, we extracted the region features from Faster R-CNN using the original implementation³ used by \mathcal{M}^2 Transformer and another implementation⁴ used by VinVL.

Inference Time Comparison Figure 3.1 shows the breakdown of the inference time for the three methods. GRIT reduces the time for feature extraction by a factor of **10 and more** compared with the others. Similar to \mathcal{M}^2 Transformer, GRIT has a lightweight caption generator and thus spends much less time than VinVL for generating a caption after receiving the visual features. GRIT can run with minibatch size up to 64 on a single V100 GPU, while others cannot afford large minibatch. With minibatch size ≥ 32 , the per-image inference time decreases to about 32ms.

It is seen that VinVL and \mathcal{M}^2 Transformer spend considerable time on feature extraction due to the forward pass through the CNN backbone with high resolution

¹<https://github.com/aimagelab/meshed-memory-transformer>

²<https://github.com/pzzhang/VinVL>

³<https://github.com/peteanderson80/bottom-up-attention>

⁴https://github.com/microsoft/scene_graph_benchmark

inputs and the computationally expensive regional operations. It is also noted that VinVL introduced class-agnostic NMS operations, which reduce a great amount of time consumed by class-aware NMS operations in the standard Faster R-CNN. On the other hand, we employ a Deformable DETR-based detector to extract region features without using all such operations. Table 3.7 shows the comparison on feature extraction.

Table 3.7: The inference time on feature extraction of different methods.

Method	Backbone	Detector	Regional Operations	Inference Time
VinVL _{large} [58]	ResNeXt-152	Faster R-CNN	Class-Agnostic NMS RoI Align, etc	304 ms
\mathcal{M}^2 Trans. [55]	ResNet-101	Faster R-CNN	Class-Aware NMS RoI Align, etc	736 ms
GRIT	Swin-Base	DETR-based	-	31 ms

Regarding caption generation, all the methods use beam search as the decoding strategy, with beam size of 5 and the maximum caption length of 20. Both \mathcal{M}^2 Transformer and GRIT employ a lightweight caption generator (caption decoder) having only 3 transformer layers with hidden dimension of 512 while VinVL_{large} has 24 transformer layers with hidden dimension of 1024; see Table 3.8. Thus, with the visual features as inputs, \mathcal{M}^2 Transformer and GRIT spend less inference time generating words than VinVL_{large} in the autoregressive manner.

Table 3.8: The inference time on caption generation of different methods.

Method	No. of Layers	Hidden Dim.	Inference Time
VinVL _{large} [58]	24	1024	542 ms
\mathcal{M}^2 Transformer [55]	3	512	174 ms
GRIT	3	512	138 ms

3.4.9 Qualitative Results

Figure 3.4, 3.5, 3.6, and 3.7 show some examples of the captions generated by our proposed method (GRIT) and another region-based method (\mathcal{M}^2 Transformer) given the same input images from the COCO test split. It is observed that the generated captions from GRIT are qualitatively better than those generated by the baseline method in terms of detecting and counting objects as well as describing their relationships in the given images. The inaccuracy of the captions generated by the baseline method might be due to the drawbacks of the region features extracted by a frozen pretrained object detector which produces wrong detection and lacks of contextual information.

3.5 Summary and Conclusion

In this chapter, we have proposed a Transformer-based architecture for image captioning named GRIT. It integrates the region features and the grid features extracted from an input image to extract richer visual information from input images. Previous SOTA methods employ a CNN-based detector to extract region features, which prevents the end-to-end training of the entire model and makes to high computational costs. Using the Swin Transformer for a backbone extracting the initial visual feature, GRIT resolves these two issues by employing a DETR-based detector. Furthermore, GRIT obtains grid features by updating the feature from the same backbone using a self-attention Transformer, aiming to extract richer context information complementing the region feature. These two features are fed to the caption generator equipped with a unique cross-attention mechanism, which computes and applies attention from the dual features on the generated caption sentence. The integration of all these components led to significant performance improvement. The experimental results validated our approach, showing that GRIT outperforms all published methods by a large margin in inference accuracy and speed.



GT-1: a child is brushing her hair in the mirror
GT-2: a little girl is brushing her hair in a bathroom
M²: a young girl holding a baseball bat in a
GRIT: a little girl brushing her hair with a brush



GT-1: an elephant walking not to far from a rhino in a forest
GT-2: an elephant and a rhino share a field with a pond
M²: a group of elephants grazing in a field
GRIT: an elephant and a rhino standing in a field



GT-1: a bike is parked alongside the lake shore
GT-2: a bike is parked on the grass in front of the lake
M²: a bicycle leaning against a bridge over the water
GRIT: a bike parked next to a bridge on the water



GT-1: 2 female tennis players standing with their rackets
GT-2: a pair of young women hold tennis balls and rackets
M²: a woman hitting a tennis ball with a tennis racket
GRIT: 2 people hold tennis rackets and balls on a court



GT-1: a cat holding a toothbrush in its mouth
GT-2: a cat chewing on a packaged pink toothbrush
M²: a cat laying on top of a pair of scissors
GRIT: a cat with a toothbrush in its mouth on



GT-1: the boy is playing video games in his bedroom
GT-2: a young man is sitting in a chair playing a video game
M²: a young man sitting in a chair holding a wii remote
GRIT: a man sitting in a chair playing a video game



GT-1: a woman is taking a turkey out of the oven
GT-2: a woman is taking the cooked turkey out of the oven.
M²: a woman taking a pizza out of an oven with a
GRIT: a woman taking a turkey out of an oven with



GT-1: a giraffe standing outside of a building next to a tree.
GT-2: a giraffe standing in a small piece of shade.
M²: two giraffes are standing in a zoo enclosure
GRIT: a giraffe standing in the dirt next to a building



GT-1: bowls on a table with meat and vegetables.
GT-2: four plates of different kind of food sitting on a table
M²: three plates of food on a wooden table with a
GRIT: four bowls of food and a spoon on a table

Figure 3.4: Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.



GT-1: a white cat is laying on a black skateboard
GT-2: A cat is sleeping on a skateboard.
M²: a kitten laying on the floor next to a skateboard
GRIT: a cat laying on a skateboard on the floor



GT-1: A baby elephant looking at a white duck
GT-2: A small elephant standing next to a white bird
M²: an elephant in a field with two birds in the
GRIT: a baby elephant walking in a field of grass



GT-1: Two children wrapped in blankets reading on a bed.
GT-2: Two children reading while lying in their bed
M²: two people laying in a bed with a
GRIT: two young boys sitting on a bed reading a book



GT-1: a kitchen with a refrigerator next to a sink.
GT-2: a red bucket sits in a sink next to an open refrigerator
M²: an open refrigerator with the door open in a kitchen
GRIT: a kitchen with a sink and an open refrigerator



GT-1: a woman pulling her luggage past an fire hydrant.
GT-2: a woman pulls a wheeled suitcase past a fire hydrant
M²: a person riding a skateboard down a street with a
GRIT: a person pulling a suitcase next to a fire hydrant



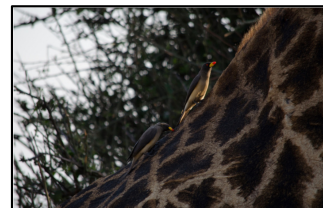
GT-1: two zebras in an animal park behind a wire fence
GT-2: two zebras in a zoo, behind a wire fence
M²: a zebra standing next to a fence in a
GRIT: two zebras standing behind a fence in a zoo



GT-1: a small teddy bear is wedged into an opening in a car dashboard
GT-2: little teddy bear attached to the dashboard of the car
M²: a stuffed teddy bear sitting in the back of a car
GRIT: a teddy bear sitting on the dashboard of a car



GT-1: horses racing on a race track with jockeys
GT-2: a group of jockeys ride horses on a track
M²: a group of people riding horses in a
GRIT: a group of jockeys riding horses on a track



GT-1: two birds going up the back of a giraffe.
GT-2: two birds sitting on the the back of a giraffe.
M²: a bird on the neck of a giraffe with a
GRIT: two birds sitting on the back of a giraffe

Figure 3.5: Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.



GT-1: An elderly man looks at a cell phone.

GT-2: An old man holding up a cell phone to his face.

M²: a man is taking a picture of himself on a motorcycle

GRIT: a man sitting in a chair holding a cell phone



GT-1: A bagel sandwich with scrambled egg and bacon.

GT-2: A poppy seed bagel sandwich with eggs and meat.

M²: a stack of pancakes on a white plate with a

GRIT: a bagel sandwich with meat and egg on a plate



GT-1: An ostrich and zebra fenced in with each other.

GT-2: An ostrich standing in a zoo pin near some zebras.

M²: a group of chickens and a fence in a field

GRIT: two zebras and an ostrich standing in a zoo



GT-1: a table top with some plates of food on it

GT-2: Two plates of breakfast foods on a restaurant table.

M²: a plate of food with eggs and meat on a table

GRIT: two plates of food on a table with a fork



GT-1: there are many people in the beach playing volleyball

GT-2: some males on some sand are playing volleyball

M²: a group of people playing soccer on the beach

GRIT: a group of men playing volleyball on the beach



GT-1: A polar bear playing with a ball in a small pond area.

GT-2: A bear is playing with a ball in the zoo

M²: a group of ducks swimming in the water with a

GRIT: two polar bears playing with a ball in the water



GT-1: A woman is paddle boarding down the river.

GT-2: A woman on a paddle board with people in the background.

M²: a woman standing on a boat in the water

GRIT: a woman standing on a paddle board in the water



GT-1: A wet brown dog in a bath tub.

GT-2: A wet dog in the tub getting a bath

M²: two dogs standing in the water with a

GRIT: a group of jockeys riding horses on a track



GT-1: an image of a woman sitting down on a couch with laptop

GT-2: A lady sitting on a couch with a laptop

M²: a woman laying on a bed with a

GRIT: a woman sitting on a couch with a laptop computer

Figure 3.6: Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.



GT-1: A dried black flower in a long, tall black & white vase.
GT-2: Thin black and white vase with black flowers.
M²: two white vases with a flower in them on a
GRIT: a black and white vase with a flower in it



GT-1: The bushels of bananas on display are purple
GT-2: A pile of black bananas and other fruit
M²: a bunch of fruits and vegetables in a basket
GRIT: a pile of bananas and other fruit on display



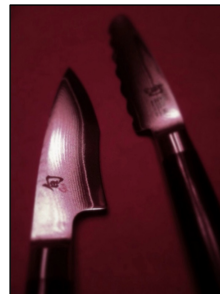
GT-1: A doll sitting at a table with fake food
GT-2: The doll is posed at the table eating a meal
M²: a young child sitting at a table with a plate of food
GRIT: a doll sitting at a table with a plate of food



GT-1: A woman throwing a frisbee outside at a park
GT-2: a woman is throwing a disk outside in the sun
M²: a woman holding a blue umbrella in the street
GRIT: a woman is throwing a frisbee in the street



GT-1: Two frisbees laying on the ground next to a sports water bottle.
GT-2: Two flying disks on the ground next to a water bottle
M²: a knife and a knife on a table with a
GRIT: two frisbees laying on the ground next to a bottle



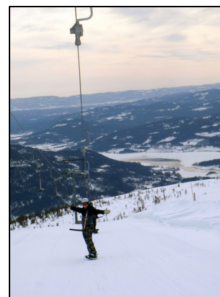
GT-1: Two knives are lying on a dark red surface.
GT-2: Two knives placed on a dining table
M²: a close up of a red tie with a
GRIT: two knives are on a red table with



GT-1: A woman laying in bed reading a book while wearing purple socks
GT-2: A woman is laying in bed reading a book
M²: a dog is looking at a person on a bed
GRIT: a woman laying on a bed with a book



GT-1: A zombie walking down a street covered in blood
GT-2: A man dressed like a zombie with other zombies around him.
M²: a man in a suit and tie walking with a group of people
GRIT: a man dressed as zombies walking down a street



GT-1: A person is standing near a ski-lift with a view of mountains
GT-2: A man stands beside a ski lift on a mountain
M²: a person riding a snowboard down a snow covered slope
GRIT: a person on a ski lift on a snowy mountain

Figure 3.7: Qualitative examples from our method (GRIT) and a region-based method (\mathcal{M}^2 Transformer) on the COCO test images. Zoom in for better view.

Chapter 4

LTMI: Lightweight Transformer for Many Inputs in Visual Dialog

4.1 Introduction

Recently, an increasing amount of attention has been paid to problems lying at the intersection of the vision and language domains. Many pilot tasks in this intersecting region have been designed and introduced to the research community, together with datasets. Visual dialog has been developed aiming at a higher level of vision-language interactions [18], as compared with VQA (visual question answering) [17] and VCR (visual commonsense reasoning). It extends VQA to multiple rounds; given an image and a history of question-answer pairs about the image, an agent is required to answer a new question. For example, to answer the question ‘*What color are they?*’, the agent needs to understand the context from a dialog history to know what ‘*they*’ refers to and look at the relevant image region to find out a color.

In recent studies of vision-language tasks, a primary concern has been to design an attention mechanism that can effectively deal with interactions between the two modalities. In the case of visual dialog, it becomes further necessary to consider interactions between an image, a question, and a dialog history or additionally multiple question-answer pairs in the history. Thus, the key to success will be how to deal with such interactions between three and more entities. Following a recent study [86], we will use the term *utility* to represent each of these input entities for clarity, since

the term *modality* is inconvenient to distinguish between the question and the dialog history.

Existing studies have considered attention from one utility to another based on different hypotheses, such as “question \rightarrow history \rightarrow image” path in [87, 88], and “question \rightarrow image \rightarrow history \rightarrow question” path in [89, 90], etc. These methods cannot take all the interactions between utilities into account, although the missing interactions could be crucial. Motivated by this, a recent study tries to capture all the possible interactions by using a factor graph [86]. However, building the factor graph is computationally inefficient, which seemingly hinders the method from unleashing the full potential of modeling all the interactions, especially when the dialog history grows long.

The Transformer [3] has become a standard neural architecture for various tasks in the field of natural language processing, especially since the huge success of its pre-trained model, BERT [12]. Its basic mechanism has recently been extended to the bi-modal problems of vision and language, yielding promising results [91–95]. Then, it appears to be natural to extend it further to deal with many-to-many utility interactions. However, it is not easy due to several reasons. As its basic structure is designed to be deal with self-attention, even in the simplest case of bi-modality, letting X and Y be the two utilities, there are four patterns of attention, $X \rightarrow Y$, $Y \rightarrow X$, $X \rightarrow X$, and $Y \rightarrow Y$; we need an independent Transformer block for each of these four. When extending this to deal with many-to-many utility interactions, the number of the blocks and thus of their total parameters increases proportionally with the square of the number of utilities, making it computationally expensive. Moreover, it is not apparent how to aggregate the results from all the interactions.

To cope with this, we propose a neural architecture named *Lightweight Transformer for Many Inputs* (LTMI) that can deal with all the interactions between many utilities. While it has a block structure similar to the Transformer and shares the core design of attention computation, it differs in the following two aspects. One is the difference in the implementation of multi-head attention. Multi-head attention in the Transformer projects the input feature space linearly to multiple lower-dimensional spaces, enabling to handle multiple attention maps, where the linear mappings are represented with learnable parameters. In the proposed model, we instead split the

input feature space to subspaces mechanically according to its indexes, removing all the learnable parameters from the attention computation.

The other difference from the Transformer is that LTMI is designed to receive multiple utilities and compute all the interactions to one utility from all the others including itself. This yields the same number of attended features as the input utilities, which are then concatenated in the direction of the feature space dimensions and then linearly projected back to the original feature space. We treat the parameters of the last linear projection as only learnable parameters in LTMI. This design makes it possible to retain sufficient representational power with a much fewer number of parameters, as compared with a natural extension of the Transformer block to many utilities. By using the same number of blocks in parallel as the number of utilities, we can deal with all the interactions between the utilities; see Fig. 4.2 for example. Assuming three utilities and the feature space dimensionality of 512, a layer consisting of LTMI has 2.38M parameters, whereas its counterpart based on naive Transformer extension has 28.4M parameters.

4.2 Related Work

4.2.1 Attention Mechanisms for Vision-Language Tasks

Attention mechanisms are currently indispensable to build neural architectures for vision-language tasks, such as VQA [96–103] and visual grounding [104–106], etc. Inspired by the recent success of the Transformer for language tasks [3, 12], several studies have proposed its extensions to bi-modal vision-language tasks [91–95, 107]. Specifically, for VQA, it is proposed to use intra-modal and inter-modal attention blocks and stack them alternately to fuse question and image features [92]; it is also proposed to use a cascade of modular co-attention layers that compute the self-attention and guided-attention of question and image features [95]. The method of pre-training a Transformer model used in BERT [12] is employed along with Transformer extension to bi-modal tasks for several vision-language tasks [91, 93, 94]. They first pre-train the models on external datasets, such as COCO Captions [108] or Conceptual Captions dataset [109], and then fine-tune them on several target tasks.

4.2.2 Visual Dialog

The task of visual dialog has recently been proposed by two groups of researchers concurrently [18, 110]. De Vries et al. introduced the GuessWhat?! dataset, which is built upon goal-oriented dialogs held by two agents to identify unknown objects in an image through a set of yes/no questions [110]. Das et al. released the VisDial dataset, which is built upon dialogs consisting of pairs of a question and an answer about an image that are provided in the form of natural language texts [18]. Kottur et al. recently introduced CLEVR-Dialog as the diagnostic dataset for visual dialog [111].

Most of the existing approaches employ an encoder-decoder architecture [112]. They can be categorized into the following three groups by the design of the encoder: i) fusion-based methods, e.g., LF [18] and HRE [18], which fuses the inputs by their concatenation followed by the application of a feed-forward or recurrent network, and Synergistic [113], which fuses the inputs at multiple stages; ii) attention-based methods that compute attended features of the input image, question, and history utilities, e.g., MN [18], CoAtt [90], HCIAE [88], Synergistic [113], ReDAN [89], FGA [86], and CDF [114]; ReDAN compute the attention over several reasoning steps, FGA models all the interactions over many utilities via a factor graph; iii) methods that attempt to resolve visual co-reference, e.g., RvA [115] and CorefNMN [116], which use neural modules to form an attention mechanism, DAN [87], which employs a network having two attention modules, and AMEM [117], which utilizes a memory mechanism for attention. As for the decoder, there are two designs: i) discriminative decoders that rank the candidate answers using the cross-entropy loss [18] or the n-pair loss [88]; and ii) generative decoders that yield an answer by using a MLE loss [18], weighted likelihood estimation [118], or a combination with adversarial learning [88, 90], which trains a discriminator on both positive and negative answers, then transferring it to the generator with auxiliary adversarial learning.

Other approaches include GNN [119], which models relations in a dialog by an unknown graph structure; the employment of reinforcement learning [120, 121]; and HACAN [122] which adopts policy gradient to learn the impact of history by intentionally imposing the wrong answer into dialog history. In [123, 124], pre-trained vision-language models are adopted, which consist of many Transformer blocks with hundreds of millions parameters, leading to some performance gain. Qi et al. [125]

present model-agnostic principles for visual dialog to maximize performance.

4.3 Lightweight Transformer for Many Utilities

4.3.1 Attention Mechanism of Transformer

As mentioned earlier, the Transformer has been applied to several bi-modal vision-language tasks, yielding promising results. The Transformer computes and uses attention from three types of inputs, Q (query), K (key), and V (value). Its computation is given by

$$\mathcal{A}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d}} \right) V, \quad (4.1)$$

where Q , K , and V are all collection of features, each of which is represented by a d -dimensional vector. To be specific, $Q = [q_1, \dots, q_M]^\top \in \mathbb{R}^{M \times d}$ is a collection of M features; similarly, K and V are each a collection of N features, i.e., $K, V \in \mathbb{R}^{N \times d}$. In Eq.(4.1), V is attended with the weights computed from the similarity between Q and K .

The above computation is usually multi-plexed in the way called multi-head attention. It enables to use a number of attention distributions in parallel, aiming at an increase in representational power. The outputs of H ‘heads’ are concatenated, followed by linear transformation with learnable weights $W^O \in \mathbb{R}^{d \times d}$ as

$$\mathcal{A}^M(Q, K, V) = \left[\text{head}_1, \dots, \text{head}_H \right] W^O. \quad (4.2)$$

Each head is computed as follows:

$$\text{head}_h = \mathcal{A}(QW_h^Q, KW_h^K, VW_h^V), \quad h = 1, \dots, H, \quad (4.3)$$

where $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d \times d_H}$ each are learnable weights inducing a linear projection from the feature space of d -dimensions to a lower space of $d_H (= d/H)$ -dimensions. Thus, one attentional block $\mathcal{A}^M(Q, K, V)$ has the following learnable weights:

$$(W_1^Q, W_1^K, W_1^V), \dots, (W_H^Q, W_H^K, W_H^V) \text{ and } W^O. \quad (4.4)$$

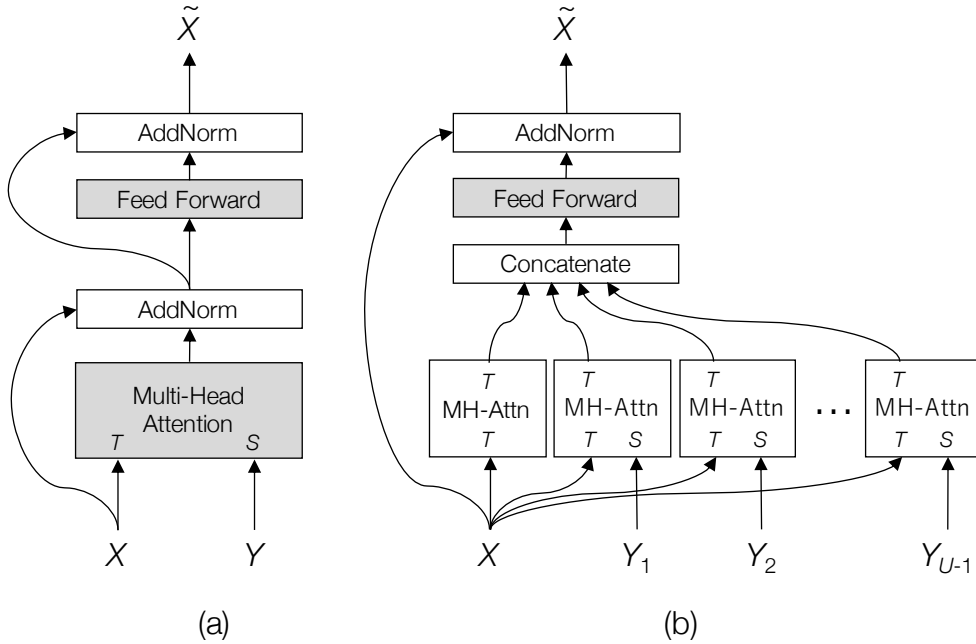


Figure 4.1: (a) Source-to-target attention for bi-modal problems implemented by the standard Transformer block; the source Y is attended by weights computed from the similarity between the target X and Y . (b) The proposed block that can deal with many utilities; the source features $\{Y_1, \dots, Y_{U-1}\}$ are attended by weights computed between them and the target X . Shaded boxes have learnable weights

4.3.2 Application to Bi-Modal Tasks

While Q , K , and V in NLP tasks are of the same modality (i.e., language), the above mechanism has been extended to bi-modality and applied to vision-language tasks in recent studies [91–95, 107]. They follow the original idea of the Transformer, considering attention from source features Y to target features X as

$$\mathcal{A}_Y(X) = \mathcal{A}^M(X, Y, Y). \quad (4.5)$$

In MCAN [95], language feature is treated as the source and visual feature is as the target. In [93] and others [91, 92, 94, 107], co-attention, i.e., attention in the both directions, is considered. Self-attention, i.e., the attention from features to themselves, is given as a special case by

$$\mathcal{A}_X(X) = \mathcal{A}^M(X, X, X). \quad (4.6)$$

In the above studies, the Transformer block with the source-to-target attention and that with the self-attention are independently treated and are stacked, e.g., alternately or sequentially.

4.3.3 Lightweight Transformer for Many Inputs

Now suppose we wish to extend the above attention mechanism to a greater number of utilities¹; we denote the number by U . If we consider every possible source-target pairs, there are $U(U - 1)$ cases in total, as there are U targets, for each of which $U - 1$ sources exist. Then we need to consider attention computation $\mathcal{A}_Y(X)$ over $U - 1$ sources Y 's for each target X . Thus, the straightforward extension of the above attention mechanism to U utilities needs $U(U - 1)$ times the number of parameters listed in Eq.(4.4). If we stack the blocks, the total number of parameters further increases proportionally.

To cope with this, we remove all the weights from Eq.(4.5). To be specific, for each head $h(= 1, \dots, H)$, we choose and freeze (W_h^Q, W_h^K, W_h^V) as

$$W_h^Q = W_h^K = W_h^V = \underbrace{[O_{d_H}, \dots, O_{d_H}, I_{d_H}]_{(h-1)d_H}}_{(h-1)d_H}, \underbrace{[O_{d_H}, \dots, O_{d_H}]_{(H-h)d_H}}_{(H-h)d_H}, \quad (4.7)$$

where O_{d_H} is a $d_H \times d_H$ zero matrix and I_{d_H} is a $d_H \times d_H$ identity matrix. In short, the subspace for each head is determined to be one of H subspaces obtained by splitting the d -dimensional feature space with its axis indexes. Besides, we set $W^O = I$, which is the linear mapping applied to the concatenation of the heads' outputs. Let $\bar{\mathcal{A}}_Y(X)$ denote this simplified attention mechanism.

Now let the utilities be denoted by $\{X, Y_1, \dots, Y_{U-1}\}$, where $X \in \mathbb{R}^{M \times d}$ is the chosen target and others $Y_i \in \mathbb{R}^{N_i \times d}$ are the sources. Then, we compute all the source-to-target attention as $\bar{\mathcal{A}}_{Y_1}(X), \dots, \bar{\mathcal{A}}_{Y_{U-1}}(X)$. In the standard Transformer block (or rigorously its natural extensions to bi-modal problems), the attended features are simply added to the target as $X + \mathcal{A}_Y(X)$, followed by normalization and subsequent computations. To recover some of the loss in representational power due to the simplification yielding $\bar{\mathcal{A}}_Y(X)$, we propose a different approach to aggregate

¹As we stated in Introduction, we use the term *utility* here to mean a collection of features.

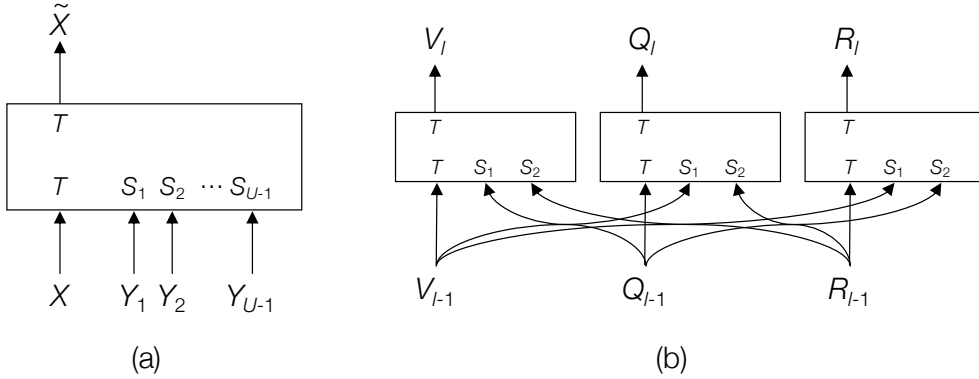


Figure 4.2: (a) Simplified symbol of the proposed block shown in Fig. 4.1(b). (b) Its application to Visual Dialog

$\bar{\mathcal{A}}_{Y_1}(X), \dots, \bar{\mathcal{A}}_{Y_{U-1}}(X)$ and X . Specifically, we concatenate all the source-to-target attention plus the self-attention $\bar{\mathcal{A}}_X(X)$ from X to X as

$$X_{\text{concat}} = [\bar{\mathcal{A}}_X(X), \bar{\mathcal{A}}_{Y_1}(X), \dots, \bar{\mathcal{A}}_{Y_{U-1}}(X)], \quad (4.8)$$

where $X_{\text{concat}} \in \mathbb{R}^{M \times U d}$. We then apply linear transformation to it given by $W \in \mathbb{R}^{U d \times d}$ and $b \in \mathbb{R}^d$ with a single fully-connected layer, followed by the addition of the original X and layer normalization as

$$\tilde{X} = \text{LayerNorm}(\text{ReLU}(X_{\text{concat}} W + \mathbf{1}_M \cdot b^\top) + X), \quad (4.9)$$

where $\mathbf{1}_M$ is M -vector with all ones. With this method, we aim at recovery of representational power as well as the effective aggregation of information from all the utilities.

Memory features When computing $\bar{\mathcal{A}}_Y(X)$, we perform the following form of computation

$$\mathcal{A}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V,$$

where we compute a matrix product QK^\top as above. In the computation of $\bar{\mathcal{A}}_X(Y)$, we need another matrix product, but it is merely the transposed matrix KQ^\top due to the symmetry between X and Y . For the computational efficiency, we perform computation of $\bar{\mathcal{A}}_Y(X)$ and $\bar{\mathcal{A}}_X(Y)$ simultaneously; see `MultiHeadAttention(X, Y)`

in our code. Further, following [100], we also pad X and Y with two d -dimensional vectors that are randomly initialized with He normal initialization. This implements “no-where-to-attend” features as memory features in the computation of $\bar{\mathcal{A}}_Y(X)$ and $\bar{\mathcal{A}}_X(Y)$.

4.3.4 Interactions between All Utilities

We have designed a basic block as shown in Fig. 4.1(b) that deals with attention from many sources to a single target. We wish to consider all possible interactions between all the utilities, not a single utility being the only target. To do this, we use U basic blocks to consider all the source-to-target attention. Using the basic block as a building block, we show how an architecture is designed for visual dialog having three utilities, visual features V , question features Q , and dialog history features R , in Fig. 4.2(b).

4.4 Implementation Details for Visual Dialog

4.4.1 Problem Definition

The problem of Visual Dialog is stated as follows. An agent is given the image of a scene and a dialog history containing T entities, which consists of a caption and question-answer pairs at $T - 1$ rounds. Then, the agent is further given a new question at round T along with 100 candidate answers for it and requested to answer the question by choosing one or scoring each of the candidate answers.

4.4.2 Representation of Utilities

We first extract features from an input image, a dialog history, and a new question at round T to obtain their representations. For this, we follow the standard method employed in many recent studies. For the image utility, we use the bottom-up mechanism [38], which extracts region-level image features using the Faster R-CNN [8] pre-trained on the Visual Genome dataset [126]. For each region (i.e., a bounding box = an object), we combine its CNN feature and geometry to get a d -dimensional vector v_i ($i = 1, \dots, K$), where K is the predefined number of regions. We then

define $V = [v_1, v_2, \dots, v_K]^\top \in \mathbb{R}^{K \times d}$. For the question utility, after embedding each word using an embedding layer initialized by pre-trained GloVe vectors, we use two-layer Bi-LSTM to transform them to q_i ($i = 1, \dots, N$), where N is the number of words in the question. We optionally use the positional embedding widely used in NLP studies. We examine its effects in an ablation test. We then define $Q = [q_1, \dots, q_N]^\top \in \mathbb{R}^{N \times d}$. For the dialog history utility, we choose to represent it as a single utility here. Thus, each of its entities represents the initial caption or the question-answer pair at one round. As with the question utility, we use the same embedding layer and a two-layer Bi-LSTM together with the positional embeddings for the order of dialog rounds to encode them with a slight difference in formation of an entity vector r_i ($i = 1, \dots, T$), where T is the number of Q&A plus the caption. We then define $R = [r_1, \dots, r_T]^\top \in \mathbb{R}^{T \times d}$.

Image Utility The image utility is represented by the standard method employed in many recent studies. It is based on the bottom-up mechanism [38], which extracts region-level image features using the Faster R-CNN pre-trained on the Visual Genome dataset [126]. For each input image, we select the top K objects, and represent each of them by a visual feature $v_i^r \in \mathbb{R}^{2048}$ and a bounding box expressed by $(x_{i,1}, x_{i,2})$ and $(x_{i,3}, x_{i,4})$ (the coordinates of the upper-left and lower-right corners.)

The feature vector v_i^r is then converted into another vector $v_i^f \in \mathbb{R}^d$ as follows. We introduce the following notation to express a single FC layer with ReLU, to which dropout regularization is applied:

$$\text{MLP}_{k \rightarrow d}(x) \equiv \text{Dropout}(\text{ReLU}(W^\top x + b)), \quad (4.10)$$

where $x \in \mathbb{R}^k$ is the input and $W \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^d$ are the weights and biases. Then, v_i^f is obtained by

$$v_i^f = \text{LayerNorm}(\text{MLP}_{2048 \rightarrow d}(v_i^r)), \quad (4.11)$$

where LayerNorm is the layer normalization [127] applied to the output.

The bounding box geometry is converted into $v_i^b \in \mathbb{R}^d$ in the following way. First, the image is resized to 600×600 pixels and the bounding box geometry is transformed

accordingly. Then, representing each of the four coordinates by a one-hot vector of size 600, we convert them into the embedding vectors $\hat{x}_{i,1}, \dots, \hat{x}_{i,4} (\in \mathbb{R}^d)$ using four different embedding layers. Then, we obtain v_i^b as below

$$v_i^b = \sum_{j=1}^4 \text{LayerNorm}(\text{MLP}_{d \rightarrow d}(\hat{x}_{i,j})). \quad (4.12)$$

Finally, v_i^f encoding the visual feature and v_i^b encoding the spatial feature are aggregated by adding and normalizing as

$$v_i = \text{LayerNorm}(v_i^f + v_i^b). \quad (4.13)$$

The resulting v_i 's for the K objects ($i = 1, \dots, K$) comprise a matrix $V = [v_1, v_2, \dots, v_K]^\top \in \mathbb{R}^{K \times d}$, which gives the representation of the visual utility.

Optional Image Feature Enrichment. In the experiment of comparing ensembles on the test split of Visdial v1.0, we enrich the image features for further improvement. To be specific, for each object, we also obtain a class label with highest probability (e.g. ‘cat’, ‘hair’, and ‘car’) and the top 20 attributes for each class label (e.g., ‘curly’, ‘blond’, ‘long’, and so on, for the label ‘hair’). These can be extracted from the Faster R-CNN along with the above CNN features and bounding box geometry. We incorporate these into the image utility representation in the following way.

The class label for the i -th object is first encoded into an embedding vector $e_i^c \in \mathbb{R}^{300}$ using the same embedding layer as the question. Then, we convert e_i^c into a d -dimensional vector v_i^c by

$$v_i^c = \text{LayerNorm}(\text{MLP}_{300 \rightarrow d}(e_i^c)). \quad (4.14)$$

Similarly, for the top 20 attributes of each object i , we encode them into embedding vectors of size 300, i.e. $e_{i,1}^a, \dots, e_{i,20}^a$, and then convert them further into $v_i^a \in \mathbb{R}^d$ as

$$v_i^a = \sum_{j=1}^{20} \text{LayerNorm}(\text{MLP}_{300 \rightarrow d}(e_{i,j}^a)w_{i,j}^a), \quad (4.15)$$

where $w_{i,j}^a$ is the confidence score extracted from the Faster R-CNN for attribute j of the i -th object. Then, the visual feature v_i^f , the spatial feature v_i^b , the class feature v_i^c , and the attribute feature v_i^a are aggregated by their addition followed by normalization as

$$v_i = \text{LayerNorm}(v_i^f + v_i^b + v_i^c + v_i^a). \quad (4.16)$$

We then use these vectors to form the matrix V instead of Eq.(4.13).

Question Utility The question utility is also obtained by the standard method but with one exception, the employment of positional embedding used in NLP studies. Note that we examine its effects in an ablation test shown in the main paper. A given question sentence is first fit into a sequence of N words; zero-padding is applied if necessary. Each word w_i ($i = 1, \dots, N$) is embedded into a vector e_i of a fixed size using an embedding layer initialized with pre-trained GloVe vectors [128]. They are then inputted into two-layer Bi-LSTM, obtaining two d -dimensional vectors \vec{h}_i and \overleftarrow{h}_i as their higher-layer hidden state:

$$\begin{aligned} \vec{h}_i &= \text{LSTM}(e_i, \overrightarrow{h_{i-1}}), \\ \overleftarrow{h}_i &= \text{LSTM}(e_i, \overleftarrow{h_{i+1}}). \end{aligned} \quad (4.17)$$

Their concatenation, $h_i = [\vec{h}_i^\top, \overleftarrow{h}_i^\top]^\top$, is then projected back to a d -dimensional space using a linear transformation, yielding a vector q_i^f . Positional embedding q_i^p from the paper [3] is added to get the final representation $q_i \in \mathbb{R}^d$ of w_i as

$$q_i = \text{LayerNorm}(q_i^f + q_i^p). \quad (4.18)$$

The representation of the question utility is given as $Q = [q_1, \dots, q_N]^\top \in \mathbb{R}^{N \times d}$.

Dialog History Utility In this study, we choose to represent the dialog history as a single utility. Each of its entities represents the question-answer pair at one round. As with previous studies, the caption is treated as the first round of $2N$ -word which is padded or truncated if necessary. For each round $t > 1$, the word sequences of the question and the answer at the round is concatenated into $2N$ -word sequence with zero padding if necessary. As with the question utility, after embedding each

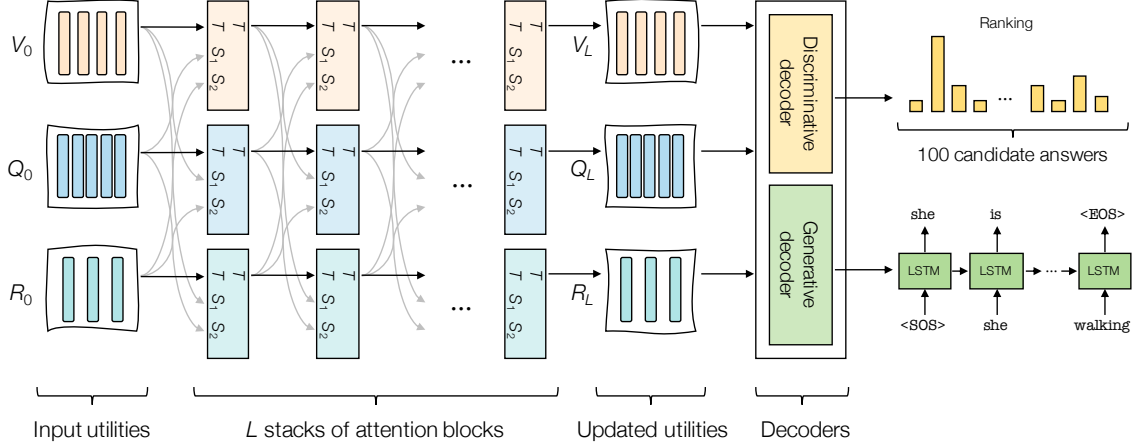


Figure 4.3: The entire network built upon the proposed LTMI for Visual Dialog

word into a GloVe vector, the resulting sequence of $2N$ embedded vectors is inputted to two-layer Bi-LSTM, from which only their last (higher-layer) hidden states are extracted to construct $2d$ -dimensional vector $[\vec{h}_0^\top, \overleftarrow{h}_{2N}^\top]^\top$. We then project it with a linear transform to a d -dimensional space, yielding $r_t^f \in \mathbb{R}^d$. For the linear projection, we use different learnable weights from the question utility. As in Eq.(4.18), we add positional embedding, which represents the order of rounds, and then apply layer normalization, yielding a feature vector of the round t question-answer pair. The history utility is then given by $R = [r_1, \dots, r_T]^\top \in \mathbb{R}^{T \times d}$.

4.4.3 Overall Network Design

Figure 4.3 shows the entire network. It consists of an encoder and a decoder. The encoder consists of L stacks of the proposed attention blocks; a single stack has U blocks in parallel, as shown in Fig. 4.2(b). We set $V_0 = V$, $Q_0 = Q$, and $R_0 = R$ as the inputs of the first stack. After the l -th stack, the representations of the image, question, and dialog history utilities are updated as V_l , Q_l , and R_l , respectively. In the experiments, we apply dropout with the rate of 0.1 to the linear layer inside every block. There is a decoder(s) on top of the encoder. We consider a discriminative decoder and a generative decoder, as in previous studies. Their design is explained below.

4.4.4 Design of Decoders

Decoders receive the updated utility representations, V_L , Q_L , and R_L at their inputs. We convert them independently into d -dimensional vectors c_V , c_Q , and c_R , respectively. This conversion is performed by a simple self-attention computation. We take c_V as an example here. First, attention weights over the entities of V_L are computed by a two-layer network as

$$a_V = \text{softmax}(\text{ReLU}(V_L W_1 + \mathbf{1}_K b_1^\top) W_2 + \mathbf{1}_K b_2), \quad (4.19)$$

where $W_1 \in \mathbb{R}^{d \times d}$, $W_2 \in \mathbb{R}^{d \times 1}$, $b_1 \in \mathbb{R}^d$, $b_2 \in \mathbb{R}^1$, and $\mathbf{1}_K$ is K -vector with all ones. Then, c_V is given by

$$c_V = \sum_{i=1}^K v_{L,i}^\top a_{V,i}, \quad (4.20)$$

where $v_{L,i}$ is the i -th row vector of V_L and $a_{V,i}$ is the i -th attention weight (a scalar). The others, i.e., c_Q and c_R , can be obtained similarly.

These vectors are integrated and used by the decoders. In our implementation for visual dialog, we found that c_R does not contribute to better results; thus we use only c_V and c_Q . Note that this does not mean the dialog utility R is not necessary; it is interacted with other utilities inside the attention computation, contributing to the final prediction. The two d -vectors c_V and c_Q are concatenated as $[c_V^\top, c_Q^\top]^\top$, and this is projected to d -dimensional space, yielding a context vector $c \in \mathbb{R}^d$.

We design the discriminative and generative decoders following the previous studies. Receiving c and the candidate answers, the two decoders compute the score of each candidate answer in different ways.

Discriminative Decoder A discriminative decoder outputs the likelihood score for each of 100 candidate answers for the current question at round T in the following way. We use a similar architecture to the one used to extract question features in Sec. 4.4.2 to convert each candidate answer (indexed by $i(= 1, \dots, 100)$) to a feature vector $a_i \in \mathbb{R}^d$. Specifically, it is two-layer Bi-LSTM receiving a candidate answer at its input, on top of which there is a linear projection layer followed by layer normalization. Using the resulting vectors, the score p_i for i -th candidate answer is

computed by

$$p_i = \text{logsoftmax}_i(a_1^\top c, \dots, a_{100}^\top c). \quad (4.21)$$

In the test phase, we sort the candidate answers using these scores. In the training phase, the cross-entropy loss \mathcal{L}_D between $p = [p_1, \dots, p_{100}]^\top$ and the ground truth label encoded by a one-hot vector y is minimized:

$$\mathcal{L}_D = - \sum_{i=1}^{100} y_i p_i. \quad (4.22)$$

When relevance scores $s = [s_1, \dots, s_{100}]^\top$ over the answer candidates are available (called dense annotation in the VisDial dataset) rather than a single ground truth answer, we can use them by setting $y_i = s_i$ for all i 's and minimize the above loss. We employ dropout with rate of 0.1 for the LSTM.

Generative Decoder Following [18], we also consider a generative decoder to score the candidate answers using the log-likelihood scores. The generative decoder consists of a two-layer LSTM to generate an answer using the context vector c as the initial hidden state. In the training phase, we predict the next token based on the current token from the ground truth answer. In details, we first append the special token ‘‘SOS’’ at the beginning of the ground truth answer, then embedding all the sentence into the embedding vectors $a_{gt} = [w_0, w_1, \dots, w_N]$ where w_0 is the embedding vector of ‘‘SOS’’ token. The hidden state $h_n \in \mathbb{R}^d$ at the n -th timestep (extracted from the higher-layer LSTM) is computed given w_{n-1} and h_{n-1} as follows:

$$h_n = \text{LSTM}(w_{n-1}, h_{n-1}), \quad (4.23)$$

where h_0 is initialized by c . Thus, we compute p_n , the log-likelihood of n -th word as

$$p_n = \text{logsoftmax}_j(W_n^\top h_n + b_n), \quad (4.24)$$

where $W_n \in \mathbb{R}^{d \times |V|}$ and $p_n \in \mathbb{R}^{|V|}$, where $|V|$ is the vocabulary size; and j is the index of n -th word in the vocabulary.

In the training phase, we minimize \mathcal{L}_G , the summation of the negative log-

likelihood defined by

$$\mathcal{L}_G = - \sum_{n=1}^N p_n. \quad (4.25)$$

In the validation and test phase, for each candidate answer $A_{T,i}$, we compute $s_i = \sum_{n=1}^N p_n^{(A_{T,i})}$ where $p_n^{(A_{T,i})}$ is the log-likelihood of the n -th word in the candidate answer $A_{T,i}$ which is computed similarly as in Eq.(4.24). Then, the rankings of the candidate answers are derived as $\text{softmax}_i(s_1, \dots, s_{100})$. We employ dropout with rate of 0.1 for the LSTM.

4.4.5 Multi-Task Learning

We observe in our experiments that accuracy is improved by training the entire network using the two decoders simultaneously. This is simply done by minimizing the sum of the losses, \mathcal{L}_D for the discriminative one and \mathcal{L}_G for the generative one (we do not use weights on the losses):

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_G. \quad (4.26)$$

The increase in performance may be attributable to the synergy of learning two tasks while sharing the same encoder. Details will be given in Sec. 4.5.3.

4.5 Experiments on Visual Dialog

4.5.1 Experimental Setup

Dataset We use the VisDial v1.0 dataset in our experiments which consists of the train 1.0 split (123,287 images), the val 1.0 split (2,064 images), and test v1.0 split (8,000 images). Each image has a dialog composed of 10 question-answer pairs along with a caption. For each question-answer pair, 100 candidate answers are given. The val v1.0 split and 2,000 images of the train v1.0 split are provided with dense annotations (i.e., relevance scores) for all candidate answers. Although the test v1.0 split was also densely annotated, the information about the ground truth answers and the dense annotations are not publicly available.

Evaluation metrics From the visual dialog challenge 2018, normalized discounted cumulative gain (NDCG) has been used as the principal metric to evaluate methods on the VisDial v1.0 dataset. Unlike other classical retrieval metrics such as R@1, R@5, R@10, mean reciprocal rank (MRR), and mean rank, which are only based on a single ground truth answer, NDCG is computed based on the relevance scores of all candidate answers for each question, which can properly handle the case where each question has more than one correct answer, such as ‘*yes it is*’ and ‘*yes*’; such cases do occur frequently.

Other configurations We employ the standard method used by many recent studies for the determination of hyper-parameters etc. For the visual features, we detect $K = 100$ objects from each image. For the question and history features, we first build the vocabulary composed of 11,322 words that appear at least five times in the training split. The captions, questions, and answers are truncated or padded to 40, 20, and 20 words, respectively. Thus, $N = 20$ for the question utility Q . T for the history utilities varies depending on the number of dialogs. We use pre-trained 300-dimensional GloVe vectors [128] to initialize the embedding layer, which is shared for all the captions, questions, and answers.

For the attention blocks, we set the dimension of the feature space to $d = 512$ and the number of heads H in each attention block to 4. We mainly use models having two stacks of the proposed attention block. We train our models on the VisDial v0.9 and VisDial v1.0 dataset using the Adam optimizer [27] with 5 epochs and 15 epochs respectively. The learning rate is warmed up from 1×10^{-5} to 1×10^{-3} in the first epoch, then halved every 2 epochs. The batch size is set to 32 for the both datasets.

Table 4.1 shows the hyper-parameters used in our experiments, which are selected following the previous studies. We perform all the experiments on a GPU server that has four Tesla V100-SXM2 of 16GB memory with CUDA version 10.0 and Driver version 410.104. It has Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz of 80 cores with the RAM of 376GB memory. We use Pytorch version 1.2 [129] as the deep learning framework.

Table 4.1: Hyper-paramters used in the training procedure.

Hyper-parameter	Value
Warm-up learning rate	1e-5
Warm-up factor	0.2
Initial learning rate after the 1st epoch	1e-3
β_1 in Adam	0.9
β_2 in Adam	0.997
ϵ in Adam	1e-9
Weight decay	1e-5
Number of workers	8
Batch size	32

Table 4.2: Comparison of the performances of different methods on the validation set of VisDial v1.0 with discriminative and generative decoders.

Model	Discriminative						Generative					
	NDCG \uparrow	MRR \uparrow	R@1 \uparrow	R@5 \uparrow	R@10 \uparrow	Mean \downarrow	NDCG \uparrow	MRR \uparrow	R@1 \uparrow	R@5 \uparrow	R@10 \uparrow	Mean \downarrow
MN [18]	55.13	60.42	46.09	78.14	88.05	4.63	56.99	47.83	38.01	57.49	64.08	18.76
CoAtt [90]	57.72	62.91	48.86	80.41	89.83	4.21	59.24	49.64	40.09	59.37	65.92	17.86
HCIAE [88]	57.75	62.96	48.94	80.5	89.66	4.24	59.70	49.07	39.72	58.23	64.73	18.43
ReDAN [89]	59.32	64.21	50.6	81.39	90.26	4.05	60.47	50.02	40.27	59.93	66.78	17.4
LTMI	62.72	62.32	48.94	78.65	87.88	4.86	63.58	50.74	40.44	61.61	69.71	14.93

4.5.2 Comparison with State-of-the-art Methods

Compared methods We compare our method with previously published methods on the VisDial v0.9 and VisDial v1.0 datasets, including LF, HRE, MN [18], LF-Att, MN-Att (with attention) [18], SAN [101], AMEM [117], SF [130], HCIAE [88] and Sequential CoAttention model (CoAtt) [90], Synergistic [113], FGA [86], GNN [119], RvA [115], CorefNMN [116], DAN [87], and ReDAN [89], all of which were trained without using external datasets or data imposition. Unless noted otherwise, the results of our models are obtained from the output of discriminative decoders.

Results on the val v1.0 split We first compare single-model performance on the val v1.0 split. We select here MN, CoAtt, HCIAE, and ReDAN for comparison, as their performances from the both decoders in all metrics are available in the literature. To

be specific, we use the accuracy values reported in [89] for a fair comparison, in which these methods are reimplemented using the bottom-up-attention features. Similar to ours, all these methods employ the standard design of discriminative and generative decoders as in [18]. Table 4.2 shows the results. It is seen that our method outperforms all the compared methods on the NDCG metric with large margins regardless of the decoder type. Specifically, as compared with ReDAN, the current state-of-the-art on the VisDial v1.0 dataset, our model has improved NDCG from 59.32 to 62.72 and from 60.47 to 63.58 with discriminative and generative decoders, respectively.

Results on the test-standard v1.0 split We next consider performance on the test-standard v1.0 split. In our experiments, we encountered a phenomenon that accuracy values measured by NDCG and other metrics show a trade-off relation (see the supplementary material for details), depending much on the choice of metrics (i.e., NDCG or others) for judging convergence at the training time. This is observed in the results reported in [89] and is attributable to the inconsistency between the two types of metrics. Thus, we show two results here, the one obtained using NDCG for judging convergence and the one using MRR for it; the latter is equivalent to performing early stopping.

Table 4.3 shows single-model performances on the blind test-standard v1.0 split. With the outputs from the discriminative decoder, our model gains improvement of 3.33pp in NDCG from the best model. When employing the aforementioned early stopping, our model achieves at least comparable or better performance in other metrics as well.

Many previous studies report the performance of an ensemble of multiple models. To make a comparison, we create an ensemble of 16 models with some differences, from initialization with different random seeds to whether to use sharing weights across attention blocks or not, the number of attention blocks (i.e. $L = 2, 3$), and the number of objects in the image (i.e. $K = 50, 100$). Aiming at achieving the best performance, we also enrich the image features by incorporating the class label and attributes of each object in an image, which are also obtained from the pre-trained Faster R-CNN model. Details are given in the supplementary material. We take the average of the outputs (probability distributions) from the discriminative decoders of these models to rank the candidate answers. Furthermore, we also test fine-tuning

Table 4.3: Comparison in terms of **single-model** performance on the blind test-standard v1.0 split of the VisDial v1.0 dataset. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger .

Model	NDCG \uparrow	MRR \uparrow	R@1 \uparrow	R@5 \uparrow	R@10 \uparrow	Mean \downarrow
LF [18]	45.31	55.42	40.95	72.45	82.83	5.95
HRE [18]	45.46	54.16	39.93	70.45	81.50	6.41
MN [18]	47.50	55.49	40.98	72.30	83.30	5.92
MN-Att [18]	49.58	56.90	42.42	74.00	84.35	5.59
LF-Att [18]	49.76	57.07	42.08	74.82	85.05	5.41
FGA [86]	52.10	63.70	49.58	80.97	88.55	4.51
GNN [119]	52.82	61.37	47.33	77.98	87.83	4.57
CorefNMN [116]	54.70	61.50	47.55	78.10	88.80	4.40
RvA [115]	55.59	63.03	49.03	80.40	89.83	4.18
Synergistic [113]	57.32	62.20	47.90	80.43	89.95	4.17
DAN [87]	57.59	63.20	49.63	79.75	89.35	4.30
LTMI\star	59.03	64.08	50.20	80.68	90.35	4.05
LTMI	60.92	60.65	47.00	77.03	87.75	4.90

each model with its discriminative decoder on the available dense annotations from the train v1.0 and val v1.0, where the cross-entropy loss with soft labels (i.e. relevance scores) is minimized for two epochs. Table 4.4 shows the results. It is observed that our ensemble model (w/o the fine-tuning) achieves the best NDCG = 66.53 in all the ensemble models.

With optional fine-tuning, our ensemble model further gains a large improvement in NDCG, resulting in the third place in the leaderboard. The gap in NDCG to the first place (VD-BERT) is only 0.25pp, while our model yields performance that is better in all the other metrics, i.e, by 2.14pp, 5.67pp, and 3.37pp in MRR, R@5, and R@10, respectively, and 5.36% reduction in Mean.

Table 4.5 shows the number of parameters of the multi-modal attention mechanism employed in the recent methods along with their NDCG scores on the VisDial v1.0 test-standard set. We exclude the parameters of the networks computing the input

Table 4.4: Comparison in terms of **ensemble-model** performance on the blind test-standard v1.0 split of the VisDial v1.0 dataset. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger .

Model	NDCG \uparrow	MRR \uparrow	R@1 \uparrow	R@5 \uparrow	R@10 \uparrow	Mean \downarrow
FGA [86]	52.10	67.30	53.40	85.28	92.70	3.54
Synergistic [113]	57.88	63.42	49.30	80.77	90.68	3.97
DAN [87]	59.36	64.92	51.28	81.60	90.88	3.92
ReDAN [89]	64.47	53.73	42.45	64.68	75.68	6.63
LTMI	66.53	63.19	49.18	80.45	89.75	4.14
P1_P2 [125] \dagger	74.91	49.13	36.68	62.98	78.55	7.03
VD-BERT [123] \dagger	75.13	50.00	38.28	60.93	77.28	6.90
LTMI \dagger	74.88	52.14	38.93	66.60	80.65	6.53

utilities and the decoders, as they are basically shared among these methods. ‘Naive Transformer’ consists of two stacks of transformer blocks with simple extension to three utilities as mentioned in Sec. 4.1. The efficiency of our models can be observed. Note also that the gap between (Q, V) and (Q, V, R) is small, contrary to the argument in [125].

4.5.3 Ablation Study

To evaluate the effect of each of the components of our method, we perform the ablation study on the val v1.0 split of VisDial dataset. We evaluate here the accuracy of the discriminative decoder and the generative decoder separately. We denote the former by D-NDCG and the latter by G-NDCG, and the accuracy of their averaged model by A-NDCG (i.e., averaging the probability distributions over the candidate answers obtained by the discriminative and generative decoders). The results are shown in Table 4.6 and Table 4.7.

The first block of Table 4.6 shows the effect of the number of stacks of the proposed attention blocks. We observe that the use of two to three stacks achieves good performance on all three measures. More stacks did not bring further improvement, and thus are omitted in the table.

Table 4.5: Comparison in terms of the number of parameters of the attention mechanism. The result obtained by early stopping on MRR metric is denoted by \star and those with fine-tuning on dense annotations are denoted by \dagger .

Model	# params	MRR \uparrow	NDCG \uparrow
DAN [87]	12.6M	63.20	57.59
RvA [115]	11.9M	63.03	55.59
Naive Transformer	56.8M	62.09	55.10
LTMI* (MRR-based)	4.8M	64.08	59.92
LTMI (Q, V)	4.8M	60.65	60.92
LTMI (Q, V, R)	4.8M	60.76	61.12

The second block of Table 4.6 shows the effect of self-attention, which computes the interaction within a utility, i.e., $\bar{\mathcal{A}}_X(X)$. We examine this because it can be removed from the attention computation. It is seen that self-attention does contribute to good performance. The third block shows the effects of how to aggregate the attended features. It is seen that their concatenation yields better performance than their simple addition. The fourth block shows the impact of sharing the weights across the stacks of the attention blocks. If the weights can be shared as in [131], it contributes a further decrease in the number of parameters. We observe that the performance does drop if weight sharing is employed, but the drop is not very large.

The first block of Table 4.7 shows the effect of how to aggregate the context features c_V , c_Q , and c_R in the decoder(s), which are obtained from the outputs of our encoder. As mentioned above, the context vector c_R of the dialog history does not contribute to the performance. However, the context vector c_v of the image is important for achieving the best performance. The second block of Table 4.7 shows the effects of simultaneously training the both decoders (with the entire model). It is seen that this contributes greatly to the performance; this indicates the synergy of learning two tasks while sharing the encoder, resulting better generalization as compared with those trained with a single decoder.

We have also confirmed that the use of fewer objects leads to worse results. Besides, the positional embedding for representing the question and history utilities as

Table 4.6: Ablation study on the components of our method on the val v1.0 split of VisDial dataset. \uparrow indicates the higher the better.

(a)

Component	Details	A-NDCG \uparrow	D-NDCG \uparrow	G-NDCG \uparrow
Number of attention blocks	1	65.37	62.06	62.95
	2	65.75	62.72	63.58
	3	65.42	62.48	63.22
Self-Attention	No	65.38	61.76	63.31
	Yes	65.75	62.72	63.58
Attended features aggregation	Add	64.12	60.28	61.49
	Concat	65.75	62.72	63.58
Shared Attention weights	No	65.75	62.72	63.58
	Yes	65.57	62.50	63.24

well as the spatial embedding (i.e., the bounding box geometry of objects) for image utility representation have a certain amount of contribution.

4.5.4 Qualitative Results

Visualization of Generated Attention Figure 4.4 shows attention weights generated in our model on two rounds of Q&A on two images. We show here two types of attention. One is the self-attention weights used to compute the context vectors c_V and c_Q . For c_V , the attention weights a_V are generated over image regions (i.e., bounding boxes), as in Eq.(4.19). Similarly, for c_Q , the attention weights are generated over question words. These two sets of attention weights are displayed by brightness of the image bounding-boxes and darkness of question words, respectively, in the center and the rightmost columns. It can be observed from these that the relevant regions and words are properly highlighted at each Q&A round.

The other attention we visualize is the source-to-target attention computed inside the proposed block. We choose here the image-to-question attention $\bar{\mathcal{A}}_V(Q)$ and the history-to-question attention $\bar{\mathcal{A}}_R(Q)$. For each, we compute the average of the

Table 4.7: Ablation study on the components of our method on the val v1.0 split of VisDial dataset. \uparrow indicates the higher the better.

(b)				
Component	Details	A-NDCG \uparrow	D-NDCG \uparrow	G-NDCG \uparrow
Context feature aggregation	[Q]	65.12	61.50	63.19
	[Q, V]	65.75	62.72	63.58
	[Q, V, R]	65.53	62.37	63.38
Decoder Type	Gen	-	-	62.35
	Disc	-	61.80	-
	Both	65.75	62.72	63.58
The number of objects in an image	36	65.25	62.40	63.08
	50	65.24	62.29	63.12
	100	65.75	62.72	63.58
Positional and spatial embeddings	No	65.18	61.84	62.96
	Yes	65.75	62.72	63.58

attention weights over all the heads computed inside the block belonging to the upper stack. In Fig. 4.4, the former is displayed by the red boxes connected between an image region and a question word; only the region with the largest weight is shown for the target word; the word with the largest self-attention weight is chosen for the target. The history-to-question attention is displayed by the Q&As highlighted in blue color connected to a selected question word that is semantically ambiguous, e.g., *'its'*, *'he'*, and *'his'*. It is seen that the model performs proper visual grounding for the important words, *'hair'*, *'shorts'*, and *'tusks'*. It is also observed that the model properly resolves the co-reference for the words, *'he'* and *'its'*.

We provide additional examples of the results obtained by our method in Figs. 4.5-4.8. They are divided into two groups, results for which the top-1 prediction coincides with the ground truth answer (Figs. 4.5-4.6) and those for which they do not coincide (Figs. 4.7-4.8). For each result, we show the attention maps created on the input image and question, respectively.

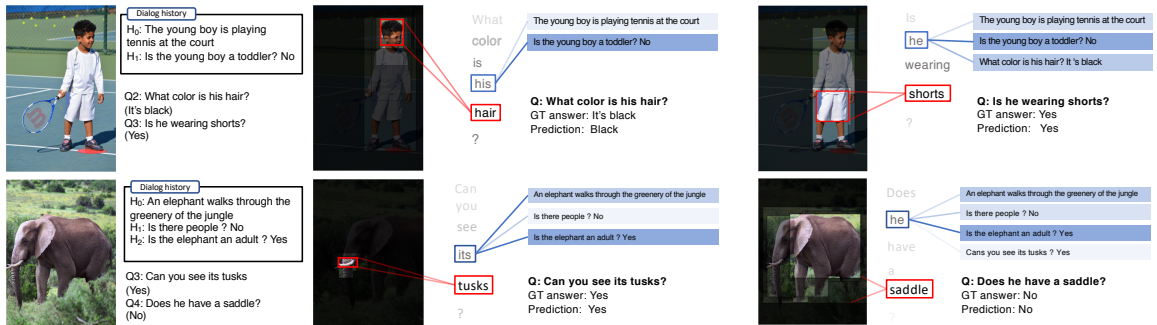


Figure 4.4: Examples of visualization for the attention weights generated in our model at two Q&A rounds on two images. See Sec. 4.5.4 for details.

Table 4.8: Comparison of response generation evaluation results with objective measures.

Model	Video Feat.	CIDEr	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROUGE_L
Baseline [132]	VGG	0.618	0.231	0.141	0.095	0.067	0.102	0.259
Ours	VGG	0.841	0.266	0.172	0.118	0.086	0.117	0.296
Baseline [132]	I3D	0.727	0.256	0.161	0.109	0.078	0.113	0.277
Ours	I3D	0.851	0.277	0.178	0.122	0.088	0.119	0.302

4.6 Experiments on Audio Visual Scene-aware Dialog

To test the generality of the proposed method on other tasks as well as its performance on a greater number of utilities, we additionally apply it to the Audio Visual Scene-aware Dialog (AVSD) task [132]. This task requires a system to generate an answer to a question about events seen in a video given with a previous dialog. AVSD provides more utilities than Visual Dialog, i.e., audio features and video features, such as VGG or I3D features (I3D RGB sequence and I3D flow sequence). We build a network by simply replacing the multi-modal attention mechanism in the baseline model of [132] with a simple extension of the proposed attention mechanism. Details are given below.

4.6.1 Network Design

Following the baselines [132], we extract the question utility Q using a two-layer LSTM. We separate the caption from the dialog history and feed it into another two-layer LSTM to obtain the caption utility C . Similar to [132], the dialog history consisting of previous question-answer pairs is inputted into a hierarchical LSTM network; specifically, we encode each question-answer pair with one LSTM and summarize the obtained encodings with another LSTM, yielding a final vector representation c_r . All LSTMs used for language encoding have d units. We convert words into vectors with a shared embedding layer initialized with GLoVe vectors.

The video provides two sources of features, i.e., video features and audio features. We use the audio features extracted from the pre-trained VGGish model [132], which are fed to a projection layer, providing the audio utility A ; it is represented as a collection of d -dimensional vectors. For video processing, following [132], we consider two models with different features: i) VGG features extracted from four uniformly sampled frames in the video, giving the video utility V , and ii) I3D features extracted by the I3D network pre-trained on an action recognition task, which are forwarded to projection layers to obtain an I3D-rgb utility and an I3D-flow utility denoted by V and F .

To compute the multi-modal attention between U utilities, we add a stack of U proposed attention blocks; $U = 4$ for the model (i) and $U = 5$ for (ii). To make the designs of two models (i) and (ii) similar, we use only A utility to attend language utilities; and only Q and C are allowed to attend audio and video utilities. After obtaining the updated representations of all utilities, we summarize each utility into a single vector by the self-attention mechanism, in which the summarized vector of question utility is denoted by c_q . We concatenate all these vectors together with c_r , projecting it into a d -dimensional vector of context representation c .

The decoder architecture is similar to the generative decoder described in Sec. 4.4.4 except that the input of the decoder at the i -th step is the concatenation of w_{i-1} , c_q , and c_r . At the time of inference, we use the beam search technique to efficiently find the most likely hypothesis generated by the decoder.

4.6.2 Experimental Setup

Following [132], we perform the experiment on the AVSD prototype which is split into training, validation, and test sets with 6172, 732, and 733 videos, respectively. Each video is collected from the Charades dataset, annotated with a caption and 10 dialog rounds. The hidden size d is set to 512; the GLoVe vectors are 300-dimensional. We train the models in 15 epochs using the Adam optimizer with initial learning rate 1×10^{-3} in all the experiments. The dropout with rate of 0.2 is applied for the LSTMs.

4.6.3 Experimental Results

Table 4.8 shows the results, which include evaluation on a number of metrics to measure the quality of generated answers, i.e. CIDEr, BLEU, METEOR, ROUGE_L. It is seen that our models outperform the baselines presented in [132] over all the metrics; specifically, it improves the CIDEr score by 22.3% (from 0.618 to 0.841) with VGG features and by 12.4% (from 0.727 to 0.851) with I3D features.

4.7 Summary and Conclusion

In this chapter, we have proposed LTMI (Lightweight Transformer for Many Inputs) that can deal with all the interactions between multiple input utilities in an efficient way. As compared with other methods, the proposed architecture is much simpler in terms of the number of parameters as well as the way of handling inputs (i.e., their equal treatment), and nevertheless surpasses the previous methods in accuracy; it achieves the new state-of-the-art results on the VisDial datasets, e.g., high NDCG scores on the VisDial v1.0 dataset. Thus, we believe our method can be used as a simple yet strong baseline.

Q&A at a round		Q&A at another round	
 <p>Q2: What color is the truck ? GT: The truck is orange , black , and white</p>	 <p>What color is the truck ? Pred: The truck is orange , black , and white</p>	 <p>Q8: Are there any tree 's or grass ? GT: Yes , there are trees</p>	 <p>Are there any tree 's or grass ? Pred: Yes , there are trees</p>
 <p>Q2: What is the tennis player wearing ? GT: White tennis dress</p>	 <p>What is the tennis player wearing ? Pred: White tennis dress</p>	 <p>Q4: Is she wearing a hat ? GT: A visor</p>	 <p>Is she wearing a hat ? Pred: A visor</p>
 <p>Q7: Is there a fence ? GT: Yes</p>	 <p>Is there a fence ? Pred: Yes</p>	 <p>Q9: Can you see a ball ? GT: Yes</p>	 <p>Can you see a ball ? Pred: Yes</p>
 <p>Q1: What color is the tennis court ? GT: Green</p>	 <p>What color is the tennis court ? Pred: Green</p>	 <p>Q4: What color is the ball ? GT: Green</p>	 <p>What color is the ball ? Pred: Green</p>

Figure 4.5: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

















Q&A at a round		Q&A at another round	
 <p>Q4: Is it sunny ? GT: Yes</p>	 <p>Q4: Is it sunny ? Pred: Yes</p>	 <p>Q7: Can you see any signs ? GT: Yes , a stop sign and a railroad crossing sign</p>	 <p>Q7: Can you see any signs ? Pred: Yes , a stop sign and a railroad crossing sign</p>
 <p>Q4: Are there any animals ? GT: Just the bird</p>	 <p>Q4: Are there any animals ? Pred: Just the bird</p>	 <p>Q7: What color is her shirt ? GT: Mauve</p>	 <p>Q7: What color is her shirt ? Pred: Mauve</p>
 <p>Q1: Is the light lit up ? GT: Yes</p>	 <p>Q1: Is the light lit up ? Pred: Yes</p>	 <p>Q2: Are there cars ? GT: Yes</p>	 <p>Q2: Are there cars ? Pred: Yes</p>
 <p>Q4: Is anyone holding a bat ? GT: Yes</p>	 <p>Q4: Is anyone holding a bat ? Pred: Yes</p>	 <p>Q5: Can you see a ball ? GT: Yes</p>	 <p>Q5: Can you see a ball ? Pred: Yes</p>

Figure 4.6: Examples of results for which the top-1 prediction is the same as the ground truth answer on the validation split of Visdial v1.0. Each row shows selected two rounds of Q&A for one image.

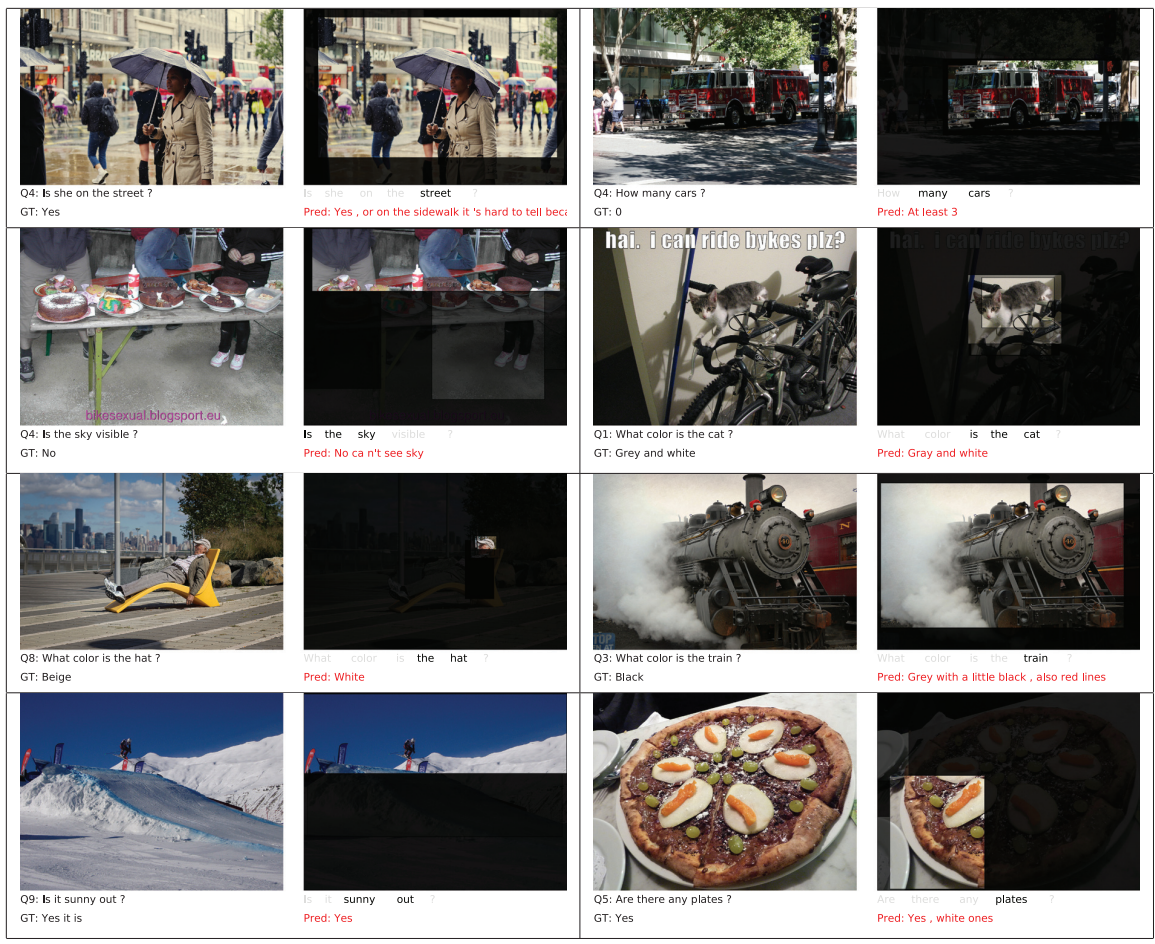


Figure 4.7: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

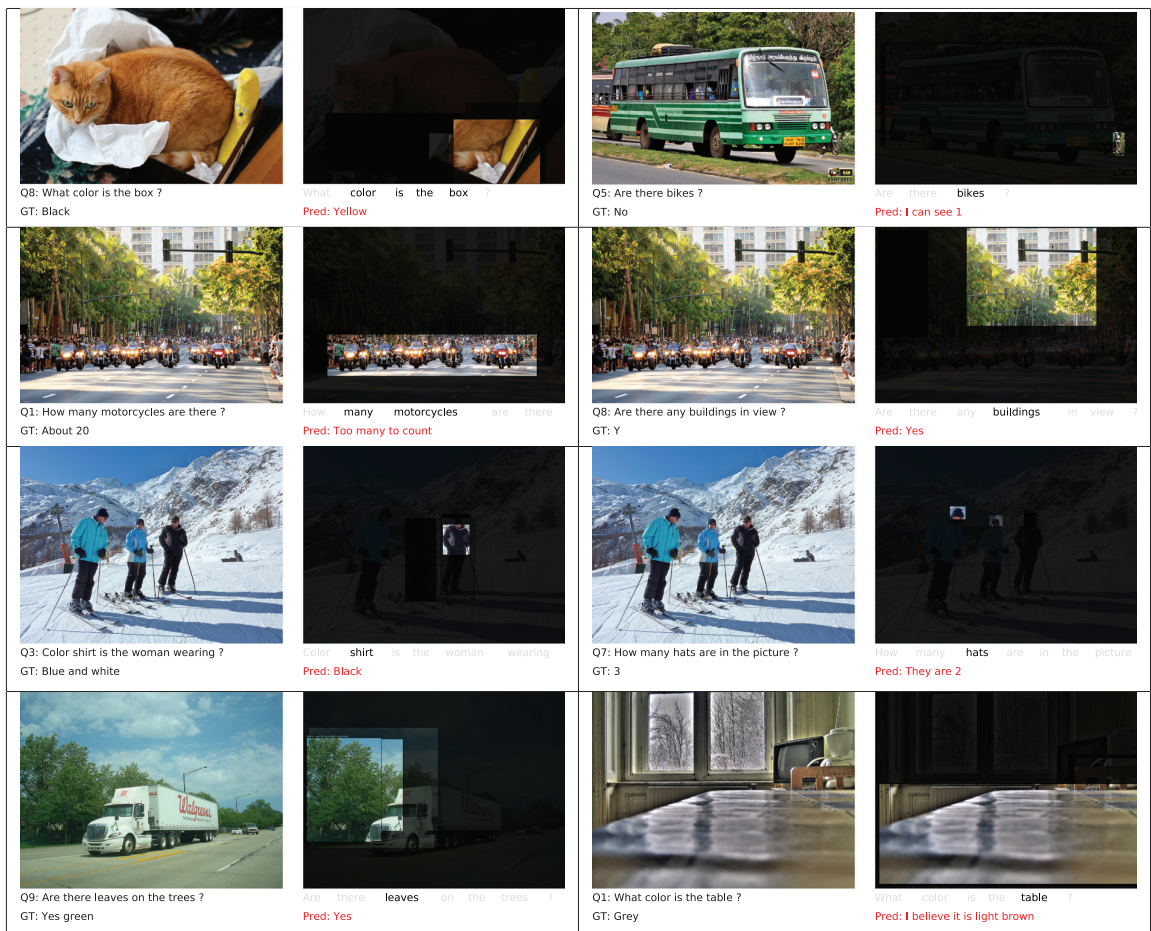


Figure 4.8: Examples of results for which the top-1 prediction is different from the ground truth answer on the validation split of Visdial v1.0.

Chapter 5

LWIT: Improving Performance on Instruction Following Tasks

5.1 Introduction

There is a growing interest in the community in making an embodied AI agent perform a complicated task following natural language directives. Recent studies of vision-language navigation tasks (VLN) have made significant progress [19–21]. However, these studies consider navigation in static environments, where the action space is simplified, and there is no interaction with objects in the environment.

To consider more complex tasks, a benchmark named ALFRED was developed recently [1]. It requires an agent to accomplish a household task in interactive environments following given language directives. Compared with VLN, ALFRED is more challenging as the agent needs to (1) reason over a greater number of instructions and (2) predict actions from larger action space to perform a task in longer action horizons. The agent also needs to (3) localize the objects to manipulate by predicting the pixel-wise masks. Previous studies (e.g., [1]) employ a Seq2Seq model, which performs well on the VLN tasks [23]. However, it works poorly on ALFRED. Overall, existing methods only show limited performance; there is a huge gap with human performance.

In this chapter, we propose a new method that leads to significant performance improvements. It is based on several ideas. Firstly, we propose to choose a single

instruction to process at each timestep from the given series of instructions. This approach contrasts with previous methods that encode them into a single long sequence of word features and use soft attention to specify which instruction to consider at each timestep implicitly [1, 133, 134]. Our method chooses individual instructions explicitly by learning to predict when the agent completes an instruction. This makes it possible to utilize constraints on parsing instructions, leading to a more accurate alignment of instructions and action prediction. Secondly, we propose a two-stage approach to the interpretation of the selected instruction. In its first stage, the method interprets the instruction without using visual inputs from the environment, yielding a tentative prediction of an action-object sequence. In the second stage, the prediction is integrated with the visual inputs to predict the action to do and the object to manipulate. The tentative interpretation makes it clear to interact with what class of objects, contributing to an accurate selection of objects to interact with.

Moreover, we acquire multiple agent egocentric views of a scene as visual inputs and integrate them using a hierarchical attention mechanism. This allows the agent to have a wider field of views, leading to more accurate navigation. To be specific, converting each view into an object-centric representation, we integrate those for the multiple views into a single feature vector using hierarchical attention conditioned on the current instruction.

Besides, we propose a module for predicting precise pixel-wise masks of objects to interact with, referred to as the mask decoder. It employs the object-centric representation of the center view, i.e., multiple object masks detected by the object detector. The module selects one of these candidate masks to specify the object to interact with. In the selection, self-attention is applied to the candidate masks to weight them; they are multiplied with the tentative prediction of the pairs of action and an object class and the detector’s confidence scores for the candidate masks.

The experimental results show that the proposed method outperforms all the existing methods by a large margin and ranks first in the challenge leaderboard as of the time of submission. A preliminary version of the method won an international competition held last year. The present version further improved the task success rate in unseen and seen environments to 8.37% and 29.16%, respectively, which are significantly higher than the previously published SOTA (0.39% and 3.98%, respec-

tively) [1].

5.2 Related Work

5.2.1 Embodied Vision-Language Tasks

Many studies have been recently conducted on the problem of making an embodied AI agent follow natural language directives and accomplish the specified tasks in a three-dimensional environment while properly interacting with it. Vision-language navigation (VLN) tasks have been the most extensively studied, which require an agent to follow navigation directions in an environment.

Several frameworks and datasets for simulating real-world environments have been developed to study the VLN tasks. The early ones lack photo-realism and/or natural language directions [135–137]. Recent studies consider perceptually-rich simulated environments and natural language navigation directions [19, 138, 139]. In particular, since the release of the Room-to-Room (R2R) dataset [19] that is based on real imagery [140], VLN has attracted increasing attention, leading to the development of many methods [20, 23, 141–143].

Several variants of VLN tasks have been proposed. A study [144] allows the agent to communicate with an adviser using natural language to accomplish a given goal. In a study [145], the agent placed in an environment attempts to find a specified object by communicating with a human by natural language dialog. A recent study [146] proposes the interactive environments where users can collaborate with an agent by not only instructing it to complete tasks, but also acting alongside it. Another study [147] introduces a continuous environment based on the R2R dataset that enables an agent to take more fine-grained navigation actions. A number of other embodied vision-language tasks have been proposed such as visual semantic planning [148, 149] and embodied question answering [150–153].

5.2.2 Existing Methods for ALFRED

As mentioned earlier, ALFRED was developed to consider more complicated interactions with environments, which are missing in the above tasks, such as manipulat-

ing objects. Several methods for it have been proposed so far. A baseline method [1] employs a Seq2Seq model with an attention mechanism and a progress monitor [23], which is prior art for the VLN tasks. In [134], a pre-trained Mask R-CNN is employed to generate object masks. It is proposed in [133] to train the agent to follow instructions and reconstruct them. In [154], a modular architecture is proposed to exploit the compositionality of instructions. These methods have brought about only modest performance improvements over the baseline. A concurrent study [155] proposes a modular architecture design in which the prediction of actions and object masks are treated separately, as with ours. Although it achieves notable performance improvements, the study’s ablation test indicates that the separation of the two is not the primary source of the improvements. Closely related to ALFRED, ALFWorld [156] has been recently proposed to combine TextWorld [157] and ALFRED for creating aligned environments, which enable transferring high-level policies learned in the text world to the embodied world.

5.3 Proposed Method

The proposed model consists of three decoders (i.e., instruction, mask, and action decoders) with the modules extracting features from the inputs, i.e., the visual observations of the environment and the language directives. We first summarize ALFRED and then explain the components one by one.

5.3.1 Summary of ALFRED

ALFRED is built upon AI2Thor [136], a simulation environment for embodied AI. An agent performs seven types of tasks in 120 indoor scenes that require interaction with 84 classes of objects, including 26 receptacle object classes. For each object class, there are multiple visual instances with different shapes, textures, and colors.

The dataset contains 8,055 expert demonstration episodes of task instances. They are sequences of actions, whose average length is 50, and they are used as a ground truth action sequence at training time. For each of them, language directives annotated by AMT workers are provided, which consist of a goal statement G and a set of step-by-step instructions, S_1, \dots, S_L . The alignment between each instruction and

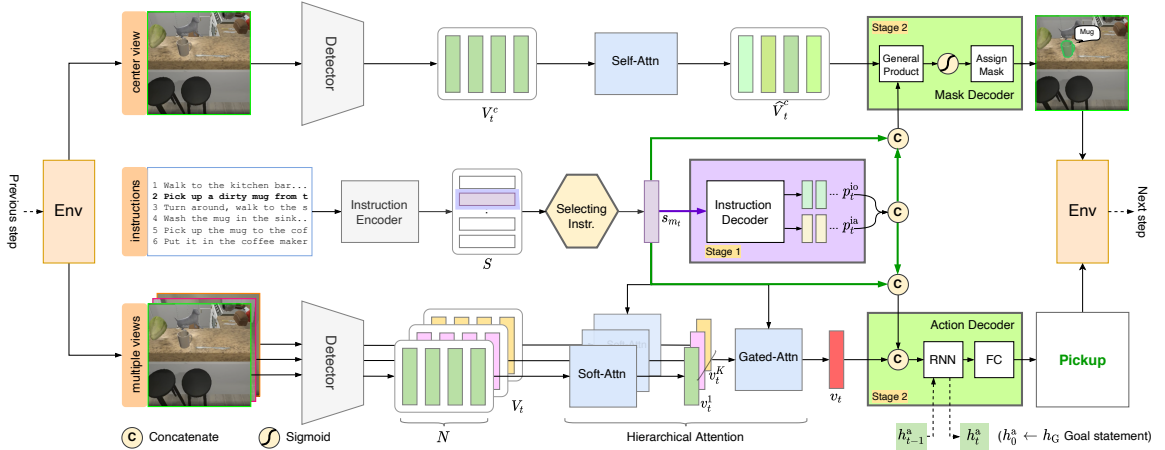


Figure 5.1: Architecture overview of the proposed model. It consists of the modules encoding the visual inputs and the language directives (Sec. 5.3.2), the instruction decoder with an instruction selector (Sec. 5.3.3), the action decoder (Sec. 5.3.4), and the mask decoder (Sec. 5.3.5).

a segment of the action sequence is known. As multiple AMT workers annotate the same demonstrations, there are 25,743 language directives in total.

We wish to predict the sequence of agent’s actions, given G and S_1, \dots, S_L of a task instance. There are two types of actions, navigation actions and manipulation actions. There are five navigation actions (e.g., `MoveAhead` and `RotateRight`) and seven manipulation actions (e.g., `Pickup` and `ToggleOn`). The manipulation actions accompany an object. The agent specifies it using a pixel-wise mask in the egocentric input image. Thus, the *outputs* are a sequence of actions with, if necessary, the object masks.

5.3.2 Feature Representations

Object-centric Visual Representations

Unlike previous studies [1, 133, 134], we employ the object-centric representations of a scene [158], which are extracted from a pre-trained object detector (i.e., Mask R-CNN [9]). It provides richer spatial information about the scene at a more fine-grained level and thus allows the agent to localize the target objects better. Moreover, we make the agent look wider by capturing the images of its surroundings, aiming to enhance its navigation ability.

Specifically, at timestep t , the agent obtains visual observations from K egocentric

views. For each view k , we encode the visual observation by a bag of N object features, which are extracted the object detector. Every detected object is associated with a visual feature, a mask, and its confidence score. We project the visual feature into \mathbb{R}^d with a linear layer, followed by a ReLU activation and dropout regularization [159] to obtain a single vector; thus, we get a set of N object features for view k , $V_t^k = (v_{t,1}^k, \dots, v_{t,N}^k)$. We obtain V_t^1, \dots, V_t^K for all the views.

Language Representations

We encode the language directives as follows. We use an embedding layer initialized with pre-trained GloVe [128] vectors to embed each word of the L step-by-step instructions and the goal statement. For each instruction $i(= 1, \dots, L)$, the embedded feature sequence is inputted to a two-layer LSTM [33], and its last hidden state is used as the feature $s_i \in \mathbb{R}^d$ of the instruction. We use the same LSTM for all the instructions with dropout regularization. We encode the goal statement G in the same manner using an LSTM with the same architecture different weights, obtaining $h_G \in \mathbb{R}^d$.

5.3.3 Instruction Decoder

Selecting Instructions

Previous studies [1, 133, 134] employ a Seq2Seq model in which all the language directives are represented as a *single sequence* of word features, and soft attention is generated over it to specify the portion to deal with at each timestep. We think this method could fail to correctly segment instructions with time, even with the employment of progress monitoring [23]. This method does not use a few constraints on parsing the step-by-step instructions that they should be processed in the given order and when dealing with one of them, the other instructions, especially the future ones, will be of little importance.

We propose a simple method that can take the above constraints into account, which explicitly represents which instruction to consider at the current timestep t . The method introduces an integer variable $m_t(\in [1, L])$ storing the index of the instruction to deal with at t .

To update m_t properly, we introduce a virtual action representing the *completion of a single instruction*, which we treat equally to the original twelve actions defined in ALFRED. Defining a new token COMPLETE to represent this virtual action, we augment each instruction’s action sequence provided in the expert demonstrations always end with COMPLETE. At training time, we train the action decoder to predict the augmented sequences. At test time, the same decoder predicts an action at each timestep; if it predicts COMPLETE, this means completing the current instruction. The instruction index m_t is updated as follows:

$$m_t = \begin{cases} m_{t-1} + 1, & \text{if } \operatorname{argmax}(p_{t-1}^a) = \text{COMPLETE} \\ m_{t-1}, & \text{otherwise,} \end{cases} \quad (5.1)$$

where p_{t-1}^a is the predicted probability distribution over all the actions at time $t - 1$, which will be explained in Sec. 5.3.4. The encoded feature s_{m_t} of the selected instruction is used in all the subsequent components, as shown in Fig. 5.1.

Decoder Design

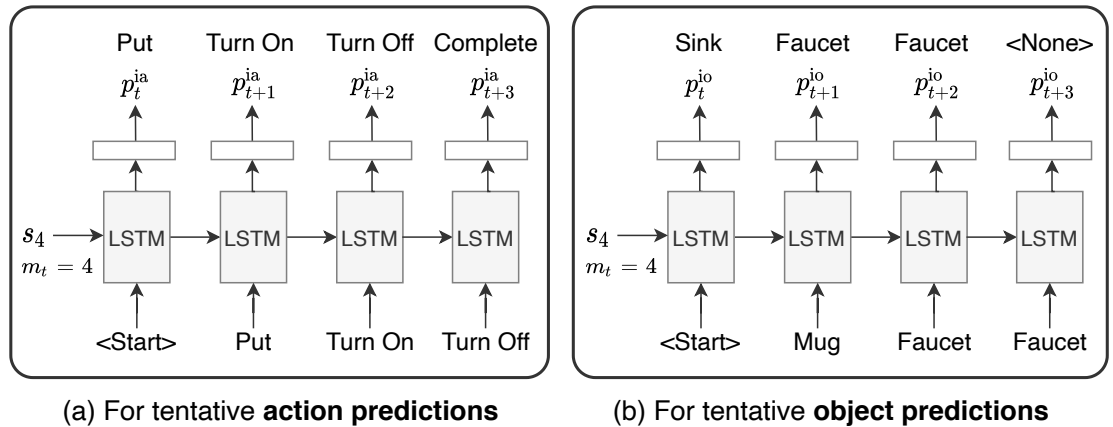


Figure 5.2: An example illustrates how we reinitialize the hidden states of the two LSTMs in the instruction encoder by s_{m_t} when $m_t = m_{t-1} + 1$ ($m_t = 4$).

As explained earlier, our method employs a two-stage approach for interpreting the instructions. The instruction decoder (see Fig. 5.1) runs the first stage, where it interprets the instruction encoded as s_{m_t} *without any visual input*. To be specific, it transforms s_{m_t} into the sequence of action-object pairs without additional input. In

this stage, objects mean the *classes* of objects.

As it is not based on visual inputs, the predicted action-object sequence has to be tentative. The downstream components in the model (i.e., the mask decoder and the action decoder) interpret s_{m_t} again, yielding the final prediction of an action-object sequence, which are grounded on the visual inputs. Our intention of this two-stage approach is to increase prediction accuracy; we expect that using a prior prediction of (action, object class) pairs helps more accurate grounding.

In fact, many instructions in the dataset, particularly those about interactions with objects, are sufficiently specific so that they are uniquely translated into (action, object class) sequences with a perfect accuracy, even without visual inputs. For instance, “Wash the mug in the sink” can be translated into (**Put**, **Sink**), (**TurnOn**, **Faucet**), (**TurnOff**, **Faucet**), (**PickUp**, **Mug**). However, this is not the case with navigation instructions. For instance, “Go straight to the sink” may be translated into a variable number of repetition of **MoveAhead**; it is also hard to translate “Walk into the drawers” when it requires to navigate to the left/right. Therefore, we separately deal with the manipulation actions and the navigation actions. In what follows, we first explain the common part and then the different parts.

Given the encoded feature s_{m_t} of the selected instruction, the instruction decoder predicts the action and the object class to choose at t . To be precise, it outputs the probability distributions $p_t^{\text{ia}} (\in \mathbb{R}^{N_a})$ and $p_t^{\text{io}} (\in \mathbb{R}^{N_o})$ over all the actions and the object classes, respectively; N_a and N_o are the numbers of the actions and the object classes.

These probabilities p_t^{ia} and p_t^{io} are predicted separately by two LSTMs in an autoregressive fashion. The two LSTMs are initialized whenever a new instruction is selected; to be precise, we reset their internal states as $h_{t-1}^{\text{ia}} = h_{t-1}^{\text{io}} = s_{m_t}$ for t when we increment m_t as $m_t = m_{t-1} + 1$ (see the example in Fig. 5.2). Then, p_t^{ia} and p_t^{io} are predicted as follows:

$$p_t^{\text{ia}} = \text{softmax}(W_{\text{ia}} \text{LSTM}(E_a(p_{t-1}^{\text{ia}}), h_{t-1}^{\text{ia}}) + b_{\text{ia}}), \quad (5.2a)$$

$$p_t^{\text{io}} = \text{softmax}(W_{\text{io}} \text{LSTM}(E_o(p_{t-1}^{\text{io}}), h_{t-1}^{\text{io}}) + b_{\text{io}}), \quad (5.2b)$$

where $W_{\text{ia}} \in \mathbb{R}^{N_a \times d}$, $b_{\text{ia}} \in \mathbb{R}^{N_a}$, $W_{\text{io}} \in \mathbb{R}^{N_o \times d}$, and $b_{\text{io}} \in \mathbb{R}^{N_o}$ are learnable parameters; E_a maps the most likely action into the respective vectors according to the last pre-

dictionaries p_{t-1}^{ia} using a dictionary with $N_a \times d$ learnable parameters; E_o does the same for the object classes. The predicted p_t^{ia} and p_t^{io} are transferred to the input of these LSTMs at the next timestep and also inputted to the downstream components, the mask decoder and the action decoder.

Now, as they do not need visual inputs, we can train the two LSTMs in a supervised fashion using the pairs of instructions and the corresponding ground truth action-object sequences. We denote this supervised loss, i.e., the sum of the losses for the two LSTMs, by \mathcal{L}_{aux} . Although it is independent of the environment and we can train the LSTMs offline, we simultaneously train them along with other components in the model by adding \mathcal{L}_{aux} to the overall loss. We think this contributes to better learning of instruction representation s_{m_t} , which is also used by the mask decoder and the action decoder.

As mentioned above, we treat the navigation actions differently from the manipulation actions. There are three differences. First, we simplify the ground truth action sequence for the navigation actions if necessary. For instance, suppose an instruction “Turn left, go ahead to the counter and turn right” with a ground truth action sequence “RotateLeft, MoveAhead, MoveAhead, MoveAhead, MoveAhead, RotateRight”. The repetition of MoveAhead reflects the environment and cannot be predicted without visual inputs. Thus, by eliminating the repeated actions, we convert the sequence into the minimum-length one, “RotateLeft, MoveAhead, RotateRight”, and regard it as the ground truth sequence, training the instruction decoder. Second, as there is no accompanied object for the navigation actions, we use the object-class sequence “None, None, None” as the ground truth. Third, in the case of navigation actions, we do not transfer the outputs p_t^{ia} and p_t^{io} to the mask decoder and the action decoder and instead feed constant (but learnable) vectors $p_{\text{nav}}^{\text{ia}} \in \mathbb{R}^{N_a}$ and $p_{\text{nav}}^{\text{io}} \in \mathbb{R}^{N_o}$ to them. As the instruction decoder learns to predict the minimum-length action sequences as above, providing such predictions will be harmful for the action decoder. We avoid this by feeding $p_{\text{nav}}^{\text{ia}}$ and $p_{\text{nav}}^{\text{io}}$.

5.3.4 Action Decoder

The action decoder receives four inputs and predicts the action at t . The inputs are as follows: the encoded instruction s_{m_t} , the output p_t^{ia} and p_t^{io} of the instruction

decoder¹ and aggregated feature v_t of visual inputs, which will be described below.

Hierarchical Attention over Visual Features

As explained in Sec. 5.3.2, we use the multi-view object-centric representation of visual inputs. To be specific, we aggregate $N \times K$ outputs of Mask R-CNN from K ego-centric images, obtaining a single vector v_t . The Mask R-CNN outputs for view $k(= 1, \dots, K)$ are the visual features $(v_{t,1}^k, \dots, v_{t,N}^k)$ and the confidence scores $(\rho_{t,1}^k, \dots, \rho_{t,N}^k)$ of N detected objects.

To do this feature aggregation, we employ a hierarchical approach, where we first search for the objects relevant to the current instruction in each view and then merge the features over the views to a single feature vector. In the first step, we compute and apply soft-attentions over N objects for each view. To be specific, we compute attention weights $\alpha_s^k \in \mathbb{R}^N$ across $v_{t,1}^k, \dots, v_{t,N}^k$ guided by s_{m_t} as

$$\alpha_{s,n}^k = \text{softmax}((v_{t,n}^k)^\top W_s^k s_{m_t}), \quad (5.3)$$

where $W_s^k \in \mathbb{R}^{d \times d}$ is a learnable matrix, for $k = 1, \dots, K$. We then apply the weights to the N visual features multiplied with their confidence scores for this view, yielding a single d -dimensional vector as

$$v_t^k = \sum_{n=1}^N \alpha_{s,n}^k v_{t,n}^k \rho_{t,n}^k, \quad (5.4)$$

where $\rho_{t,n}^k$ is the confidence score associated with $v_{t,n}^k$.

In the second step, we merge the above features v_t^1, \dots, v_t^K using *gated-attention*. We compute the weight $\alpha_g^k (\in \mathbb{R})$ of view $k(= 1, \dots, K)$ guided by s_{m_t} as

$$\alpha_g^k = \text{sigmoid}((v_t^k)^\top W_g s_{m_t}), \quad (5.5)$$

where $W_g \in \mathbb{R}^{d \times d}$ is a learnable matrix. Finally, we apply the weights to $\{v_t^k\}_{k=1, \dots, K}$

¹These are replaced with $p_{\text{nav}}^{\text{ia}}$ and $p_{\text{nav}}^{\text{ia}}$ if $\text{argmax}(p_t^{\text{ia}})$ is not a manipulation action, as mention above.

to have the visual feature $v_t \in \mathbb{R}^d$ as

$$v_t = \sum_{k=1}^K \alpha_g^k v_t^k. \quad (5.6)$$

As shown in the ablation test in the supplementary, the performance drops significantly when replacing the above gated-attention by soft-attention, indicating the necessity for merging observations of different views, not selecting one of them.

Decoder Design

The decoder predicts the action at t from v_t , s_{m_t} , p_t^{ia} and p_t^{io} . We employ an LSTM, which outputs the hidden state $h_t^a \in \mathbb{R}^d$ at t from the previous state h_{t-1}^a along with the above four inputs as

$$h_t^a = \text{LSTM}([v_t; s_{m_t}; p_t^{\text{ia}}; p_t^{\text{io}}], h_{t-1}^a), \quad (5.7)$$

where $[\cdot]$ denotes concatenation operation. We initialize the LSTM by setting the initial hidden state h_0^a to h_G , the encoded feature of the goal statement; see Sec. 5.3.2. The updated state h_t^a is fed into a fully-connected layer to yield the probabilities over the $N_a + 1$ actions including COMPLETE as follows:

$$p_t^a = \text{softmax}(W_a h_t^a + b_a), \quad (5.8)$$

where $W_a \in \mathbb{R}^{(N_a+1) \times d}$ and $b_a \in \mathbb{R}^{N_a+1}$. We choose the action with the maximum probability for the predicted action. In the training of the model, we use cross entropy loss $\mathcal{L}_{\text{action}}$ computed between p_t^a and the one-hot representation of the true action.

5.3.5 Mask Decoder

To predict the mask specifying an object to interact with, we utilize the object-centric representations $V_t^c = (v_{t,1}^c, \dots, v_{t,N}^c)$ of the visual inputs of the central view ($k = c$). Namely, we have only to select one of the N detected objects. This enables more accurate specification of an object mask than predicting a class-agnostic binary mask as in the prior work [1].

To do this, we first apply simple self-attention to the visual features V_t^c , aiming at capturing the relation between objects in the central view. We employ the attention mechanism inside the light-weight Transformer with a single head proposed in [160] for this purpose, obtaining $\bar{\mathcal{A}}_{V_t^c}(V_t^c) \in \mathbb{R}^{N \times d}$. We then apply linear transformation to $\bar{\mathcal{A}}_{V_t^c}(V_t^c)$ using a single fully-connected layer having weight $W \in \mathbb{R}^{N \times d}$ and bias $b \in \mathbb{R}^d$, with a residual connection as

$$\hat{V}_t^c = \text{ReLU}(W\bar{\mathcal{A}}_{V_t^c}(V_t^c) + \mathbf{1}_K \cdot b^\top) + V_t^c, \quad (5.9)$$

where $\mathbf{1}_K$ is K -vector with all ones.

We then compute the probability $p_{t,n}^m$ of selecting n -th object from the N candidates using the above self-attended object features along with other inputs s_{m_t} , p_t^{ia} , and p_t^{io} . We concatenate the latter three inputs into a vector $g_t^m = [s_{m_t}; p_t^{\text{ia}}; p_t^{\text{io}}]$ and then compute the probability as

$$p_{t,n}^m = \text{sigmoid}((g_t^m)^\top W_m \hat{v}_{t,n}^c), \quad (5.10)$$

where $W_m \in \mathbb{R}^{d+N_a+N_o \times d}$ is a learnable matrix. We select the object mask with the highest probability (i.e., $\text{argmax}_{n=1,\dots,N}(p_{t,n}^m)$) at inference time. At training time, we first match the ground truth object mask with the object mask having the highest IoU. Then, we calculate the BCE loss $\mathcal{L}_{\text{mask}}$ between the two.

5.4 Experiments

5.4.1 Experimental Configuration

Dataset. We follow the standard procedure of ALFRED; 25,743 language directives over 8,055 expert demonstration episodes are split into the training, validation, and test sets. The latter two are further divided into two splits, called *seen* and *unseen*, depending on whether the scenes are included in the training set.

Evaluation metrics. Following [1], we report the standard metrics, i.e., the scores of Task Success Rate, denoted by **Task** and Goal Condition Success Rate, denoted by

Table 5.1: Task and Goal-Condition Success Rate. For each metric, the corresponding path weighted metrics are given in (parentheses). The highest values per fold and metric are shown in **bold**.

Model	Validation				Test			
	Seen		Unseen		Seen		Unseen	
	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond
Shridhar et al. [1]	3.70 (2.10)	10.00 (7.00)	0.00 (0.00)	6.90 (5.10)	3.98 (2.02)	9.42 (6.27)	0.39 (0.80)	7.03 (4.26)
Legg et al. [133]	-	-	-	-	3.85 (1.50)	8.87 (5.52)	0.85 (0.36)	7.68 (4.31)
Singh et al. [134]	4.50 (2.20)	12.20 (8.10)	0.70 (0.30)	9.50 (6.10)	5.41 (2.51)	12.32 (8.27)	1.50 (0.7)	8.08 (5.20)
MOCA [155]	19.15 (13.60)	28.50 (22.30)	3.78 (2.00)	13.40 (8.30)	22.05 (15.10)	28.29 (22.05)	5.30 (2.72)	14.28 (9.99)
Ours (single view)	18.90 (13.90)	26.80 (21.90)	3.90 (2.50)	15.30 (10.90)	15.20 (11.79)	23.95 (20.27)	4.45 (2.37)	14.71 (10.88)
Ours (multiple views)	33.70 (28.40)	43.10 (38.00)	9.70 (7.30)	23.10 (18.10)	29.16 (24.67)	38.82 (34.85)	8.37 (5.06)	19.13 (14.81)
Ours (winning entry) ^o	14.30 (10.80)	22.40 (19.60)	4.60 (2.80)	11.40 (8.70)	12.39 (8.20)	20.68 (18.79)	4.45 (2.24)	12.34 (9.44)
Human	-	-	-	-	-	-	91.00 (85.80)	94.50 (87.60)

Goal-Cond. The **Goal-Cond** score is the ratio of goal conditions being completed at the end of an episode. The **Task** score is defined to be one if all the goal conditions are completed, and otherwise 0. Besides, each metric is accompanied by a path-length-weighted (PLW) score [161], which measures the agent’s efficiency by penalizing scores with the length of the action sequence.

Implementation details. We use $K = 5$ views: the center view, *up* and *down* views with the elevation degrees of $\pm 15^\circ$, and *left* and *right* views with the angles of $\pm 90^\circ$. We employ a Mask R-CNN model with ResNet-50 backbone that receives a 300×300 image and outputs $N = 32$ object candidates. We train it before training the proposed model with 800K frames and corresponding instance segmentation masks collected by replaying the expert demonstrations of the training set. We set the feature dimensionality $d = 512$. We train the model using imitation learning on the expert demonstrations by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{aux}}. \quad (5.11)$$

We use the Adam optimizer with an initial learning rate of 10^{-3} , which is halved at epoch 5, 8, and 10, and a batch size of 32 for 15 epochs in total. We use a dropout with the dropout probability 0.2 for the both visual features and LSTM decoder hidden states.

Table 5.2: Sub-goal success rate. All values are in percentage. The agent is evaluated on the Validation set. Highest values per fold are indicated in **bold**.

Sub-goal	[1]		[155]		Ours	
	Seen	Unseen	Seen	Unseen	Seen	Unseen
Goto	51	22	54	32	59	39
Pickup	32	21	53	44	84	79
Put	81	46	62	39	82	66
Slice	25	12	51	55	89	85
Cool	88	92	87	38	92	94
Heat	85	89	84	86	99	95
Clean	81	57	79	71	94	68
Toggle	100	32	93	11	99	66
Average	68	46	70	47	87	74

5.4.2 Experimental Results

Table 5.1 shows the results. It is seen that our method shows significant improvement over the previous methods [1, 133, 134, 155] on all metrics. Our method also achieves better PLW (path length weighted) scores in all the metrics (indicated in the parentheses), showing its efficiency. Notably, our method attains **8.96%** success rate on the unseen test split, improving approximately 20 times compared with the published result in [1]. The higher success rate in the unseen scenes indicates its ability to generalize in novel environments. Detailed results for each of the seven task types are shown in the supplementary.

The preliminary version of our method won an international competition, whose performance is lower than the present version. It differs in that (p_t^{ia}, p_t^{io}) are not forwarded to the mask decoder and the action decoder and the number of Mask R-CNN’s outputs is set to $N = 20$. It is noted that even with a single view (i.e., $K = 1$), our model still outperforms [1, 133, 134] in all the metrics.

Sub-goal success rate. Following [1], we evaluate the performance on individual sub-goals. Table 5.2 shows the results. It is seen that our method shows higher

success rates in almost all of the sub-goal categories.

Performance by Task Type Table 5.3 shows the success rates across the 7 task types achieved by the existing methods including ours on the validation set of AL-FRED. It is seen that our method outperforms others by a large margin in both seen and unseen environments.

Task-Type	[1]		[155]		Ours	
	Seen	Unseen	Seen	Unseen	Seen	Unseen
Pick & Place	7.0	0.0	29.5	5.0	40.1	13.0
Stack & Place	0.9	0.0	5.2	1.8	17.4	11.9
Pick Two	0.8	0.0	11.2	1.1	21.8	1.1
Clean & Place	1.8	0.0	22.3	2.4	40.2	15.0
Heat & Place	1.9	0.0	15.8	2.7	41.2	9.6
Cool & Place	4.0	0.0	26.1	0.7	40.0	13.8
Examine	9.6	0.0	20.2	13.2	34.4	12.9
Average	3.7	0.0	18.6	3.8	33.6	11.0

Table 5.3: Success rate across 7 task types. All values are in percentages. The agent is evaluated on the validation set. Highest values per split are indicated in **bold**.

5.4.3 Ablation Study

We conduct an ablation test to validate the effectiveness of the components by incrementally adding each component to the proposed model. The results are shown in Table 5.4.

The model variants 1-4 use a single-view input ($K = 1$); they do not use multi-view inputs and the hierarchical attention method. Model 1 further discards the instruction decoder by replacing it with the soft-attention-based approach [1], which yields a different language feature s_{att} at each timestep. Accordingly, p_t^{io} and p_t^{ia} are not fed to the mask/action decoders; we use $g_t^m = [s_{att}; h_t^a]$. These changes will make the method almost unworkable. Model 2 retains only the instruction selection module, yielding s_{m_t} . It performs much better than Model 1. Model 3 has the instruction

Table 5.4: Ablation study for the components of the proposed model. We report the success rate (Task score) on the validation seen and unseen splits. The **X** mark denotes that a corresponding component is removed from the proposed model.

Model	Components				Validation
	Instruction Selection	Two-stage Interpretation	Multi-view Hier. Attn	Mask Decoder	Seen / Unseen
1	X	X	X	✓	2.8 / 0.5
2	✓	X	X	✓	12.9 / 2.9
3	✓	✓	X	✓	18.9 / 3.9
4	✓	✓	X	X	3.8 / 0.7
5	✓	✓	✓	✓	33.7 / 9.7

decoder, which feeds p_t^{io} and p_t^{ia} to the subsequent decoders. It performs better than Model 2 by a large margin, showing the effectiveness of the two-stage method.

Model 4 replaces the mask decoder with the counterpart of the baseline method [1], which upsamples a concatenated vector $[g_t^m; v_t]$ by deconvolution layers. This change results in inaccurate mask prediction, yielding a considerable performance drop. Model 5 is the full model. The difference from Model 3 is the use of multi-view inputs with the hierarchical attention mechanism. It contributes to a notable performance improvement, validating its effectiveness.

Table 5.5 shows the full results of the ablation test reported in the main paper. We also provide additional results in Table 5.6 with different activation functions (i.e. sigmoid or softmax) in the second step of the proposed hierarchical attention mechanism, and with different K 's; K is selected from 1 (only ‘center’ view), 3 (‘center’, ‘left’, and ‘right’ views), or 5 (‘center’, ‘left’, ‘right’, ‘up’, and ‘down’ views)). The results show that the use of *gated-attention* in Eq.(5) (of the main paper) is essential. We also confirm the number of views also affect the success rate.

Components				Validation-Seen		Validation-Unseen	
Instruction	Two-stage	Multi-view	Mask	Task	Goal-Cond.	Task	Goal-Cond.
Selection	Interpretation	Hier. Attn	Decoder				
x	x	x	✓	2.8 (1.3)	9.7 (6.5)	0.5 (0.2)	9.2 (5.4)
✓	x	x	✓	12.9 (9.4)	21.6 (17.3)	2.9 (1.6)	13.1 (9.4)
✓	✓	x	✓	18.9 (13.9)	26.8 (21.9)	3.9 (2.5)	15.3 (10.9)
✓	✓	✓	x	3.8 (2.4)	14.9 (11.2)	0.7 (0.3)	10.4 (6.9)
✓	✓	✓	✓	33.7 (28.4)	43.1 (38.0)	9.7 (7.3)	23.1 (18.1)

Table 5.5: Results of an ablation test for examining the effectiveness of each component of the proposed model. The path weighted scores are reported in the parentheses.

Configurations		Validation-Seen		Validation-Unseen	
		Task	Goal-Cond.	Task	Goal-Cond.
Activation Function	Softmax	11.9 (9.3)	20.8 (17.3)	4.1 (2.2)	14.0 (10.2)
	Sigmoid	33.7 (28.4)	43.1 (38.0)	9.7 (7.3)	23.1 (18.1)
Ego-centric views	center	18.9 (13.9)	26.8 (21.9)	3.9 (2.5)	15.3 (10.9)
	center, left, right	25.9 (21.2)	34.4 (30.0)	6.2 (3.8)	17.0 (12.3)
	center, left, right, up, down	33.7 (28.4)	43.1 (38.0)	9.7 (7.3)	23.1 (18.1)

Table 5.6: Results of experiments comparing activation functions in the module for aggregating and encoding multi-view visual inputs. The path weighted scores are reported in the parentheses.

5.4.4 Qualitative Results

Entire Task Completion

Figures 5.3-5.6 show the visualization of how the agent completes one of the seven types of tasks. These are the results for the unseen environment of the validation set. Each panel shows the agent’s center view with the predicted action and object mask (if existing) at different time-steps.

Mask Prediction for Sub-goal Completion

Figure 5.8 shows an example of the mask prediction by the baseline [1] and the proposed method. It shows our method can predict a more accurate object mask



Figure 5.3: Our agent completes an **Examine** task “*Examine an empty box by the light of a floor lamp*” in an unseen environment.

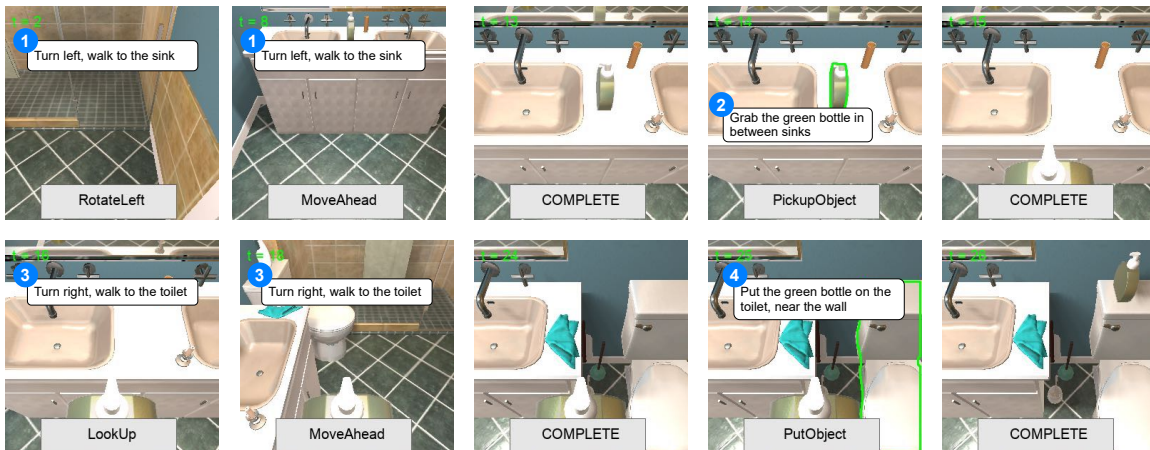


Figure 5.4: Our agent completes a **Pick & Place** task “*Place the green bottle on the toilet basin*” in an unseen environment.

when performing **Slice** sub-goal. More examples are shown in the supplementary material. Overall, our method shows better results, especially for difficult sub-goals like **Pickup**, **Put**, and **Clean**, for which a target object needs to be chosen from a wide range of candidates.

We also provide seven video clips as independent files, which contain several examples of the agent’s entire task completion for seven above task instances in unseen environments.



Figure 5.5: Our agent completes a **Pick Two & Place** task “*To move two bars of soap to the cabinet*” in an unseen environment.



Figure 5.6: Our agent completes a **Cool & Place** task “*Put chilled lettuce on the counter*” in an unseen environment.

5.5 Analyses of Failure Cases

We analyze the failure cases of our method using the results on the validation splits. We categorize them into navigation failures and manipulation failures.

5.5.1 Navigation Failures

It is seen from the sub-goal results of Table 2 in the main paper that the **Goto** sub-goal is the most challenging. Failures with it tend to make it hard to complete the entire goal, since they will inevitably affect the subsequent actions to take. We think there are three major cases for the navigation failures.

The first case, which occurs most frequently, is that the agent follows a navigation

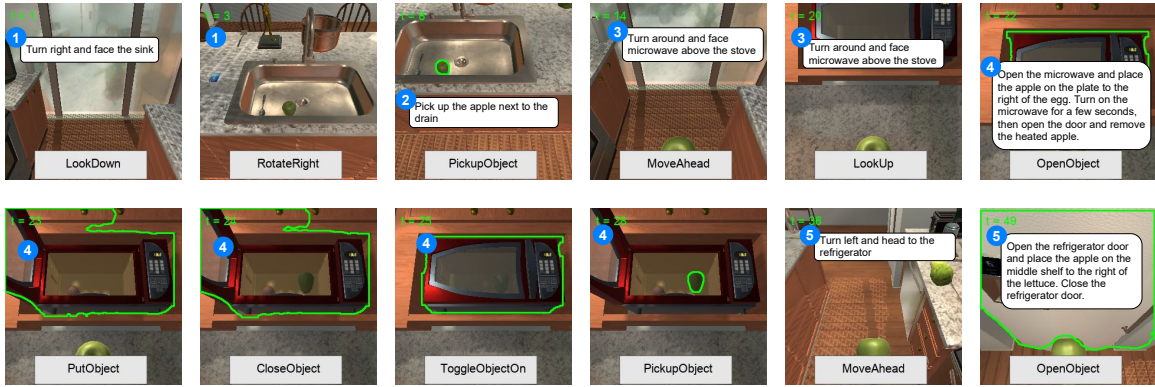
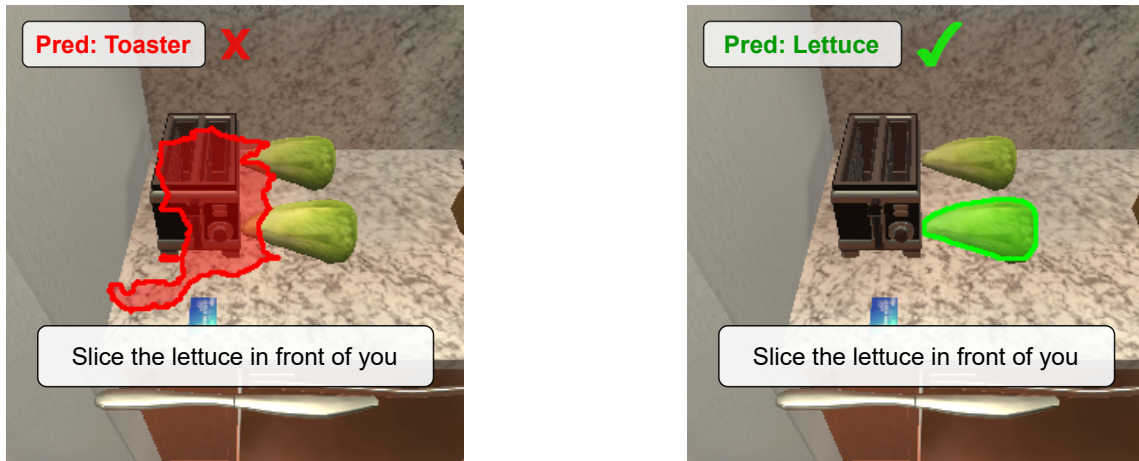


Figure 5.7: Our agent completes a **Heat & Place** task “Put a heated apple next to the lettuce on the middle shelf in the refrigerator” in an unseen environment.



(a) Shridhar et al. [1]

(b) Ours

Figure 5.8: The prediction masks generated by Shridhar *et al.* and our method where the agents are moved to the same location to accomplish **Slice** sub-goal.

instruction and reaches a position that should be fine as far as the instruction goes; nevertheless, it is not the right position for the next manipulation action to take. For instance, following the instruction “Go to the table,” the agent goes to the table. The next instruction is “Pickup the remote control at the table,” but the remote lies on the other side of the table. This is counted as a failure of completing the Goto sub-goal.

The second case is when the instructions are either abstract or misleading. An example is that when the agent has to take several left and right turns together with multiple **MoveAhead** steps to reach the destination, e.g., a drawer, the provided

instruction is simply “Go to the drawer.”

The third case, which occurs less frequently, is that while there is an obstacle in front of the agent, e.g., wall, it attempts to take the **MoveAhead** action. This occurs because of the lack of proper visual inputs. This is demonstrated by the fact that when we reduce the number of views, the task success rate drops significantly, as shown in the second block of Table 5.6.

5.5.2 Manipulation Failures

As shown in Table 2, after it has moved to the ideal position right before performing any interaction sub-goals (i.e., all the sub-goals but **Goto**), the agent can manipulate objects with high success rates of 91% and 69% in the seen and unseen environments, respectively. However, the success rates for completing the **Goto** sub-goal in the seen and unseen environments are only 59% and 39%, respectively. Therefore, the primary cause of the manipulation failures is that the agent cannot find the target object because it fails to reach the right destination due to a navigation failure.

Even if the agent has successfully navigated to the right destination, it can fail to detect the target object. This seems to happen mostly because the object is either too small or indistinguishable from the surroundings. The agent tends to fail to detect, for example, a small knife placed on the steel/metal-made sink of the same color.

The agent also fails to detect an object that has not been seen in the training. This is confirmed by the fact that the performance drops considerably in unseen environments for some interaction sub-goals (including **Put**, **Clean**, and **Toggle**). There are also a small number of cases where failures are attributable to bad instructions, e.g., incorrect statement of objects.

5.6 Summary and Conclusion

This chapter has presented a new method for interactive instruction following tasks and applied it to ALFRED. The method is built upon several new ideas, including the explicit selection of one of the provided instructions, the two-stage approach to the interpretation of each instruction (i.e., the instruction decoder), the employ-

ment of the object-centric representation of visual inputs obtained by hierarchical attention from multiple surrounding views (i.e., the action decoder), and the precise specification of objects to interact with based on the object-centric representation (i.e., the mask decoder). The experimental results have shown that the proposed method achieves superior performances in both seen and unseen environments compared with all the existing methods. We believe this study provides a useful baseline framework for future studies.

Chapter 6

Conclusion

In this chapter, we summarize the contributions of our work and discuss some remaining problems as well as the future directions.

Over the past decades, we have experienced unprecedented progress in Artificial Intelligence, especially in Computer Vision and Natural Language Processing. Recently, there has been an increasing interest in solving problems that integrate visual and linguistic learning to unlock many practical applications potentially. In this dissertation, we have developed the models to understand visual and linguistic information and interact with humans and physical environments.

Concretely, in Chapter 3, we proposed a better and faster image captioning model. It is arguably believed that extracting good visual representations from the input image plays a crucial role in generating captions. We then pointed out that region-based features extracted by an object detector, such as Faster R-CNN, employed by previous methods, have some limitations: 1) lack of contextual information, 2) the risk of false detection, and 3) the high computational cost. We observed that grid-based features could alleviate the first two issues, while DETR-based detectors could help overcome the third issue. Thus, we introduced GRIT, which effectively utilizes the two visual features to generate better captions. Under the same training condition, we found that our proposed method outperforms previous methods significantly in terms of accuracy and speed.

We believe that integrating the two visual features would also benefit other tasks in which the contextual information provides further clues for more accurate predictions.

The tasks include but are not limited to several vision-language tasks (e.g., image-text retrieval, VQA, multi-modal verification, visual dialog, etc.), referring expression, scene graph generation, etc. Although this method may produce promising results for image captioning and maybe other tasks without vision-language pre-training, it is unclear how to utilize these two features in pre-training large models on large-scale datasets. First, using these two elements simultaneously complicates the model design. Second, the datasets with object detection annotations appearing marginal mitigate the advantage of region features compared to pre-training simple grid-based models on large-scale datasets of image-text pairs. We hope future research will leverage the two visual features for pre-training on large-scale datasets and fine-tuning on downstream tasks.

In Chapter 4, we studied the visual dialog task, which requires agents to answer multiple questions in the form of dialog with humans. Previous methods considered attention from one input to another based on different hypotheses, such as “question \rightarrow history \rightarrow image” path in [87, 88], and “question \rightarrow image \rightarrow history \rightarrow question” path in [89, 90], etc. These methods cannot take all the interactions between inputs into account. We argued that a plausible approach to solving the problem is effectively modeling all the interactions between inputs. Therefore, we proposed LTMI, a neural architecture that can efficiently deal with all the interactions between multiple such inputs in visual dialog. It treats all the utilities equally and simultaneously computes all their interactions. It has a design structure similar to Transformers, yet is much light-weight with less than one-tenth of the number of parameters. We found that the models built upon the proposed LTMI block achieve considerable performance improvement in Visual Dialog and Audio Visual Scene-aware Dialog with more inputs.

It is worth mentioning that recent state-of-the-art methods [123, 124] for Visual Dialog employ pre-trained vision-language transformers. Specifically, these methods concatenate the dialog history and the question into a unified text before forwarding them into the pre-trained transformers and fine-tuning for the task. We expect that placing additional LTMI layers on the top of transformer outputs may disentangle the inputs and capture their interactions for better prediction. Since the LTMI layer is light-weight, its introduction adds only minimal computation. We leave it for future research.

In Chapter 5, we tackled the ALFRED, an interactive instruction-following benchmark that requires performing household tasks by following human instructions. We pointed out that the previous methods that perform well on related tasks, Vision Language Navigation (VLN), fail to achieve good performance on ALFRED due to a number of challenging problems in ALFRED compared with VLN. We proposed LWIT to overcome the shortcomings of previous methods, which outperforms them by a large margin. We found several ideas that contribute to better performance: 1) selecting explicitly one instruction from a sequence of given instructions to accomplish at a time; 2) interpreting each instruction in two steps which refine the initial predictions; 3) employing region-based features obtained from multiple surrounding views and specifying objects to interact with rather than generating pixel-wise predictions.

Despite significantly improving the ALFRED tasks, our method performs much worse than humans. However, the method serves as a strong baseline for ALFRED for further development. We strongly believe that several ideas, such as selecting masks instead of pixel-wise predictions and two-stage interpretation, are still applicable to future research to improve the accuracy. It is also noted that in this work, we assume that agents perform low-level actions (i.e., navigation and interaction actions with objects and environments) with perfect accuracy, which is beyond our research scope. We suspect that building a complete intelligent robot that works in real environments is much more complicated than simulation. Research on simulation-to-real embodied agents is still active and attracting attention from the community [162–164].

Bibliography

- [1] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [2] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.
- [3] 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.
- [4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.
- [5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*, 2014.
- [6] 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*, pages 770–778, 2016.
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg

- Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *arXiv:2010.11929*, 2020.
- [8] 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*, pages 91–99, 2015.
- [9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988. IEEE Computer Society, 2017.
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2009.
- [11] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. In *CoRR*, volume abs/1902.06162, 2019.
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*, 2018.
- [13] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *Technical report*. OpenAI, 2018.
- [14] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *arXiv preprint arXiv:2005.14165*, 2020.
- [15] Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

- [16] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.
- [17] 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.
- [18] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2017.
- [19] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. 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*, 2018.
- [20] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, 2018.
- [21] F. Zhu, Y. Zhu, X. Chang, and X. Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [22] Aditya Mogadala. Polylingual multimodal learning. In *ECML PKDD Doctoral Consortium*, page 155. Citeseer, 2015.
- [23] C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *Proceedings of International Conference on Learning Representations*, 2019.
- [24] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

- [25] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. In *Ussr computational mathematics and mathematical physics*, volume 4, pages 1–17. Elsevier, 1964.
- [26] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.
- [27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.
- [28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of machine learning research*, volume 12, 2011.
- [29] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.
- [30] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [32] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. In *Cognitive modeling*, volume 5, page 1, 1988.
- [33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural computation*, volume 9, pages 1735–1780, 1997.
- [34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10012–10022, 2021.

- [35] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of International Conference on Machine Learning*, pages 2048–2057, 2015.
- [36] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024, 2017.
- [37] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 375–383, 2017.
- [38] 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.
- [39] Yunpeng Luo, Jiayi Ji, Xiaoshuai Sun, Liujuan Cao, Yongjian Wu, Feiyue Huang, Chia-Wen Lin, and Rongrong Ji. Dual-level collaborative transformer for image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2286–2293, 2021.
- [40] Tiantao Xian, Zhixin Li, Canlong Zhang, and Huifang Ma. Dual global enhanced transformer for image captioning. In *Neural Networks*, volume 148, pages 129–141, 2022.
- [41] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, pages 213–229, 2020.
- [42] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai.

- Deformable detr: Deformable transformers for end-to-end object detection. In *Proceedings of International Conference of Learning Representations*, 2021.
- [43] 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*, pages 740–755. Springer, 2014.
- [44] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. Simvlm: Simple visual language model pretraining with weak supervision. In *arXiv:2108.10904*, 2021.
- [45] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.
- [46] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10267–10276, 2020.
- [47] Xuying Zhang, Xiaoshuai Sun, Yunpeng Luo, Jiayi Ji, Yiyi Zhou, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Rstnet: Captioning with adaptive attention on visual and non-visual words. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 15465–15474, 2021.
- [48] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. In *Advances in Neural Information Processing Systems*, 2021.
- [49] Hwanjun Song, Deqing Sun, Sanghyuk Chun, Varun Jampani, Dongyoon Han, Byeongho Heo, Wonjae Kim, and Ming-Hsuan Yang. Vidt: An efficient and effective fully transformer-based object detector. In *arXiv:2110.03921*, 2021.

- [50] Haiyang Xu, Ming Yan, Chenliang Li, Bin Bi, Songfang Huang, Wenming Xiao, and Fei Huang. E2e-vlp: End-to-end vision-language pre-training enhanced by visual learning. In *arXiv:2106.01804*, 2021.
- [51] Xu Yang, Hanwang Zhang, and Jianfei Cai. Learning to collocate neural modules for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4250–4260, 2019.
- [52] Guang Li, Linchao Zhu, Ping Liu, and Yi Yang. Entangled transformer for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8928–8937, 2019.
- [53] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4634–4643, 2019.
- [54] Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. X-linear attention networks for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10971–10980, 2020.
- [55] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10578–10587, 2020.
- [56] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. In *Advances in Neural Information Processing Systems*, 2019.
- [57] Longteng Guo, Jing Liu, Xinxin Zhu, Peng Yao, Shichen Lu, and Hanqing Lu. Normalized and geometry-aware self-attention network for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10327–10336, 2020.
- [58] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations

- in vision-language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5579–5588, 2021.
- [59] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. In *International Journal of Computer Vision*, volume 123, pages 32–73, 2017.
- [60] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. In *International Journal of Computer Vision*, volume 128, pages 1956–1981, 2020.
- [61] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8430–8439, 2019.
- [62] Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. nocaps: novel object captioning at scale. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8948–8957, 2019.
- [63] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2018.
- [64] Karpathy. Karpathy/neuraltalk: Neuraltalk is a python+numpy project for learning multimodal recurrent neural networks that describe images with sentences.
- [65] Panos Achlioptas, Maks Ovsjanikov, Kilichbek Haydarov, Mohamed Elhoseiny, and Leonidas J Guibas. Artemis: Affective language for visual art. In *Pro-*

- ceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11569–11579, 2021.
- [66] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
- [67] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.
- [68] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.
- [69] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.
- [70] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 382–398, 2016.
- [71] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.
- [72] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Representation Learning*, 2015.
- [73] Luwei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13041–13049, 2020.

- [74] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Proceedings of the European Conference on Computer Vision*, pages 121–137, 2020.
- [75] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4894–4902, 2017.
- [76] Lei Ke, Wenjie Pei, Ruiyu Li, Xiaoyong Shen, and Yu-Wing Tai. Reflective decoding network for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8888–8897, 2019.
- [77] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European Conference on Computer Vision*, pages 684–699, 2018.
- [78] Yu Qin, Jiajun Du, Yonghua Zhang, and Hongtao Lu. Look back and predict forward in image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8367–8375, 2019.
- [79] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.
- [80] Jiayi Ji, Yunpeng Luo, Xiaoshuai Sun, Fuhai Chen, Gen Luo, Yongjian Wu, Yue Gao, and Rongrong Ji. Improving image captioning by leveraging intra- and inter-layer global representation in transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1655–1663, 2021.
- [81] Zhihao Fan, Zhongyu Wei, Siyuan Wang, Ruize Wang, Zejun Li, Haijun Shan, and Xuanjing Huang. Tcic: Theme concepts learning cross language and vision for image captioning. In *arXiv:2106.10936*, 2021.
- [82] Weixuan Wang, Zhihong Chen, and Haifeng Hu. Hierarchical attention network for image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8957–8964, 2019.

- [83] Alexander Mathews, Lexing Xie, and Xuming He. Senticap: Generating image descriptions with sentiments. In *Proceedings of the AAAI conference on artificial intelligence*, pages 3574–3580, 2016.
- [84] 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.
- [85] Wonjae Kim, Son Bokyung, Kim Ildoo, and Wonjae Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *Proceedings of International Conference on Machine Learning*, 2021.
- [86] Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G Schwing. Factor graph attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2039–2048, 2019.
- [87] Gi-Cheon Kang, Jaeseo Lim, and Byoung-Tak Zhang. Dual attention networks for visual reference resolution in visual dialog. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2024–2033, 2019.
- [88] Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324, 2017.
- [89] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. Multi-step reasoning via recurrent dual attention for visual dialog. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 6463–6474, 2019.
- [90] Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. Are you talking to me? reasoned visual dialog generation through adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6106–6115, 2018.

- [91] 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. In *arXiv preprint arXiv:1909.11740*, 2019.
- [92] 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.
- [93] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. In *arXiv preprint arXiv:1908.03557*, 2019.
- [94] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *arXiv preprint arXiv:1908.02265*, 2019.
- [95] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6281–6290, 2019.
- [96] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. In *arXiv preprint arXiv:1511.05960*, 2015.
- [97] Ilija Ilievski, Shuicheng Yan, and Jiashi Feng. A focused dynamic attention model for visual question answering. In *arXiv preprint arXiv:1604.01485*, 2016.
- [98] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574, 2018.
- [99] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems*, pages 289–297, 2016.

- [100] Duy-Kien Nguyen and Takayuki Okatani. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6087–6096, 2018.
- [101] 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*, pages 21–29, 2016.
- [102] 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.
- [103] Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. In *IEEE Transactions on Neural Networks and Learning Systems*, volume 29, pages 5947–5959. IEEE, 2018.
- [104] Chaorui Deng, Qi Wu, Qingyao Wu, Fuyuan Hu, Fan Lyu, and Mingkui Tan. Visual grounding via accumulated attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7746–7755, 2018.
- [105] 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.
- [106] Bohan Zhuang, Qi Wu, Chunhua Shen, Ian Reid, and Anton van den Hengel. Parallel attention: A unified framework for visual object discovery through dialogs and queries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4252–4261, 2018.
- [107] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

- [108] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. In *arXiv preprint arXiv:1504.00325*, 2015.
- [109] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2556–2565, 2018.
- [110] Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512, 2017.
- [111] Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. Clevr-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. In *arXiv preprint arXiv:1903.03166*, 2019.
- [112] 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.
- [113] Dalu Guo, Chang Xu, and Dacheng Tao. Image-question-answer synergistic network for visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10434–10443, 2019.
- [114] Hyounghun Kim, Hao Tan, and Mohit Bansal. Modality-balanced models for visual dialogue. In *arXiv preprint arXiv:2001.06354*, 2020.
- [115] Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. Recursive visual attention in visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6679–6688, 2019.
- [116] Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. Visual coreference resolution in visual dialog using neural mod-

- ule networks. In *Proceedings of the European Conference on Computer Vision*, pages 153–169, 2018.
- [117] Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. Visual reference resolution using attention memory for visual dialog. In *Advances in Neural Information Processing Systems*, pages 3719–3729, 2017.
- [118] Heming Zhang, Shalini Ghosh, Larry Heck, Stephen Walsh, Junting Zhang, Jie Zhang, and C-C Jay Kuo. Generative visual dialogue system via weighted likelihood estimation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1025–1031, 2019.
- [119] Zilong Zheng, Wenguan Wang, Siyuan Qi, and Song-Chun Zhu. Reasoning visual dialogs with structural and partial observations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6669–6678, 2019.
- [120] Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. Evaluating visual conversational agents via cooperative human-ai games. In *Proceedings of AAAI Conference on Human Computation and Crowdsourcing*, 2017.
- [121] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2951–2960, 2017.
- [122] Tianhao Yang, Zheng-Jun Zha, and Hanwang Zhang. Making history matter: History-advantage sequence training for visual dialog. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2561–2569, 2019.
- [123] Yue Wang, Shafiq Joty, Michael R Lyu, Irwin King, Caiming Xiong, and Steven CH Hoi. Vd-bert: A unified vision and dialog transformer with bert. In *arXiv preprint arXiv:2004.13278*, 2020.

- [124] Vishvak Murahari, Dhruv Batra, Devi Parikh, and Abhishek Das. Large-scale pretraining for visual dialog: A simple state-of-the-art baseline. In *arXiv preprint arXiv:1912.02379*, 2019.
- [125] Jiaxin Qi, Yulei Niu, Jianqiang Huang, and Hanwang Zhang. Two causal principles for improving visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10860–10869, 2020.
- [126] 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. In *International Journal of Computer Vision*, volume 123, pages 32–73. Springer, 2017.
- [127] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. In *arXiv preprint arXiv:1607.06450*, 2016.
- [128] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.
- [129] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In <https://pytorch.org>, 2017.
- [130] Unnat Jain, Svetlana Lazebnik, and Alexander G Schwing. Two can play this game: visual dialog with discriminative question generation and answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2018.
- [131] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *arXiv preprint arXiv:1909.11942*, 2019.
- [132] Chiori Hori, Huda Alamri, Jue Wang, Gordon Wichern, Takaaki Hori, Anoop Cherian, Tim K Marks, Vincent Cartillier, Raphael Gontijo Lopes, Abhishek

- Das, et al. End-to-end audio visual scene-aware dialog using multimodal attention-based video features. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2352–2356, 2019.
- [133] L. Yeung, Y. Bisk, and O. Polozov. Alfred speaks: Automatic instruction generation for egocentric skill learning. In *https://askforalfred.com/EVAL*, 2020.
- [134] K. P. Singh, S. Bhambri, B. Kim, , and J. Choi. Improving mask prediction for long horizon instruction following. In *https://askforalfred.com/EVAL*, 2020.
- [135] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, 2016.
- [136] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. In *arXiv:1712.05474*, 2017.
- [137] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *Proceedings of International Conference on Learning Representations*, 2018.
- [138] H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [139] K. M. Hermann, M. Malinowski, P. Mirowski, A. Banki-Horvath, K. Anderson, and R. Hadsell. Learning to follow directions in street view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
- [140] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proceedings of International Conference on 3D Vision*, 2017.

- [141] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [142] H. Tan, L. Yu, and M. Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.
- [143] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [144] K. Nguyen, D. Dey, C. Brockett, and B. Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [145] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. Vision-and-dialog navigation. In *Proceedings of Conference on Robot Learning*, 2020.
- [146] Alane Suhr, Claudia Yan, Jacob Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions. *arXiv preprint arXiv:1910.03655*, 2019.
- [147] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [148] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [149] D. Gordon, D. Fox, and A. Farhadi. What should i do now? marrying reinforcement learning and symbolic planning. In *arXiv:1901.01492*, 2019.

- [150] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [151] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [152] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [153] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [154] R. Corona, D. Fried, C. Devin, D. Klein, and T. Darrell. Modularity improves out-of-domain instruction following. In *arXiv:2010.12764*, 2020.
- [155] K. P. Singh, S. Bhambri, B. Kim, R. Mottaghi, and J. Choi. Moca: A modular object-centric approach for interactive instruction following. In *arXiv:2012.03208*, 2020.
- [156] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *Proceedings of International Conference on Learning Representations*, 2021.
- [157] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. In *CoRR*, volume abs/1806.11532, 2018.
- [158] C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep object-centric representations for generalizable robot learning. In *IEEE International Conference on Robotics and Automation*, 2018.

- [159] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *The journal of machine learning research*, volume 15, pages 1929–1958, 2014.
- [160] V. Q. Nguyen, M. Suganuma, and T. Okatani. Efficient attention mechanism for visual dialog that can handle all the interactions between multiple inputs. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [161] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On evaluation of embodied navigation agents. In *arXiv:1807.06757*, 2018.
- [162] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020.
- [163] Homanga Bharadhwaj, Zihan Wang, Yoshua Bengio, and Liam Paull. A data-efficient framework for training and sim-to-real transfer of navigation policies. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 782–788. IEEE, 2019.
- [164] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021.

Acknowledgments

I owe a debt of gratitude to many people for making my years as a Ph.D. student meaningful and memorable. This journey would not have been so memorable, and I would not have completed this dissertation without them. First and foremost, I must express my gratitude to my supervisor, Professor Takayuki Okatani. We have discussed numerous meetings and emails over the last three years. He has patiently listened to all of my progress reports, regardless of whether they were a mixture of messed-up text, invalid results, or inferior ideas. He has constantly given me comments and asked important questions about my research. Professor Okatani spent days and weeks carefully reading my manuscripts during the submission deadline period, and I can recall it vividly. He then distilled the essence, articulated and contextualized ideas, and transformed all manuscripts into greater ones. I admire his pursuit of excellence and clarity and acknowledge his support for students like myself. It must be said that, as of now, I have achieved many fruitful results thanks to his constant support. I am grateful to him for serving as my advisor.

During my preliminary and final defenses, other jury members on my dissertation committee, Professor Koichi Hashimoto, Professor Kentaro Inui, and Associate Professor Shingo Kagami, provided insightful comments and suggestions.

In addition, I'd like to thank Assistant Professor Masanori Suganuma. He has been an outstanding collaborator and communicator and has assisted me with my research on numerous occasions. He has also offered me a lot of advice on research, career, and everyday life. Mrs. Sakane Akemi, the laboratory secretary, has helped assist other students and me with our paperwork requirements. With her assistance, things have become simpler. I have had the good fortune to meet and become friends with numerous others. All my labmates, including Dang Anh Chuong, Kang Jun Liu, Earth, Kitto, and Luo, have shared joy and happiness with me during our time together. I value all my university and non-university friends, and I cherish the time we spent together in Sendai, where I studied for the past five years.

I would like to thank the Japanese government and Tohoku University for selecting me as a MEXT scholarship recipient during my five years at Tohoku University.

Dinh Thu Hien, my sweetheart, deserves my heartfelt gratitude. She has made this long journey more enjoyable and meaningful by providing companionship and

encouragement. Furthermore, I want to express my enduring gratitude to my parents and siblings for their unconditional love and support throughout the writing of this thesis and my life in general. They have been my inspiration and motivation for every step I have taken.

Finally, I dedicate this work to myself, who has previously studied and worked tirelessly through ups and downs. Any errors or deficiencies in my dissertation that may still exist are entirely my fault and responsibility.