MONASH University

Low Data Visual Question Answering

Narjes Askarian

A thesis submitted for the degree of *Doctor of Philosophy* at

Monash University in 2022

Faculty of Information Technology

This thesis is dedicated to my parents,

for their endless love, support, and encouragement and,

to my beloved husband, Amir

for always being on my side over the last ten years.

Acknowledgements

I would like to thank the following people who played important roles in the different dimensions of my life as a PhD student.

I am deeply thankful to my main PhD supervisor, Assoc. Prof. Reza Haffari. Thank you for accepting me as your student and for your constant advice and support. You have patiently guided me to develop my thoughts and skills as a researcher. I admire your ability to analyse facts and find solutions to seemingly unsolvable challenges. I would also like to thank my co-supervisors Prof. Ingrid Zukerman and Prof. Wray Buntine. Ingrid's feedback helped me well-organize my research notes and improve my writing. Wray joined my supervisory team when I was truly feeling desperate and changed the game. Thank you not only for being an amazing supervisor, but also for caring about your students as though they are your own children. I always mention you as a miracle when talking about my PhD life. I am also immensely grateful to Dr Ehsan Abbasnejad, a knowledgeable and kind researcher who I collaborated with over last two years. Surprisingly, Ehsan always had a precise answer for all of my computer vision questions.

I would like to thank all the members of my panel committee, Dr Mario Boley, Dr Lan Du, and Dr Munawar Hayat for their valuable feedback at my PhD milestones. Furthermore, I would like to thank Professor Danette Deriane, and Helen Cridland from the FIT Postgraduate Research Team, for their continued support from the very first day of my candidature. I

am also very grateful to Julie Holden for her generous support in academic writing and her great passion for teaching us to write professionally and precisely.

I am also grateful to Monash University for providing my PhD scholarship, conference travel funding, and Graduate Research Completion Award. Also, thanks to the support team at Monash Advanced Research Computing Hybrid (MonARCH) and Multi-modal Australian ScienceS Imaging and Visualization Environment (MASSIVE) for all their technical help using computational resources.

I would like to extend my gratitude to my amazing colleagues, Snow Situ, Najam Zaidi, Fahimeh Saleh, Sameen Maruf, Trang Vu, Poorya Zaremoodi, Xuanli He, and Michelle Zhao, for their empathy, trustworthiness, and encouragement during the time of study we had together.

I would also like to express my sincere thanks to my life-long, beloved friend, Hadiseh Farahbakhsh whose friendship has grown stronger regardless of the distance that separates us or the time that passes. I will never forget the times in the middle of my PhD, that you called me every night to ask about my day's progress to encourage me.

My heartfelt thanks go to my inspiring parents, who set me off on the road to this PhD a long time ago. Your love, support and encouragement are worth more than I can express on paper. There have been many times that I could not see a ray of hope for the future but, when I heard your warm voices, the world would turn as brilliant as a gleaming dawn. This accomplishment would not have been possible without your support. Thanks too, to my dear sisters Najmeh, Saemeh, and Sahar for their emotional support, and their endless love. Thanks to my sweet young nieces Hosna and Fatemeh Zahra, whose adorable and amusing videos let me unwind during my stressful days. I also like to thank my parents in-law for always sending their prayers my way.

Finally, I want to express my heartfelt gratitude to my wonderful spouse, Amir. When it comes to speaking about your unwavering support, generosity and care throughout my intensive studies in Australia, I am moved beyond words. Your support did not let me worry

about any aspect of our life during last years; helping me focus on my research. Thank you for bearing with me the whole time I was exhausted and stressed. Aside from that, you helped me a lot in developing this thesis. I will never forget how you worked in my room sitting on the floor away from your fancy office throughout the last months of my PhD, just to keep me motivated to write. Almost all the diagrams, figures and references in this thesis are revised and edited by you. Asking me about my plan in the mornings and my progress at nights helped me organize my busy mind and kept me working as fast as possible. You are not only my beloved husband, but also an incredible friend who has always been there for me and pushed me to achieve my goals.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: .....

Print Name: NARJES ASKARIAN

Date : 19/07/2022

Abstract

Visual question answering is a multimodal task in artificial intelligence where, given an image and a natural language question about the image a machine automatically generates an answer. It is a primary step towards developing agents that can see and communicate their understanding of their surroundings through natural language. Such agents can be helpful in a wide range of high-impact applications; from aiding visually impaired people to rescue missions in dangerous situations. Although it may seem trivial for humans, VQA as a task at the intersection of natural language processing and computer vision, requires image understanding and language comprehension.

This thesis studies VQA in low data scenarios where the training data is limited compared to the currently available large-scale datasets, as collecting a large amount of data is infeasible in many domains. In the absence of sufficient data, current VQA models do not maintain their high performance. We aim to improve VQA performance in low data settings by injecting inductive biases so the model has access to them in a data-efficient manner. We use an inductive bias that a typical learner acquires by training on natural language tasks related to the inherent compositionality of human language, *e.g.,* a complex sentence can be understood by understanding its simpler chunks. The meaning of the resulting chunks are generally easier to capture and provides a powerful foundation for understanding complex sentences.

Inspired by the fact that a complex question can be learned from its basic concepts, we augment the training set of complex questions with simpler questions to help the model. We particularly include simpler questions that, if learned could lead to better representations in the VQA model. We take a data augmentation approach and enlarge the initial small training set by automatically generating simple question-answer pairs for images. After successfully applying data augmentation, we focus on pretraining a VQA model in a self-supervised manner without using any additional labeled data. With this approach, we have the model learn basic visual representations required for answering complex questions using a contrastive learning algorithm. The basic visual representations can be viewed as answers to the sub-questions of a complex question.

In the next stage of our research, we encourage the model to build a knowledge foundation before answering complex questions. We use a curriculum learning approach to supervise the order in which data examples are exposed to the model to begin the learning from easy examples, and gradually consider harder ones, rather than using examples in a random sequence. Our experiments confirm that our approaches have been successful in improving VQA performance when training data is limited.

Our last research stage focuses on adapting the distribution of basic concepts to answer a target complex question. We use examples similar to the target questions in the training set and a meta-learning approach to tune the model parameters accordingly where the model learns how to answer a question by visiting a number of similar examples.

To the best of our knowledge, ours is the first work to investigate improving VQA performance in low data settings and to study low data VQA in general.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI . . . . . Artificial Intelligence

NLP . . . . . Natural Language Processing

VQA . . . . . Visual Question Answering

RNN . . . . . Recurrent Neural Network

CL . . . . . Curriculum Learning

NMN . . . . . Neural Module Network

List of Publications

Some parts of this thesis have been published previously in the form of conference papers. This applies to:

[WACV2022] Chapter 3 was presented previously in:

Narjes Askarian, Ehsan Abbasnejad, Ingrid Zukerman, Wray Buntine, and Reza Haffari. *Inductive Biases for Low Data VQA: A Data Augmentation Approach.* In Proceedings of the 2022 Workshop on Applications of Computer Vision.

[ALTA2021] Chapter 5 was presented previously in:

Narjes Askarian, Ehsan Abbasnejad, Ingrid Zukerman, Wray Buntine, and Reza Haffari. *Curriculum Learning Effectively Improves Low Data VQA*. In Proceedings of the 19th Annual Workshop of the Australasian Language Technology Association.

1 | Introduction

One of the ultimate goals of Artificial Intelligence (AI) (Turing, 1950) is to create systems that have the ability to *see* (*i.e.,* understand the contents of a visual scene: who is where, when doing what?) and *talk* (*i.e.,* convey the understanding to humans in natural language). Humanity can benefit from such systems in different ways, including: 1) educating students, especially children, through conversational demonstrations, 2) assisting analysts in making judgments based on vast amounts of surveillance data, 3) communicating with personal AI assistants such as Alexa and Siri, and 4) using in robotic applications *e.g.,* search and rescue missions.

A primary step toward developing agents that can communicate their understanding of visual content through natural language is visual question answering (VQA) (Antol et al., 2015), where providing a natural language question associated with an image, the machine automatically produces the answer in natural language typically in a few words or a short phrase (see Figure 1.1). A variety of high-impact societal applications requiring human-machine collaboration to extract information from visual data becomes possible with VQA. Such applications range from aiding visually impaired people (*e.g., "is it safe to cross the road?"*) to robotics and digital personal assistants. While seemingly trivial for a human, VQA combines the areas of computer vision and natural language processing (NLP), as it requires image understanding and textual comprehension.

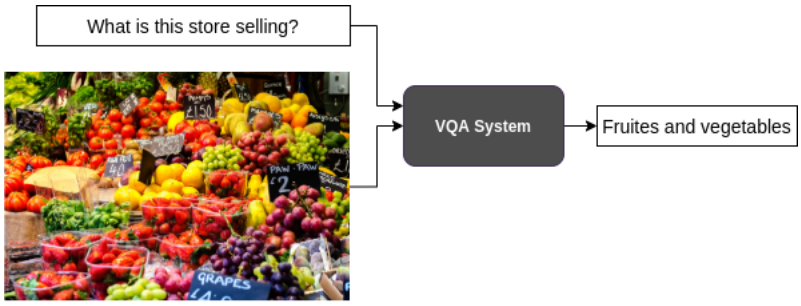


Figure 1.1: An example of a VQA system.

In addition, a VQA system requires the ability of reasoning and grounding a question in an image to infer the answer. This feature makes VQA an important tool for evaluating an algorithm's capacity to extract high-level information from images and to perform reasoning based on information. This ability is referred to as deep visual understanding which is a major goal in the computer vision field. VQA cases contain a wide range of complexity as questions are free-from and open-ended asking for a different set of operations to reason an answer.

1.1 Motivation

Since its introduction, VQA has attracted significant attention and achieved impressive results (Hudson & Manning, 2019b; Teney, Liu, & Van Den Hengel, 2017; Agrawal, Batra, Parikh, & Kembhavi, 2018). This progress is partly because VQA models are commonly trained on large-scale datasets to achieve state-of-the-art performance (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017; Antol et al., 2015; Hudson & Manning, 2019a). Relying on large-scale datasets is not realistic since in many settings collecting labeled data and creating such datasets is either prohibitively expensive or infeasible. In addition, the objective of VQA in a sense is rather ambitious as there are potentially infinite numbers of questions to be asked about an image. As such, we argue that VQA is generally a low-data problem. We demonstrate that in the absence of adequate data, current VQA systems do not maintain their high performance. Motivated by this observation, in this thesis, we study VQA in low-data scenarios. We shed light on the performance of current modular

VQA models under data scarcity conditions and propose approaches to enhance their performance. To the best of our knowledge, this is the first study to investigate VQA models in a low-data regime.

Poor generalization is one of the main reasons for the low performance of VQA models. Generalization becomes a crucial challenge in low data settings. One of the important factors in generalization is capturing appropriate and useful inductive biases (A. Goyal & Bengio, 2021). These are defined as underlying assumptions that a learner uses to predict labels of new data. According to primitives of reasoning theory in logic, induction is the process of inferring a rule from observed cases and their corresponding results (Y. Wu et al., 2021). In machine learning terms, it can be interpreted as the acquisition of inductive biases from the training examples and their labels.

Having access to a large amount of data, a learner can eventually capture inductive biases by being frequently exposed to many examples with similar patterns during training. It provides the learner with the opportunity to develop predictive assumptions by gradually adapting the learned patterns and biases. In contrast, in a low data setting, generalization becomes a critical challenge due to the fact that inductive biases are restricted to the extracted knowledge from a small amount of data. Such biases are very likely to be inaccurate when it comes to new data. Simply put, the model cannot generalize. This research aims to explore different ways of injecting inductive biases thus improving upon VQA generalization in low data settings. Inspired by the inherent characteristics of the task, we select a set of inductive biases and propose methods to introduce them to a VQA system. We also endeavor to analyze the extent to which these methods enhance VQA performance.

1.2 Aims and Objectives

For a long time, generalization under data scarcity has been a challenging problem in the machine learning, and general approaches can be found in the literature. Many approaches have been used to improve the performance of deep learning models when training on lim-

ited data; ranging from data augmentations (X. Zhang, Wang, Liu, & Ling, 2019) and pre-training (Erhan, Courville, Bengio, & Vincent, 2010) to semi-supervised learning (Kingma, Rezende, Mohamed, & Welling, 2014) and transfer learning (Raina, Battle, Lee, Packer, & Ng, 2007). However in a low data regime, even these approaches cannot capture all variances in input data. Furthermore, these works mostly deal with the scarcity of labeled data by leveraging available unlabeled data, or by transferring knowledge from similar domains. Unlike prior work, we do not require any labeled or unlabeled data in addition to a small training set. In other words, we aim to investigate how far we can push generalization in a low data setting without exploiting extra data by meticulously extracting helpful information from the training data. Using additional data from other sources in low data VQA could indeed be an interesting research direction but it does not fall within the scope of this research.

As stated, the overarching idea of introducing inductive biases to the system to increase generalization ability has steered our attention to the problem of poor VQA performance when data is restricted. We concentrate on the inductive biases that arise from the compositionality inherent in natural language. Compositionality is a key function of the human mind allowing people to solve a large number of new problems using a limited number of basic skills. It can also allow a VQA system to cope with novel unseen cases while trained on a small amount of data.

In VQA terms, compositionality suggests that complex questions are made up of simple concepts contained in simple questions. Complex questions form a large part of VQA datasets. A VQA model requires a large amount of labeled data to capture such complexity, which can be challenging to achieve in low data circumstances. According to compositionality, a complex question may be divided into smaller parts, making it simpler to understand with less data. This idea leads to the following key inductive biases which this research relies on:

1. A learner can understand simple questions easier than complex questions

2. Simple questions contain basic concepts
3. A learner understands the complex questions on the basis of underlying basic concepts.

Now we provide an overview of the specific contributions of this research in the ensuing sections.

1.2.1 Augmenting Low Data VQA with Simple Questions (Chapter 3)

Data augmentation is one of the widely accepted approaches to the problem of insufficient labeled data (Krizhevsky, Sutskever, & Hinton, 2012) using various techniques such as oversampling and data warping (Ruprecht & Muller, 1995; Shorten & Khoshgoftaar, 2019). However, there are only a few studies covering data augmentation in VQA owing to the challenge of maintaining the correct relationship between question, image, and answer (Kafle, Yousefhussien, & Kanan, 2017; Ray, Sikka, Divakaran, Lee, & Burachas, 2019). As VQA datasets are typically made up of complex questions, addressing them entails identifying multiple objects in the image and comprehending their relationships. A learner cannot capture the complexities in a low data regime when it does not have access to large variations of labeled data. As such, we believe that supplementing complex questions with simpler ones will facilitate the acquisition of essential concepts. The learned concepts could then be utilized as a basis for learning complex concepts with a limited amount of annotated data.

1.2.2 Transferring basic knowledge (Chapter 4)

Transfer learning is a machine learning technique in which a model that has been trained for one task is repurposed for a different task. When modeling the new task, it allows for faster learning or enhanced performance. Transfer learning is popular in deep learning given the massive resources required to train a neural network. However, this technique only works if the domain of the first task is close to that of the new task and the learned representations are sufficiently general to be suitable for the new task. We hypothesize that transferring

basic knowledge compensates for the lack of data and helps the model to learn general and powerful biases for comprehending complex questions.

1.2.3 Encouraging the learning of basic concepts (Chapter 5)

Many studies argue that not all examples in a training set have the same value (Canevet & Fleuret, 2015; Katharopoulos & Fleuret, 2018; Shrivastava, Gupta, & Girshick, 2016; Zhou, Wang, & Bilmes, 2020). These works show that investing training time and model capacity on informative samples can lead to significant improvements in performance. Although the definitions of what constitutes informative may vary, they share the principle that making use of efficient examples can reduce the need for a large amount of annotated data. Motivated by such studies, we hypothesize that if we prioritize the training examples so that the model focuses on learning basic concepts at early training and shifts the focus to the complex questions later, we may improve VQA performance.

1.2.4 Employing basic concepts in learning complex questions (Chapter 6)

Applying the learned knowledge from basic questions to complex questions is not always trivial. The issue is that the distribution of the compositions in the training set might be different from those compositions in the test set. For instance, suppose that the probability of a *sphere* object appearing is only 3% in the training set and the model faces a question about *sphere* in the test time. It is very likely the module that learned to identify *sphere* objects does not perform well at the test time. This is due to the distributional mismatch between the trained and target questions which can hinder performance. To tackle this problem, we propose that the reasoning process of the questions in the training set that are similar to the target questions could help the model change the distribution in favor of the target question.

1.3 Contributions

In this thesis, we:

1. introduce and study the task of low data VQA
2. develop novel data augmentation and pretraining techniques based on compositionality of questions to address the poor performance of VQA models in low data settings
3. use curriculum learning as a tool for building knowledge foundations for understanding complex questions
4. cast VQA as a meta-learning problem and introduce VQA pseudo tasks.

1.4 Thesis Outline

In this section, we provide an outline of the remainder of the thesis. The primary contribution of this thesis is offered in four chapters: Chapters 3, 4, 5 and 6. The outline and summary of each chapter are as follows:

- **Chapter 2:** This chapter gives an overview of the foundations for the research described in this thesis, including state-of-the-art VQA approaches and a detailed description of previous work on the intersection of vision and language. We also discussed the VQA datasets that are currently being employed widely in this field
- **Chapter 3:** In this chapter, we present our data augmentation method for generating simple questions and use them to train a VQA model. Experimental results are reported showing drastic improvement in answer prediction accuracy. This chapter also includes a detailed description of the background and related works in data augmentation in general, as well as in VQA
- **Chapter 4:** This chapter describes a self-supervised pretraining strategy that exploits the compositionality of the questions to break them down and learn the components one at a time. The experimental findings for three variations of our pretraining method

demonstrate that it is considerably successful in increasing VQA performance, even with minimal data

- **Chapter 5:** We introduce three curriculum learning methods for training a VQA model with limited data. We also present the results of the experiments on four different training sets and analyze the results by conducting various ablation studies. We demonstrated that one of the proposed curriculum learning methods significantly outperforms the baseline
- **Chapter 6:** We introduce a meta-learning approach for VQA where the model exploits the most similar examples in the training set to predict a target example. We test our proposed approach with four different similarity criteria and demonstrate improvements with respect to the baseline
- **Chapter 7:** This chapter summarises our findings and contributions and highlights potential directions for future research.

2 | Literature Review

This chapter provides a thorough overview of the foundations for the research described in this thesis, including various neural approaches, models and datasets in VQA. We start off by describing the evolution of vision+language tasks in Section 2.1, where we introduce the most common multimodal problems at the intersection of language and vision. This helps locating VQA amongst other related tasks, as well as providing a foreground for describing the basics of VQA neural models. This is followed by a detailed description of the state-of-the-art VQA architectures falling into the two major categories of modular and non-modular approaches. Then, in Section 2.3, we shed light on the popular VQA datasets and their characteristics. Note that the background information and literature review related to the specific techniques are provided as a separate section in the related chapter.

2.1 Vision+Language Tasks

In the last several years, we have witnessed considerable progress in machine visual perception (K. He, Zhang, Ren, & Sun, 2016; Redmon, Divvala, Girshick, & Farhadi, 2016) on the one hand, and language understanding (Saha, Aralikatte, Khapra, & Sankaranarayanan, 2018; Luo et al., 2018) on the other. The progress in both fields has inspired researchers to develop agents that can see and communicate in natural language. Building such systems has been considered an ambitious goal. However, since 2014, there has been great progress

in developing systems with such abilities (Vinyals, Toshev, Bengio, & Erhan, 2015; Das et al., 2017; Antol et al., 2015). Researchers believe that the next generation of intelligent systems will need to master the ability to communicate visual content for a variety of applications. At the intersection of vision and language some tasks have received a considerable amount of attention: image and video captioning and visual question answering, as well as visual dialog. In the following, I briefly review the first two tasks and elaborate on the last.

Figure 2.1: Examples of image captioning [1].

2.1.1 Image Captioning

Image captioning is the task of generating a description for a given image. See Figure 2.1 for an example. It can also be formulated as a retrieval problem to extract the best fitting description from a pool of possible captions (X. Liu, Li, Shao, Chen, & Wang, 2018). As the caption space is huge, constructing a dataset of captions that is sufficient for describing a reasonably large fraction of images seems challenging. Some studies formulate this task as a sequence-to-sequence problem in which a sequence of pixels has to be translated into a sequence of words (Vinyals et al., 2015; Sharma, Agrahari, Singh, Firoj, & Mishra, 2020). For instance, Vinyals et al. (2015) use an encoder-decoder architecture. On the encoder side, they train a convolutional neural network (CNN) to produce an image representation, and on the decoder side, they train a recurrent neural network (RNN) to generate the image caption given the image representation. Encoding the visual content of an image into an effective representation is a critical step of image captioning. Modern studies leverage various ap-

[1]The images are from https://cs.stanford.edu/people/karpathy/sfmltalk.pdf

proaches for it ranging from attention over visual regions (Qin, Du, Zhang, & Lu, 2019; Ke, Pei, Li, Shen, & Tai, 2019; Huang, Wang, Xia, & Chen, 2019; L. Wang, Bai, Zhang, & Lu, 2020) to graph-based encoding (Yao, Pan, Li, & Mei, 2018; Yang, Tang, Zhang, & Cai, 2019) and self-attention (F. Liu et al., 2020; Guo et al., 2020; Touvron et al., 2021). On the decoder side, recent trend suggests using pretrained transformer-based language models that are discussed in Section 2.2.6.

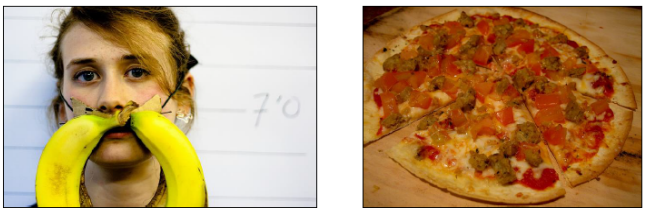


Figure 2.2: Examples of visual question answering[2].

2.1.2 Visual Question Answering (VQA)

A VQA task is defined as answering a question about a provided image (see Figure 2.2 for VQA examples). In fact, VQA is a fusion of image understanding and textual comprehension. VQA may be simplistically considered as a visual extension of textual question answering, however, the visual supporting capability adds significant challenges since images are of much higher dimension and generally noisier than text. Moreover, images do not follow the structure and grammatical rules language does. Compared to image captioning, VQA complexity is partly attributed to the requirement of information that is not present in the image. Such information is usually commonsense knowledge or encyclopedic information about a specific object in the image (Zellers, Bisk, Farhadi, & Choi, 2019; P. Wang, Wu, Shen, Dick, & van den Hengel, 2018). On the other hand, evaluating VQA is easier as VQA answers are typically a few words while long captions are difficult to compare with the ground truths.

Studies on VQA have taken advantage of mature techniques in both NLP and computer

[2]The images are from (Antol et al., 2015)

vision, as well as the availability of large datasets. Therefore, a large body of literature on VQA has been generated over the last few years. For instance, many different deep models based on multimodal representation (Y.-C. Chen et al., 2020; Malinowski, Rohrbach, & Fritz, 2015), attention (Zhu, Groth, Bernstein, & Fei-Fei, 2016; Xu & Saenko, 2016), and memory networks (Das et al., 2017; A. Goyal, Wang, & Deng, 2018) have been developed for this task using a variety of datasets (Zhu et al., 2016; Antol et al., 2015; P. Wang et al., 2018). Despite these studies, it remains difficult to assess the reasoning capabilities of current VQA systems to distinguish them from the memorization of training set patterns.

2.1.3 Visual Dialog

Given an image and the history of a dialog composed of a sequence of question-answer pairs, the visual dialog task is to generate, or to answer, a natural language question. Figure 2.3 shows an example of visual dialog. Despite significant recent progress, it is obvious that we are still far from the goal of having agents which can conduct a dialog about a visual content. In image captioning, the system produces a description of the image without any input from a human. VQA takes a step forward and takes a question as input from a human, but it is still a single round of a dialog. While visual dialog largely resembles the VQA

Visual Dialog
Q: How many people are on wheelchairs ?
A: Two
Q: What are their genders ?
A: One male and one female
Q: Which one is holding a racket ?
A: The woman

Visual Dialog
Q: What is the gender of the one in the white shirt ?
A: She is a woman
Q: What is she doing ?
A: Playing a Wii game
Q: Is that a man to her right
A: No, it's a woman

Figure 2.3: Two examples of visual dialog[3].

[3]The images are from (Das et al., 2017)

task, taking the history of the dialog into account makes a significant difference. To clarify, consider the visual dialog examples in Figure 2.3. The question *"what is the gender of the one in the white shirt?"* requires the machine to attend to a specific region of the image. *"what is she doing?"* requires co-reference resolution to detect to whom the pronoun *"she"* refers. *"is that a man to her right?"* requires a visual memory to remember what object in the image we are talking about. Moreover, the system needs to be consistent with its output. For example, consider the dialog *"how many people are in the picture"*, *"two"*, *"what are their genders?"*, *"one male and the other female"*. Note that the number of genders specified in the second answer should be consistent with the number of people in the first answer. Such challenges make this problem different from VQA and interesting to the researchers.

2.2 Neural Approaches in VQA

One of the earliest attempts at open-world VQA tried to combine semantic text parsing with image segmentation in a Bayesian framework, and used nearest neighbors in the training set to answer the questions (Malinowski & Fritz, 2014). Another work by Tu, Meng, Lee, Choe, and Zhu (2013) was based on jointly parsing text and videos in the form of a graph. All of these early approaches are restricted to predefined forms of questions while modern approaches mainly focus on free-form open-ended questions.

With the advances in the computing power of systems available today, many modern approaches are using neural networks to solve a variety of tasks in the fields of computer vision and natural language processing, e.g., image classification (Krizhevsky et al., 2012; T. He et al., 2019; Bhojanapalli et al., 2021) and machine translation (Bahdanau, Cho, & Bengio, 2015; Vaswani et al., 2018; Freitag et al., 2021). VQA received attraction after deep learning approaches had gained wide popularity due to their state-of-the-art performance on various vision and NLP tasks. As a result, almost all modern studies on VQA involve deep learning approaches. We categorize methods based on four criteria: joint embedding, attention mechanism, modular models, and knowledge-based enhancement. Obviously, some

methods fall into multiple categories as they use a combination of strategies.

2.2.1 Joint Embedding Approaches

The idea of jointly embedding images and text was first employed for the task of image captioning. It is further emphasized in VQA as it needs to reason on both modalities to answer questions. Representing both text and images in a common space allows easy interaction between the two modalities and consequently performing inference over the image and question. Creating join embedding practically involves three steps: 1) image representation; 2) text (question) representation and 3) combining the representations in a common embedding space. In the following, I explain the steps in more detail.

Image Representation To extract image features, most studies use a convolutional neural network CNN pre-trained on the ImageNet dataset (Deng et al., 2009) and object recognition task. ImageNet is a large-scale image database designed for use in visual object recognition tasks. It includes over 14 million images with objects in the pictures annotated by humans[4]. A CNN common architecture for extracting image features is the VGG model (Simonyan & Zisserman, 2015) [5]. VGG architecture is substantially deeper than prior studies increasing the depth to 16 to 19 convolutional layers, resulting in a significant improvement. Figure 2.4 shows a macro-view of the VGG architecture. ResNet (K. He et al., 2016) is another popular architecture that was introduced after VGG and achieved higher performance. The core idea in ResNet is based on skip-connections that skip one or more layers. This strategy is introduced as a solution to the vanishing gradient problem in deep CNNs. Residual blocks (shown in Figure 2.4) contain skip connections enabling the model to learn residual functions with reference to the input. The network consists of stacked residual blocks allowing hundreds of layers.

A recent trend is on the adoption of transformers in vision models to further improve representations obtained by CNNs (B. Wu et al., 2020; Carion et al., 2020; Dosovitskiy

[4]ImageNet in Wikipedia: https://en.wikipedia.org/wiki/ImageNet#cite_note-1
[5]http://www.robots.ox.ac.uk/~vgg/research/very_deep/

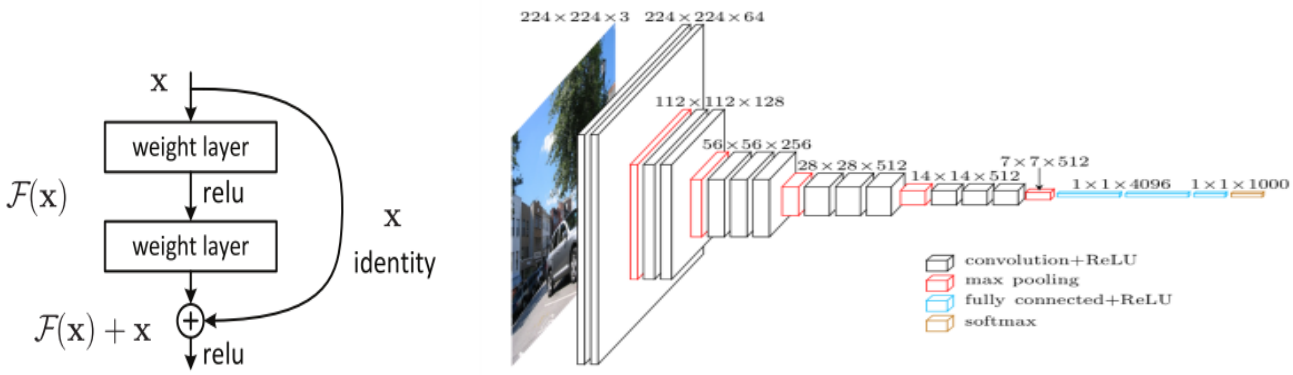


Figure 2.4: Left: A residual block with skip connections. Right: a macro-view of VGG architecture.

et al., 2021). Transformer models are based on attention mechanism that is introduced in Section 2.2.2.

Text Representation In VQA, question representations are obtained by word embedding networks pretrained on large text corpora. Deep neural models process the textual inputs by replacing each word w_i with a high-dimensional feature vector representation as word embedding $e(w_i)$. Word embedding practically maps the words to a semantic space in which the distances reflect the semantic similarities (Bengio, Ducharme, Vincent, & Jauvin, 2003). It is typically a lookup table with a row for each word, and parametrized by a matrix learned like other parameters in the network. The parameters are optimized using the back-propagated errors from the objective function during end-to-end training. To encode the sequence of words, studies use different architectures, mostly RNNs, long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014).

Recent studies widely use pretrained language models specially transformer-based models such as **BERT** (Devlin, Chang, Lee, & Toutanova, 2019) and **GPT3** (Brown et al., 2020) to represent questions. Such models are pretrained in self-supervised manners using massive datasets and it is shown that they can provide strong text representations, helpful to many downstream applications. As an example, we use the Sentence-BERT (Reimers &

Gurevych, 2019) model to represent questions in Chapter 6. See Section 6.4.1. Pretrained models are covered in Section 2.2.6 with more details and examples.

Multimodal Representation One of the important challenges in multimodal machine learning is integrating representations from multiple modalities in order to predict an outcome. Multimodal representation has three main advantages. First, assuming each modality views the same phenomenon from a different perspective, having access to multiple modalities may lead to a more robust prediction. Second, taking advantage of multiple modalities may result in capturing complementary information about a phenomenon. For instance, if something is not evident in an image, it might be captured from text. Third, even if one modality is missing, a multimodal system may still function, although at a lower level of performance.

To create a joint embedding of images and text, various approaches have been proposed in the VQA literature. Some studies adopt a simple concatenation of both representations and then linearly transform the result to a desired length vector in the joint space (Das et al., 2017). Some other studies utilize an element-wise summation or pooling methods (Antol et al., 2015; Fukui et al., 2016). Fukui et al. (2016), for instance, suggest a pooling method to produce a joint embedding by randomly projecting the image and text features to a higher-dimensional space. Then both vectors are efficiently convolved in a Fourier space with multiplication. Malinowski and Fritz (2014) feed question and image features together to an LSTM encoder to produce a fixed-size feature vector which is then passed to an LSTM decoder to produce a variable-length answer. Ren, Kiros, and Zemel (2015) similarly create the fixed-size feature vector but feed it to a classifier to produce single-word answers. Both of these works use bidirectional LSTMs to better capture the relations between distant words.

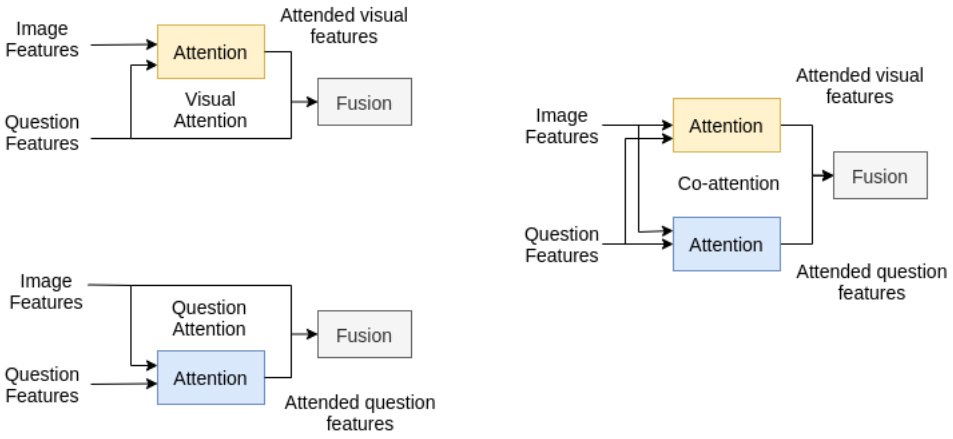


Figure 2.5: **Top-Left:**Visual Attention guided by question features. **Bottom-Left:**Question Attention guided by visual features. **Right:**Co-attention mechanism where attention weights are calculated for both modalities in parallel.

2.2.2 Attention Mechanisms

Joint embedding approaches are limited to using question and image representations holistically. The attention mechanism allows focusing on important words in a question or important regions of the image. It facilitates interaction between visual and textual features and is proved to be efficient. The idea behind the attention mechanism is to compute a context vector of a modality guided by the information obtained from the feature vector of the other modality. Thus, attention in VQA can be implemented as question attention, visual attention and co-attention. Visual attention uses the question feature vector Q as guidance to calculate the attention weights a^v. This attention weight matrix is further used to compute the attended visual features $V^a = a^v V$. The most typical method for obtaining attended visual features is to look for a correlation between visual features and question features.

A softmax function then normalizes the weights. As an example, Zhu et al. (2016) propose to add spatial attention to the standard LSTM model. This study calculates the

LSTM unit as follows:

$$\begin{aligned} i_t =& \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{zi}Z_t + b_i) \\ f_t =& \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{zf}Z_t + b_f) \\ o_t =& \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{zo}Z_t + b_o) \\ g_t =& \tanh(W_{xg}x_t + W_{hg}h_{t-1} + W_{zg}Z_t + b_g) \\ c_t =& f_t \odot c_{t-1} + i_t \odot g_t \\ h_t =& o_t \odot \tanh(c_t) \end{aligned} \tag{2.1}$$

where $\sigma(.)$ is a sigmoid nonlinearity, and i_t, f_t, c_t, o_t are the input, forget, memory and output gates of the LSTM. $\odot$ is the element-wise multiplication operator. x_t is the input (e.g. a word of the question) and h_t is the hidden state of the time step t. z_t represents the attention mechanism which is a weighted average of convolutional features. It can be formulated as the following:

$$\begin{aligned} e_t =& w_a^T \tanh(W_{he}h_{t-1} + W_{ce}C(I)) + b_a \\ a_t =& softmax(e_t) \\ z_t =& a_t^T C(I) \end{aligned} \tag{2.2}$$

where $C(I)$ represents the convolutional feature map of the image I. The attention term a_t determines the significance of each convolutional feature at the t-th time step. Larger values in a_t indicate greater relevance of the corresponding region to the question. All the Ws and bs are trainable parameters. Visual attention is used by Zhu et al. (2016); Anderson et al. (2018); L. Yu et al. (2018); D. Yu, Fu, Mei, and Rui (2017); L. Yu et al. (2018) among others. Zhuang, Wu, Shen, Reid, and Hengel (2018) use a dual attention mechanism in a visual reference resolution task. They aimed to recognize a target object in an image using the information from a sequence of question-answer pairs in the form of a dialog. The first attention is image-level which attends to the relevant region of the whole image, while the other attention mechanism is object-level which attends to each object segmentation separately. The use of object-level attention is to weight different candidate proposals when the referring expression is associated with multiple objects.

The model mentioned above uses the attention mechanism over images. Similarly, some studies attend over questions. Attended question features Q^a can be obtained as $Q^a = a^q Q$ where a^q is guided by visual features. In (Das et al., 2017), a hierarchical recurrent encoder makes a joint representation in two levels; utterance level and dialog level. In the first layer, an LSTM embeds the question and image jointly, and another LSTM embeds each round of the dialog history H_t. Then, a concatenation of these representations is passed to an RNN on top of the first layer. An attention mechanism over the dialog history enables the RNN to attend to the round of history relevant to the current question.

The co-attention mechanism computes the attention weights for both visual and question features in the same manner and in parallel (Tan & Bansal, 2019; Lu, Yang, Batra, & Parikh, 2016). For instance, (Lu et al., 2016) describe their proposed co-attention mechanism as follows:

$$\boldsymbol{C} = \tanh(\boldsymbol{Q}\boldsymbol{W}_b\boldsymbol{V}) \tag{2.3}$$

where C is the correlation matrix between visual and question features. Other studies may use $\boldsymbol{C} = \boldsymbol{W_v V} + \boldsymbol{W_q Q}$ such as (Zhu et al., 2016) or $\boldsymbol{C} = sigmoid(\boldsymbol{W_q Q})\boldsymbol{V}$ such as (K. Chen et al., 2015) instead of using the product of Q and V in the above equation. The co-attention matrices are then calculated using the following equations:

$$\begin{aligned} \boldsymbol{H}^v = \tanh(\boldsymbol{W}_v\boldsymbol{V} + (\boldsymbol{W}_q\boldsymbol{Q})\boldsymbol{C}), \quad \boldsymbol{H}^q = \tanh(\boldsymbol{W}_q\boldsymbol{Q} + (\boldsymbol{W}_v\boldsymbol{V})\boldsymbol{C}^T) \\ \boldsymbol{a}^v = \text{softmax}(\boldsymbol{w}_{hv}^T\boldsymbol{H}^v), \quad \boldsymbol{a}^q = \text{softmax}(\boldsymbol{w}_{hq}^T\boldsymbol{H}^q) \end{aligned} \tag{2.4}$$

where all Ws are learnable weights. The image and question attention vectors are further calculated based on the above attention weight as the weighted sum of the image features and question features, i.e.,

$$\boldsymbol{v}^a = \sum_{n=1}^{N} a_n^v \boldsymbol{v}_n, \quad \boldsymbol{q}^a = \sum_{t=1}^{T} a_t^q \boldsymbol{q}_t \tag{2.5}$$

2.2.3 Dynamic Memory Network

The Dynamic Memory Network (DMN) is a neural network with a modular architecture proposed in (A. Kumar et al., 2016) to address the problem of the small size of memory in RNNs and LSTMs. This limitation becomes especially apparent when dealing with a very long sequence of data. DMN tackles this by storing the input data representation called fact and using the attention mechanism to retrieve them. This allows the network to refer back to the input sequence, instead of forcing it to encode all information into one fixed-length vector. A number of DMN variants are introduced in (Bordes, Usunier, Chopra, & Weston, 2015; Sukhbaatar, Szlam, Weston, & Fergus, 2015) but we focus on (Xiong, Merity, & Socher, 2016) which modifies memory networks for VQA tasks. The model in (Xiong et al., 2016) contains an input module for extracting image features which are then fed to a GRU to linearly traverse the image while a question module creates question representations. The episodic memory module retrieves the facts required to answer the questions allowing multiple passes over the facts. To select the relevant facts, the GRU utilizes attention mechanism and based on those facts updates memory representations to a new status.

Das et al. (2017) propose a DMN encoder for a visual dialog task that stores each round of the dialog history as a 'fact' in its memory bank. The encoder learns an attention mechanism to refer to these facts and the image to answer the question. To encode the question and the dialog history, this work uses separate LSTMs to get a 512-d vector for the question and a $t \times 512$ matrix as the history embedding. The attention is over the history and is defined as an inner product of question vector with each history round which is fed to a softmax to obtain attention probabilities.

2.2.4 Compositional Architectures

Modern approaches in VQA attempt to learn mapping from input, question-image pairs, to output, answers, in an end-to-end manner (Antol et al., 2015; Zhu et al., 2016) They do

not explicitly manage the reasoning process. Direct mapping fails to generalize well on novel examples that require a deep understanding of the vast space of objects, attributes, relationships, and interactions. Such systems typically learn dataset biases, preventing them from successfully performing complex reasoning tasks on unseen examples. To alleviate this problem, generating compositional reasoning using linguistically informed approaches has received attention in the VQA area. In this section, we mention two approaches in this direction: MAC recurrent network, and neural module network.

MAC Recurrent Network

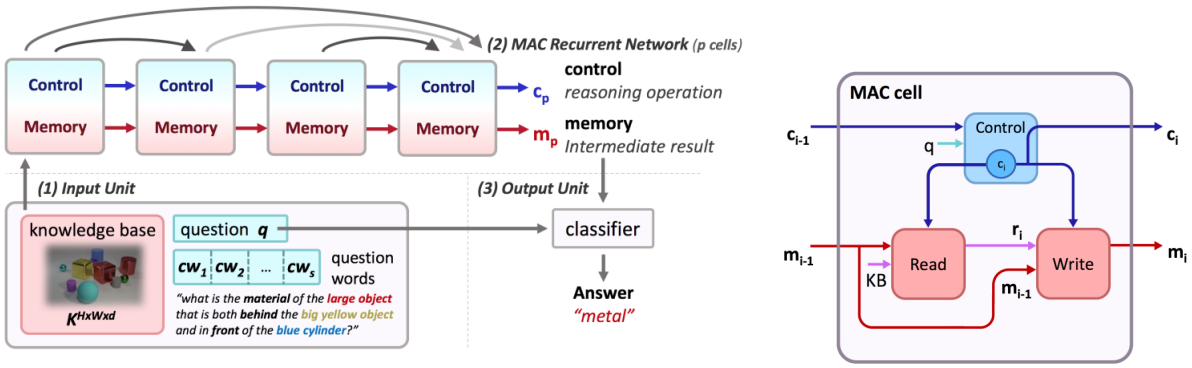


Figure 2.6: **Left:** Overview of MAC recurrent network which consists of an input unit, recurrent MAC cells and a classifier. **Right:** MAC cell architecture that includes a control unit, read unit, and write unit communicating using control c_i and memory m_i hidden states. Figures adapted from (Hudson & Manning, 2018)

Hudson and Manning (2018) introduce an interesting compositional architecture that decomposes a VQA task into a series of attention-based reasoning steps. As seen in Figure 2.6, the model consists of an input unit that generates input representations and a recurrent network that performs the reasoning processing. The output is produced by a classifier using the question and the final state of the recurrent network. Each cell of the recurrent network is called Memory, Attention, and Composition (MAC) cell and contains three units: a control unit, a read unit and a write unit. Given a question-image pair, the control unit (CU) determines a series of required operations for answer prediction by decomposing the question and successively attending to different parts of it. The read unit (RU) extracts the information from the knowledge base (here the image) guided by the control unit. The write

unit (WR) takes the retrieved information and integrates it into the current state to generate a new intermediate result. To communicate, the units operate on *control* and *memory* hidden states. Later, we discuss that, compared to black box models, the modular architecture of the MAC model increases the interpretability of the reasoning process, though not as much as neural module models.

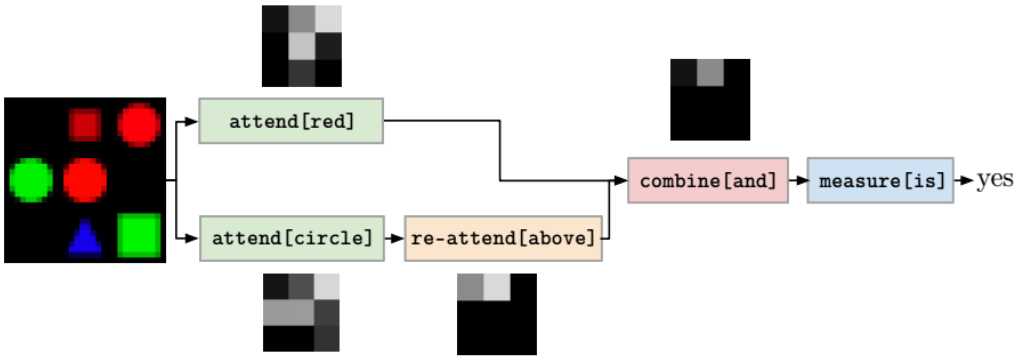


Figure 2.7: The Neural Module Networks (NMN), Section 2.2.4, take advantage of compositional structure of questions to create a modular reasoning chain. In this example, the questions is *"is there a red shape above a circle?"*. The semantic parsing of the question results in a layout of assembling modules. The `attend` modules find *red* shapes and *circles*, `re-attend[above]` moves the attention above the circle, `combine` determines the intersection of attended areas, and `measure[is]` produces the answer *yes* if the final attention in non-empty. (Figure adapted from (Andreas et al., 2016b)).

Neural Module Networks

As one of the first attempts in this direction, Andreas et al. (2016b) introduced Neural Memory Network (NMN) and extend in (Andreas, Rohrbach, Darrell, & Klein, 2016a). NMNs are assembled on-the-fly for each instance to an architecture that reflects the complexity of the questions. For instance, the resulting model for a question such as *"is this a bike?"* is simpler than the corresponding model of the question *"how many red balls are to the right of the tree?"* which requires multiple processing steps. Andreas et al. (2016b) make use of ad-hoc hand-written rules to generate a network consisting of a set of predefined basic *'modules'*. The network is guided by the semantic structure of the input question obtained

[5]The figure adapted from https://www.cs.toronto.edu/~frossard/post/vgg16/

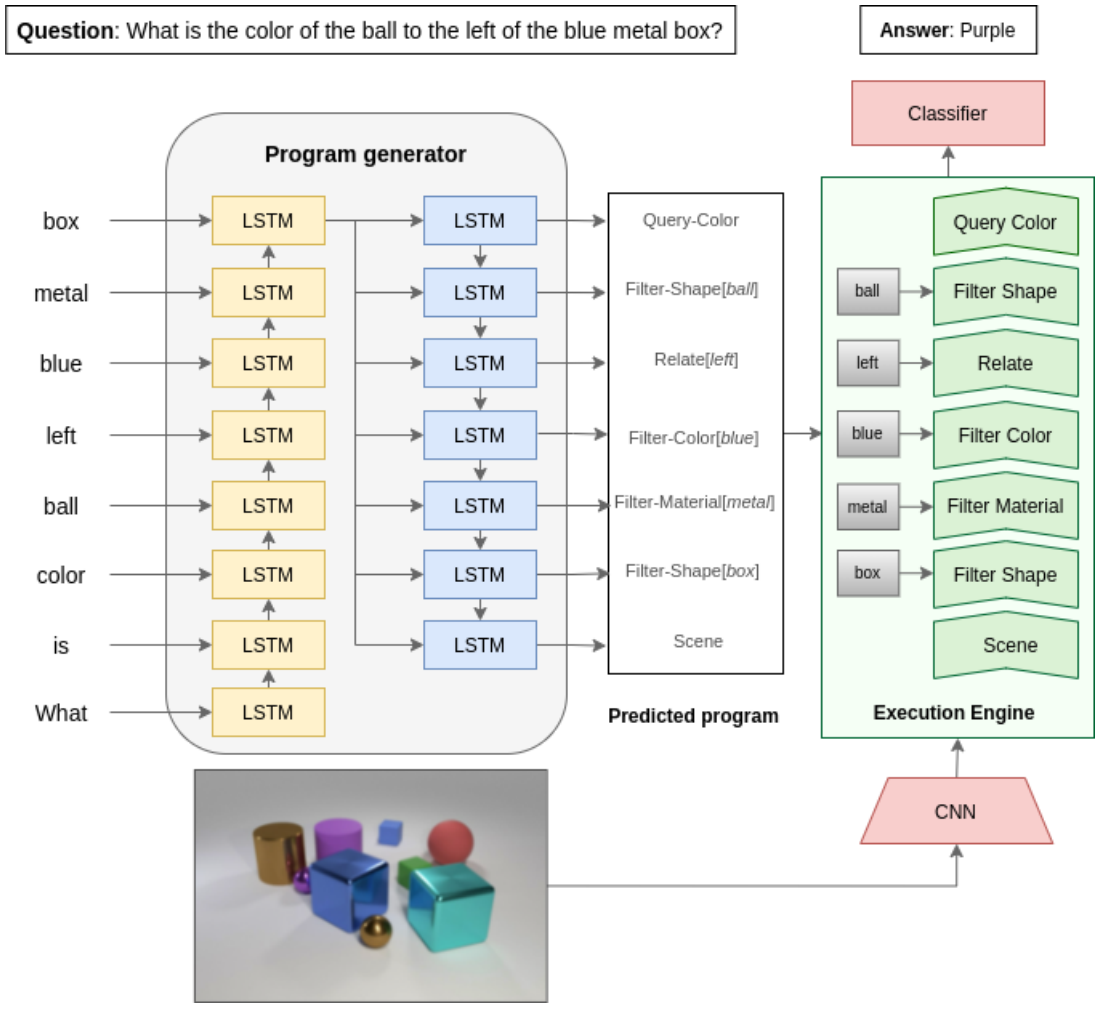


Figure 2.8: An overview of a modular VQA system consisting of a *program generator* and an *execution engine*.

from a dependency parser. Thus the computation performed for each instance will be different from the others. In such a structure, each module functions as a single step of reasoning while the network represents the whole reasoning process. Figure 2.7 depicts an example of such networks.

The use of an off-the-shelf parser prohibits end-to-end training of the entire architecture which makes it inefficient in training. To tackle this defect, further studies try to learn to predict the reasoning network (Andreas et al., 2016a; Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017; Hu, Andreas, Darrell, & Saenko, 2018). They add an additional

component to the architecture as a task-specific parser to generates the network assembling layouts. More specifically, in (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017), a component called *program generator* $\mathcal{G}$ takes a question and generates a module layout known as *program*. For instance, the question *"what is the color of the ball to the left of the blue metal box?"* corresponds to the program "`Scene` → `Filter-Shape[`*`box`*`]` → `Filter-Material[`*`metal`*`]`→ `Filter-Color[`*`blue`*`]` → `Relate[`*`left`*`]` → `Filter-Shape[`*`ball`*`]` → `Query-Color`". This program is further fed to another components named *execution engine* $\mathcal{E}$ that aims to assemble the modules accordingly. The resulting network is executed on the input image to predict the answer. Figure 2.8 shows the model architecture. The modules are small neural networks treated as single-task functions that are combined into a larger network to undertake a complex job. They share a generic architecture allowing them to easily connect to each other. A module with n inputs receives n features map of shape $C \times H \times W$ and outputs a feature map of the same shape. The architecture of unary modules consists of a standard residual block (K. He et al., 2016) with two 3×3 convolutional layers. Binary modules have a similar architecture except the two inputs are concatenated before feeding to the residual block. The *program generator* is first trained in a supervised manner using a small set of questions and the corresponding hand-crafted programs. It is then combined with the *execution engine* to train in a reinforcement learning setting in an end-to-end manner. We use *execution engine* in this thesis as the base model.

Modular networks are generally proved to outperform their competitors on complex questions that require multi-hop reasoning such as locating an object and identifying its attributes. However, all the mentioned approaches are primarily based on the assumption that a *fixed* predefined set of the basic modules is provided which may cause oversimplification of the questions that ignore some grammatical clues. They are also evaluated on synthetically generated datasets that are visually simple with a limited variation in the question structure which is different from real world scenes and linguistically diverse human-generated questions. Modular approaches naturally have a strong potential for interpretability. Hu et al. (2018) evaluate the interpretability of their modular network compared to the non-modular

model in (Hudson & Manning, 2018), which uses a multi-hop reasoning method without explicitly extracting the reasoning procedure. Then, they visualized intermediate models' outputs, in addition to the final one to be evaluated by humans. The evaluation scores show that the users can more clearly understand the reasoning steps in modular models.

2.2.5 Using External Knowledge Bases

Understanding a question and an image in VQA may require (extra) information not included in the training data such as commonsense knowledge or encyclopedic information. For example, answering the question *"what color is the mammal in the image?"* involves understanding the word *"mammal"* and knowing that animals belong to this category. Since neural networks capture their knowledge from the training data, it is clear that expecting them to cover all real-world scenarios is unreasonable. Moreover, the limited capacity of learning inevitably prevents neural networks from capturing less frequently-used information. Decoupling reasoning from the knowledge storage is an alternative to this problem that seems possible with the availability of large-scale knowledge bases such as Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), ConceptNet (H. Liu & Singh, 2004) and DBpedia (S. Auer et al., 2007).

A general idea is to map question-image pairs to *queries* over the knowledge bases and use the result of the query to obtain the final answer (P. Wang et al., 2018; Z. Su et al., 2018; S. Shah, Mishra, Yadati, & Talukdar, 2019; Vickers, Aletras, Monti, & Barrault, 2021). As an example, (Q. Wu, Wang, Shen, Dick, & Hengel, 2016) extracts the semantic attributes from a given image and retrieves the related knowledge pieces to the attributes from DBpedia. The knowledge is then embedded into vectors which serve as input to an LSTM along with questions. The final answer is generated using the output of the LSTM. Vickers et al. (2021) integrate facts extracted from KBs to improve the reasoning performance of multimodal pretrained model Vision+Language Bert (Y.-C. Chen et al., 2020).

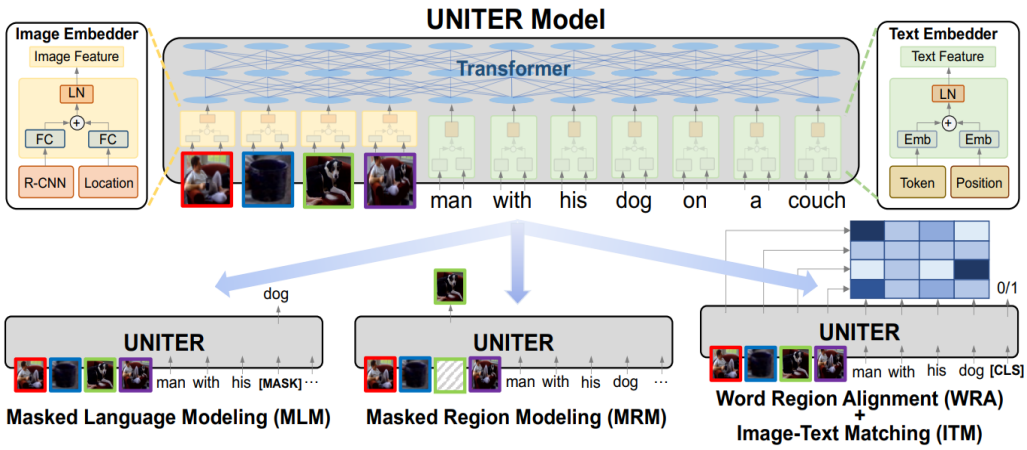


Figure 2.9: Overview of the UNITER model. Figure adapted from (Y.-C. Chen et al., 2020)

2.2.6 Multimodal Pretrained Models

Recently, the advent of pretrained language models such as **BERT** (Devlin et al., 2019) , **GPT3** (Brown et al., 2020), and **RoBERTa** (Y. Liu et al., 2019), that commonly employ self-supervised learning, have offered a significant advance in the variety of NLP tasks showing the advantage of transfer learning. Inspired by such models, there has been a surge in interest in creating pretrained models for multimodal tasks in a similar self-supervised manner. Multimodal pretrained models aim to provide rich joint embedding on large-scale image/video and text pairs for downstream tasks. For instance, **VideoBERT** (Sun, Myers, Vondrick, Murphy, & Schmid, 2019) and **CBT** (Sun, Baradel, Murphy, & Schmid, 2019) learned a join distribution over linguistic tokens and video frames using BERT. **ViLBERT** (Lu, Batra, Parikh, & Lee, 2019) and **LXMERT** (Tan & Bansal, 2019) proposed a two-stream architecture for independently learning visual and textual embedding using two transformers where a third transformer is used to fuse the embedding later on. In contrast, **VisualBERT** (L. H. Li, Yatskar, Yin, Hsieh, & Chang, 2019), **VL-BERT** (W. Su et al., 2020), **B2T2** (Alberti, Ling, Collins, & Reitter, 2020), and **CLIP** (Radford et al., 2021) employ a single transformer to apply to both modalities in a single stream architecture.

Among all the multimodal pretrained models, let us investigate **UNITER** (Y.-C. Chen et al., 2020) further as it has outperformed the other models in many downstream tasks. Fig-

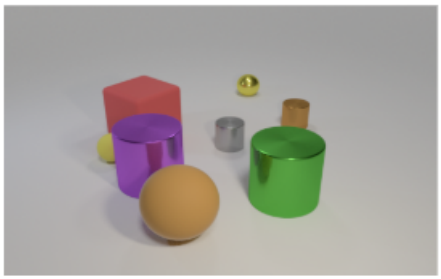
ure 2.9 depicts an overview of the model. As seen, it consists of an *Image Embedder* which applies a Fast R-CNN (Girshick, 2015) pretrained on visual Gnome with image regions to extract visual features. The location of each region is also encoded in a vector. Both visual and location features are fused and mapped to the same embedding space using a fully-connected layer. *Text Embedder*, on the other hand, tokenizes the input sentence to the words and obtains word embedding as well as their encoded position. These embeddings are then fed into a multi-layer transformer where cross-modality embeddings are learned over visual regions and textual tokens using four main pretraining tasks.

The pretraining tasks are introduced as Masked Language Modeling (MLM), Masked Region Modeling (MRM), Image-Text Matching (ITM), and Word-Region Alignment (WRA). The first two tasks, *i.e.,* MLM and MRM, are implemented by randomly masking some words or regions from the input and learning to recover the masked parts in the transformer output. In order to mask a word, it is replaced with `[MASK]` token while a masked region of an image is zeroed out. These tasks are also used in prior works such as (Devlin et al., 2019; Lu et al., 2019). ITM aims to develop an instance-level alignment between the whole sentence and image. For this, positive and negative image-sentence pairs are fed to the model where it learns to predict their matching score. Via the WRA task, the model learns to transport the contextualized image embedding to word embedding space at the minimum cost. The UNITER model is pretrained by training on one randomly-sampled task per mini-batch.

2.3 Datasets for VQA

Diverse and large-scale datasets are key components of deep learning models. Various VQA datasets have recently been created and published for different purposes. In this section, we focus on the most common datasets. **VQA 2.0** dataset is released by (Antol et al., 2015) containing 105,904 open-ended questions about 204,721 images associated with open-ended or multiple choice answers. This dataset is characterised by statistical biases. To mitigate these biases, **CLEVR** dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017) was introduced with the goal of testing the reasoning ability of the models by focusing on

VQA Dataset
Question: What color is the man's shirt?
Answer: Blue



CLEVR Dataset
Question: There is a small ball that is the same color as the tiny matte object; what material is it?
Answer: Metal

FVQA Dataset:
Question: Which transportation was in this image is cheaper than taxi?
Answer: Bus
Support Fact: Buses are cheaper than taxi.

GQA Dataset:
Question: Is the giraffe is behind the man?
Answer: Yes

Figure 2.10: Examples of four popular VQA datasets.

different aspects of visual reasoning such as counting, logic and attribute recognition. It contains $\sim 700k$ complex questions asking about synthetic images which require multi-step reasoning to produce the answer. Objects in the images have the following attributes: size (*large* or *small*), shape (*cube*, *cylinder*, or *sphere*), color (*cyan*, *blue*, *red*, *purple*, *yellow*, *brown*, *green* or *gray*) and material (*metal*, or *rubber*). The number of objects in the images is limited and answer options contain a single word. In this thesis, we use CLEVR as our main source of data.

Hudson and Manning (2019a) released the **GQA** dataset which contains open-ended questions about real images focusing on the use of scene graphs and intermediate answers in answering questions of various degrees of compositionality. The data set contains scene graphs that contain objects, their attributes and relationships in addition to the images, questions and answers. **Visual7W** (Zhu et al., 2016) and **Visual Genome** (R. Krishna et al., 2017) aim to improve the visual understanding and systematically evaluate model capability in spatial reasoning. There are also datasets that have been developed to enhance knowledge-based reasoning tasks and provide a benchmark for evaluation. Among them let us mention **FVQA** (P. Wang et al., 2018), the Fact-based VQA dataset that includes support facts in addition to images, questions, and answers. The facts are provided in external KBs as structured representations of knowledge that are required for answering questions. The dataset contains 2,190 images sampled from **MS COCO** (T.-Y. Lin et al., 2014) and 5,826 questions corresponding to 4,216 facts. Figure 2.10 shows examples from different datasets, while Table 2.1 compares the characteristics of the abovementioned and more VQA datasets.

Dataset	Number of questions	Number of images	Knowledge based?	Goal	Answer type	Avg. answer length	Avg. question length
DAQUAR	12,468	1,449	*No*	visual: counts, colors, objects	Open	1.1	11.5
Visual Madlibs	360,001	10,738	*No*	visual: scene, objects, person	Fill-in-the-blank/ Multi-choice	2.8	4.9
Visual 7W	327,939	47,300	*No*	visual: object-grounded questions	Multi-choice	2.0	6.9
VQA (v2)	1.1M	200K	*No*	visual understanding	Open/ Multi-choice	1.2	6.1
MovieQA	14,944	408V	*No*	text + visual story comprehension	Multi-choice	5.3	9.3
CLEVR	999,968	100,000	*No*	logical reasoning	Open	1.0	18.4
KB-VQA	2,402	700	*Yes*	visual reasoning with given KB	Open	2.0	6.8
FVQA	5,826	2,190	*Yes*	visual reasoning with given KB	Open	1.2	9.5
OK-VQA	14,055	14,031	*Yes*	visual reasoning with open knowledge	Open	1.3	8.1

Table 2.1: Comparison of common VQA datasets including **DAQUAR**(Malinowski & Fritz, 2014), **Visual Madlibs**(L. Yu et al., 2015), **Visual 7W**(Zhu et al., 2016), **VQA (v2)**(Y. Goyal et al., 2019), **MovieQA**(Tapaswi et al., 2016), **CLEVR**(Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017), **KB-VQA**(P. Wang et al., 2017), **FVQA**(P. Wang et al., 2018), **OK-VQA**(Marino et al., 2019).

3 | A Data Augmentation Approach for Injecting Inductive Biases

VQA models have recently achieved remarkable results when trained on large scale dataset. Current VQA datasets mostly contain complex questions and usually lack simple questions. Relying on complex questions does not pose any problem for a model when there is access to a large amount of data. However, when it comes to low data scenarios learning only from complex questions may introduce a serious challenge for generalization. In this chapter we address this problem by explicitly injecting the inductive biases at the data level. Our solution rely mainly on the inductive bias which states "simple questions provide basic concepts". Our goal is to demonstrate that a model can better learn complex questions in a low data regime if it is given a chance to learn basic concepts from simple questions.

3.1 Introduction

VQA datasets tend to contain complex questions in order to evaluate the capability of current multimodal comprehension models. In other words, as one of its main applications, VQA is used as a proxy task to test models' capacities to understand visual scenes and ground language to them. Answering simple questions is not sufficiently challenging for such a purpose, while answering complex questions requires identifying multiple objects and understanding their relationships.

To understand an unseen question and correctly predict the answer, a model requires capturing the compositionality inductive biases. Understanding an image scene with a rich number of objects is a lot easier for the model when training on a large dataset. In fact, the model implicitly captures the inductive biases by repeatedly seeing a large variety of data when training on a large-scale dataset. However, capturing complicated relationships from small datasets is challenging. A small training set does not offer a large variety of labeled data for learning the underlying biases which harms generalization.

The main goal of this chapter is to improve the generalisation of VQA models by injecting inductive biases so the model can explicitly have access to them in a data efficient manner. Inductive biases are of particularly high significance to questions as they impact the answers in VQA significantly. An inductive bias that a typical learner acquires by training on natural language tasks is related to the inherent compositionality of the human language, *e.g.,* a complex sentence can be understood by understanding its simpler chunks. With the resulting chunks, meaning is normally easier to capture, providing a powerful foundation for understanding complex sentences.

Inspired by the fact that a complex question can be learned on the basis of the basic concepts, we hypothesized that augmenting the training set of complex questions with simpler questions will help the model. We applied the notion that the simplicity of a question can be defined based on different criteria, including syntactic and semantic dimensions. In the VQA context, we consider simplicity as the number of reasoning steps required for answering a question. Thus, the simplest possible question requires identifying a single object and reasoning about it. We particularly include simpler questions that if learned could lead to better representations in the VQA model.

We take a data augmentation approach and enlarge the initial small training set by automatically generating simple question-answer pairs for images. We hypothesise that basic concepts can be learned from simple questions, enabling the model to better learn the structure of more complex questions.

Data augmentation strategies have proven to be particularly useful in a variety of computer vision applications, including images classifications (Krizhevsky et al., 2012). Not only they can be helpful in overcoming the problem of insufficiently labeled data, they are also used to reduce overfitting and class imbalance problems (Shorten & Khoshgoftaar, 2019). Current data augmentation techniques use data warping or oversampling to increase the size of the training dataset (Ruprecht & Muller, 1995; Shorten & Khoshgoftaar, 2019). Data warping is a technique for transforming data while maintaining its labels. Typically, the examples are transformed by geometric and color transformations, random erasing, neural style transfer and adversarial training.

In contrast to the computer vision area, data augmentation in VQA is under-explored due to the challenge of correctly preserving the semantic relation of the *<image, questions, answer>* triplet during transformation. Geometric transform, and random cropping of the image cannot guarantee to preserve the answer. For instance, the answer to *"what color is the thing on the left side of the cube?"* may be flipped if the image is vertically transformed. Random cropping can result in missing the number of objects when counting to answer a *how many* question.

In this chapter, we present a data augmentation method that automatically generates simple questions. The proposed method is an automatic template-based approach which only requires having access to the superficial annotations of an image scene and does not use any additional labeled data. The annotations give some information about the appearance of the objects in the image. The answers to the questions are also automatically generated at no cost of human effort. The method is generic and applicable to any dataset if the scene information is available. The experimental results and analysis demonstrate that our method is very effective in improving VQA performance, and significantly outperform the baselines' result by a large margin in terms of accuracy.

3.2 Related Work

3.2.1 Data Augmentation

It is widely accepted that using larger datasets in training yields stronger DNN models. However, in many domains such as medical applications, limited datasets present common challenges due to the manual effort of collecting and annotating data. One of the main problems of training on insufficient data particularly in deep learning is overfitting. Many methods try to solve this problem by focusing on the model's architecture (K. He et al., 2016) or regularization methods (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014; Ioffe & Szegedy, 2015). In contrast to these, data augmentation tackles overfitting from the root *i.e.,* by manipulating the training set. This lies on the idea that data augmentation can extract more information from the original dataset. It generally consists of artificially increasing the size and the diversity of training examples by automatically creating additional augmented data based on the available data. Besides the problem of limited datasets, data augmenting has also been considered in the class imbalance distribution problem.

Data augmentation has received most of its attention in computer vision (CV). Many widely-used augmentation techniques are introduced to improve the generalization ability of CNN models in vision tasks such as image classification, object detection and image segmentation. These techniques are most applicable to images but not other types of data. In other areas such as text processing or more specifically Natural Language Processing (NLP), data augmentation has not been well explored. Although VQA is a multimodal problem involves in both image and text, however, due to some restrictions it cannot easily benefit the data augmentation advances in both image and text modalities. The ensuing sections elaborate on the related studies on data augmentation in CV and NLP as a foreground for VQA data augmentation. We will explore data augmentation studies in VQA later.

Visual Data Augmentation Image augmentation techniques are either a kind of data warping or oversampling. Warping techniques in computer vision transform images while

preserving labels. The techniques generally used for warping include geometric transformation, random erasing, color transformation, adversarial training, and neural style transfer. Oversampling is different from data warping, but they are not mutually exclusive. It generates synthetic training data by mixing images, or features space augmentation, or even simply replicating examples.

Lecun, Bottou, Bengio, and Haffner (1998) is one of the first works that used data warping for handwritten digit classification. Data augmentation has also been explored in oversampling applications to use re-sampling for solving the imbalanced class distribution problem. (Krizhevsky et al., 2012) proposed AlexNet CNN architecture which revolutionized image classification. They employed data augmentation in the experiments to increase the dataset size. This is done in a multi-step approach by randomly cropping 224x224 patches from the original photos, flipping them horizontally, then applying PCA color augmentation to change the intensity of the RGB channels. The authors claim that data augmentation helped reduce overfitting during training a DNN as well as the model's error rate by more than 1%.

Textual Data Augmentation. Compared to vision, data augmentation in language has barely been studied. There have only been a few attempts to employ textual augmentation to help with classification problems. Wei and Zou (2019) provide a thorough study on the use of text editing techniques for NLP data augmentation and could demonstrate improvement in text classification. However, Tang, Ma, Zhang, Wu, and Yang (2020) show that those techniques can adversely impact the performance of a VQA model. Other works generate new data by paraphrasing (X. Zhang & LeCun, 2015) or introducing noise to text data (Bittlingmayer, 2018; Goodfellow, Bengio, & Courville, 2016). In modern NLP, studies widely finetune pretrained language models to automatically generate noisy or paraphrased data (K. Krishna, Wieting, & Iyyer, 2020; S. Wang, Thompson, & Iyyer, 2021).

VQA Data Augmentation Data augmentation for VQA is covered in fewer works, *e.g.,* (Kafle et al., 2017; Ray et al., 2019; Bitton, Stanovsky, Schwartz, & Elhadad, 2021). Kafle

and Kanan (2017) are the first to use semantic annotations on images to create new questions. Ray et al. (2019) leverages knowledge in the Visual Genome dataset (R. Krishna et al., 2017) to create QA pairs that quantitatively evaluate the consistency of a VQA model. The idea is that, if a model answers *"red"* to *"what color is the ball?"*, it should answer *"yes"* if asked *"is the ball red?"* to correctly preserve the notions of entailment. They automatically create a set of logically consistent QA pairs from a source QA pair and also collect a human-annotated set of consistent QA pairs based on common-sense *e.g., "is the ball the same color as a tomato?"*. Mirzaee, Rajaby Faghihi, Ning, and Kordjamshidi (2021) automatically generate synthetic complex questions and answers based on NLVR dataset (Suhr, Lewis, Yeh, & Artzi, 2017) focusing on spatial reasoning. For this, similar to our approach in this chapter, they leverage scene structure of images. However, to infer the answers to the synthetic questions, they design spacial reasoning rules and novel CFGs.

M. Shah, Chen, Rohrbach, and Parikh (2019) presents a cyclic-consistent training strategy in which the model is trained to predict a consistent answer for a source question and its rephrasing version. To enhance the model's robustness against semantic visual changes, their method uses a GAN-based re-synthesis methodology to automatically eliminate items. Agarwal, Shetty, and Fritz (2020) use data augmentation to enhance the model's robustness against semantic visual modifications. They employ a GAN-based re-synthesis approach to automatically eliminate the objects.

3.2.2 Scene Knowledge in VQA

Scene knowledge describes what exists in an image in terms of the entities, relations and actions. This information can be expressed in different forms with varying levels of details and granularity. It can be simply represented in the form of a list that reports the objects available in the image, or a more sophisticated form such as a graph. More formally, a scene graph is a structured representation that clearly shows the objects, attributes, and relationships between the objects in the scene. Scene information specifically in the form of a scene graph has served in many applications such as image generation (Y. Li et al., 2019),

image and video captioning (Yang et al., 2019), image-text retrieval (S. Wang, Wang, Yao, Shan, & Chen, 2020), and VQA.

A VQA task requires the understanding of both questions and images, grounding the questions in the image, and finally reasoning and producing the answer. Provided by many current VQA datasets, scene knowledge has drawn researcher attention as a tool for facilitating image understanding as well as reasoning (F. Yu et al., 2021; C. Zhang, Chao, & Xuan, 2019; Damodaran et al., 2021). Teney et al. (2017) propose to build a scene graph using information provided in the dataset, as well as text graph over the questions and feed both graphs into a neural network for reasoning. Similarly, Hudson and Manning (2019b) construct a probabilistic scene graph and treat it as a state machine traversing its states to reason. Hildebrandt, Li, Koner, Tresp, and Günnemann (2020) developed a reinforcement agent that learns to generate paths by navigating over scene graphs leading to the answer.

Although many studies use scene graphs to answer questions, a few works attempt to exploit such information for generating questions. (Bitton et al., 2021) proposed to generate contrast questions for the GQA dataset. The contrast sets proposed in (Gardner et al., 2020) aim to perturb a small subset of the test instances in a meaningful way, typically changing the ground truth label in order to evaluate a model's true capabilities. More specifically, contrast sets challenge a model's decision boundaries in local views. For instance, a contrasting example for QA pair *"is there a fence near the puddle? Yes"* is automatically generated using the scene graph as *"is there a wall near the puddle? No"*. (Ray et al., 2019) consider scene graphs to create consistent QA pairs that can be derived based on simple notions of logic. For instance, they create a QA like *"is the sofa black? No"* according to the relational triplet of <*sofa, is, white*> derived from the scene graph and using a simple logic like *"a white thing is not black."* They focus mainly on attribute, existential and relational consistency.

Similar to the above work, we automatically generate the QA pairs using scene knowledge, but in contrast, our proposed method does not rely on the relationships in a scene graph or any logic, rather it uses the attributes of the objects in images which can be easily

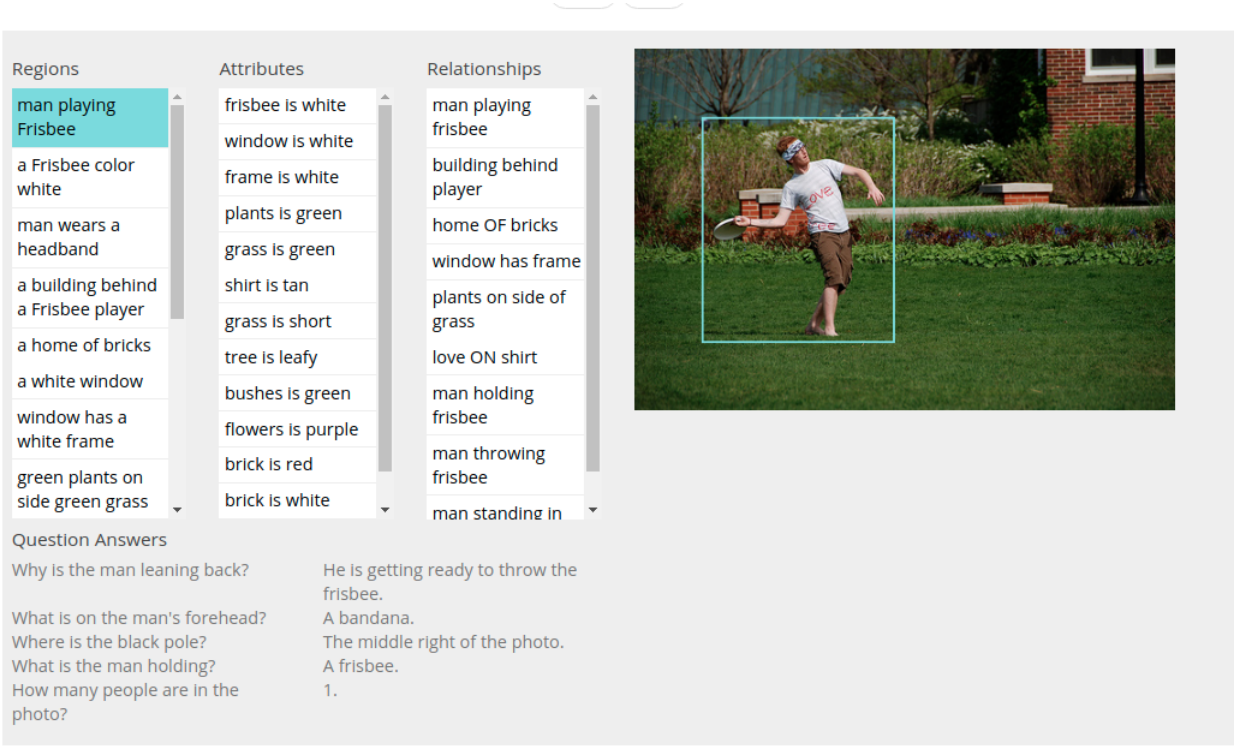


Figure 3.1: An example of scene information from Visual Gnome dataset.

generated by an object detection model.

3.2.3 VQA Datasets with Scene Knowledge

many of the VQA datasets introduced in Section2.3 include the scene information of the images. Visual Gnome (R. Krishna et al., 2017) includes the objects' region, their attributes and relationships. Figure 3.1 shows an example of the Visual Gnome dataset. The VQA dataset (Antol et al., 2015) provides scene information for both real and abstract images. CLEVR is a synthetic dataset that also contains detailed scene information in terms of the objects' spatial coordinates, attributes, and relations. Figure 3.4 depicts a sample of CLEVR scenes. GQA was introduced to offer clear and sophisticated scene graphs as the authors believe that scene graphs can be used to link the symbolic reasoning in classic AI to current deep neural network approaches. There are also other datasets (P. Zhang, Goyal, Summers-Stay, Batra, & Parikh, 2016) providing detailed information from the image scene which we

have not mentioned for the sake of brevity.

3.3 Current VQA Model Performance in Low Data Setting

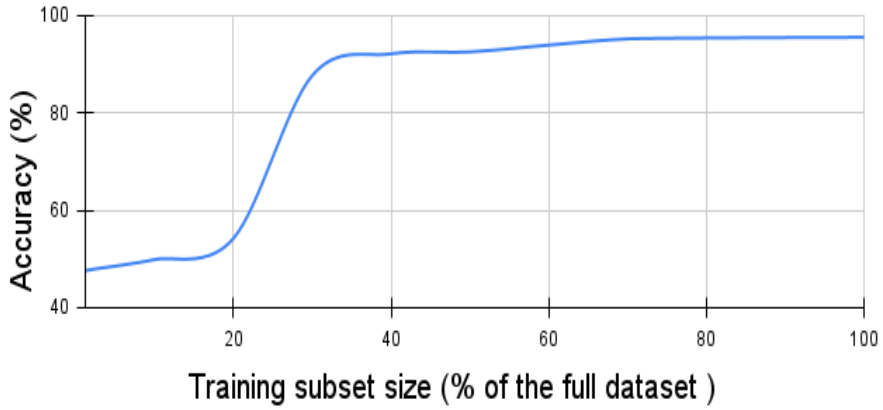


Figure 3.2: Accuracy of vanilla training of the *execution engine* on CLEVR `val` when trained on different sized subsets from CLEVR `train` set. Note that these experiments do not use the *program generator* . Instead ground truth programs are used as the input to the *execution engine*.

In a VQA task, a model receives as input a pair $(\mathbf{x}, q)$ of image $\mathbf{x}$ and a question q about the image. The model learns to select an answer $a \in \mathcal{A}$ to the questions from a set $\mathcal{A}$ of possible answers. We use the VQA model (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017) to test its performance when decreasing the size of the training set. This model includes two main components: a *program generator* $\mathcal{G}$ and an *execution engine* $\mathcal{E}$. The *program generator* predicts a program p to address a question q. The *execution engine* combines the modules according to the program, and executes the obtained network on the image to produce an answer. See Section 2.2.4 for more details about the model.

Johnson, Hariharan, van der Maaten, Hoffman, et al. (2017) train the model using a semi-supervised learning approach. They demonstrate that the *program generator* can produce acceptable programs while training on only a small fraction of possible programs ($\leq 4\%$). To evaluate $\mathcal{E}$'s performance in a low data regime, we conducted a set of experiments with different sized training sets. In these experiments, we train the model with randomly

selected subsets of CLVER `train` set in a vanilla supervised manner. We sample the training subsets with different percentages of the full training set, *e.g.,* 5%, 20% and 70%. Note that we use ground truth programs in addition to images as the inputs to $\mathcal{E}$ in all experiments in this thesis.

Figure 3.2 shows the best accuracy of the experiments on CLEVR's `val` set while the *execution engine* is trained on the subsets of the CLEVR's train set. The accuracy of the full training set (100%) is 95.55, close to the result reported in the original paper. Moving left, the accuracy steadily decreases as the training sets become smaller. However, the performance drastically drops where the training set size becomes lower than 30% so that the accuracy is 87.70% for the 30% subset and 54.24% for the 20% subset. This may be explained as the point that the information that the training data provides are not sufficient to tune the model weights regarding the model size. Generally speaking, the results verify *execution engine*'s poor performance on the small sized training subsets.

3.4 Our Proposed Data Augmentation Approach

This section explains our method for automatically generating simple questions using superficial information from the image scene. We use only the basic attributes of the objects present in an image including size, color, material, and shape; and disregard object's coordination and spatial relationship between objects.

3.4.1 The Notion of Simplicity of Questions

The first objective of this work is to generate simple questions. As discussed earlier, simplicity is an abstract and relative concept that can be defined in different ways. One may view it from the linguistic point and incorporate lexical and semantic criteria such as the length of the questions, while others may interpret simplicity from a different angle according to their goals. In order to focus on simple questions, we inevitably must formulate simplicity as a measurable criterion. Based on the general focus of this work which is reasoning us-

ing compositionality, we translate simplicity as the number of reasoning steps required to answer a question. In this sense, a question with a longer reasoning chain is considered a harder question relative to other questions with a shorter chain.

Since we aim to generate simple questions, we first specify the characteristics of the simplest possible question that can be created. According to the above definition, the simplest question means the one with the shortest reasoning chain. One of the most important things in comprehending questions is identifying the target object. In many cases, an object is detectable through its spatial relationship with other objects, *e.g., "the sphere behind the green box"* in Figure 3.3. Such cases involve locating multiple objects and, as a result, the reasoning chain would be longer. As a result, demand for finding a single object in the image can be taken as the first characteristic of the simplest questions.

The second reason for long reasoning chain even if it detects a single object, is a long referring expression to the target. The referring expression is the phrase by which we can uniquely locate the object in the image. Every word in a referring expression will be translated to a filtering step in the reasoning chain. For instance, the expression *"red sphere"* is converted to *"1) filtering the red colors, 2) filtering the sphere shapes on top of the result of the first step"*. To keep a question as simple as possible, the referring expression also needs to be short.

3.4.2 The Notion of Ambiguity of Questions

The objective here is to generate the questions automatically, *i.e.,* without human effort. Generally speaking, human intervention in question generation may be required for two main purposes: either supervising the quality of generated questions or producing answers. Question quality is a broad concept that embraces meaningfulness, and unambiguity in addition to linguistic factors such as grammar and fluency. Every work may narrowly define question quality according to its task and aims.

In our VQA setting, question quality narrows down to unambiguity. Since our method

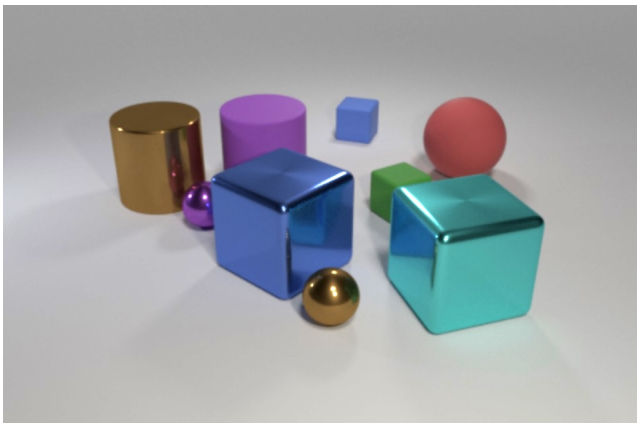


Figure 3.3: The question *"what color is the object behind the blue cube?"* is ambiguous since there are two blue cubes in the image. To create an unambiguous question, the target object must be uniquely identifiable by the referring expression in the question. A unique attribute combination ensures that the expression refers to a unique object such as *"red"* or *"red sphere"* and that both refer to the red sphere in the top right corner.

is a template-based question generation approach, other aspects of question quality are automatically considered in the templates (see Section 3.4.4 for more details). Therefore, human intervention may be needed to filter out the ambiguous questions. According to our definition, ***ambiguity*** occurs when the target object cannot be uniquely identified by the attributes stated in the question. For example, in Figure 3.3 the question *"what color is the object behind the blue cube?"* is ambiguous because firstly the expression *"the blue cube"* can refer to either the large shiny blue cube or the small matt cube at the back. Secondly, if supposedly *"the big shiny blue cube"* was the only *"the blue cube"* in the image, the questions would still be ambiguous because there is more than one object behind it whose color needs to be captured as the answer. The reader should note that the task is a QA task, not a dialog in which the agent can ask for more clarification by responding, *i.e., "which object behind the blue cube?"*. On the other hand, an unambiguous question can be easily answered by locating the target object and looking up the required attribute among its attributes from the scene knowledge.

As a result of the discussion in the above paragraphs, the concern of eliminating human effort is reduced to posing unambiguous questions. For this purpose, we introduce **unique attribute combinations** to guarantee that answering the generated questions does

not require any human effort by ensuring that they refer to a unique object in images. The following section explains unique attribute combinations in detail whereafter we elaborate on the templates used for generating questions.

3.4.3 Unique attribute combinations

A **unique attribute combination** is defined as a referring expression consisting of a set of attributes that can uniquely identify an object in an image. For instance, a unique attribute combination for the red sphere at the top right corner of Figure 3.3 can be *"red"* because it is the only red object in the image and it can be uniquely identified if one refers to it as *"the red object"*. Although it is the shortest unique attribute combination for the red sphere, there are other unique combinations for this object including *"large sphere"*, *"matt sphere"*, and *"large red sphere"*. As we are interested in short questions, we only produce short attribute combinations of the lengths 1 and 2.

Now, we explain how we create unique attribute combinations. Let uac_l be the unique attribute combination of length l. We begin by creating combinations of length 1 for all objects in the image, $uac_{l=1}$ by comparing the values of the similar attributes in all objects of the image *e.g.,* the colors or shapes of the objects; and then record the attribute values that appear in only one object such as *"red"* in Figure 3.3. This process is shown in Algorithm 1 line 4–9.

A general method to create unique combinations with length $l > 1$ is generating all possible combinations of length l for all objects in an image. Then we compare the combinations and remove those that refer to more than one object by eliminating the repeated combinations. However, as illustrated in Algorithm 1 line 10–19, we modify this method to reduce computational time. We join the values in $uac_{l=1}$ to other attribute values of the corresponding object, *e.g.,* we attach *"red"* as a uac_1 to *"sphere"* to create *"red sphere"* or *"red large"* where $l = 2$. This strategy produces longer unique combinations at a fast rate but at the cost of missing a number of possibilities. Once the uacs are generated, it is time to move on to the next step using templates for question generation. Let us now introduce

Algorithm 1 Creating Unique Attribute Combinations

I: Image list
A: List of attributes *i.e.,* {color, shape, size, material}
procedure CREATE-UAC(I)
    #Obtaining *uac* where l==1
    for $im \in I$ **do**
        for $obj \in im$ **do**
            for $attrib \in$ A **do**
                if $obj.attrib.value$ is unique **then**
                    $uac_1 \leftarrow obj.attrib.value$
    #Obtaining *uac* where l≥1
    if $l \geq 1$ **then**
        for $im \in I$ **do**
            for $obj \in im$ **do**
                for $u1 \in im.obj.uac_1$ **do**
                    #Choosing *l-1* attribute
                    from all *obj* attributes
                    for *attrib-comb* $\in \binom{l-1}{obj.attibutes}$ **do**
                        if $u1 \notin$ *attrib-comb* **then**
                            $uac_l \leftarrow$ Concat($u1$, *attrib-comb*)

our templates and the generation details.

3.4.4 Generating Template-based Questions

We synthesize two types of questions from scene knowledge: *query-attribute* and *existential* questions. As their names suggest, *query-attribute* questions begin with *"what"* and ask about the value of an attribute in the target object, *e.g., "the color of the cube"* or *"the shape of the big object"* while *existential* questions start with *"is"* and ask whether the target object is present in the image. Table 3.1 shows the templates along with corresponding examples. [attribute$_q$] indicates a placeholder which must be replaced with the queried attribute name that can be any of the attributes from the attribute set *i.e.,* $\{size, color, material, shape\}$; while [attribute$_{uac}$] is the attribute name being presented in the uac. [uac] indicates a uac of the desired object.

The phrase inside the braces creates a sequence of filters with the same length as uac when being repeated for each attribute value in uac and combined together. In fact, the

Query-attribute Type		
	Template	**Example**
Question	What [attribute$_q$] is the [uac] object?	What size is the red object?
Program	$scene \rightarrow \left\{ filter\text{–}[attribute_{uac^i}][[uac^i]] \rightarrow \right\}_{i=1}^{l}$ $query\text{–}[attribute_q]$	$scene \rightarrow filter\text{–}\texttt{color}[red] \rightarrow query\text{–}size$

Existential Type		
	Template	**Example**
Question	Is there a [uac] object?	Is there a red object?
Program	$scene \rightarrow \left\{ filter\text{–}[attribute_{uac^i}][[uac^i]] \rightarrow \right\}_{i=1}^{l} exist$	$scene \rightarrow filter\text{–}\texttt{color}[red] \rightarrow exist$

Table 3.1: The question and program templates for two types of questions is used in this work along with an example for each. The top table shows the the templates and example for *query-attribute* type where the question begins with *what* and asks about an attribute value of the target object. The lower table details the *existential* type where the questions asks if the target object presents in the image.

examples provided in the table use a uac with $l = 1$, *i.e., "red"*; however in case the uac is longer, *e.g., "red sphere"* then identification process of the target object is performed in a multi-step filtering; for instance firstly filtering the *red* `color` and secondly filtering the *sphere* `shape`. This process is formalized as the following phrase: $filter\text{–}\texttt{color}[red] \rightarrow filter\text{–}\texttt{shape}[sphere]$. It is also noteworthy to mention that we let it be possible that a question asks about one of the attributes that constitute the uac, *e.g., "what color is the red sphere?"*, although the answer is very obvious and already appears in the question itself. In other words, the attribute that is chosen to fill the [attribute] placeholder may be one of the attributes whose value is included in the [uac].

The *existential* template can be used to produce *yes*/ *no* questions. Replacing a target object's uac with [uac] in the template can easily result in a *yes* answer while for a question with an answer *no* an invalid attribute should be joined to uac. Any attribute value which does not match the target object is considered as an invalid attribute for it. For instance, assume *"is there a red sphere in the image?"* as a positive-answered question, then *"is*

*there a red **small** sphere?"* makes a negative-answered question because ***small*** is an invalid attribute for the target object and hence, if attached to uac creates a referring expression with no visual equivalence in the image. The invalid attribute must be selected among the attributes that don't already contribute in the uac, otherwise the referring expression becomes ambiguous.

As may be clear, to select an answer to a generated question, we simply determine the target object and draw its scene information, then pick the value of the requested attribute. In the case of the *yes/no* answers, the answer is determined by the questions generation process based on whether an invalid attribute is used or not. Let us emphasize again that utilizing uac ensures that the question refers to a unique object and as the result, the answer will be distinct so that it can be automatically selected or generated.

3.5 Experiments

To simulate a low-data scenario, we select four small subsets from the training set with different sizes denoted as a percentage of the full dataset. We refer to these subsets as s-CLEVR$_x$ where $x \in \{5, 10, 20, 30\}$ indicates the size of the subset, *e.g.,* s-CLEVR$_{20}$ is the subset chosen to be as small as 20% of the full training set with $\sim 140k$ *(image, question, answer)* tuples. We use the scene information of the images from s-CLEVR$_x$ denoted as I_x to generate augmented questions.

We conducted our experiments in two stages: ***data augmentation*** and ***training***. The first stage includes extracting uacs and generating QA pairs using the proposed data augmentation method. The uacs are extracted from the scene information of a selected set of images (see details in Section 2.3). To emphasise simplicity, we only use uac_1 for generating *query-attribute* questions and uac_2 for *existential* questions. uac_2, as described in the preceding section, are built upon uac_1s. For each uac_1, we generate *existential* questions with a probability of $p = 0.25$ where the chance of *yes* answers equals *no* ones. We refer to the generated QA set as Aug^{simple} .

Answer Category	# Classes	Answer Classes
Size	2	Large, Small
Material	2	Metal (Shiny) , Rubber (Matt)
Shape	3	Cube, Cylinder, Sphere
Color	8	Brown, Gray, Red, Yellow, Purple, Cyan, Green, Blue
Yes/no	2	Yes, No
Numbers	11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Total	28	

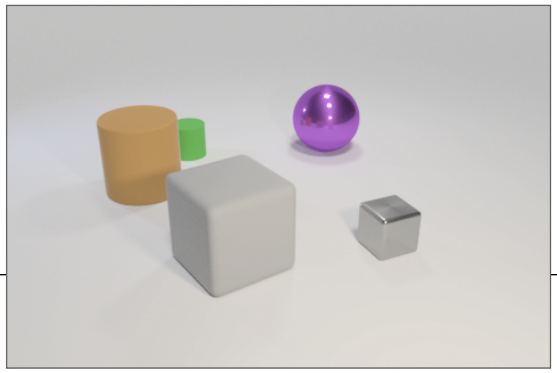
Table 3.2: The classes of answers in the CLEVR dataset.

In the second stage, we add the augmented questions to the training set to create an augmented training set *i.e.,* s-CLEVR$_x$+Aug^{simple} . We use the *execution engine* $\mathcal{E}$ described in Section 3.3 as the model in the experiments. We train $\mathcal{E}$ on the augmented training set from scratch and evaluate on `val`.

In addition to the questions, the model requires the image features to produce the answer. The image features are the output of $conv4$ of ResNet-101 (K. He et al., 2016) pretrained on ImageNet (Deng et al., 2009). We then test the model on the validation set and compare the results with two baselines. The following sections explain the setup details for both stages.

3.5.1 Setup

Dataset Although the data augmentation method is generic and applicable to many VQA datasets, for generating augmented questions, we make use of the scene information from the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017). This information is available in the form of objects' attributes and their spatial relationships as seen in Figure 3.4. Our proposed method aims to use the superficial attributes of objects, thus we only consider color, size, material and shape from all the information provided about the scene of an image. We also use the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017) for training. It provides a training set with $70k$ images, $700k$ *(image, question, answer)* tuples. The answers comes from 28 classes including 8 colors, 2 sizes, 3 shapes, 2 materials, 11 numbers as well as yes and no all listed in Table 3.2. To simulate



Scene Information

image_index: 0,

objects: [

{"color": "brown", "size": "large", "rotation": 178.92387258999463, "shape": "cylinder",
"3d_coords": [-1.4937210083007812, -1.9936031103134155, 0.699999988079071],
"material": "rubber", "pixel_coords": [119, 131, 10.801968574523926]},

{"color": "gray", "size": "large", "rotation": 243.405459279722, "shape": "cube",
"3d_coords": [1.555708646774292, -2.104736566543579, 0.699999988079071],
"material": "rubber", "pixel_coords": [198, 190, 8.60103988647461]},

{"color": "green", "size": "small", "rotation": 230.45235024165092, "shape": "cylinder",
"3d_coords": [-2.342184543609619, -0.5205014944076538, 0.3499999940395355],
"material": "rubber", "pixel_coords": [161, 118, 12.372727394104004]},

{"color": "purple", "size": "large", "rotation": 31.654351858799153, "shape": "sphere",
"3d_coords": [-0.8073106408119202, 1.914123773574829, 0.699999988079071],
"material": "metal", "pixel_coords": [282, 100, 12.495001792907715]},

{"color": "gray", "size": "small", "rotation": 42.183287560575, "shape": "cube",
"3d_coords": [2.6763813495635986, 0.03453871235251427, 0.3499999940395355],
"material": "metal", "pixel_coords": [337, 195, 9.161211967468262]}

],

relationships: {

"right": [[1, 2, 3, 4], [3, 4], [1, 3, 4], [4], []],
"behind": [[2, 3], [0, 2, 3, 4], [3], [], [0, 2, 3]],
"front": [[1, 4], [], [0, 1, 4], [0, 1, 2, 4], [1]],
"left": [[], [0, 2], [0], [0, 1, 2], [0, 1, 2, 3]]

},

image_filename: "CLEVR_val_000000.png",

split: "val",

directions: {

"right": [0.6563112735748291, 0.7544902563095093, -0.0],
"behind": [-0.754490315914154, 0.6563112735748291, 0.0],
"above": [0.0, 0.0, 1.0],
"below": [-0.0, -0.0, -1.0],
"left": [-0.6563112735748291, -0.7544902563095093, 0.0],
"front": [0.754490315914154, -0.6563112735748291, -0.0]}

}

Figure 3.4: An example of scene information from CLEVR `val` set. Among all provided detail, we use only the `size, color, material` and `shape` attributes of the objects for data augmentation.

a low-data scenario, we select a small subset from the training set referred as s-CLEVR$_{I20}$. The subset is chosen to be as small as 20% of the full training set with $140k$ *(image, question, answer)* tuples. We use the scene information of the images from s-CLEVR$_{I20}$ for generating augmented questions.

Figure 3.5 confirms that the CLEVR dataset mainly includes the complex questions and only a small proportion of the training set consists of simple questions. It shows the length distribution of the questions and programs in the CLEVR `train` set. 91.42% of the questions are longer than 10 in length. The longer a question is, the more complex it is likely to be because it may be involved in more objects and their relationships. A program shows the reasoning chain for predicting the answer to a questions. As such, length in programs indicates the number of steps in the reasoning chain. As seen in the right diagram of Figure 3.5, the majority of programs indicate a long reasoning chain, *e.g.,* about 85.80% of the program lengths are higher than 6. Note that a question is normally translated to a program with a shorter length as not all of the question tokens can be translated to a reasoning step.

Avoiding Reproducing Questions A concern in our setting when working with a synthetic dataset like CLEVR is that by automatically generating questions we may end up replicating the questions in the dataset. Since CLEVR questions have also been produced using templates, we should ensure that we did not reproduce the questions of the other $(100-x)\%$ of the dataset when augmenting s-CLEVR$_x$ with automatically-generated questions. Otherwise, comparing the outcomes when using augmented data to that of the original dataset is not fair.

To address this concern, first we randomly select a subset of $x\%$ of the CLEVR images, I_x. then collect all the questions referring to those images, and use them as the $x\%$ subset of the dataset, s-CLEVR$_x$. In CLEVR the rate of the question referring to a specific image is almost fixed to 10, so the set of questions about I_x roughly equals $x\%$ of the full dataset. We only make use of the scene information of I_x for data augmentation.

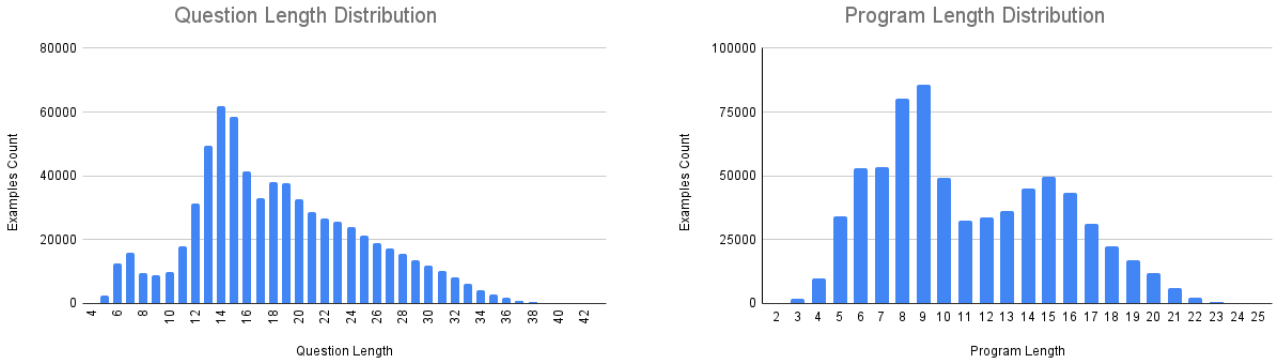


Figure 3.5: The length distribution of questions and programs in the CLEVR `train` set. As seen in the left diagram, the majority of the questions are of a length above 10 which means the dataset mostly consists of complex questions. Since lengths of programs indicate the number of reasoning steps, the right diagram shows only a small number of examples can be considered as simple with few reasoning steps, *e.g.,* less that six steps.

Baselines We use the *execution engine* $\mathcal{E}$, to assess the proposed data augmentation impact on the VQA's performance in comparison to two baselines. As a baseline, the model is trained on s-CLEVR$_{I20}$ which includes only the subset of CLEVR questions about the images in I_{20} denoted as s-CLEVR$_{I20}$. This subset is assumed to mainly contain complex questions since it is selected from the VQA dataset CLEVR. The second baseline will be obtained by training the model on only generated questions denoted as Aug^{simple} which only includes simple and short questions. We do not compare with the state of the art, because the goal of our research is to study VQA in a low-data regime, and to the best of our knowledge, there is no other work that conducts similar research. Thus, we focus on improving the performance of our baseline models.

3.5.2 Results and Discussion

Data Augmentation Statistics. Table 3.3 shows some statistics of the data augmentation experiment. As seen in the upper part of the table, from $\sim 70k$ images in the CLEVR dataset of which we randomly select $x\%$ of images for each subset. In other words, the size of Ix for each subset in our experiments is as shown in the table, *e.g.,* in the case of $I20$ it is 13,917. Then, all uac_1 were extracted from the image scenes which can be further divided

	# images	# unique colors	# unique shapes	#unique materials	# unique size
5%	3,480	10,198	2,604	862	835
10%	6,961	20,288	5,204	1,727	1,673
20%	13,917	40,520	10,331	3,490	3,365
30%	20,859	60,927	15,575	5,179	5,145

	Exist Questions	**Attribute Questions**	**All Questions**
Subcategories	yes, no	color, shape, size, material	-
Questions length	6	6	6
Program Length	4	4	4
Count in 5%	10,817 (16%)	57,996 (84%)	68,813
Count in 10%	21,715 (16%)	115,568 (84%)	137,283
Count in 20%	43,375 (16%)	230,824 (84%)	274,199
Count in 30%	65399 (16%)	347,304 (84%)	412,703

Table 3.3: The statistics of generated questions for each training subset. **Top:** the number of images that are chosen to generate augmented questions and the number of unique attributes, *i.e.,* uac_1, that are extracted from those images. **Bottom:** The length and number of generated questions per question type and in total.

into colors, shapes, materials, and sizes. The statistics neatly correlate with the number of values of each attribute. For instance in CLEVR dataset, the attribute `color` includes eighth values of *blue, cyan, green, gray, yellow, brown, red,* and *purple*. Thus it is more likely for a certain object in an image to have a color different from other objects in the image. As a result, the number of unique colors are presented in the images $\in Ix$ are largely higher than other attributes. On the other hand, `size` has only two values of *large* and *small*. So an object is less probable to be the only `small` or `large` object in the image particularly in the current VQA datasets in which the images are relatively rich in terms of the number of objects.

The lower part of Table 3.3 reports the statistics of the generated questions from the selected subsets. For instance, from the extracted uac_1 of I_{20}, we generated 274,199 questions in total of which 43,375 are of *existential* type while the remaining are from *query-attribute* type. The number of *existential* questions with *yes* and *no* answers is almost equal contain-

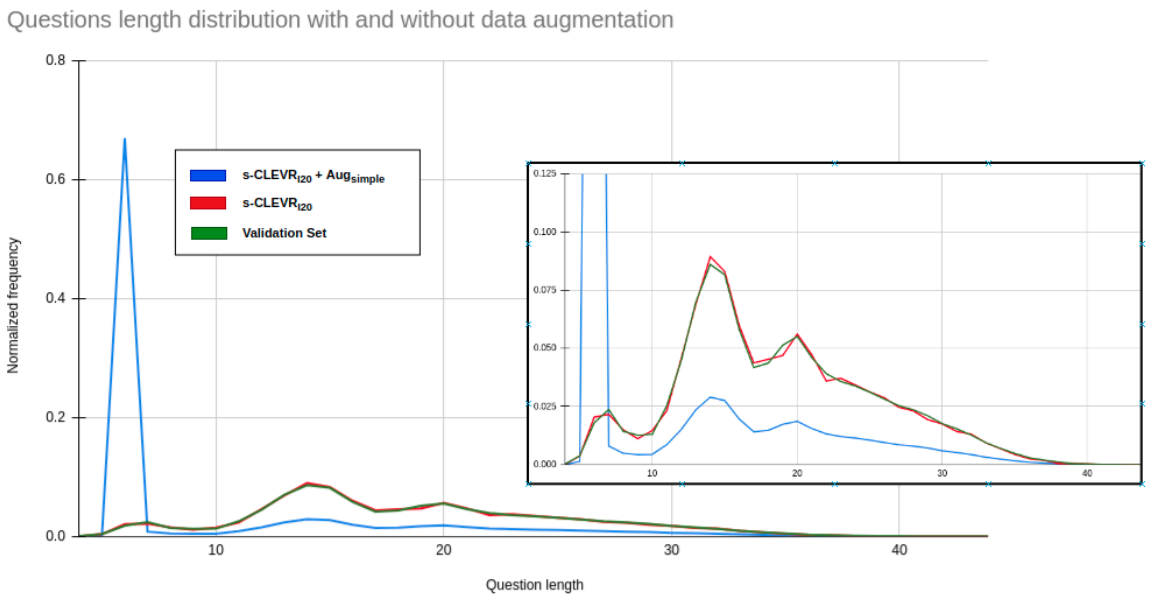


Figure 3.6: Comparison of the distributions of questions length with and without data augmentation. s-CLEVR$_{20}$ and `val` set are sampled from the dataset original distribution while s-CLEVR$_{20}$+Aug^{simple} shows the distribution of the augmented training set.

ing 21,665 and 21,710 questions respectively. Considering that the size of training sets are $35k$ for s-CLEVR$_5$, $70k$ for s-CLEVR$_{10}$, $140k$ for s-CLEVR$_{20}$, and $210k$ for s-CLEVR$_{30}$, we can say that the number of generated questions is roughly twice the original sets size. The rest of 203,824 questions ask about four attributes of `color`, `shape`, `size`, and `material` each 57,706 questions. As for every uac_1, four questions corresponding to the four attributes are generated, hence the count of the questions for all attributes is the same. We use only uac_1 for *what-[attribute]* questions and uac_2 for *existential* questions, according to templates in Table 3.1, the question's length will be 6 and the length of their corresponding programs will be 4.

How does the augmentation change the training set distributions? Figure 3.6 compares the distribution of questions length in s-CLEVR$_{20}$ and Aug^{simple} as well as the `valid` set. From the close-up shot of the curves, we can see that the distribution of s-CLEVR$_{20}$ and `valid` set is very similar. This is due to the fact that the distribution of the `train` and `valid` set are similar in the CLEVR dataset and the s-CLEVR$_{20}$ is a random subset of

`train` set which means they share the same distribution. As seen, augmenting s-CLEVR$_{20}$ with Aug$_{20}^{simple}$ dramatically changes the distribution of questions in terms of length and type. One may hypothesise that such distribution dissimilarity will adversely affect the performance. However, the results demonstrate that in our setting it significantly enhances the model performance in predicting answers.

Training. Table 3.4 depicts the results of model validation when training on (1) generated simple examples, Aug$_{x}^{simple}$, (2) a subset of complex questions, s-CLEVR$_x$, and (3) the augmented sets, s-CLEVR$_x$+Aug$_{x}^{simple}$. As seen our data augmentation method, *i.e.,* training on s-CLEVR+Aug^{simple} outperforms the baselines. The best performance of our approach is achieved by training on and s-CLEVR$_{10}$+Aug$_{10}^{simple}$ and s-CLEVR$_{20}$+Aug$_{20}^{simple}$ with 34 scores accuracy increase. This large improvement confirms our initial hypothesis about lacking the basic concepts in a small training set with only complex questions. Adding simple questions to the training set helps the model to learn the basic concepts as the basis of complex ones. Then the model learns how to combine those simple concepts by training on a small set of complex questions. These two skills together make the model effectively generalize on unseen examples and it is the reason for producing 34% more accurate answers on `val` set in the cases of 10% and 20% subsets.

This analysis also explains the results of other subsets. In the case of s-CLEVR$_5$ the set of complex questions is not large enough for the model to learn all variations of combining the basic concepts. So the improvement is less than the next two larger subset

Training Set	5%	10%	20%	30%
Aug^{simple}	31.69	30.18	30.14	30.17
s-CLEVR	46.91	49.90	54.24	87.70
s-CLEVR$_+$Aug^{simple}	69.23	83.06	87.81	91.26

Table 3.4: The accuracy on CLEVR `val` set when training on different sets.

s-CLEVR$_{30}$+Aug$_{30}^{simple}$ increase only 5% improvement in the accuracy. It is because s-CLEVR$_{30}$ contains almost 200k of complex examples. The numbers of complex examples

are quite large so that the model can implicitly infer many basic concepts by repeatedly visiting close examples. In this case, explicitly introducing basic concepts cannot largely enhance the performance.

It is noteworthy that the poor results on Aug^{simple} sets are as expected since the training sets are not challenging enough for a model to learn how to deal with complex questions. Therefore many predicted answers on `val` set are incorrect.

How does the proposed augmentation approach generalize on realistic detasets? We investigated to what extent the method proposed in this chapter is applicable to real world images. For this, we picked a random data example from GQA dataset (Hudson & Manning, 2019a) shown in Figure 3.7 and generated some simple questions. Similar to many other VQA datasets, GQA contains scene graph information. As you can see, the figure contains the name of the object along with their attributes, however, unlike CLEVR the type of attributes is not clear. Fortunately, the approach proposed in this chapter is general enough to be used for such datasets as well, although depending on the characteristic of the dataset small modifications in Algorithm 1 might be required. For instance, we can simply put the extracted uacs in Existential Type template in Table 3.1. Among the yellow-highlighted attributes in the following example, some of the uacs include *"dark pants"*, *"long-sleeve shirts"*, *"metal floor"*, and *"green tree"*. Using Existential Type template the simple questions such as *"is there a metal floor?"* can be generated.

There is a limitation in using Query-attribute template in Table 3.1 as the dataset does not include the type of attributes. A small modification can fix this problem. It is suggested that hand-crafted lists of common attributes such as color and material are used for looking up the type of attributes. Having done so, we can use the Query-attribute template to generate the questions such as *"What color is the tree?"* or *"what material is the floor?"* To summarize, generally speaking, the approach is applicable to synthetic and real-world VQA datasets, however, the main limitation is that Algorithm 1 should be modified depending on the structure of datasets and the information they provided.

An example of scene Info from GQA dataset


```
"2410049": {"width": 375, "objects": {
"227269": {"name": "head", "h": 87, "relations": [{"object": "227273", "name": "to the right of"}, {"object": "227260",
      "name": "to the right of"}], "w": 64, "attributes": [], "y": 59, "x": 196},
 "227260": {"name": "door", "h": 354, "relations": [{"object": "227263", "name": "to the left of"}, {"object": "227257",
      "name": "to the left of"}, {"object": "227274", "name": "to the left of"}, {"object": "227264", "name": "to the left of"},
      {"object": "227258", "name": "to the left of"}, {"object": "227262", "name": "to the left of"}, {"object": "227276",
      "name": "to the left of"}, {"object": "227256", "name": "to the left of"}, {"object": "227269", "name": "to the left of"},
      {"object": "227270", "name": "to the left of"}], "w": 105, "attributes": ["open"], "y": 0, "x": 42},
"227256": {"name": "man", "h": 278, "relations": [{"object": "227257", "name": "reading"}, {"object": "227263", "name":
      "wearing"}, {"object": "227258", "name": "wearing"}, {"object": "227262", "name": "wearing"}, {"object": "227257",
      "name": "holding"}, {"object": "227261", "name": "on"}, {"object": "227260", "name": "to the right of"}, {"object":
      "227261", "name": "in"}, {"object": "227273", "name": "to the right of"}, {"object": "227267", "name": "to the right
      of"}, {"object": "227264", "name": "wearing"}], "w": 160,  "attributes": ["crouched"], "y": 57, "x": 133},
 "227257": {"name": "book", "h": 40, "relations": [{"object": "227260", "name": "to the right of"}, {"object": "227267",
      "name": "to the right of"}], "w": 57, "attributes": ["open"], "y": 164, "x": 133},
"227263": {"name": "sandals", "h": 34, "relations": [{"object": "227260", "name": "to the right of"}], "w": 80, "attributes":
      [], "y": 294, "x": 175}, "227267": {"name": "lock", "h": 67, "relations": [{"object": "227270", "name": "to the left of"},
      {"object": "227262", "name": "to the left of"}, {"object": "227256", "name": "to the left of"}, {"object": "227257",
      "name": "to the left of"}], "w": 77, "attributes": [], "y": 152, "x": 43},
 "227258": {"name": "eye glasses", "h": 21, "relations": [{"object": "227260", "name": "to the right of"}, {"object":
      "227273", "name": "to the right of"}], "w": 26, "attributes": ["black"], "y": 108, "x": 198},
"227259": {"name": "floor", "h": 241, "relations": [], "w": 289, "attributes": ["metal", "gray"], "y": 258, "x": 27},
"227270": {"name": "hands", "h": 22, "relations": [{"object": "227260", "name": "to the right of"}, {"object": "227262",
      "name": "to the left of"}, {"object": "227267", "name": "to the right of"}], "w": 19, "attributes": [], "y": 179, "x": 141},
"227264": {"name": "pants", "h": 91, "relations": [{"object": "227260", "name": "to the right of"}], "w": 132, "attributes":
      ["dark"], "y": 210, "x": 138},
 "227274": {"name": "seat", "h": 63, "relations": [{"object": "227260", "name": "to the right of"}], "w": 154, "attributes": [],
      "y": 283, "x": 151},
 "227262": {"name": "shirt", "h": 158, "relations": [{"object": "227267", "name": "to the right of"}, {"object": "227270",
      "name": "to the right of"}, {"object": "227260", "name": "to the right of"}], "w": 147, "attributes": ["brown", "striped",
      "long sleeved"], "y": 126, "x": 154},
"227273": {"name": "window", "h": 127, "relations": [{"object": "227256", "name": "to the left of"}, {"object": "227269",
      "name": "to the left of"}, {"object": "227276", "name": "to the left of"}, {"object": "227258", "name": "to the left of"}],
      "w": 41, "attributes": [], "y": 1, "x": 85}, "227261": {"name": "train car", "h": 492, "relations": [], "w": 366, "attributes":
      [], "y": 4, "x": 8},
 "227276": {"name": "trees", "h": 77, "relations": [{"object": "227260", "name": "to the right of"}, {"object": "227273",
      "name": "to the right of"}], "w": 61, "attributes": ["green"], "y": 8, "x": 217}}, "height": 500},
```


Figure 3.7: An example of scene information from GQA dataset.

Since the questions are artificially created, is it easier for the program generator to translate them compared to natural questions? While it is true that program generation may have some issues in a new domain, our assumption is that it remains an easier task than learning the neural module network. Thus our focus is on improving the learning of this network. Moreover, experimenting with a different dataset always has the risk of posing new challenges to a defined method and in our case, a real dataset may introduce a number of challenges that a synthetic dataset may not. From the program generation point of view, the vocabulary of the program languages is limited. So it may not be able to properly transfer all types of human-generated questions to corresponding programs that can affect reasoning quality. However, Andreas et al. (2016b) and Johnson, Hariharan, van der Maaten, Hoffman, et al. (2017) showed that such programs can interpret a wide range of human-created questions.

3.6 Summary

This chapter explores VQA in low data settings motivated by the low performance of VQA models in the absence of sufficient data. To improve the performance, we propose a data augmentation method aiming to explicitly inject certain inductive biases. The inductive biases are based on the inherent compositionality of the questions which allows a complex question to be decomposed into smaller parts. We utilize this feature to augment the training set with basic questions where the learning can occur easier. The proposed data augmentation approach relies solely on the existing training set without seeking help from any other data resource. The results show that our method outperforms the baseline in all cases by a large margin.

4 | Self-supervised Pretraining of Intermediate Visual Representations in Low-Data VQA

In the previous chapter, we highlighted a common VQA datasets feature that causes generalization problems in low data scenarios. VQA datasets consist mainly of complex questions to challenge models' understanding and reasoning ability. This feature does not normally cause any problem in the presence of large-scale datasets because the model can learn complex inductive biases by repeatedly seeing a lot of examples. However, in low data scenarios, understanding complexity from a limited number of examples is difficult. We proposed a data augmentation solution in Chapter 2 relying on the assumption that simple questions can facilitate the learning process by bridging the gap between the fundamental concepts and complex questions. This happens because of two advantages of simple questions: including basic concepts and being easy to understand. As a result, we supplement the training set with simple questions using our proposed data augmentation method.

So far, the reader should know that basic concepts, which are the building blocks of complex questions, must be understood before a model can grasp the entire question. In contrast to the previous chapter where we inject the basic concepts into the training set through simple questions, in this chapter we use the compositional feature of a question to learn its constituents. These two approaches are orthogonal and they can be employed as complementary approaches, together making the learning strategy stronger. This chapter

presents a self-supervised pretraining approach to low data VQA which uses the compositionality of questions to break them down and learn the components one at a time. This stage aims to prepare the model for learning complex questions during training. Our experiments show that the proposed pretraining approach significantly outperforms the baselines.

4.1 Introduction

VQA was introduced and has received much attention the last few years due to the remarkable success of deep vision and language models. Similar to other deep learning approaches, current VQA models require large-scale datasets. Training VQA models in low data settings is overshadowed by the excitement associated with large VQA datasets targeting a new task in the VQA area. However, the unavoidable fact is that not many areas can provide a large amount of data. A good example is medical tasks in which the amount of data for many diseases is limited (See (Z. Lin et al., 2022) for examples of VQA application in medical domain). This thesis focuses on low data VQA and devising solutions for the poor performance of current VQA models when not enough data is accessible.

To compensate for the lack of data, we hypothesize that pretraining basic knowledge helps the model to learn general and powerful biases for comprehending complex questions. Specifically, we are proposing to use questions' compositionality inherent in natural language. Natural language is known to be enormously productive, as it allows for a theoretically unlimited number of potential expressions. This productivity comes from the compositional nature of language. Compositionality, or the capacity to build larger linguistic statements by combining smaller components, is nearly unanimously agreed upon by linguists (Baroni, 2020). Language compositionality is typically defined by the focus on semantic in the sense that the meaning of a linguistic expression is determined by the meaning of its constituents and the rules employed to combine them. However, it is more beneficial to consider a more general notion of compositionality such as its syntactic aspect when studying generalization in neural networks. We use compositionality in its general sense which encompasses both semantic and syntactic aspects.

In this chapter, we break down the compositional structure of the questions into smaller chunks and the model learns the components before trying to understand the entire question. This idea intuitively makes sense because the distinctive human ability to generalize also allows learning small things and then using those to understand new complex things (Lake, Linzen, & Baroni, 2019), *e.g.,* assume a child who has learnt the concepts of red and ball. It is not surprising that she can effortlessly identify a red ball amongst the blue balls when first seeing the balls. Similarly, our approach suggests learning the basic components in a pretraining phase, then the model can be trained on the real training set with complex questions. In other words, instead of giving complex supervision on the composition, we are trying to provide cleaner and simpler supervision to the parts to help learn the whole.

For most deep learning applications, a recent trend suggests that well-tailored model pretraining methods (weakly-supervised, semi-supervised and self-supervised) can dramatically enhance performance on downstream tasks. This has been shown in NLP (Devlin et al., 2019; Radford et al., 2019), Speech Recognition (Schneider, Baevski, Collobert, & Auli, 2019; Riviere, Joulin, Mazaré, & Dupoux, 2020), and Computer Vision (Mahajan et al., 2018). The success of such models is based on two main factors: pretraining on massive datasets, and using the models with billions of parameters, *e.g.,* GPT-3, the language model in (Brown et al., 2020), has 175 B parameters pretrained on 300B words. Contrary to this trend, we consider the low data setting for the pretraining phase in addition to training. Therefore, in our experiments, both phases share the same training set. This is to emphasize the situation that a limited amount of data is accessible for a certain task. We also want to investigate how far we can push VQA model performance without using additional data.

The advantage of pretraining has mostly been demonstrated in the scope of datasets that are originally curated for supervised or weakly supervised learning. Not only are these datasets limited, they represent only a small portion of the actual distribution of Internet-scale data. A clever solution to this problem is self-supervised learning in which no labeled data is required. In other words, the learning algorithm self-labels the raw data on the fly using a self-made fork to support feature learning. The initial attempts of self-supervised

training on uncurated data in computer vision could use millions of images. This work uses a self-supervised technique to learn intermediate representations for sub-questions.

Splitting a question into smaller chunks is implicitly creating sub-questions that are easier to understand and answer. A complex question, is composed of several sub-questions, each question is asked based on the answer of the previous, *e.g., "what is the size of the red ball behind the box?"* consists of the following sub-questions: *"where is the box?"* $\rightarrow$ *"where is the area behind it?"* $\rightarrow$ *"where are the balls in that area?"* $\rightarrow$ *"which one is red?"* $\rightarrow$ *"what size is it?"*. The answers to each of these sub-questions are referred to as intermediate answers as they serve as stepping stones to the final answer.

If provided, the intermediate answers effectively facilitate learning to answer complex questions. However, collecting human-provided annotations is very costly. Instead, we propose to learn the visual representation of the intermediate answers using contrastive learning as one of the most powerful approaches in self-supervised learning. The main goal of contrastive learning is to develop representations that keep similar samples close together in the embedding space while dissimilar ones are distant. We employ the SimCLR algorithm (T. Chen, Kornblith, Norouzi, & Hinton, 2020), a contrastive learning framework, in our experiments to pretrain our model using images from an unlabeled training set. A modular VQA model is used in the experiments to further investigate the effect of our approach. This allows us to reflect the compositionality in the model architecture in addition to the input data. We investigate three versions of a self-supervised approach for pretraining modules including pretraining single modules, random sequence of modules, and increasing-length sampled sequence of modules. The model is then fine-tuned on the VQA task and tested on $10K$ VQA samples. According to the findings, our pretraining method improves the VQA model's performance by a significant margin.

4.2 Problem from Vanishing Gradient Point of View

In modular neural models, a complex question when processed normally leads to a long chain of modules. When combined, these modules compose a deep network where the vanishing gradient problem becomes serious. As defined in (Sussillo & Abbott, 2015) *"The term **vanishing gradient** refers to the fact that in a feedforward network (FFN) the back-propagated error signal typically decreases (or increases) exponentially as a function of the distance from the final layer"*. We hypothesized that, due to the vanishing gradient problem, the back-propagated signal would gradually weaken as it flows back toward the beginning of a module chain. This implies that the modules at the initial positions of the chains cannot benefit from the gradient signal for learning.

One possibility is that a module receives different strength error signals during training, and stronger gradients compensate for weaker ones. This idea originates in the assumption that the positions of the modules in programs are not static, and change depending on the question. As a result, a module that is located at the initial positions of the program may appear in later positions in other samples. Consequently, when it appears closer to the end of the program, a module receives a stronger learning signal. This interpretation is not necessarily true as the likelihood of a modules' appearance in positions of a program is not uniform. Our ablation experiments show that certain modules are more likely to appear at the initial positions (see Section 2.3). The modules tend to appear at the initial modules often receive weak feedback for learning effective visual representation. To alleviate this issue, we propose to shorten the chain by breaking down the questions based on their compositional feature. Different approaches have been studied and applied to train deep networks to address vanishing gradients, ranging from pretraining (Hinton & Salakhutdinov, 2006), random initialization scaling (Glorot & Bengio, 2010; Sussillo & Abbott, 2015), to employing specific architectures (Krizhevsky et al., 2012), and optimization methods (Martens, 2010). Our solution for this problem is a self-supervised pretraining approach described in the next sections.

4.3 Related Work

4.3.1 Transfer Learning

Transfer learning seeks to improve a target learners' performance on target domains by transferring information from various but related source domains. In this approach, the reliance on a large amount of target-domain data for training target learners may be minimized. Transfer learning has become a prominent and promising field in machine learning due to its broad application possibilities. It studies how a large and general source dataset can be used to enhance performance on down stream activities (Raina et al., 2007; Quattoni, Collins, & Darrell, 2008; Bengio, 2012; Devlin et al., 2019; K. He, Fan, Wu, Xie, & Girshick, 2020). This chapter employs a typical technique in transfer learning where model weights pretrained on source data are used to initialize training in the target task (Yosinski, Clune, Bengio, & Lipson, 2014) with the difference that the same set of data is used in both stages. The resulting model performance is generally attributed to the size of the source dataset and the degree of similarity of the source and target data (Raghu, Zhang, Kleinberg, & Bengio, 2019; Neumann, Pinto, Zhai, & Houlsby, 2019).

A fundamental challenge in this field is heterogeneous transfer learning where the aim is to obtain high performance on target data when it is not similar to the source data (Day & Khoshgoftaar, 2017). As a solution to this challenge, Puigcerver et al. (2021) propose to first pretrain a base model on the upstream data and then create multiple expert models each trained on a different subset of labels, and finally using the target data to select the best expert model for fine-tuning. Similarly, Ngiam et al. (2018) score the significance of source data using the target data.

4.3.2 Self-supervised Pretraining

Self-supervised learning is a type of unsupervised learning that acquires the intrinsic properties of data without requiring human-annotated labels (Doersch & Zisserman, 2017; Do-

ersch, Gupta, & Efros, 2015; Gidaris, Singh, & Komodakis, 2018). In this work, we use *contrastive learning* (K. He et al., 2020; Z. Wu, Xiong, Yu, & Lin, 2018), a form of self-supervised pretraining which trains a network by predicting whether two input images are the visually augmented versions of the same original image. Contrastive learning has shown better results than supervised pretraining on a number of tasks (T. Chen, Kornblith, Swersky, Norouzi, & Hinton, 2020; K. He et al., 2020), leading to increased use in many applications such as vision tasks (X. Chen, Fan, Girshick, & He, 2020; Reed et al., 2021) and NLP tasks (Pfeiffer et al., 2020; Gururangan et al., 2020). In this work, we use the SimCLR algorithm which is a contrastive-based algorithm in pretraining (T. Chen, Kornblith, Norouzi, & Hinton, 2020). Let us explore this algorithm in more detail.

Contrastive Learning Contrastive learning tries to bring similar samples closer together while keeping different samples apart. Human learning patterns provide the primary incentive for contrastive learning. Humans are capable of recognizing items without recalling all of the information. For example, we can quickly recognize a table in an image based on its color, form, and other characteristics. In a nutshell, we develop mental representations of new items which we subsequently utilize to recognize them. The primary goals of self-supervised and contrastive learning are to build and generalize these representations, respectively. A key component of a contrastive learning algorithm is the contrastive loss function. The following paragraph gives a clear view of this loss function.

Contrastive Loss Contrastive loss (Chopra et al. 2005) was one of the first training objectives employed in contrastive deep metric learning. Let $y_i \in \{1, ..., L\}$ be the corresponding labels of input samples x_i among L classes. The goal is to learn a function $f_\theta(.) : \mathcal{X} \to \mathbb{R}^d$ that encodes x_i such that the embedding vector of the samples from the same class have close similar embeddings and those from a different class have different embedding. Therefore, a contrastive loss inputs a pair of samples (x_i, x_j) and minimize the distance when they share the same class, and maximizes it otherwise. Although the early versions of contrastive loss were formulated to take only one positive and one negative sample as inputs,

the recent trend suggests objective functions involving multiple positive and negative pairs. Equation 4.1 shows a simple form of contrastive loss with two input samples where ϵ is a hyperparameter defining the lower bound distance between samples of different classes.

$$\mathcal{L}_{cont}(x_i, x_j, \theta) = 1[y_i = y_j]||f_{\theta(x_i)}||_2^2 + 1[y_i \neq y_j]max(0, \epsilon - ||f_{\theta(x_i)}||_2)^2 \tag{4.1}$$

SimCLR SimCLR uses a contrastive loss to maximize agreement between the representations of two transformed views of the same sample. To produce the correlated views of a sample, a data augmentation module transforms a given data example by simple augmentation techniques such as cropping and Gaussian blur. The augmented data examples are then fed to an encoder that extracts the representations vectors, and finally a small neural network projects representations to a space where contrastive loss is applied.

4.3.3 Self-supervised Pretraining in VQA

Pretraining has proven to make significant improvements in many visual-textual tasks. ViLBERT Lu et al. (2019) extends BERT to a multimodal architecture and uses a self-supervised learning approach to pretrain their model on some multimodal auxiliary tasks in analogy to the standard BERT approach. For example *masked multimodal modeling* where the model is asked to reconstruct the masked words of input text or regions of the input image. Another concurrent work with ViLBERT is (Sun, Myers, et al., 2019) which derives self-supervised tasks from cooking videos paired with text and uses a unified BERT model for both visual and textual modalities. Similar to the mentioned studies, Tan and Bansal (2019) focuses on learning alignment and relationships between the two modalities and pretrain their large-scale transformer model on five different tasks such as masked language modeling, masked object prediction, and cross-modality matching. A similar study is presented in (L. H. Li et al., 2019). Unlike these works, we use an image-based self-supervised approach to pretrain neural modules in our model.

4.3.4 Image Transformation Techniques

The earliest studies show the success of data augmentation roots in simple transformations such as random cropping and horizontal flipping. More complicated strategies were introduced in further investigations *e.g.,* based on neural networks and GAN models. In this section, we briefly explore different techniques of image transformation which are used to generate augmented versions of an image in many self-supervised approaches.

Geometric Transformations

Geometric transformations cover a large number of transformations that manipulate a basic image and are easy to implement. These techniques are typically described in the context of the *safety* of an application, in that using them as data augmentation methods preserves the label post-transformation (Cubuk, Zoph, Mane, Vasudevan, & Le, 2019). For instance, in a cat-versus-dog classification task, flipping or rotation does not change the label while these augmentation techniques are not safe in digit classification *e.g.,* a 180-degree rotation can turn 6 to 9. As this example suggests, the safety of a transformation is domain/task dependent and important to consider. As at some distortion magnitude every transformation may result in changing the label, a data-specific design of augmentation may be needed for many problems. We discuss safety notions in our VQA problem in the next sections and explain how we select a safe set of augmentation techniques for the task at hand. In this work, we mainly use geometric transformations described below for the sake of easy designing of safety criteria.

- **Flipping** is one of the easiest augmentations to implement and has shown to be effective on datasets such as ImageNet (Deng et al., 2009) and CIFAR-10 (Krizhevsky, Nair, & Hinton, 2010), while on digit and text recognition datasets such as MNIST (Lecun & Cortes, 2010) it is not label preserving. Horizontal flipping is more common than vertical flipping.
- **Cropping** can be done by cropping a central region of each image, especially for

mixed height and width dimension images. Random cropping can be used as a simple transformation that may not preserve the label depending on the reduction threshold.

- **Rotating** is done by rotating the image left or right with a degree between 1 and 359. The safety of rotation is specified by the degree parameter. In this work, we carefully determine the degree of rotation in order to preserve the label.
- **Noise Injection** consists of introducing a matrix of random values to the image (Moreno-Barea, Strazzera, Jerez, Urda, & Franco, 2018). The matrix is usually derived from Gaussian distribution. This technique helps CNN-based models learn robust representations of images.

4.4 Invariance Set

Unlike computer vision tasks, in VQA not any transformation type can be applied to the image when creating a transformed version of it. The reason is that the correct relationship between question, image, and answer in an example that must be preserved over transformation. A simple geometric transformation such as horizontal flip does not cause any problem in an image classification or segmentation task, while it may do so in a VQA task. Consider the example in Figure 4.1 where the question is *"what is the color of the ball to the left of the blue metal box?"* Here the target object is the small purple sphere between the cube and the two cylinders and thus the answer is *"purple"*. A horizontal flip on the image results in the image shown at the top left side of Figure 4.1 and causes the target object to be either the small brown sphere or the large red sphere when grounding the question on the transformed image. Therefore, the answer to the questions would be different from the true answer or as in this example, the questions will be ambiguous.

To avoid such problems, we create a set of valid transformations for each unique module in the reasoning chain. Recall that a module is a neural network implementing a reasoning step. A transformation is valid for a module if the module is invariant to that transformation. This means that if the transformation is applied on the input image, *e.g.,* flipping the

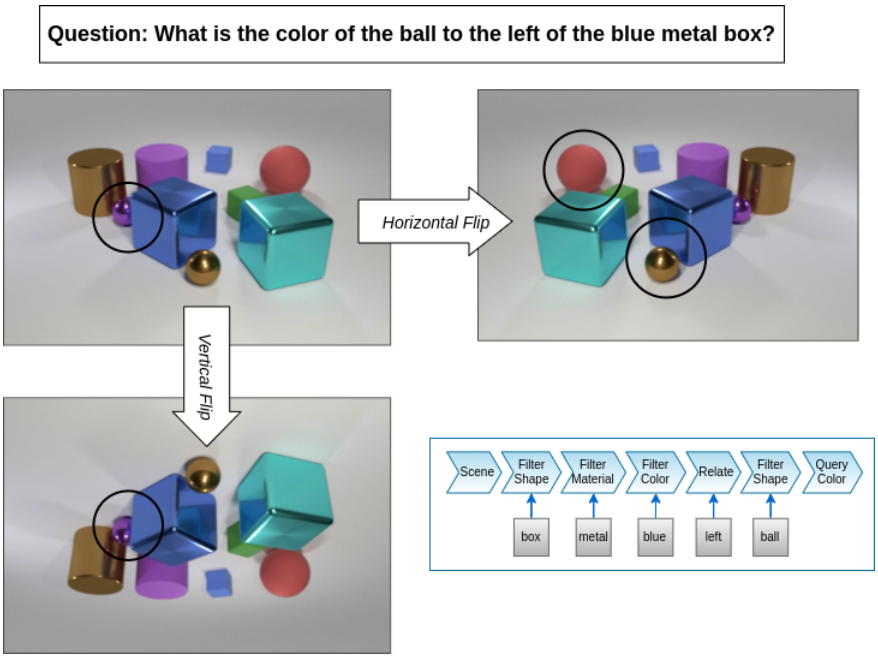


Figure 4.1: The target object and thus the answer do not change by vertical transformation which means the questions is invariant to the vertical transformation, while they change when applying horizontal translation on the original image.

image, the output of the module will not change. To formulate it, let m be the module, t the transformation function, and x the image. Transformation invariance can be mathematically expressed as $m(t(x)) = m(x)$. A module invariance set IV_m includes all the transformations to which the module m is invariant; *i.e.*, $IV_m = \{t|m(t(.)) = m(.)\}$. The invariance set of each module may be different from that of the other modules. We manually create the invariant set for all modules from a set of transformations.

To produce the answer to it, a question is first translated to a reasoning chain which is a sequence of modules. A transformation is valid for a sequence if all modules involved in the sequence are invariant to it. We make the invariant set of a sequence by taking the intersection of the invariant sets of all modules in the sequence. More formally, $IV_s = \bigcap_{m=1}^{n} IV_m$ where s is a sequence of modules representing the reasoning chain and n is the number of modules in the sequence.

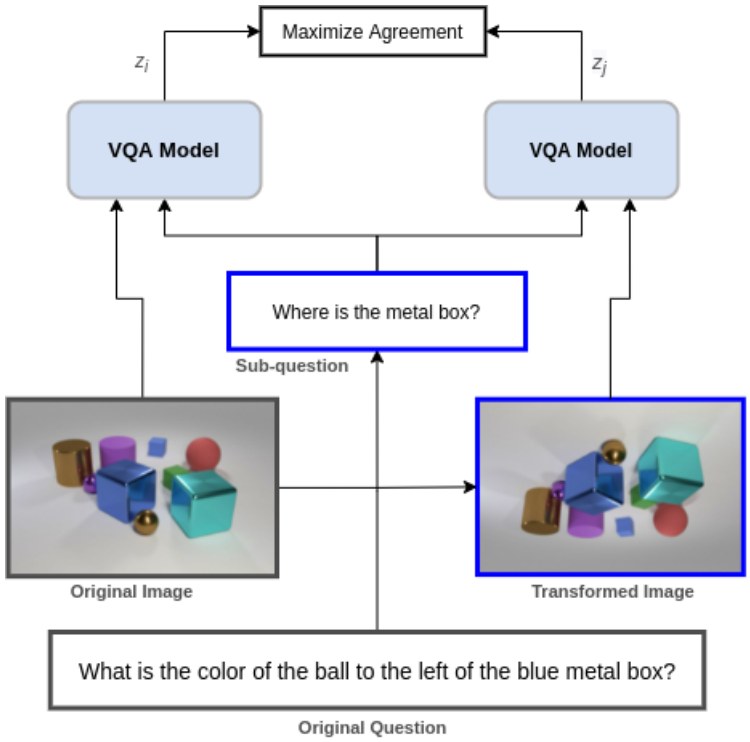


Figure 4.2: Overview of our self-supervised pretraining framework.

4.5 Self-supervised Pretraining

In this section, we present our method of self-supervised pretraining in two levels: first, as a general method that relies on the compositional nature of questions and is applicable to any VQA model, and second as a more specific method that takes advantage of architectural features of modular VQA models.

4.5.1 Using Questions' Compositionality

We use the contrastive learning framework proposed in (T. Chen, Kornblith, Norouzi, & Hinton, 2020) instructing the VQA model to learn effective visual representations of the images by maximizing the agreement between the images and their transformed versions. The framework consists of four components:

- A data augmentation unit that transforms the input image x and produces two aug-

mented views of the image denoted $\tilde{x}_i$ and $\tilde{x}_j$. As mentioned earlier a random augmentation technique may not maintain the correct relationship between questions, images, and answers. In this work, we propose to create a list of valid augmentation methods for each input question (see Section 4.4 for more details). Selecting an augmentation method from the valid transformation list results in a positive pair where their learned representations are expected to be close in the embedding space.

- A VQA model that encodes the augmented images into representation vectors. This framework allows using different choices of VQA models with no architectural constraint. The output of the penultimate layer (the last layer before classification layer) in a VQA model can be used to obtain , $h_i = VQA(\tilde{x}_i)$ where $h_i \in \mathbb{R}^d$ is the representation of $\tilde{x}_i$.

- A projection head which is a small neural network that maps the representations h_i and h_j to the space of contrastive loss function resulting in z_i and z_j. An MLP with one hidden layer is used as the projection head.

- A contrastive loss function as:

$$l_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(sim(z_i, z_k)/\tau)}$$

where N is the number of samples and sim indicates a function that measures the similarity of two samples. At this stage, the VQA model can be of any kind, with no architectural constraints. The inputs to the framework are transformed images and subquestions as shown in Figure 4.2.

The typical inputs to a VQA model are a question and an image, however, toward the goal of learning the basic concepts, we input sub-questions to the model instead of full questions. To create sub-questions, one can simply feed the question to a syntactic parser and generate the question syntax tree. Each noun phrase (NP) or adjective phrase (AP) in the syntax tree is a potential sub-question when placed in simple templates such as

"where is [ADJP]" and *"what is [NP]"*. Given a sub-question and an image, a VQA model is pretrained using the contrastive learning framework. The augmentation unit transforms the image and the model learns a similar representation for the image and its transformed version.

4.5.2 Using Model Modularity

Modular VQA models are made up of a series of small neural network components. Modules are basically independent and may appear anywhere in the combination. However, to produce the answer to a question, they are combined according to a layout guided by the question, and form a sequence to process the input image. Therefore, the compositionality of the model allows the structure of a question to be explicitly reflected in the model's architecture. We utilize this property to expand the idea of pretraining by sub-questions to pretraining by sub-models. Let us define sub-models as shorter sequences of modules that mirror sub-questions in the model's architecture. Sub-models are supposed to learn basic concepts of complex questions faster because i) need to understand only a limited number of concepts such as the color of red or the shape of large, and ii) modules in a shorter sequence received stronger learning signals. We investigate pretraining three variations of module sequences: an individual module, a random sequence, and a sampled sequence of modules. More details are provided in the next sections.

Pretraining Individual Modules Pretraining the modules individually seems a trivial solution to the lack of data and can also provide strong feedback to the network due to the short distance between the network and loss function. For this, we pretrain a module by randomly picking it from a module pool and placing it as a VQA model in the contrastive framework. The module is individually pretrained on a batch of random images for a certain number of epochs. A single module is equivalent to a basic sub-question which asks about a single property, *e.g., "where the color is red in the image?"* In our setting the input to a module sequence is images only, so we disregard questions at this stage.

Pretraining Random Sequences of Modules Modules as the basic building blocks of a compositional VQA model can be trained independently, however, when combined they learn to interact by tuning their parameters and cooperate to achieve the objective which is predicting the answer. Putting them into a sequence provides a setting for a model to learn how to interact with other modules. Given a length, we randomly select modules and combine them to create a sequence with the specified length. A simple scheduler controls the length starting from short lengths while gradually increasing it. Similar to pretraining individual modules, the selected sequence is placed in the framework and pretrained on a random batch of images. The augmentation unit assures that the transformation technique is valid for all the modules in the sequence.

Pretraining Sampled Sequences of Modules As a meaningful sentence in a natural language is not created by putting random words together, a module sequence, which aims to mirror a question structure into the model architecture, contains an inherent notion of order. It is very likely that randomly creating sequences of modules does not follow the true distributions of the modules. To highlight the sequentiality in modules orders, we select a random question and create its corresponding module sequence. However, since a complex question usually results in a long sequence, the problem of a weak learning signal resurfaces here. We use a similar length scheduler, as that described before, to cut the head of the module sequence according to the specified length and ignore the tail, *e.g.,* we keep the first ℓ modules in the sequence where the determined length is ℓ. The pretraining algorithm then follows a similar process to a random sequence. Algorithm 2 summarizes the above-mentioned procedure of pretraining a sampled sequence of modules.

4.6 Experiments

Dataset We use the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017) for training. It provides a training set with $70k$ images, $\sim 700k$ *(image, question, answer)* tuples. The answers come from 28 classes including eight colors, two sizes, three

Algorithm 2 Pretraining Sampled Sequence of Modules

$\mathcal{G}$: program generator
$\mathcal{C}$: curriculum of sequence length increasing gradually
$\mathcal{I}$: Image set used as the training set in contrastive pretraining
$SimCLR$: self-supervised contrastive learning framework
procedure PRETRAIN-SAMPLED-SEQUENCE
    while not done **do**
        sampled-sequence = $\mathcal{G}$.sample()
        for len $\in \mathcal{C}$ **do**
            sub-sequence = *sample-program*[0:len]
            model = **combine-modules**(*sub-sequence*)
            model = **load-pretrained-weights**(*model*)
            valid-transforms = **get-valid-transforms**(*sub-sequence*)
            augmented-$\mathcal{I}$ = **transform**($\mathcal{I}$, *valid-transforms*)
            $SimCLR$.train(*model*, *augmented-*$\mathcal{I}$)
        end for
    end while
end procedure

shapes, two materials and 11 numbers as well as yes and no. The `val` split contains $\sim 150k$ questions and $15k$ unique images. We conducted our evaluation on $10k$ samples from this set.

4.6.1 Setup

To simulate low-data scenarios, we select five subsets from the training set with various sizes to investigate the effect of training data size. These subsets are referred as s-CLEVR$_x$ where $x \in \{5, 10, 15, 20, 30\}$ denotes the size of the subset in terms of the percentage of the full training set *i.e.,* s-CLEVR$_{20}$ contains $\sim 140k$ training examples which is 20% of training set size. We only use the images that participate in the subset for pretraining when training on s-CLEVR$_x$. The experiments are conducted in two stages: **self-supervised pretraining** and **training**. The first stage includes sampling a single or a sequence of modules and pretraining it using the self-supervised algorithm. In the case of sequences of modules, we determine the invariant set of transformations for the sequence according to which the valid transformations are applied to the input images. The images are selected from s-CLEVR$_x$ depending on the subset currently under experiment. In the second stage, we initialize the

execution engine $\mathcal{E}$ weights with pretrained module weights and fine-tune on the given subset. A module weight may be initialized randomly if the module was not sampled during the pretraining stage. Finally, we evaluate the model in $\sim 10k$ examples from CLEVR `val` and compare the results with the baseline described later. The model requires extracting the augmented image features for the pretraining stage and the original images for the training stage. We use the output of $conv4$ of ResNet-101 (K. He et al., 2016) pretrained on ImageNet (Deng et al., 2009) as the features in both stages.

Baselines We use the *execution engine* $\mathcal{E}$ to assess our proposed pretraining approach on VQA performance in comparison to some baselines. We use *vanilla* training of the *execution engine* in s-CLEVR$_x$ subsets as the first baseline where the model is trained from scratch and the weights are randomly initialized. Furthermore, we consider each method of pretraining as a basis of comparison for the other two methods, *i.e.,* pretraining single modules Pre_{single} can be seen as a baseline for a random sequence Pre_{rand} and sampled sequence of modules Pre_{samp}.

4.6.2 Results and Discussion

	5%	10%	15%	20%	30%
$Vanilla$	46.91 ± 0.52	49.90 ± 0.37	52.32 ± 0.30	54.24 ± 0.26	87.70 ± 0.14
Pre_{single}	$\mathbf{50.47 \pm 0.52}$	$\mathbf{61.59 \pm 0.36}$	$\mathbf{79.83 \pm 0.24}$	$\mathbf{85.66 \pm 0.18}$	$\mathbf{89.05 \pm 0.13}$
Pre_{rand}	50.03 ± 0.52	55.27 ± 0.37	71.31 ± 0.27	83.79 ± 0.19	88.18 ± 0.14
Pre_{samp}	49.91 ± 0.52	53.92 ± 0.37	60.93 ± 0.29	80.30 ± 0.21	87.67 ± 0.14

Table 4.1: The accuracy on CLEVR `val` split for different methods of pretraining and different training set. Best accuracy for each training set is shown in bold font. Error bars give 95% confidence bound for estimated accuracy from training.

Table 4.1 shows the experimental results evaluating the VQA model when the model parameters are initialized with pretrained weights and then fine-tuned. Each row of the table shows the accuracy of a module selection method in the pretraining stage on different training sets except for the *vanilla* row which does not contain any pretraining. The header

row indicates the subset of data that is used for both pretraining and fine-tuning stages. As can be seen from the table, pretraining improves the VQA performance in almost all cases at no cost of using additional data. All the reported accuracy scores are statistically significant compared to the *vanilla* baseline except the Pre_{rand} and Pre_{samp} results of s-CLEVR$_{30}$ that are close to the baseline performance. Pre_{rand} pretraining of s-CLEVR$_{30}$ shows a slight improvement $\sim 0.5\%$ in accuracy while the result of Pre_{samp} is 0.03% less than that of the *vanilla* baseline which can be statistically considered insignificant. Among different methods, single module pretraining shows consistent improvements higher than others. Closer inspection of the table shows that the best result has been obtained by the Pre_{single} selection method on s-CLEVR$_{20}$ where the accuracy improves by **31.42** scores compared to the *vanilla* baseline.

As explained in Section 2.2.6, the impressive effects of fine-tuning pretrained models for downstream tasks have been shown by many recent studies. Contrary to those works that use a massive amount of data for pretraining, this study is designed to determine how far self-supervised pretraining can push the performance in low data settings using only limited data for both pretraining and fine-tuning. Our findings also confirm that pretraining is significantly helpful in enhancing the performance even when a limited amount of data is available, and even if the same set of data is used for both pretraining and fine-tuning.

Single module pretraining versus sequence pretraining

Surprisingly, the results of Table 4.1 reveals that pretraining single modules *i.e.,* Pre_{single} outperforms the methods of pretraining sequences of modules *i.e.,* Pre_{rand} and Pre_{sample}. We expected that, during pretraining sequences, modules learn how to interact between themselves and hence work better at the finetuning stage. However, our findings suggest that even pretraining random sequences yield higher performance than sampled sequences of modules. A simple explanation for this might be that single module pretraining focuses on a specific module, allowing the module to learn better visual representations compared to sequence pretraining. Another possible explanation might be that in a sequence of modules,

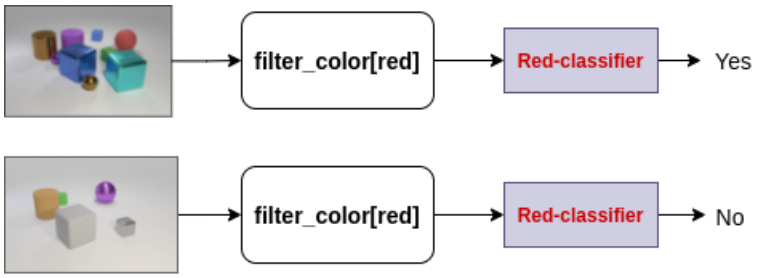


Figure 4.3: The output of a module such as `filter-color[red]` is assumed to provide appropriate information for a hypothetical binary classifier that aims to determine if the module has fulfilled its duty *e.g.,* if the red color exists in the image. However, such assumption seems to oversimplify the problem considering that a module output needs to represent not only the existence of an attribute such as red color but also its spatial positions in the image.

error may propagate and be reinforced through the sequence that eventually reduces the effect of knowledge gained by pretraining.

We hypothesize that such an error may have its root in our underlying assumptions for selecting transformation methods during pretraining. As shown in Figure 4.3, we assumed that, given an image a module should represent it in a way that a binary classifier can predict whether the module fulfilled its duty or not. For example, the module `filter-color[red]` aims to filter and represent red objects in the image. Therefore, the output of this module should provide enough information to a binary classifier determining whether a red object exists in the image. Based on such an assumption, we create a list of valid transformations for each module and produce the augmented images during pretraining. Table 4.2 shows the list of transformations we consider valid for every module at the pretraining stage.

However, such an assumption seems to oversimplify the problem because in practice modules produce an attention matrix over the image to represent the desired objects according to their specific task. By definition, an attention matrix focuses on important areas of an image which means the spatial position is a critical aspect that causes all modules to be sensitive to spatial differences. The SimCLR contrastive algorithm that is used in our pretraining aims a module to learn a similar representation for an image and its augmented version. Now, consider the module `filter-color[red]` is asked to represent an image

Module	Invariance Set (Valid Transformations)
`Filter-color`	vertical-flip, horizontal-flip, rotation(30), colorjitter, affine, perspective, zoom-out, blur
`Filter-materil`	vertical-flip, horizontal-flip, rotation(30), colorjitter, grayscale, affine, perspective, zoom-out
`Relate[left/right]`	vertical-flip, colorjitter, grayscale, affine, perspective, zoom-out, blur
`Relate[behind/front]`	horizontal-flip, colorjitter, grayscale, perspective, zoom-out, blur
Other	vertical-flip, horizontal-flip, rotation(30), colorjitter, grayscale, affine, perspective, zoom-out, blur

Table 4.2: The invariance set for each module shows the valid image transformation that is used in SimCLR, a contrastive learning algorithm, during pretraining stage. The last row contains all other modules that share similar transformation methods including `count`, `equal-color`, `equal-integer`, `equal-material`, `equal-shape`, `equal-size`, `exist`, `filter-shape[cube/cylinder/sphere]`, `filter-size[small/large]`, `greater-than`, `intersect`, `less-than`, `query-color`, `query-material`, `query-shape`, `query-size`, `same-color`, `same-material`, `same-shape`, `same-size`, `scene`, `union`, and `unique`.

and its vertically flipped version similarly resulting in discarding the spatial features of red objects in the image. Such representation with limited spatial information fails to convey the appropriate features required for other modules like `relate[right]`.

How generalizable and scalable is the proposed method in this chapter if we have a realistic dataset? In this chapter we propose a two-part method for self-supervised pretraining. The first part is based on a question's compositionality which is a more general method and applicable to any VQA model and the second part uses a model's modularity which is limited to neural module networks. Since modular networks are a specific type of generic neural network, we only use the question's compositionality to investigate how this method is applicable to a realistic dataset. For this, we picked a random question from GQA dataset and used Stanford unlexicalized PCFG parser (Klein & Manning, 2002) [1] to parse the questions as follows. The input question is *"is there any surfboard to the right of the man the people are standing by?"* and the resulted parse tree is shown in Figure 4.4. To generate sub-questions using the question's compositionality according to Section 4.5.1. Let's pick samples of NPs from the above parse tree, for instance, *"the man"* or *"any surfboard"*. Hav-

[1] http://nlp.stanford.edu:8080/parser/

Parse

```
(ROOT
  (SQ (VBZ Is)
    (NP (EX there))
    (NP
      (NP (DT any) (NN surfboard))
      (PP (IN to)
        (NP
          (NP (DT the) (NN right))
          (PP (IN of)
            (NP
              (NP (DT the) (NN man))
              (SBAR
                (S
                  (NP (DT the) (NNS people))
                  (VP (VBP are)
                    (VP (VBG standing)
                      (PP (IN by)))))))))))
    (. ?)))
```

Figure 4.4: An example of parse tree for question *"is there any surfboard to the right of the man the people are standing by?"* from GQA dataset.

ing the NPs, we can simply create simple questions such as *"where is the man?"*, *"where is any surfboard?"*.

4.7 Summary

In this chapter, we investigated the self-supervised pretraining approach to address the problem of poor performance of low data VQA. We proposed a novel idea which used question compositionality to learn intermediate visual representations for answering questions. Intermediate representations can be viewed as the answers to the sub-questions of a target question allowing the model to answer complex questions by learning to answer the smaller sub-questions. We employed a contrastive algorithm to pretrain the modules of a compositional VQA model and fine-tune the pretrained parameters to answer given questions. Using contrastive learning encourages the model to learn a similar representation for an image and a transformed version of it where we carefully select the valid transformation methods for every VQA example. We experimented with pretraining single modules, a random sequence of modules, and a sampled sequence of modules. According to the experimental results, our

proposed pretraining method significantly outperforms the baseline performance in almost all cases. This confirms that pretraining even with limited data, can enhance performance.

5 | Curriculum Learning for learning Basic Concepts as a Foundation for Answering Complex Questions

VQA models, particularly modular ones, are commonly trained on large-scale datasets to achieve state of the art performance. However, such datasets are sometimes not available. Further, it has been shown that training these models on small datasets significantly reduces their accuracy. In this chapter, we propose a curriculum-based learning (CL) regime to increase the accuracy of VQA models trained on small datasets. Specifically, we offer three criteria to rank the samples in these datasets, and propose a training strategy for each criterion. Our results show that, for small datasets, our CL approach yields more accurate results than those obtained when training with no curriculum.

5.1 Introduction

VQA models are commonly trained on large-scale datasets to achieve state of the art performance (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017; Antol et al., 2015; Hudson & Manning, 2019a). Modular VQA models, in particular, require large data sets for training. These models dynamically combine a number of neural networks according to a pre-specified layout (Andreas et al., 2016b; Johnson, Hariharan, van der Maaten, Hoffman,

et al., 2017; L. Yu et al., 2018) to form a new larger network that produces an answer to an input question. The layout, or program, is generated for each question on the fly. As a consequence, the architecture of the resulting network varies according to the program.

Combining neural networks often leads to a wide and deep network. Training such a large-sized network with a varying architecture calls for a massive amount of labeled data, which is either expensive or very limited in many realistic settings. With insufficient data, a large and complex network can perform unsuccessfully. An example of this is our experience in training the VQA model in (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017) with only 20% of the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017). Our results showed only 55.24% accuracy compared to the accuracy of 96.90% on the full dataset reported in the original paper. Motivated by this experience, the work presented in this chapter studies VQA in low data scenarios, and sheds light on the performance of the current modular VQA models under data scarcity conditions. To the best of our knowledge, this is the first study to investigate VQA models in a low-data regime.

Many approaches have been investigated to improve the performance of deep learning models when trained on limited data, ranging from data augmentations (X. Zhang et al., 2019) and pre-training (Erhan et al., 2010) to semi-supervised learning (Kingma et al., 2014) and transfer learning (Raina et al., 2007). However, these works mostly deal with the scarcity of labeled data by obtaining help from available unlabeled data, or by transferring knowledge from similar domains. Unlike these, our goal is to train a modular VQA model from scratch using only a small amount of labeled data without using any other resources. This is basically because we aim to investigate our approach in isolation and see how far it can improve the system's performance.

Specifically, we take a curriculum learning (CL) approach to tackle the problem of VQA models' low performance under low data conditions. CL was introduced as a method to supervise the order in which data examples are exposed to the model (Bengio, Louradour, Collobert, & Weston, 2009). Our hope is to maximize the usage of training samples by performing supervision on the order of training data fed into the model.

The underlying idea of CL is to start learning from easy examples, and gradually consider harder ones, rather than using examples in a random sequence. To rank training examples from easy to hard, CL must define the concepts of easy and hard examples. Such ranking is a key challenge in CL. Many of the ranking criteria introduced in the CL literature are problem-specific heuristics (C. Liu, He, Liu, & Zhao, 2018) or automated measures based on model performance (Hacohen & Weinshall, 2019). In this work, we propose and analyze the performance of three ranking criteria: (1) a length-based criterion, which considers longer questions to be more complex than shorter questions, and ranks the examples in increasing order of their program length; (2) a criterion based on an answer hierarchy, which organizes all possible answers from coarse to fine; and (3) a criterion that relies on model loss for deciding the difficulty level of the examples and ranking them accordingly.

In addition to the ranking heuristics, in Section 5.4, we suggest a CL training strategy for each criterion. We also argue that under CL training in low data regimes, a model is very susceptible to overfitting and poor generalization. Employing a regularizer is critical to prevent the model from becoming over-confident on the training data. We demonstrate that the proposed training strategies, when coupled with *L2-norm* regularization, lead to a significant improvement in performance, in some cases over 30% increase in accuracy.

We apply our approach to the model proposed in (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017) as a modular VQA model. The model originally consists of two main components: (1) a *program generator* that takes a question and generates a program; and (2) an *execution engine* that combines neural modules according to the program in order to create a network to produce an answer from the input image. Johnson, Hariharan, van der Maaten, Hoffman, et al. (2017) demonstrate that the *program generator* can produce acceptable programs by only training on a small fraction of all possible programs ($\leq 4\%$). Thus, we focus on training the *execution engine* in a low-data setting and use ground-truth programs as input to the *execution engine*. To simulate a low data regime, we use four randomly chosen small subsets of the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017) for training. Our results show that our CL approach yields more accurate results

than those obtained when training with no curriculum.

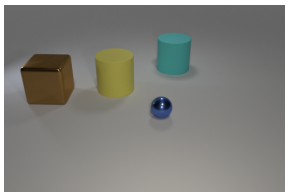
5.2 Background

Visual question answering is the task of inferring the answer by reasoning on the input question and image. Most of the current approaches map question-image pairs into a cross-modal common embedding space. A question is usually treated holistically in such approaches; thus, the reasoning process is hard to explain (Tan & Bansal, 2019; Lu et al., 2019; Selvaraju et al., 2020).

In contrast, modular approaches perform visual reasoning by semantically parsing the question and generating a reasoning chain called a program (Andreas et al., 2016b; Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017). The program represents the reasoning steps required for answering the question in the form of a layout for the modules. The algorithm then combines the modules according to the program. Modules are small neural networks treated as single-task functions that are combined into a larger network to accomplish a complex job. The resulting network is executed on the input image to predict the answer.

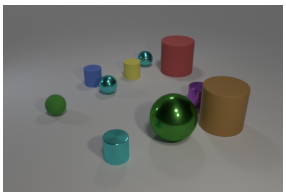
Modular approaches naturally have a strong potential for interpretability. Hu et al. (2018) showed that human evaluators can more clearly understand their modular VQA model compared to a non-modular model (Hudson & Manning, 2018). Due to this feature, we are interested in studying modular models.

Similar to other VQA models, modular approaches call for a large amount of annotated data for both the semantic parser (*program generator*) and the executor. This issue has led to recent studies on sample-efficient training strategies, ranging from multi-task learning (Hu et al., 2018) and active learning (Misra et al., 2018) to disentangling reasoning from vision and language understanding (Yi et al., 2018). For instance, Misra et al. (2018) propose an agent that, instead of operating on the training set, interactively learns by asking questions. Regarding the simulated low data setting in our work, efficiently use of training data be-



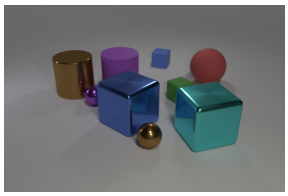
Easy Q: There is an object that is both right of the yellow rubber object and behind the large brown thing; what is its color? **A:** Cyan

(A) Easy Question



Medium Q: What number of large objects are cyan metallic spheres or yellow spheres? **A:** 0

(B) Medium Question



Hard Q: What size is the metal block right of the brown metal thing right of the blue thing in front of the small blue rubber thing? **A:** Large

(C) Hard Question

Figure 5.1: Examples of easy, medium and hard questions according to their H scores. The proposed heuristics do not always agree. According to the length-based heuristic, example A is harder than example B due to its longer question.

comes extremely important. We employ curriculum learning in Section 5.3 and Section 5.4 as a method of making best use of limited available data where a model can establish its understanding on simple concepts and gradually develop that understanding by seeing harder examples over training.

5.3 Curriculum Heuristics for VQA

Studies introduce various heuristics for measuring the hardness of examples. Some heuristics define hardness based on human judgment, in the sense that an example can be challenging for a machine if a human finds it difficult. Such criteria take features of examples into consideration such as *word frequency* and *sentence length* for texts (Spitkovsky, Alshawi, & Jurafsky, 2010; Platanios, Stretcu, Neubig, Poczos, & Mitchell, 2019; C. Liu et al., 2018) and *shape complexity* for images (Bengio et al., 2009; Duan et al., 2020). The ordering of examples provided by these heuristics is task-dependent and does not change during training. In contrast, more general criteria determine the ordering of examples by incorporating the machine's response, *e.g.,* a teacher network supervises the learning process (Hacohen & Weinshall, 2019) or the progress of a model is taken into account (M. Kumar, Packer,

& Koller, 2010; Sachan & Xing, 2016; Zhou, Wang, & Bilmes, 2021). In this study, we explore the heuristics described in the remainder of this section.

5.3.1 Curriculum by Program Length

An intuitive measure of hardness for a VQA task is based on question length, *i.e.,* longer questions are more complex to be understood and answered than shorter ones. This assumption has its root in the observation that a longer question generally involves understanding a larger number of objects and relations. We consider the length of the program corresponding to a question as an indicator of question length.

Under the program length curriculum, the network is fed with easy-to-hard ranked examples starting from shorter programs and gradually increasing programs' length.

5.3.2 Curriculum by Answer Hierarchy

Investigating the learning process of $\mathcal{E}$ while training with IID data batching, we hypothesized the model's implicit curriculum to be as follows: the model quickly learns to correctly predict the type of the answers, *e.g.,* color, size or digit. However, the more distinct the values each type includes, the longer it takes for the model to distinguish them. For instance, the model needs a longer time to distinguish between eight different color values compared to *large* and *small* as the values of size. We also assume that the model struggles to identify visual features that are hard to detect, regardless of the number of distinct values, *e.g.,* whether the material of an object is *metal* or *rubber*.

Motivated by the above observations, we define another measure based on a handcrafted answer hierarchy in order to shift the focus from questions to answers. The higher level in the hierarchy includes a coarser categorization of each answer type, and the answer types are vertically extended downward to finer classes of types. In other words, the direct link between an answer type and its values is interleaved with intermediate levels of abstraction, *e.g.,* digit at a lower level is divided into three groups, such as *'0'*, *'1'* and *many*. This

classification splits into finer groups toward the bottom of the path.

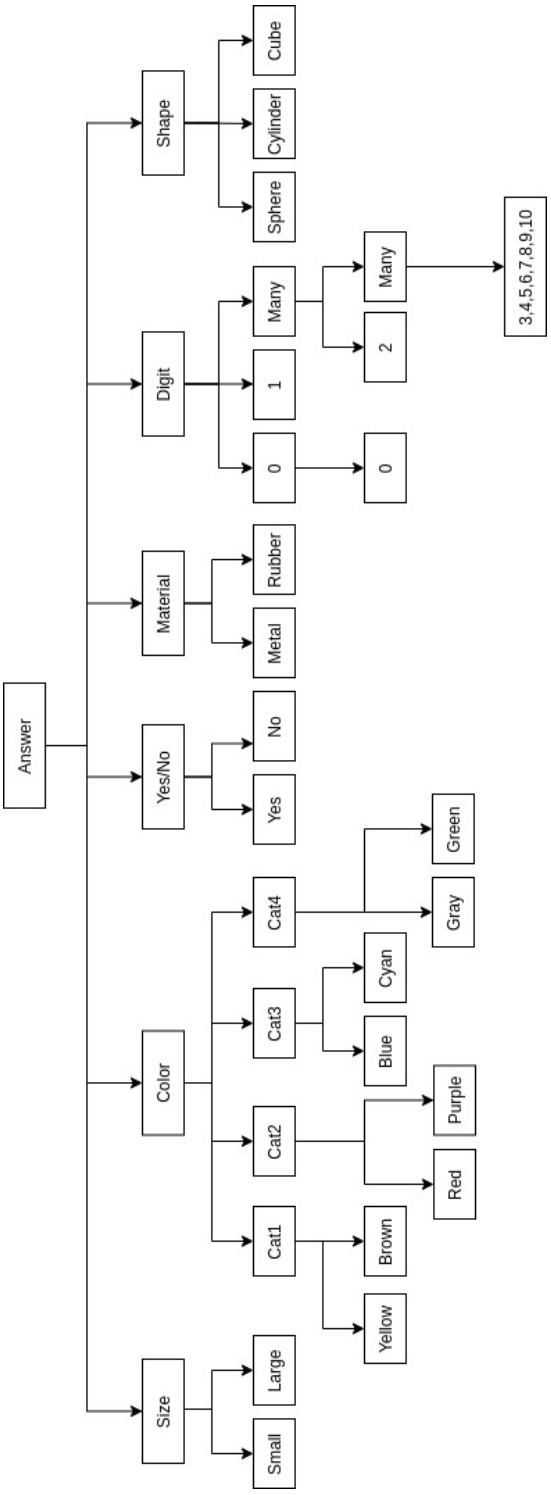


Figure 5.2: A schematic view of the answer hierarchy used as the base of a curriculum.

The answer hierarchy, shown in Figure 5.2, classifies the answers at different hierarchical levels. Specifically, we defined intermediate levels between answer types and their values. The intermediate levels are employed as the higher level pseudo answers to the questions. According to the curriculum, the algorithm maps the true answer to the higher levels pseudo answers in order to gradually guide the predicted answers from a coarse level to a more specific one. When the scheduler decides to update the curriculum, several nodes are expanded to the next level, *i.e.,* the model is exposed to the finer level of an answer type. We do not force the curriculum to simultaneously expand all of the nodes that are at a similar level of the hierarchy. Instead, we assign a number to every node that determines the expansion time in terms of curriculum update round. Specifically, a node is expanded when the count of the curriculum update is matched with its assigned number. For instance, the node *size* is expanded to its children *small* and *large* in the second round of curriculum update if number *2* is assigned to the node *size*. This provides a degree of freedom for the algorithm to gradually learn the answers. Although we statically specify these numbers in our algorithm, they can be implemented as learnable parameters, which we leave to future work. Learning expansion times helps the model move the curriculum further at its pace.

5.3.3 Curriculum by Hard Examples

The intuition behind this heuristic is to focus training on the hard examples where the learner does not perform well and consequently the loss is high. The notion of hardness is considered dynamic, as a hard problem tends to be deemed easier while it is being understood. Following Zhou et al. (2020), we employ a dynamic hardness criterion based on the running average of *instantaneous hardness*, which is defined as the loss difference between two consecutive training iterations.

Let $(\mathbf{x}_i, p_i)$ be the ith image-program pair as a training example with the ground truth answer a_i. The instantaneous hardness $r_t(i)$ of $(\mathbf{x}_i, p_i)$ at time-step t is defined as follows:

$$r_t(i) = |\ell_t(a_i - \mathcal{E}(\mathbf{x}_i, p_i; w_t)) - \ell_{t-1}(a_i - \mathcal{E}(\mathbf{x}_i, p_i; w_{t-1}))| \tag{5.1}$$

Hardness level	**Epoch**					
	1	10	25	50	75	98
Easy	0.89907262	0.80573033	1.15928526	0.93015788	1.15813931	1.12112416
Medium	5.49240219	1.87198794	2.31205007	1.40253631	1.33270774	1.26686139
Hard	11.77809829	3.56507419	1.74276955	1.0956815	0.94307758	1.39973173

Table 5.1: Hardness scores at different epochs. The hardness scores decrease as training progresses.

where t represents training epochs.

The hardness score of an example is obtained by recursively computing a running average over instantaneous hardness, which reflects the dynamics of hardness,

$$H_{t+1}(i) = \begin{cases} \gamma \times r_t(i) + (1-\gamma) \times H_t(i) & \text{if } i \in S_t \\ H_t(i) & \text{else} \end{cases} \tag{5.2}$$

where $\gamma \in [0, 1]$ is a discount factor, and $S_t \subseteq \{(\mathbf{x}_1, p1), ..., (\mathbf{x}_N, p_N)\}$ is a subset of the training set selected at each training step according to a sampling strategy. We employ the strategy introduced in Johnson, Hariharan, van der Maaten, Hoffman, et al. (2017), which uses a probability function based on the hardness score H. This function favors harder examples so long as the probability of selecting easy examples is not zero.

Once a sample is used to train the model, its H score becomes small and it stays low relative to the other samples. Thus samples' H score converges during training and remains consistent. This gives the unselected samples a higher chance to be selected by the sampling function in the future steps. Figure 5.1 shows three samples with low, medium and high H scores (denoted as easy, medium and hard questions) at the first iteration and Table 5.1 lists their corresponding H scores during training. It is clear that the H score is decreasing over training until convergence.

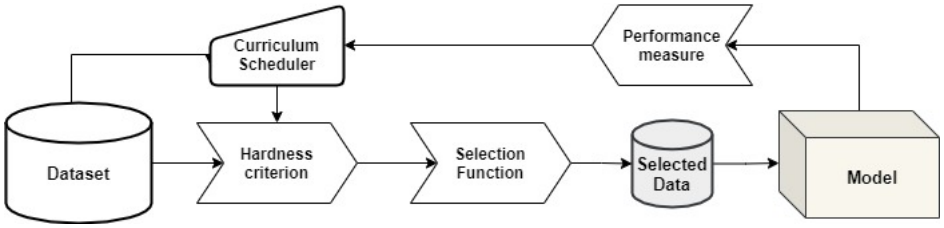


Figure 5.3: General curriculum learning pipeline.

5.4 Curriculum Learning for VQA

We now describe our training procedure. A generic curriculum learning requires a model M and a training dataset D as inputs. It also requires the existence of a hardness criterion N, a curriculum scheduler E, a selection function L and a performance measure P. Figure 5.3 depicts a high level view of the main components' workflow in a CL algorithm.

According to traditional curriculum learning, at every training iteration, the scheduler E decides when to update the curriculum. Curriculum learning is applied on top of the conventional training loop in machine learning. The output of each training loop is usually the model's performance measure, which may be used by the scheduling function L to specify the appropriate moment for modifying the curriculum. The scheduler can also decide based merely on the number of training iterations. A curriculum update typically includes re-ranking training examples according to the hardness criterion N. In the next step, the algorithm selects a subset D^* of the training set D, which to be used by the model in the next round of training. The selection function SF can utilize different approaches, *e.g.,* weighting (Liang, Jiang, Meng, & Hauptmann, 2016; Zhou et al., 2020), sampling (Zhou et al., 2021) or batching (Yong Jae Lee & Grauman, 2011).

Training by length-based curriculum. We design a CL training strategy for the length-based curriculum by equipping the CL training with a batching method as the selection function and a linear paced scheduler. The scheduler controls the curriculum update at a linear pace, *i.e.,* a hyper-parameter specifies the number of iterations for learning the curriculum.

Algorithm 3 Scheduled Training with Curriculum

1: $\mathcal{E}$: execution engine
2: $\{(\mathbf{x}_i, p_i, a_i)\}_{i=1}^{n}$: training examples
3: γ: $\in [0, 1]$, discount factor for reducing subset size
4: T: number of iterations
5: T_0: number of warm-starting iterations
6: **procedure** HEMTRAINING(X,Y)
7: **for** $t \in \{1, ..., T\}$ **do**
8: **if** $t \leq T_0$ **then** ▷ Phase1: Warm-starting
9: $S_t = [n]$
10: **else** ▷ Phase2: Hard example mining
11: **for** $i \in \{1, ..., n\}$ **do**
12: $p_i = H_t(i) + C_t(i)$
13: **end for**
14: Normalise(p_i)
15: $S_t \leftarrow$ sample k_t district elements from $Categorical(\vec{p})$
16: $w_t \leftarrow w_{t-1} + \pi\Big(\nabla_w \sum_{i \in S_t} \ell(a_i, \mathcal{E}(p_i, x_i; w_{t-1}))\Big)$
17: **end if**
18: Compute $r_t(i)$ for $i \in S_t$ using Eq. (5.1)
19: Update $H_{t+1}(i)$ using Eq.(5.2)
20: $k_{t+1} \leftarrow \gamma_k \times k_t$
21: **end for**
22: **end procedure**

Training by answer hierarchy curriculum. Our proposed training algorithm for the answer hierarchy curriculum takes advantage of a simple self-paced scheduler based on the model performance. Specifically, the scheduler updates the curriculum where the normalized difference of accuracy between two consecutive iterations goes higher than a predefined threshold.

Training by hard examples curriculum. This training strategy suggests training the model in two-phases. The first phase is a warm-up phase, where the model sweeps all training examples. The next phase is the curriculum training, where the model rank the examples according to their hardness, and learns a selected subset of them.

Algorithm 3 summarizes our training approach. To encourage diversity, we add a submodular optimization C to the hardness score in line 12, which is inspired by (Zhou & Bilmes, 2018). Since this can be any submodular function, we choose a fast greedy function

based on the similarity between examples.

$$\max_{S_t} \sum_{i \in S_t} H_t(i) + \lambda_t C(S_t) \tag{5.3}$$

where $C(S_t) = \sum_{i,j \in S_t} w_{i,j}$ and $w_{i,j}$ represents the similarity between example i and j. The preference for diversity can be controlled by λ_t. We gradually reduce it during training to further focus learning on hard examples. The input to C is a representation of a data point that can be a fusion of both text and image modalities. For this, we use the output of the model's penultimate layer as the representations of the examples.

Instead of deterministically choosing the top k samples based on H, we randomly select the examples for the next round of training with the probability $p_{t,i} \propto f(H_{t-1}(i))$ where $f(.)$ is a non-decreasing function, similar to (Zhou et al., 2020).

The selection function does not deterministically sample top k hard examples. Rather, it samples from all of the training examples with probability $p_{t,i} \propto f(H_{t-1}(i))$, where $f(.)$ is a non-decreasing function. This probability function favors hard examples, yet selecting easy ones is possible.

At early training, when the H scores are poorly estimated, $f(.)$ should encourage exploration, and move toward more exploitation as training progresses and H estimation is becoming more accurate. We balanced the trade off between exploration and exploitation using upper the confidence bandit (UCB) algorithm, similar to (P. Auer, Cesa-Bianchi, Freund, & Schapire, 2003; Zhou et al., 2020).

$$f(i,t) = Normalized\Big[H_t(i) + c\sqrt{\log T / N_t(i)}\Big] \tag{5.4}$$

where T is the number of iterations, and $N_t(i)$ is the number of times that the ith sample has been selected prior to time step t. UCB controls the degree of exploration by the hyper-parameter c which we set as 0.001 in our implementation.

5.4.1 Improved Curriculum Learning

The idea of learning the answers in a non-random ordering as happens in CL has been shown to be helpful for the learning process in many cases. However, this idea has one essential deficiency. It focuses on a particular subset of questions first, and is not exposed to a diverse set of questions. When a new question arrives, the algorithm struggles to adjust to it, as the learnt representations fit the previous questions. This problem exacerbates in low data settings. Many studies highlight the importance of selecting a diverse set of examples as a solution to this issue (Sachan & Xing, 2016; Zhou & Bilmes, 2018), and the CL algorithm generally benefits from diversity in training examples. However, as confirmed by our experiments (Section 5.5.2), it does not prevent the model from overfitting. We therefore explore the effect of other techniques of regularizing such as dropout and L2-norm.

5.5 Experiment

5.5.1 Setup

We use our implementation of the *execution engine* model (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017). A vanilla training of the model posts the lowest threshold of the performance in our setting. We also implemented and compared the three heuristics for the hardness criterion: *program length* (Section 5.3.1), *answer hierarchy* (Section 5.3.2) and *hard example* (Section 5.3.3). The *length-based* curriculum can be seen as a baseline to the *answer hierarchy* criterion, while both of them play the role of baseline for the *hard example* curriculum. We do not compare with the state of the art, because our goal is to study VQA in a low-data regime, and to the best of our knowledge, there is no other work that conducts similar research. Thus, we focus on improving the performance of our baseline models.

We assessed our baselines under the following conditions: i) **No-Reg** when no regularizer is applied, ii) **Dropout** when we apply dropout technique to the final linear layer (classification layer) in $\mathcal{E}$ and iii) **L2-norm** when *L2-norm* regularizer is applied as a weight

decay to the optimizer.

Dataset

We evaluated our approach on the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017), which provides a training set with $70k$ images, $\sim 700k$ $(\mathbf{x}, q, a)$ tuples and 32 answer classes. To simulate a low-data regime, we randomly sampled four subsets of different sizes from CLEVR `train`. The size of the subsets are chosen to be 5%, 10%, 15% and 20% of the full `train` set, which contains $35k$, $70k$, $105k$, and $140k$ $(\mathbf{x}, q, a)$ tuples respectively. We call these subsets s-CLEVR$_p$, where p denotes the percentage of the subset size in relation to `train`, *e.g.,* s-CLEVR$_{15}$ refers to the subset of size 15% of `train`. As CLEVR `train` and CLEVR `val` (the evaluation set) have similar answer distributions, to perform a fair comparison, it is important that the sampled subsets also have similar answer distributions. Our evaluation is conducted on the `val`split, which contains $\sim 150k$ questions and $15k$ unique images.

Baselines

No-CL is used as the vanilla baseline where the *execution engine* is trained with an IID sampling on s-CLEVR subsets without any curriculum. In other words, the model sees all examples training set at every iteration.

Length-CL follows a linear paced scheduler when training the *execution engine* under the length-based curriculum (Section 5.3.1).

AnswerH-CL makes use of a self-paced scheduler based performance measurement and the answer hierarchy curriculum (Section 5.3.2). More specifically, the curriculum updates if the changes in normalized accuracy between two consecutive iterations are higher than a pre-specified threshold. A batching function selects the samples for every training iteration.

HardEx-CL uses the hard example heuristic Section 5.3.3 as the criterion of ranking data and follows the algorithm 3 for training. Unless stated otherwise, we use HEM-CL in all ablation analysis experiments.

Method	**No-Reg**				**Drop-out**				**L2-norm**			
	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
No-CL	46.91	48.77	49.68	51.25	46.94	48.36	49.67	49.92	46.71	50.25	52.20	54.34
Length-CL	46.55	46.67	47.83	48.12	46.68	47.33	47.61	47.71	47.89	49.65	50.98	51.50
AnswerH-CL	47.42	48.59	49.73	51.65	47.43	47.73	48.60	50.24	48.62	49.03	48.70	48.95
HardEx-CL	47.93	50.04	51.97	53.14	48.80	49.94	51.69	56.29	48.95	51.49	53.27	$\mathbf{87.62 \pm 0.32}$

Table 5.2: The *execution engine* accuracy (%) on CLEVR `val` when training on s-CLEVR$_5$, s-CLEVR$_{10}$, s-CLEVR$_{15}$ and s-CLEVR$_{20}$ with three different choices of curriculum. The length-based (**Length-CL**) and answer hierarchy (**AnswerH**) curriculum does not improve the performance while hard example (**HardEx-CL**) outperforms the vanilla baseline (**No-CL**) in all experiments.

Implementation Details

The *execution engine* uses the images features from $conv4$ of ResNet-101 (K. He et al., 2016) pretrained on ImageNet (Deng et al., 2009). We use Adam (Kingma, 2015) with a fixed learning rate of $1e{-}4$ to optimize the first three baselines and a cyclic cosine annealing learning-rate schedule to optimize HEM-CL. In the case of the experiments that use *L2-norm*, a weight decay of $5e{-}4$ is added to the ADAM optimizer. We also use dropout $= 0.5$ for some experiments.

5.5.2 Results and Discussion

A: Curriculum heuristics' effect.

We evaluate the impact of our proposed training strategies with the three heuristics by looking at their performance on CLEVR `val` in Table 5.2 while training on s-CLEVR subsets. As the table shows, using the **length-based curriculum** yields poor accuracy in almost all cases of s-CLEVR training subsets with and without regularization. An explanation for this could be overfitting. As previously mentioned, overfitting is a serious challenge in low data training.

According to our analysis, there is a high chance for the model to overfit some mod-

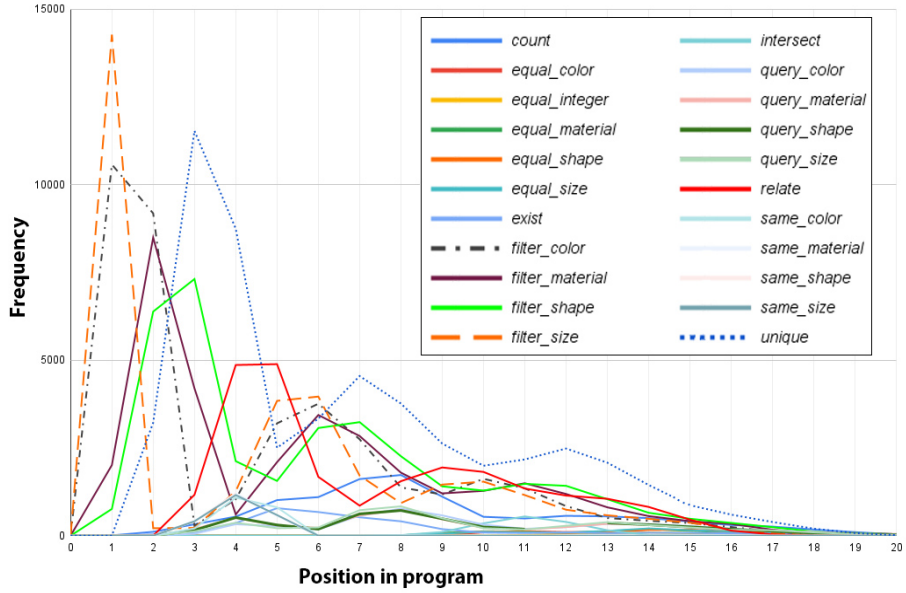


Figure 5.4: Frequency of modules appearance in different positions of programs. Some modules are more likely to appear at the first positions.

ules because they are more likely to appear in the first positions of a program. Figure 5.4 depicts the frequency of modules' appearance in various positions of programs in about $28k$ programs. These modules are commonly related to an *anchor object* in a question, where other objects are described by their relation to this object, *e.g., the yellow thing* is the anchor in the question *"what is the size of cube to the right of the yellow thing"*. To identify *the cube* and determine its *size*, one must find *the yellow thing*, and attend to the objects on its left side. Since objects are normally described by attributes such as color, size and material, attribute-related modules tend to appear at the beginning of a program.

Ranking programs by their length makes the model focus on a limited number of modules during early training, which increases the chance of overfitting. Thus, the model thus struggles with learning other modules when they appear later in longer programs. According to the results, dropout and L2 regularizations do not effectively prevent overfitting where the curriculum forces the model to over-concentrate on such structural biases in data.

Answer hierarchy curriculum makes a marginal improvement on some subsets particularly s-CLEVR$_5$. **Hard example curriculum** produces impressive results, improving

the baselines in all cases. The result verifies the effectiveness of emphasizing on hard examples in low data regimes where due to the limited size of data and its large capacity, a deep network tends to memorize easy data points without actually learning a pattern. Forcing the model to focus on hard examples induces a form of implicit regularization. Additionally, the self-pacing feature of the curriculum allows the algorithm to update the curriculum based on its progress.

Table 5.2 also shows that the **HardEx-CL** method does not produce the best accuracy per se. As the table reports the average results, it should be noted that the best accuracy we achieved in the case of **HardEx-CL** is 88.83 score in accuracy where the weights are uniformly initialized and L2-norm is used for regularization. In fact, the regularization causes a huge rise in accuracy. The next paragraphs look into the reasons that our regularization choice effectively boost the **HardEx-CL** approach.

B: Regularization impact.

To investigate the impact of different regularizers we conducted ablation studies by applying L1-norm in addition to L2 and drop-out regularization. Table 5.3 demonstrates that in contrast to dropout and L1-norm, using L2 regularization results in improved performance in almost all experiments. Although L2 regularization may have different effects in different algorithms, to investigate the role of L2 regularization in CL training, we conducted an ablation experiment on the selected examples in **HardEx-CL** algorithm with and without L2-norm. First, we record the hardness measures of selected examples at every epoch $H_t(i)$ and split the range of measures into three categories, *easy*, *medium* and *hard*. The population distribution of examples by their hardness measure has a long tail. This long tail is excluded from the splitting and categorized as *very hard*. We then calculate the proportion of each category in the selected examples at 100 epochs as plotted in Figure 5.5.

These plots provide insight into the behavior of L2 regularization. Specifically, we observe that except for the easy category, the proportion of examples from other categories is higher for all epochs. It can be explained by the fact that the **HardEx-CL** algorithm draws

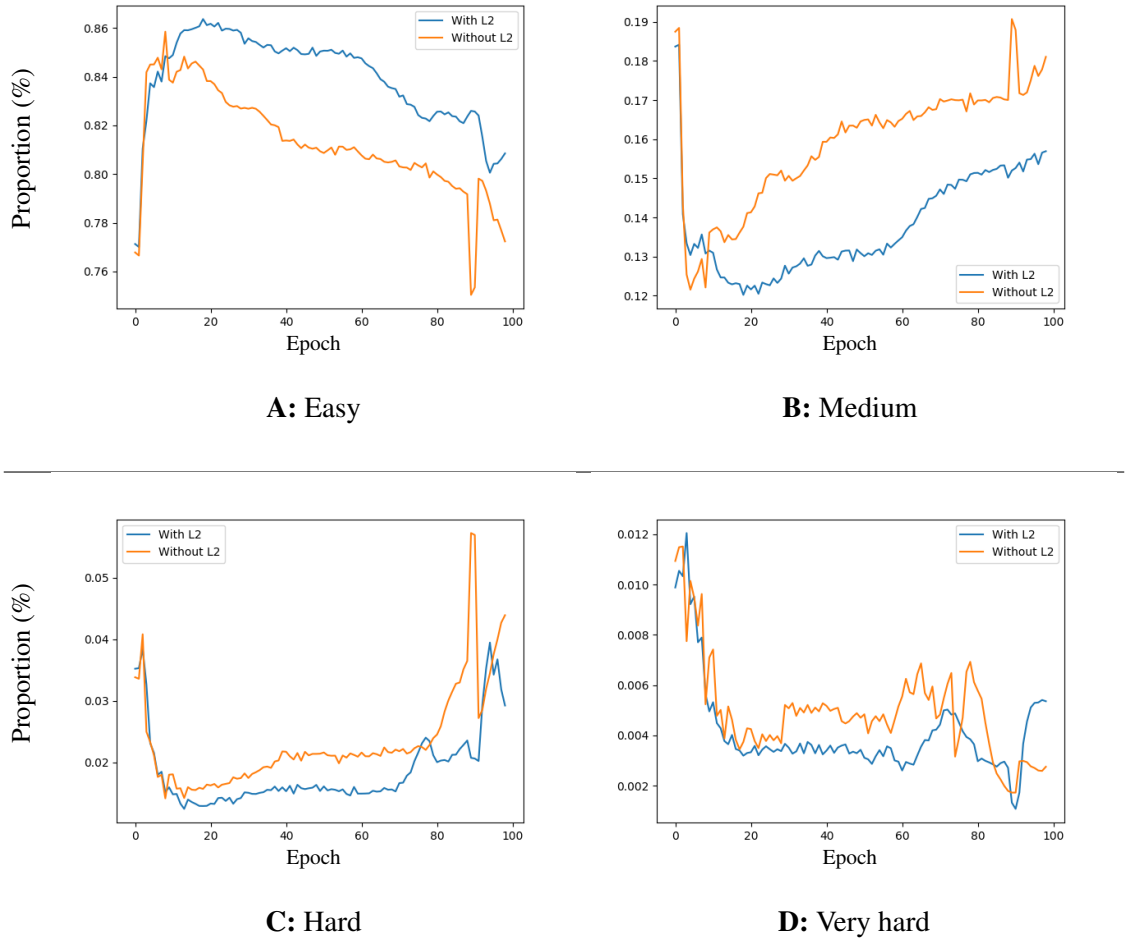


Figure 5.5: The proportion of different hardness categories in selected examples at 100 epoch in cases of with and without L2 regularization. The regularization prevents forgetting by forcing the algorithm to incorporate more easy samples in the training set.

	No Regularizer	Drop-out	L1-norm	L2-norm
No-CL	51.25	49.92	45.12	54.34
CL	53.14	56.29	46.79	86.65

Table 5.3: The impact of different regularizer on HardEX-CL accuracy when training on s-CLEVR$_{20}$.

the model's attention to hard examples during training. As the model is learning the examples, their corresponding hardness measure is decreasing so that they finally are learned and considered to be easy. Without using L2 regularization, the model overly focuses on learning hard examples and, as a consequence, forgets the learned patterns of easy examples. L2-norm protects the model from forgetting such patterns by incorporating in loss and forcing the sampling function to also samples more instances from the easy category.

C: Why is there a jump in the accuracy of HardEx-CL with L2 regularization when training on s-CLEVR$_{20}$?

An explanation for this sudden rise in accuracy could be a tipping point occurring in the training process. There are different shapes of learning curves that are defined in learning theory in psychology (Ebbinghaus, 1913; Bills, 1934). The S-curve is the idealized general form of learning where the learner slowly accumulates small steps at first followed by a steep up stage with larger steps. As the learning activity reaches it limit the smaller steps successively occur to level off the curve.

Remember that Figure 3.2 shows the accuracy of vanilla training when the size of the training set varies. Looking closely at the curve in Figure 3.2 reveals the fact that the *execution engine* performance experiences a jump in accuracy where the size of the training set becomes larger than 20%. Due to a lack of data, we do not see this gap in performance when training on s-CLEVR$_{5-20}$. L2 regularization, however, stimulates the jump to happen earlier in **HardEx-CL**. To investigate it further, we run **HardEx-CL** with four training subsets of different sizes including 15%, 20%, 25% and 30% and report the accuracy on CLEVR `val` in Figure 5.6. All settings are similar to **HardEx-CL** with L2-norm in Table 5.2 except the weights are uniformly initialized. From these experiments, we observe the jump in the

learning curve of s-CLEVR$_{15}$ in addition to the larger training sets. This shows the tipping point in the training can accrue earlier depending on the algorithm and settings.

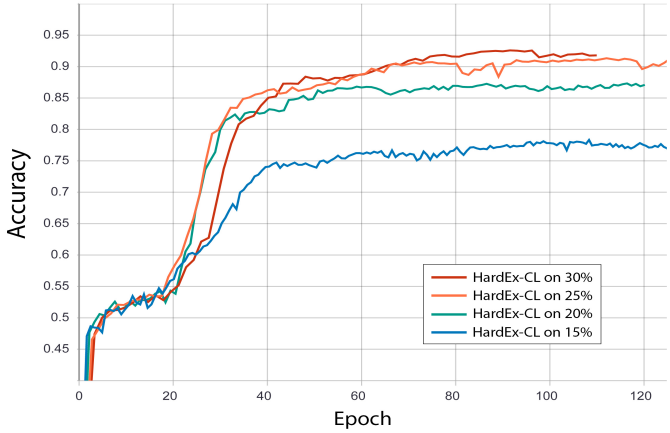


Figure 5.6: The accuracy of **HardEx-CL** algorithm on CLEVR `val` where *execution engine* weights is uniformly initialized and trained on s-CLEVR$_{15,20,25,30}$.

5.5.3 D: Could one use an automated approach instead of the hand-crafted answer hierarchy?

The hand-crafted hierarchy aims to simplify answering questions in early stages of learning by grouping the answers as broader categories. It is in fact a demo type of a broader idea of using automatic hierarchies. The general idea is to automatically create a hierarchy using the hypernymity and hyponymity relationship of categories in a knowledge base such as WordNet (Fellbaum, 2010). This approach can result in a generic hierarchy which is applicable to other datasets as well. It can also be automatically refined based on the dataset.

5.6 Summary

This work studied VQA in low data settings and shed light on the low performance of VQA models under the data scarcity condition. To improve the performance, we propose three curriculum learning approaches based on length, answer hierarchy, and hard examples. We also stressed the problem of overfitting and poor generalization that becomes crucially important in the absence of sufficient data. We explored the effect of using generalization

techniques on a model's performance in low data regimes. Our results show that the proposed CL algorithms outperform the baseline in many cases while failing in some others. However, the algorithms when coupled with L2 regularization lead to improvements.

6 | A Meta-Learning Algorithm for Low Data VQA

In the previous chapters, we proposed and tested efficient approaches for injecting inductive biases to solve the problem of answering complex questions in low-data scenarios. We stated that in the presence of sufficient data, a model learns the inductive biases during training, whereas capturing complex biases in a low data setting is challenging. This thesis focuses on the inductive biases assuming a complex question is composed of a number of basic concepts, and that learning them facilitates understanding the complex question. We employed different techniques to introduce those biases to the model in a more explicit manner, *e.g.,* data augmentation is used in Chapter 3 to enrich the training set with basic concepts, knowledge transfer is used in Chapter 4 to transfer the knowledge of basic concepts to the model, and curriculum learning is used in Chapter 5 to learn basic concepts prior to complex ones.

In this chapter, we address a problem that may arise when a model that has already learned basic concepts faces complex questions. This problem may happen because of distributional mismatch. Having demonstrated the significance of learning basic concepts as an effective foundation for complex questions, we can now focus on dealing with the distributional difference between the basic and complex questions. We propose to resolve this mismatch by seeking help from similar complex examples in the training set. In other words, the model is trained to tune the distribution for every new complex question using a few-shot learning technique. We conduct experiments to evaluate our idea and analyze the results to show its effectiveness.

6.1 Introduction

Modern VQA models, similar to many other deep neural networks, typically demand large-scale datasets. We discussed the fact that collecting a large amount of labeled data is infeasible in many settings. Moreover, VQA generally is a low data problem considering that potentially an infinite number of question can be asked about an image. Therefore, we investigate VQA in a low labeled data scenario. A learner is able to learn simple questions from a small amount of data, however, understanding complex relationships in questions is challenging in such settings. We investigate the main reasons behind the fact that when having access to a large-scale dataset, a learner eventually acquires complexity after seeing a large variety of examples. More specifically, we try to investigate the question, of "what makes a learner effectively generalize to complex examples from learning simple examples when accessing a large amount of data?" with the hope that the answer to this question suggests a solution for learning complexity with a small amount of data.

A hypothetical answer is that the learner captures a distribution over simple data in early training which may be different from that of complex data. This distribution is then adapted to fit the complex examples in addition to the simple ones by seeing a large variety of data. Distribution adaptation is less likely to successfully occur in a low data setting. Inspired by this observation, we hypothesize that helping distribution adaptation in low data scenarios can facilitate generalization on complex examples and hence enhance the performance. Similar to the previous chapters, the underlying assumption here is the compositionality feature of questions in the sense that complex questions can be divided into simple chunks which are easier to understand. A model can learn the simple chunks by learning simple examples which provides a strong foundation to learn complexities.

The generalization from simple to complex examples requires adaptation of the learned data distribution which can smoothly happen during training with a lot of data. To compensate for the lack of data and address the problem of distributional adaptation in a low data scenario, we formulate VQA as a meta-learning problem. Meta-learning or learning to learn

refers to algorithms that aim at learning good initialization that can be fine-tuned on various tasks with minimal training data (Finn, Abbeel, & Levine, 2017; Nichol, Achiam, & Schulman, 2018). Such algorithms in fact learn from the predictions on other tasks to make prediction on a new task.

In this work, the meta-learning algorithm learns to use a set of examples provided at the test time to answer a target question. Those examples selected from the training set are referred to as the *support set*. The support set is not fixed and may be different for each target question. It can expand the model's ability to predict the answer by providing additional information to the model at test time. Additionally, the possibility of providing the model with more information without retraining and the ability of exploiting that information substantially enhance the system's scalability and practicality. In this chapter, inspired by Hua, Li, Haffari, Qi, and Wu (2020), we investigate the model-agnostic meta-learning algorithm (MAML) (Finn et al., 2017) for VQA tasks. Specifically, in our meta-learning setting, we view each training example as a query task and make use of similar training examples as the support set to adapt distribution to fit the query. In other words, for each query, we create a model devoting to answering the question. Meta-learning approaches in low-resource settings typically use high resource tasks for training such as (Dou, Yu, & Anastasopoulos, 2019), In contrast to them, our approach only relies on small training data and does not require additional labeled examples. We evaluate the effectiveness of the proposed approach on the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017).

6.2 Background and Related Work

In this section, we briefly go over some fundamental concepts in meta-learning as well as their use in VQA. Meta-learning has a long history in machine learning, however its resurgence in contemporary deep learning has led to huge success in recent research. Meta-learning has the potential for implementing different techniques such as knowledge transfer, data efficiency and unsupervised learning where the main criticisms of current deep learning can be alleviated. Meta-learning is shown successful in both multitask learning, where task-

agnostic inductive biases are extracted from a set of similar tasks (Finn et al., 2017), and is used to improve learning of new tasks in single-task scenarios where a single problem is iteratively solved and improved over multiple repetitions (Hua et al., 2020).

A conventional machine learning approach is to train a task-specific model using task-specific annotated examples. In contrast, in a meta-learning framework, tasks are considered to be training examples. To solve a new task, a meta-learning algorithm requires a large number of tasks with which to train a model to adapt to all those training tasks (Hospedales, Antoniou, Micaelli, & Storkey, 2021). The model learns to solve those tasks in an inner/lower learning loop, known as *base learning* while it updates the parameters to fit the new task in an outer/upper learning loop called *meta learning*. The resulting model is expected to perform well on the new task. Similar to regular supervised learning, in meta-learning, the target task is supposed to be of the same distribution as the training tasks $p(\mathcal{T})$.

There are three common approaches to meta-learning, namely metric-based (Santoro, Bartunov, Botvinick, Wierstra, & Lillicrap, 2016; Munkhdalai & Yu, 2017), model-based (Koch, 2015; Sung et al., 2018), and optimization-based (Ravi & Larochelle, 2017; Nichol et al., 2018; Finn et al., 2017). Metric-based approaches are non-parametric learning algorithms that compare the training and validation points to predict the label of matching training points. In model-based methods, the meta-learning is synthesized in a feed-forward pass of a single model rather than iterative optimization. The model embeds the current training set into an activation state in the inner learning algorithm and makes predictions for test data based on this state. Typical architectures in such approaches include convolutional networks (Mishra, Rohaninejad, Chen, & Abbeel, 2018), recurrent networks (Ravi & Larochelle, 2017) and memory-augmented neural networks (Santoro et al., 2016; Graves, Wayne, & Danihelka, 2014). In optimization-based methods, meta-knowledge is extracted by improving optimization performance at the inner level. A famous example of such methods is the MAML algorithm (Finn et al., 2017). We exploit its optimization adjustment feature in our algorithm. Algorithm 4 summarizes the MAML approach where α is a step size hyperparameter and β is the meta step size, L indicates the loss functions and f_θ is the

objective function.

There are few works in VQA using meta-learning. Teney and Hengel (2018) formulate VQA as a meta-learning task. They separate the information required for answering questions from the method by defining a set of dynamic weights that are determined by processing the support set. At test time, the dynamic weights are retrieved based on the similarity of the test example and the support set and used to predict the answer. As another example, Nguyen et al. (2019) combine MAML with a denoising autoencoder for medical VQA.

Algorithm 4 Training procedure.

```
# Meta training stage
Pre-train model parameters θ with unlabeled datasets.
while not done do
    Sample batch of tasks {T_i} ~ p(T) from high resource tasks
    for all T_i do
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    # Meta testing stage
    Update θ with θ = θ − β∇_θ Σ_{T_i ~ p(T)} L_{T_i}(f_{θ'_i})
end while
Fine-tune θ on the target task.
```

6.3 VQA in a Meta-Learning Setting

VQA is traditionally a supervised learning problem in that a model is trained to map an input questions-image pair to scores over candidate answers. The model is trained to maximize the likelihood of the correct answer over a training dataset $\mathcal{D}$ of triplets $<q, i, \hat{a}>$ where $\hat{a} \in \mathcal{A}$ represents the ground truth answer from a set of $\mathcal{A}$ possible answers. At test time, the model is evaluated on another triplet $<q_t, i_t, \hat{a}_t>$ from an evaluation set. The model predicts the answer which is compared to the ground truth answer $\hat{a}_t$ for evaluation purposes.

Inspired by (Hua et al., 2020), we extend the above formulation of VQA to a MAML setting. In our formulation, we view each training example $<q, i, \hat{a}>$ as a query sample of

a pseudo task τ for which we aim to learn a particular model dedicated to solving the task. A pseudo task is composed of a training example $<\boldsymbol{q}_{meta}, \boldsymbol{i}_{meta}, \hat{\boldsymbol{a}}_{meta}>$ and a *support set*. We define a support set as an additional set $\mathcal{S}$ of top-N similar triplets to the query example $<\boldsymbol{q}_s, \boldsymbol{i}_s, \hat{\boldsymbol{a}}_s>$. According to our definition, the support set includes the training examples, *i.e.,* $\mathcal{S} \subset \mathcal{D}$, however, it can include novel examples at test time.

Formally, $\mathcal{T}_{pse}$ is a set of pseudo tasks τ that is defined as $\mathcal{T}_{pse} = \{\tau | <\boldsymbol{q}_{meta}, \boldsymbol{i}_{meta}, \hat{\boldsymbol{a}}_{meta}> ; \boldsymbol{s}_{meta}\}$ where $|\mathcal{T}_{pse}| = |\mathcal{D}|$ since we create a pseudo task for every training example, and $\boldsymbol{s}_{meta} = \{<\boldsymbol{q}_s, \boldsymbol{i}_s, \hat{\boldsymbol{a}}_s>\}$. In fact, $\boldsymbol{s}_{meta}$ is considered meta-training data that the learner uses to adapt a particular model, and $<\boldsymbol{q}_{meta}, \boldsymbol{i}_{meta}, \hat{\boldsymbol{a}}_{meta}>$ as the meta-testing data to assess the model. The notion of similarity of the query and the support examples can be defined based on different criteria including Euclidean distance of feature vectors in an embedding space. In this work, we use cosine similarity to select the support examples. In the meta-learning step, the model predicts the answers to $<\boldsymbol{q}_s, \boldsymbol{i}_s>$ from $\boldsymbol{s}_{meta}$ based on the current parameters θ of the model. The loss of predicted answers compared to the ground-truth answers $\hat{\boldsymbol{a}}_s$ leads to gradient updates that fine-tune the current model to obtain a task-specific model with the parameter θ'. In the meta-testing step, the answer of $<\boldsymbol{q}_{meta}, \boldsymbol{i}_{meta}>$ is produced based on θ' and is evaluated with respect to $\hat{\boldsymbol{a}}_{meta}$ to update θ. Similarly, in the inference stage, each test example $<q_t, i_t>$ is considered a new individual task. The support set of the test example is retrieved from the training dataset to form the meta-training data which is used to fine-tune parameters θ' that fit the test example and infer the answer.

6.3.1 Meta Training and Testing

Now, we explain the model's training procedure. At the beginning of a training step, a mini-batch of tasks are sampled from the pseudo task set $\mathcal{T}_{pse}$. Following the distribution of tasks for sampling makes the model generalize to unseen tasks in the training dataset. To fully use the training dataset and decrease training time, we view VQA as a problem under few-shot learning conditions. We make use of Meta-RL techniques (Finn et al., 2017) to adapt the parameters to a new task with a few training examples. Meta-RL aims to meta-learn a

Algorithm 5 Meta Training procedure.

Input: Dataset
Output: Meta-learned parameters θ^*
while not done **do**
  Sample batch of tasks $\mathcal{T}' \sim \mathcal{T}_{pse}$
  for $\tau \sim \mathcal{T}'$ **do** $\mathcal{L}_\theta \leftarrow 0$
    # Meta-training
    for each $< \boldsymbol{q}_s, \boldsymbol{i}_s, \hat{\boldsymbol{a}}_s > \in \boldsymbol{s}_{meta}$ **do**
      $a \leftarrow \mathcal{E}(\boldsymbol{q}_s, \boldsymbol{i}_s; \theta)$
      $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \log p_\theta(a)$ ▷ Evaluate Executor loss
    end for
    $\theta'_\tau \leftarrow \theta + \eta_1 \nabla_\theta \mathcal{L}_\theta$ ▷ Get adapted parameters
    # Meta-testing
    $J_{meta}(\theta'_\tau) \leftarrow \log p_{\theta'_\tau}(\mathcal{E}(\boldsymbol{q}_s, \boldsymbol{i}_s; \theta'_\tau))$
  end for
  $\theta \leftarrow \theta + \eta_2 \nabla_\theta \sum_{\tau \in \mathcal{T}'} J_{meta}(\theta'_\tau)$ ▷ Update Parameters
end while
return $\theta^* \leftarrow \theta$

model that can quickly learn optimized parameters for a new task τ. We employ gradient-based meta-learning to solve the problem such that it can obtain the optimized model after performing a few steps of vanilla gradient descent optimization. The meta-learning process is split into two steps to solve a task, namely a mini-training loop, meta-training, and a test step, meta-testing. Assume we aim to solve the pseudo task τ, which includes N meta-training examples in $\boldsymbol{s}_{meta}$ that are the most similar to the query sample $<\boldsymbol{q}_{meta}, \boldsymbol{i}_{meta}>$. The model first predicts the answers to the questions in the support set and then the model parameters are updated by N gradient descent steps for each support example according to the equation:

$$\theta'_\tau \leftarrow \theta - \eta_1 \nabla_\theta \mathcal{L}_\theta \tag{6.1}$$

where $\mathcal{L}$ is the loss function and η_1 is the gradient discount factor. During meta-testing, the answer to the query example is further produced by θ'. For the given task τ, the loss is calculated based on the adapted parameters θ'; thus the objective is:

$$J_{meta}(\theta'_\tau) \leftarrow \log p_{\theta'_\tau}(\mathcal{E}(\boldsymbol{q}_s, \boldsymbol{i}_s; \theta'_\tau)) \tag{6.2}$$

The parameters of the generic model θ are then trained by maximizing the objective $J(\theta'_\tau)$:

$$\theta \leftarrow \theta + \eta_2 \nabla_\theta \sum_{\tau \in \mathcal{T}'} J_{meta}(\theta'_\tau) \tag{6.3}$$

In each optimization step, since the support set contains N samples, N gradient adaptation steps are performed to update θ'. Meanwhile, one optimization step updates θ based on the evaluation of θ'. Algorithm 5 summarizes the meta-learning approach. At the inference stage, for each test example, we create a pseudo task, similar to the meta-learning process. The top-N examples similar to the test example are retrieved from the training dataset to form the support set s_{test}, and are used to obtain the adapted model $\theta^{*'}$, starting from the meta learned parameters θ^*. The adapted model is then used to produce the final answer.

6.4 Experiment

Dataset Similar to the previous chapters, here we evaluated our approach on the CLEVR dataset (Johnson, Hariharan, van der Maaten, Fei-Fei, et al., 2017) providing a training set with $70k$ images, and roughly 10 questions and answers for each image hence $\sim 700k$ questions in total. The questions are machine-generated associated with the images with synthetic scenes. The `val` split contains $\sim 150k$ questions and $15k$ unique images.

6.4.1 Setup

To simulate a low data scenario, we select five small subsets from the original training set with different sizes described as the percentage of the full training set size. These subsets are referred to as s-CLEVR$_x$ where $x \in \{5, 10, 15, 20, 30\}$ indicates the size of the subsets, *e.g.,* s-CLEVR$_{20}$ consists of randomly selected 20% of the training data. We conducted our experiments in two stages: *creating tasks* and *training*. The first stage includes retrieving the support set for every training and test example and forming tasks while in the second stage we use the created tasks to train the model according to Algorithm 5. We use the *execution*

engine $\mathcal{E}$ to assess the proposed meta-learning approach's impact on the VQA performance in comparison to some baselines. The first baseline is obtained by vanilla training the model on s-CLEVR subsets in a standard supervised manner. The other baselines come from training on various tasked sets with different sizes and selection methods of support sets.

As the inputs, the model requires questions/programs and images. To extract image features, we use the output of the $conv4$ of ResNet-101 (K. He et al., 2016) pretrained on ImageNet (Deng et al., 2009). We finally evaluate the trained model on $10k$ examples from the CLEVR `valset`.

Support Example Retrieving

Similarity calculation. The support examples are meant to teach the model how to answer similar query examples. The notion of similarity of examples in VQA can be defined based on different criteria including lingual and visual dimensions. We experiment with various definitions of similarity based on questions similarity as well as image similarity. The following sections explain them in detail. To calculate questions similarity, we map all questions to an embedding space using the Sentence-BERT language model (Reimers & Gurevych, 2019) which fine-tune BERT in a siamese network architecture to generate sentence embedding. Then cosine similarity measure is used to score the similarity of the questions. Image similarities are computed as follows: A ResNet-101 ($conv4$) extracts the image features which is followed by a 2D average pooling layer 5×5 to obtain down-sampled features. Similar to questions, the similarity of the images is measured by a cosine similarity score.

Support set selection. We tried four different methods of selecting support examples based on: 1) question similarity $Sim_{question}$; 2) image similarity Sim_{image}; 3) combination of question and image similarity $Sim_{question+image}$; 4) a two-stage question-image similarity $Sim_{question/image}$. For the combination similarity method, the question similarity scores of a specific example are added to the corresponding scores of the images of the same example. In the case of the two-stage method, we first select $2N$ most similar examples to the query

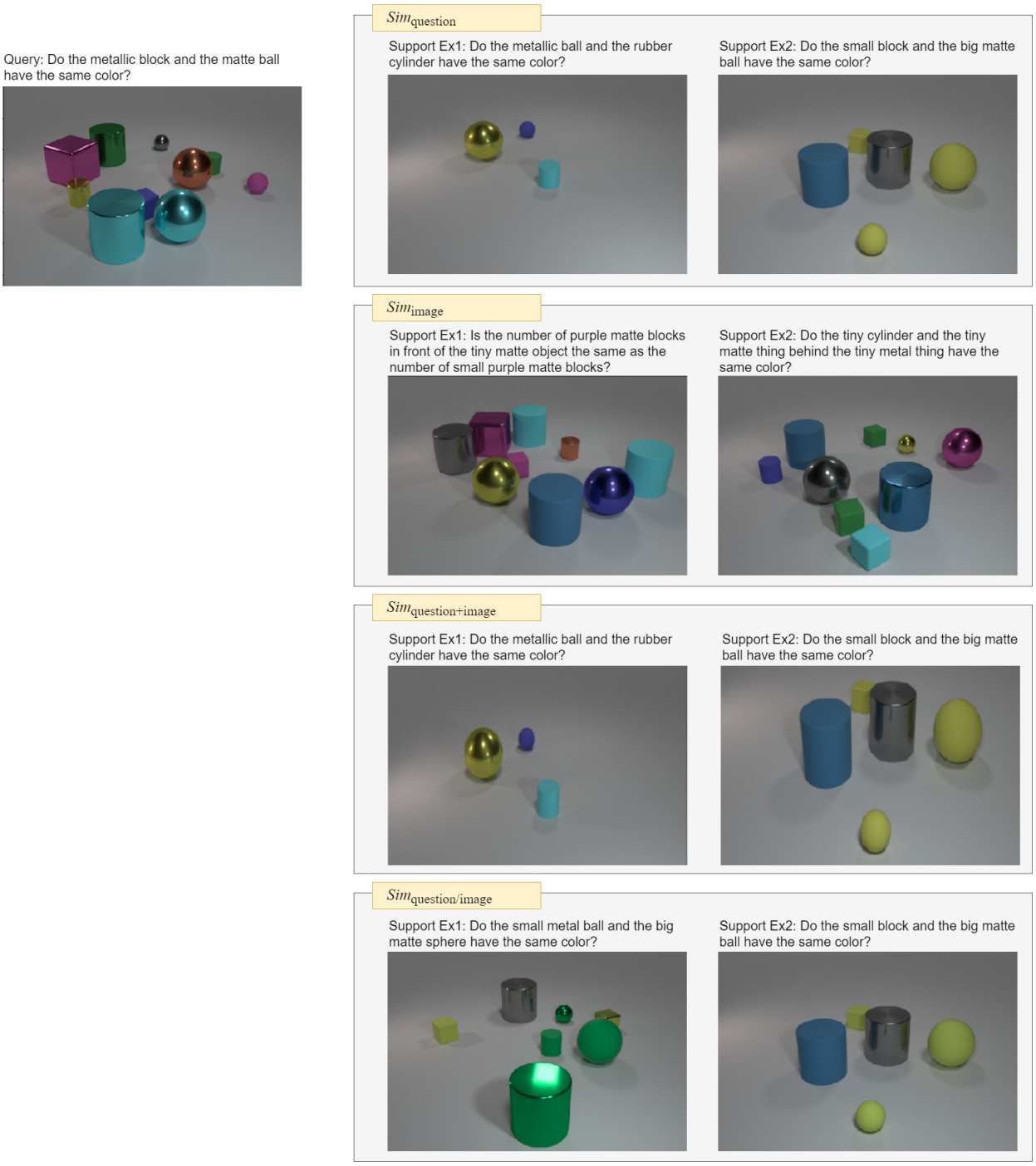


Figure 6.1: Examples from the support sets selected with different similarity criteria for a single query example.

example based on question similarity, and from those examples we pick N examples based on image similarity where N is the support set size. Figure 6.1 shows two examples of the

support set selected for a single query example with the mentioned similarity criteria. As seen, the selected examples with $Sim_{question}$ are similar to those of $Sim_{question+image}$ which suggests that question similarity has a higher impact compared to image similarity.

Creating tasks As described earlier, each task includes a set of query examples and a set of supporting examples. In all of the experiments, we consider only one query example in each task while the number of supporting examples N is given where $N \in \{1, 3, 5\}$. Every training sample is considered a potential query example for which we select a support set using the methods described above from the training set. Although it is possible, we do not use any external data examples in the support set. For each subset s-CLEVR$_x$, the query and supporting example are limited to that subset, *i.e.,* if the current low data scenario is based on s-CLEVR$_{20}$, we assume there is no training data available except s-CLEVR$_{20}$, thus all the tasks are created from the same data examples.

6.4.2 Results and Discussion

	5%	10%	15%	20%	30%
$Vanilla$	46.91 ± 0.52	49.90 ± 0.37	52.32 ± 0.30	54.24 ± 0.26	$\mathbf{87.70 \pm 0.14}$
$Sim_{question}$	$\mathbf{49.02 \pm 0.52}^*$	50.32 ± 0.37	52.09 ± 0.30	53.62 ± 0.26	80.78 ± 0.17
Sim_{image}	$48.54 \pm 0.52^*$	$\mathbf{51.56 \pm 0.37}^*$	$\mathbf{52.41 \pm 0.30}$	$\mathbf{54.30 \pm 0.26}$	80.86 ± 0.17
$Sim_{question+image}$	$48.86 \pm 0.52^*$	$50.75 \pm 0.37^*$	51.93 ± 0.30	52.81 ± 0.26	67.84 ± 0.20
$Sim_{question/image}$	$48.58 \pm 0.5^*$	49.95 ± 0.37	51.77 ± 0.30	53.13 ± 0.26	83.79 ± 0.16

Table 6.1: The accuracy on CLEVR `val` split for different methods of support set selection and different training set. Error bars give 95% confidence bound for estimated accuracy from training. Best accuracy for each training set is shown in bold font while the asterisked numbers show the statistically significant improvements. The support set size for all experiments is 3.

Table 6.1 presents the accuracy of our proposed meta-learning algorithm on different training sets and different support sets. Each row indicates a method of support set selection except for the first row that shows the results of vanilla supervised training of the model with no support set. All the experiments in the table use a support set of size 3. As seen, meta-

learning makes statistically significant improvements in some scenarios of s-CLEVR$_5$ and s-CLEVR$_{10}$ asterisked in the table whereas it marginally improves the performance in other subsets except s-CLEVR$_{30}$ where the vanilla training gains the best results. This finding suggests that the effect of support set examples should be carefully investigated. Among the support set selection methods, the image similarity criterion seems to have the highest effect in improving performance.

The error margins in Table 6.1 are calculated using $\rho \pm z^* \sqrt{\frac{\rho(1-\rho)}{n}}$ where ρ is the sample proportion and n is the sample size, and z^* is the appropriate value from the standard normal distribution for the desired confidence level. In our case that confidence interval is 95%, z^* value equals 1.96.

Task-overfitting phenomenon

The difference of the best result for the meta-learning method and for vanilla training is decreasing for all training sets from left to right, *i.e.,* 2.11 for s-CLEVR$_5$, 1.66 for s-CLEVR$_{10}$, 0.09 for s-CLEVR$_{15}$, 0.06 for s-CLEVR$_{20}$, and -3.91 for s-CLEVR$_{30}$. This suggests that as the size of the training set is increasing the effect of meta-learning is decreasing. These results may be explained by a phenomenon called *task-overfitting* introduced recently in (Yin, Tucker, Zhou, Levine, & Finn, 2020). The authors describe that it is a form of overfitting different from standard overfitting in that the task can be precisely inferred from test data alone, then the meta training data, *i.e.,* the support set examples, can be ignored while the meta training loss is still low. Such a situation provides a chance for overfitting where the meta-learner can generalize on meta-training data but fails to adapt to novel data from a new task. Yin et al. (2020) mention that current meta-learning algorithms overcome this problem implicitly by properly designing the meta-training tasks to be diverse.

Inspired by the examples provided in this study, we can argue that since the number of object attributes in the CLEVR dataset is limited and the support examples are selected to be most similar to the meta test example, the meta-learning system memorizes specific information in the training and associates it with the answers, leading it to ignore the other

information provided in the image and questions. As a result, it may achieve highly accurate answers on the meta-training set but fail to generalize on new examples effectively. (Yin et al., 2020) proposes a regularization algorithm to alleviate this problem as *meta regularization* that encourages the meta-learner to use meta-training data by softly restricting the flow of information from meta-parameters when predicting the test labels.

Effect of Support Set Size

	spt1	**spt3**	**spt5**
s-CLEVR$_{15}$	52.05	52.41	51.21
s-CLEVR$_{20}$	54.05	54.30	53.49

Table 6.2: The accuracy on CLEVR `val` split for different support set sizes of 1, 3 and 5 denoted as **spt1**, **spt3**, and **spt5** when the model is trained on the tasks from s-CLEVR$_{15}$ and s-CLEVR$_{20}$. The support examples are selected using Sim_{image} method.

To investigate the effect of the size of support sets we conduct an ablation experiment with three different sizes of support sets. We meta-train s-CLEVR$_{15}$ and s-CLEVR$_{20}$ on the tasks with 1, 3, and 5 examples in support sets. The similarity score in these experiments is based on cosine similarly scores of the images, *i.e.,* Sim_{image}. Table 6.2 shows the results of the ablation experiment. The header row indicates the size of the support sets and the next two rows are the evaluation accuracy on CLEVR `val`set. As the support sets contains the most similar examples to the query example, the support sets of size 1, *i.e.,* **spt1** are in fact a subset of the larger support sets. Note that this condition does not hold for $Sim_{question/image}$ as the selection is done in two stages. As can be seen from the table, the results are generally close that suggests exploring a wider range of support set sizes. Additionally, we observe that the accuracy is decreased in **sp5** compared to **sp1** and **sp3** in both 15% and 20% subsets. This can also be explained with the task-overfitting phenomenon in that more number of the most similar examples in the support set makes the model more confident in ignoring the support examples and only relying on the target examples since the model does not visit diverse enough examples.

6.5 Summary

In this chapter, we formulate VQA as a meta-learning task to address the problem of distributional mismatch between basic concepts and complex questions. We tackle this problem by adapting the model to a target example using its most similar examples in the training set. Every example in the training set when accompanied with a small set of similar examples – *i.e.,* the support set– is called a task which is the input to our meta-learning algorithm. Inspired by MAML, our algorithm consists of meta-training and meta-testing steps in each iteration of standard training. During meta-training the model tunes the parameters according to the provided similar examples and, at the meta-testing step, it makes a prediction for the target example of the task. We conducted experiments for four different criteria of selecting support sets. The results show that, with respect to the *vanilla* baseline, statistically significant improvement is achieved in some cases while the improvement is marginal in some others. We analyze and discuss the possible cause of the unexpected results and suggest that resolving them can lead to higher performance.

7 | Conclusions

7.1 Summary of the Thesis

In this thesis, we study a multimodal task called Visual Question Answering (VQA) where given an image and a natural language question about the image (*e.g., "what kind of animal is this?"*, *"is it safe for the kid to play here?"*), the machine aims to automatically predict a natural language answer (*"elephant"*, *"yes"*). Motivated by the poor performance of VQA models with insufficient data, we investigate VQA in low data scenarios. The primary contribution of this thesis is using the compositionality of questions to improve the task of VQA in low data setting without relying on any additional labeled or unlabeled data.

Chapter 3 We presented a data augmentation strategy aimed at explicitly injecting certain inductive biases in order to increase the VQSA performance in low data scenarios. The inductive biases are based on the question's intrinsic compositionality, which allows a complex question to be split into smaller sub-questions. using this feature, we augment the training set with basic questions that are easier to understand and learn. The data augmentation method automatically generates the questions according to a number of pre-defined templates using shallow annotation of the image scenes relying only on the existing training set with no need for additional labeled data. We demonstrate improvements over the baselines in the VQA performance of four training sets by a large margin.

Chapter 4 We proposed to take advantage of question compositionality and use self-supervised learning techniques to learn intermediate visual representations to answer questions. Intermediate representations can be considered as the answers to the sub-questions of a complex question. More specifically, we employ a contrastive algorithm to pretrain the modules of a compositional VQA model and fine-tune the pretrained modules to answer the given questions. In the pretraining phase, the modules acquire visual knowledge by learning a similar representation for an image and its transformed version. Unlike many vision tasks, the transformation methods should be chosen carefully for a VQA task in order to preserve the correct relationships between the questions, image, and answer. For this, we define an invariance set for each module containing the valid transformations. An invariance set for a sequence of modules is obtained by taking an intersection of all modules in the sequence. We experiment with three different methods of module selection for pretraining and demonstrate that our proposed method significantly outperforms the baseline performance.

Chapter 5 Since learning complexity in the absence of sufficient data is challenging, we propose to encourage the model to effectively learn basic concepts and build a strong foundation for understanding complex data. For this purpose, we use curriculum learning to order the training samples and feed them to the model according to a pre-specified ordering with the hope of maximizing the usage of training samples by performing supervision on the order of training data that are fed into the model. The underlying idea of CL is to begin learning from easy examples, and gradually consider harder ones, rather than using examples in a random sequence which means the ranking criteria plays a key role in a CL algorithm.

We study and analyze the performance of three ranking criteria: (1) a length-based criterion which considers longer questions to be more complex than shorter questions, and ranks the examples in increasing order of their program; (2) a criterion based on an answer hierarchy which organizes all possible answers from coarse to fine; and (3) a criterion that relies on model loss for deciding the hardness level of the examples and ranking them accordingly. In addition to the ranking heuristics, we propose a CL training strategy for

each criterion. We also argue that under CL training in low data regimes, a model is very susceptible to overfitting and poor generalization. Carefully employing a regularizer is crucially important to prevent the model from becoming over-confident in the training data. We demonstrate that the proposed training strategies, when coupled with *L2-norm* regularization, lead to a significant improvement in performance.

Chapter 6 We look into the problem of distributional mismatch between basic concepts and complex questions that can arise when a model with basic knowledge faces a complex example. We address this problem by taking advantage of examples similar to the target one in the training set. Specifically, the model is trained to adapt the distribution to every new question with a few-shot learning approach. Inspired by MAML, we formulate VQA as a meta-learning algorithm that consists of meta-training and meta-testing steps for each iteration of standard training tasks. We define a task as a target training example accompanied by a small set of similar examples called the support set. The meta-training step uses the support set examples to tune the distribution for predicting the answer to the target question. We create the training tasks with four different similarity criteria for selecting support set examples, and use them to train the model. The results show statistically significant improvements in some cases and marginal improvements in some others with respect to the baseline. We analyze and discuss the possible cause of the unexpected results and suggest that resolving them can lead to higher performance.

7.2 Future Directions

This section briefly mentions a number of insights and possible research directions gained from this thesis.

Low data VQA: Despite tremendous progress in VQA, the fact that VQA is basically a low data task is overshadowed in the community by the excitement of new large-scale datasets. To the best of our knowledge, this thesis is the first step toward studying low data

VQA. Further study of this problem will push VQA towards its goals, such as aiding visually impaired people or helping robots in rescue missions. In such cases, the models are highly likely to be provided with only a small number of examples of a new task requiring it to adapt quickly. Although we restricted our study to the use of a small training set with no help from additional data, future studies may investigate using available data. Recently, there has been a lot of interest in developing pretrained multimodal models that may be utilized as a warm-start for low data VQA problems. Such a warm-starting supply fundamental and general information, allowing the model to successfully fine-tune on a downstream task with minimal effort. Using pretrained models may not be as straightforward in low data settings as it is in general settings, and may pose certain issues such as domain adaptation. This is because, unlike popular approaches to the use of pretrained models, models in low data settings have access to a limited quantity of data, which may not be sufficient for optimizing the distribution of data for the new task.

Pretrained multimodal compositional models: The success of large pretrained multimodal models and their availability has gained significant attention from the VQA community recently. Hence, fewer works have studied compositional models. Although pretrained models have been proved to outperform many prior works in various vision-language tasks, they are black box models with low interpretability. The larger such models grow in size, the harder interpreting their behavior becomes. This problem exacerbates when the model fails to generalize on new tasks where determining the cause is very challenging. It is time that the VQA community starts investing its effort in creating pretrained multimodal compositional models. We think a pretrained compositional model will give researchers the chance to take advantage of individual modules according to the task at hand while understanding the reasoning process will be clear allowing a more efficient fine-tuning on novel downstream tasks. In Chapter 4 we pretrained a neural module network using the CLEVR dataset. Such compositional methods are flexible enough to be applied to a variety of VQA tasks as mentioned in Section 2.2.4. The limitation here is how a VQA task can be translated into a program that is convertible to a module sequence. Apart from that, every module in

such networks can function individually or in relation to the other modules.

Using real-world images The experiments in this thesis are performed on a synthetic dataset, *i.e.,* CLEVR. Given that the proposed ideas in this research do not pose any restrictions to a specific data characteristic, they are worth executing on the dataset with real-world images such as VQA and GQA. Unfortunately this was beyond the time schedule of this study. We hypothesized that the results on such datasets will be comparable to the results reported in this thesis. Moreover, some presented methods such as the meta-learning algorithm may achieve higher performance because a dataset with a high variety of objects and attributes in images can prevent task-overfitting.

Generating effective programs The current VQA modular approaches that produce programs as a layout for assembling modules such as (Andreas et al., 2016b) and (Johnson, Hariharan, van der Maaten, Hoffman, et al., 2017) (see Section 2.2.4 for more details) use only questions as inputs to the program generating component. However, the program can be more efficient if the program generator has access to the image in addition to the questions. Assume the question *"what color is the couch next to the table on the left of the room?"* in two examples with different images: 1) an image with only one couch in the room, and 2) an image with three couches around the room. The corresponding programs for both examples will be the same if the images are not considered whereas the reasoning process for the first example with only one couch should obviously be simpler. A program generator can produce more efficient programs if it can take the images' information into consideration.

Combining the proposed techniques in this thesis This thesis aims to show the effect of each proposed method individually. As most of them are orthogonal, they can be used in combinations with each other. This is interesting future work for expanding the proposed ideas in this thesis. For instance, the data augmentation method in Chapter 3 provides more data examples to the model in addition to the original dataset. It is a general approach that hypothetically can help any model in low data settings. This method can be combined with

the methods proposed in other chapters. However, experimentation is required to determine how impactful the combination of the methods is. Also, the resulting modules from the self-supervised pretraining method in Chapter 4 can be used to warm-start the model weights for the training methods proposed in Chapters 5 and 6.

References

Agarwal, V., Shetty, R., & Fritz, M. (2020, May). Towards Causal VQA: Revealing and Reducing Spurious Correlations by Invariant and Covariant Semantic Editing. *arXiv:1912.07538 [cs]*. Retrieved from http://arxiv.org/abs/1912.07538 (arXiv: 1912.07538)

Agrawal, A., Batra, D., Parikh, D., & Kembhavi, A. (2018, June). Don't Just Assume; Look and Answer: Overcoming Priors for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4971–4980). doi: 10.1109/CVPR.2018.00522

Alberti, C., Ling, J., Collins, M., & Reitter, D. (2020). Fusion of detected objects in text for visual question answering. In *Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 2131–2140). Association for Computational Linguistics. (Publisher: Association for Computational Linguistics)

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6077–6086). Retrieved from https://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Bottom-Up_and_Top-Down_CVPR_2018_paper.html

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016a). Learning to compose neural networks for question answering. *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016b). Neural module networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 39–48).

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., & Parikh, D. (2015). Vqa: Visual question answering. In *IEEE international conference on computer vision* (pp. 2425–2433).

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003, January). The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal on Computing*, *32*(1), 48–77. Retrieved from https://doi.org/10.1137/S0097539701398375 doi: 10.1137/S0097539701398375

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. In K. Aberer et al. (Eds.), *The Semantic Web* (pp. 722–735). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-540-76298-0_52

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate: 3rd International Conference on Learning Representations, ICLR 2015. In *3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.

Baroni, M. (2020, February). Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *375*(1791), 20190307. (Publisher: Royal Society) doi: 10.1098/rstb.2019.0307

Bengio, Y. (2012, June). Deep Learning of Representations for Unsupervised and Transfer Learning. In *ICML Workshop on Unsupervised and Transfer Learning* (pp. 17–36). JMLR Workshop and Conference Proceedings. Retrieved from https://proceedings.mlr.press/v27/bengio12a.html (ISSN: 1938-7228)

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*(Feb), 1137–1155. Retrieved

from https://www.jmlr.org/papers/v3/bengio03a.html

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In *International Conference on Machine Learning* (pp. 41–48). Montreal, Quebec, Canada: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1553374.1553380 doi: 10.1145/1553374.1553380

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., & Veit, A. (2021). Understanding Robustness of Transformers for Image Classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2021)* (pp. 10231–10241). Retrieved 2022-02-26, from https://openaccess.thecvf.com/content/ICCV2021/html/Bhojanapalli_Understanding_Robustness_of_Transformers_for_Image_Classification_ICCV_2021_paper.html

Bills, A. (1934). *General experimental psychology*. Longmans, Green and Co.

Bittlingmayer, A. (2018). *NoiseMix: data generation for natural language.* Retrieved from https://github.com/noisemix/noisemix

Bitton, Y., Stanovsky, G., Schwartz, R., & Elhadad, M. (2021). Automatic Generation of Contrast Sets from Scene Graphs: Probing the Compositional Consistency of GQA. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 94–105). Online: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/2021.naacl-main.9 doi: 10.18653/v1/2021.naacl-main.9

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008, June). Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD international conference on Management of data* (pp. 1247–1250). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1376616.1376746 doi: 10.1145/1376616.1376746

Bordes, A., Usunier, N., Chopra, S., & Weston, J. (2015). Large-scale Simple Question Answering with Memory Networks. *arXiv:1506.02075 [cs.LG]*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei,

D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Canevet, O., & Fleuret, F. (2015, February). Efficient Sample Mining for Object Detection. In *Asian Conference on Machine Learning* (pp. 48–63). PMLR. Retrieved from http://proceedings.mlr.press/v39/canevet14a.html

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *European Conference on Computer Vision (ECCV 2020)* (pp. 213–229). Cham: Springer International Publishing. doi: 10.1007/978-3-030-58452-8_13

Chen, K., Wang, J., Chen, L.-C., Gao, H., Xu, W., & Nevatia, R. (2015). ABC-CNN: An Attention Based Convolutional Neural Network for Visual Question Answering. In *CoRR*. Retrieved from http://arxiv.org/abs/1511.05960 (arXiv: 1511.05960)

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning* (pp. 1597–1607). PMLR. Retrieved from http://proceedings.mlr.press/v119/chen20j.html

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. E. (2020). Big self-supervised models are strong semi-supervised learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 22243–22255). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/file/fcbc95ccdd551da181207c0c1400c655-Paper.pdf

Chen, X., Fan, H., Girshick, R., & He, K. (2020, March). Improved Baselines with Momentum Contrastive Learning. *arXiv:2003.04297 [cs]*. Retrieved from http://arxiv.org/abs/2003.04297 (arXiv: 2003.04297)

Chen, Y.-C., Li, L., Yu, L., Kholy, A. E., Ahmed, F., Gan, Z., … Liu, J. (2020). UNITER: UNiversal Image-TExt Representation Learning. In *Computer Vision – ECCV*. Retrieved from https://arxiv.org/abs/1909.11740v3 doi: 10.1007/978-3-030-58577-8_7

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, October). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D14-1179 doi: 10.3115/v1/D14-1179

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning Augmentation Policies from Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from https://arxiv.org/pdf/1805.09501.pdf doi: 10.1109/CVPR.2019.00020

Damodaran, V., Chakravarthy, S., Kumar, A., Umapathy, A., Mitamura, T., Nakashima, Y., … Chu, C. (2021, January). Understanding the Role of Scene Graphs in Visual Question Answering. *arXiv:2101.05479 [cs]*. Retrieved from http://arxiv.org/abs/2101.05479 (arXiv: 2101.05479)

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M. F., … Batra, D. (2017). Visual Dialog. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 326–335). Retrieved from https://openaccess.thecvf.com/content_cvpr_2017/html/Das_Visual_Dialog_CVPR_2017_paper.html

Day, O., & Khoshgoftaar, T. M. (2017, September). A survey on heterogeneous transfer learning. *Journal of Big Data*, *4*(1), 29. Retrieved from https://doi.org/10.1186/s40537-017-0089-0 doi: 10.1186/s40537-017-0089-0

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009, June). ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 248–255). (ISSN: 1063-6919) doi: 10.1109/

CVPR.2009.5206848

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://aclanthology.org/N19-1423 doi: 10.18653/v1/N19-1423

Doersch, C., Gupta, A., & Efros, A. A. (2015, December). Unsupervised Visual Representation Learning by Context Prediction. In *IEEE International Conference on Computer Vision (ICCV)* (pp. 1422–1430). (ISSN: 2380-7504) doi: 10.1109/ICCV.2015.167

Doersch, C., & Zisserman, A. (2017). Multi-task Self-Supervised Visual Learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017* (pp. 2070–2079). IEEE Computer Society. doi: 10.1109/ICCV.2017.226

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2021, June). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*. Retrieved 2022-02-27, from http://arxiv.org/abs/2010.11929 (arXiv: 2010.11929)

Dou, Z.-Y., Yu, K., & Anastasopoulos, A. (2019). Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi: 10.18653/v1/D19-1112

Duan, Y., Zhu, H., Wang, H., Yi, L., Nevatia, R., & Guibas, L. J. (2020). Curriculum DeepSDF. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision – ECCV* (pp. 51–67). Cham: Springer International Publishing. doi: 10.1007/978-3-030-58598-3_4

Ebbinghaus, H. (1913). *Memory: A Contribution to Experimental Psychology*. Annals of Neurosciences, Teachers College, Columbia University.

Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010, March). Why Does Unsupervised Pre-training Help Deep Learning? In *International Conference on Artificial Intelligence and Statistics* (pp. 201–208). JMLR Workshop and Conference Proceedings.

Retrieved from http://proceedings.mlr.press/v9/erhan10a.html

Fellbaum, C. (2010). WordNet. In R. Poli, M. Healy, & A. Kameas (Eds.), *Theory and Applications of Ontology: Computer Applications* (pp. 231–243). Dordrecht: Springer Netherlands. Retrieved 2022-03-10, from https://doi.org/10.1007/978-90-481-8847-5_10 doi: 10.1007/978-90-481-8847-5_10

Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning* (pp. 1126–1135). PMLR. Retrieved from http://proceedings.mlr.press/v70/finn17a.html

Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., & Macherey, W. (2021, April). Experts, Errors, and Context: A Large-Scale Study of Human Evaluation for Machine Translation. *Journal of Machine Learning Research*. Retrieved 2022-02-26, from http://arxiv.org/abs/2104.14478 (arXiv: 2104.14478)

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., & Rohrbach, M. (2016, November). Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 457–468). Austin, Texas: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D16-1044 doi: 10.18653/v1/D16-1044

Gardner, M., Artzi, Y., Basmova, V., Berant, J., Bogin, B., Chen, S., ... Zhou, B. (2020). Evaluating Models' Local Decision Boundaries via Contrast Sets. In T. Cohn, Y. He, & Y. Liu (Eds.), *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Vol. EMNLP 2020, pp. 1307–1323). Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.117

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised Representation Learning by Predicting Image Rotations. *International Conference on Learning Representations (ICLR), International Conference on Learning Representations (ICLR)*.

Girshick, R. (2015, December). Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)* (pp. 1440–1448). (ISSN: 2380-7504) doi: 10.1109/

ICCV.2015.169

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.*

Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goyal, A., & Bengio, Y. (2021, February). Inductive Biases for Deep Learning of Higher-Level Cognition. *arXiv:2011.15091 [cs, stat]*. Retrieved from http://arxiv.org/abs/2011.15091 (arXiv: 2011.15091)

Goyal, A., Wang, J., & Deng, J. (2018). Think Visually: Question Answering through Virtual Imagery. In *the 56th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 2598–2608). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P18-1242 doi: 10.18653/v1/P18-1242

Goyal, Y., Khot, T., Agrawal, A., Summers-Stay, D., Batra, D., & Parikh, D. (2019, April). Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *International Journal of Computer Vision* (Vol. 127, pp. 398–414). Retrieved from https://doi.org/10.1007/s11263-018-1116-0 doi: 10.1007/s11263-018-1116-0

Graves, A., Wayne, G., & Danihelka, I. (2014, December). Neural Turing Machines. *arXiv:1410.5401 [cs]*. Retrieved from http://arxiv.org/abs/1410.5401 (arXiv: 1410.5401)

Guo, L., Liu, J., Zhu, X., Yao, P., Lu, S., & Lu, H. (2020). Normalized and Geometry-Aware Self-Attention Network for Image Captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)* (pp. 10327–10336). Retrieved 2022-02-23, from https://openaccess.thecvf.com/content_CVPR_2020/html/Guo_Normalized_and_Geometry-Aware_Self-Attention_Network_for_Image_Captioning_CVPR_2020_paper.html

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith,

N. A. (2020). Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *ACL*. doi: 10.18653/v1/2020.acl-main.740

Hacohen, G., & Weinshall, D. (2019, May). On The Power of Curriculum Learning in Training Deep Networks. In *International Conference on Machine Learning* (pp. 2535–2544). PMLR. Retrieved from http://proceedings.mlr.press/v97/hacohen19a.html

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum Contrast for Unsupervised Visual Representation Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 9726–9735). (ISSN: 2575-7075) doi: 10.1109/CVPR42600.2020.00975

He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). (ISSN: 1063-6919) doi: 10.1109/CVPR.2016.90

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of Tricks for Image Classification with Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)* (pp. 558–567). Retrieved 2022-02-26, from https://openaccess.thecvf.com/content_CVPR_2019/html/He_Bag_of_Tricks_for_Image_Classification_with_Convolutional_Neural_Networks_CVPR_2019_paper.html

Hildebrandt, M., Li, H., Koner, R., Tresp, V., & Günnemann, S. (2020, July). Scene Graph Reasoning for Visual Question Answering. *arXiv:2007.01072 [cs, stat]*. Retrieved from http://arxiv.org/abs/2007.01072 (arXiv: 2007.01072)

Hinton, G. E., & Salakhutdinov, R. R. (2006, July). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507. Retrieved from https://www.science.org/doi/10.1126/science.1127647 doi: 10.1126/science.1127647

Hochreiter, S., & Schmidhuber, J. (1997, November). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. Retrieved from https://doi.org/10.1162/

neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2021). Meta-Learning in Neural Networks: A Survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Retrieved from http://arxiv.org/abs/2004.05439 (arXiv: 2004.05439)

Hu, R., Andreas, J., Darrell, T., & Saenko, K. (2018). Explainable neural computation via stack neural module networks. In *European conference on computer vision (ECCV)* (pp. 53–69).

Hua, Y., Li, Y.-F., Haffari, G., Qi, G., & Wu, T. (2020, November). Few-Shot Complex Knowledge Base Question Answering via Meta Reinforcement Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 5827–5837). Retrieved from https://aclanthology.org/2020.emnlp-main.469 doi: 10.18653/v1/2020.emnlp-main.469

Huang, L., Wang, W., Xia, Y., & Chen, J. (2019). Adaptively Aligned Image Captioning via Adaptive Attention Time. In *Advances in Neural Information Processing Systems (NeurIPS 2019)* (Vol. 32). Curran Associates, Inc. Retrieved 2022-02-23, from https://papers.nips.cc/paper/2019/hash/fecc3a370a23d13b1cf91ac3c1e1ca92-Abstract.html

Hudson, D. A., & Manning, C. D. (2018, February). Compositional Attention Networks for Machine Reasoning. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=S1Euwz-Rb

Hudson, D. A., & Manning, C. D. (2019a). GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6700–6709).

Hudson, D. A., & Manning, C. D. (2019b). Learning by Abstraction: The Neural State Machine. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (pp. 5901–5914). Re-

trieved from https://proceedings.neurips.cc/paper/2019/hash/c20a7ce2a627ba838cfbff082db35197-Abstract.html

Ioffe, S., & Szegedy, C. (2015, July). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning - Volume 37* (pp. 448–456). Lille, France: JMLR.org.

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

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

Kafle, K., & Kanan, C. (2017, October). Visual Question Answering: Datasets, Algorithms, and Future Challenges. *Computer Vision and Image Understanding*, *163*, 3–20. Retrieved from http://arxiv.org/abs/1610.01465 (arXiv: 1610.01465) doi: 10.1016/j.cviu.2017.06.005

Kafle, K., Yousefhussien, M., & Kanan, C. (2017). Data Augmentation for Visual Question Answering. In *International Conference on Natural Language Generation* (pp. 198–202). Santiago de Compostela, Spain: Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/W17-3529 doi: 10.18653/v1/W17-3529

Katharopoulos, A., & Fleuret, F. (2018, July). Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *International Conference on Machine Learning* (pp. 2525–2534). PMLR. Retrieved from http://proceedings.mlr.press/v80/katharopoulos18a.html

Ke, L., Pei, W., Li, R., Shen, X., & Tai, Y.-W. (2019). Reflective decoding network for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2019)* (pp. 8888–8897).

Kingma, D. P. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA: Conference Track Proceedings. Retrieved from http://arxiv.org/abs/1412.6980

Kingma, D. P., Rezende, D. J., Mohamed, S., & Welling, M. (2014, December). Semi-supervised learning with deep generative models. In *International Conference on Neural Information Processing Systems - Volume 2* (pp. 3581–3589). Montreal, Canada: MIT Press.

Klein, D., & Manning, C. D. (2002). Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems (NIPS 2002)* (Vol. 15, pp. 3–10). Cambridge: MIT Press.

Koch, G. R. (2015). Siamese Neural Networks for One-Shot Image Recognition. In *International Conference on Machine Learning (ICML)*.

Krishna, K., Wieting, J., & Iyyer, M. (2020). Reformulating Unsupervised Style Transfer as Paraphrase Generation. In *Empirical Methods in Natural Language Processing (EMNLP)*. Retrieved 2022-02-16, from http://arxiv.org/abs/2010.05700 (arXiv: 2010.05700)

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., ... others (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, *123*(1), 32–73. (Publisher: Springer)

Krizhevsky, A., Nair, V., & Hinton, G. (2010). *CIFAR-10 (Canadian Institute for Advanced Research)*. Retrieved from http://www.cs.toronto.edu/~kriz/cifar.html

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012, December). ImageNet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems* (Vol. 1, pp. 1097–1105). Red Hook, NY, USA: Curran Associates Inc.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., ... Socher, R. (2016, June). Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *International Conference on Machine Learning* (pp. 1378–

1387). PMLR. Retrieved from https://proceedings.mlr.press/v48/kumar16.html (ISSN: 1938-7228)

Kumar, M., Packer, B., & Koller, D. (2010). Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 23). Retrieved from https://papers.nips.cc/paper/2010/hash/e57c6b956a6521b28495f2886ca0977a-Abstract.html

Lake, B. M., Linzen, T., & Baroni, M. (2019). Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998, November). Gradient-based learning applied to document recognition. *IEEE*, *86*(11), 2278–2324. doi: 10.1109/5.726791

Lecun, Y., & Cortes, C. (2010). *MNIST handwritten digit database.* Retrieved from https://www.bibsonomy.org/bibtex/2935bad99fa1f65e03c25b315aa3c1032/mhwombat

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Li, Y., Ma, T., Bai, Y., Duan, N., Wei, S., & Wang, X. (2019). Pastegan: A semi-parametric method to generate image from scene graph. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 32, pp. 3948–3958).

Liang, J., Jiang, L., Meng, D., & Hauptmann, A. (2016, July). Learning to detect concepts from webly-labeled video data. In *International Joint Conference on Artificial Intelligence* (pp. 1746–1752). New York, New York, USA: AAAI Press.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV* (pp. 740–755). Cham: Springer International Publishing. doi: 10.1007/978-3-319-10602-1_48

Lin, Z., Zhang, D., Tac, Q., Shi, D., Haffari, G., Wu, Q., ... Ge, Z. (2022, January). Medical Visual Question Answering: A Survey. *arXiv:2111.10056 [cs]*. Retrieved 2022-03-10, from http://arxiv.org/abs/2111.10056 (arXiv: 2111.10056)

Liu, C., He, S., Liu, K., & Zhao, J. (2018, July). Curriculum learning for natural answer

generation. In *International Joint Conference on Artificial Intelligence* (pp. 4223–4229). Stockholm, Sweden: AAAI Press.

Liu, F., Ren, X., Wu, X., Ge, S., Fan, W., Zou, Y., & Sun, X. (2020). Prophet attention: Predicting attention with future attention. In *Advances in Neural Information Processing Systems (NeurIPS 2020)* (Vol. 33, pp. 1865–1876).

Liu, H., & Singh, P. (2004, October). ConceptNet — A Practical Commonsense Reasoning Tool-Kit. *BT Technology Journal*, *22*(4), 211–226. Retrieved from https://doi.org/10.1023/B:BTTJ.0000047600.45421.6d doi: 10.1023/B:BTTJ.0000047600.45421.6d

Liu, X., Li, H., Shao, J., Chen, D., & Wang, X. (2018). Show, Tell and Discriminate: Image Captioning by Self-retrieval with Partially Labeled Data. In *the European Conference on Computer Vision (ECCV)*. Retrieved from http://arxiv.org/abs/1803.08314

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019, July). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. Retrieved from http://arxiv.org/abs/1907.11692 (arXiv: 1907.11692)

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Neural Information Processing Systems (NeurIPS)*.

Lu, J., Yang, J., Batra, D., & Parikh, D. (2016, December). Hierarchical question-image co-attention for visual question answering. In *International Conference on Neural Information Processing Systems* (pp. 289–297). Red Hook, NY, USA: Curran Associates Inc.

Luo, B., Feng, Y., Wang, Z., Huang, S., Yan, R., & Zhao, D. (2018, July). Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding. In *56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2083–2093). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://aclanthology.org/

P18-1194 doi: 10.18653/v1/P18-1194

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., ... Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *European conference on computer vision (ECCV)* (pp. 181–196).

Malinowski, M., & Fritz, M. (2014, December). A multi-world approach to question answering about real-world scenes based on uncertain input. In *International Conference on Neural Information Processing Systems - Volume 1* (pp. 1682–1690). Cambridge, MA, USA: MIT Press.

Malinowski, M., Rohrbach, M., & Fritz, M. (2015, October). Ask Your Neurons: A Neural-based Approach to Answering Questions about Images. In *International Conference on Computer Vision (ICCV)*. Retrieved from http://arxiv.org/abs/1505.01121 (arXiv: 1505.01121)

Marino, K., Rastegari, M., Farhadi, A., & Mottaghi, R. (2019, June). OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3190–3199). Long Beach, CA, USA: IEEE. Retrieved from https://ieeexplore.ieee.org/document/8953725/ doi: 10.1109/CVPR.2019.00331

Martens, J. (2010, June). Deep learning via Hessian-free optimization. In *International Conference on Machine Learning* (pp. 735–742). Madison, WI, USA: Omnipress.

Mirzaee, R., Rajaby Faghihi, H., Ning, Q., & Kordjamshidi, P. (2021, June). SPARTQA: A Textual Question Answering Benchmark for Spatial Reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4582–4598). Online: Association for Computational Linguistics. Retrieved 2022-02-19, from https://aclanthology.org/2021.naacl-main.364 doi: 10.18653/v1/2021.naacl-main.364

Mishra, N., Rohaninejad, M., Chen, X., & Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. In *International Conference on Learning Representations (ICLR)*. Retrieved from https://openreview.net/forum?id=B1DmUzWAW

Misra, I., Girshick, R., Fergus, R., Hebert, M., Gupta, A., & van der Maaten, L. (2018, June). Learning by Asking Questions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11–20). (ISSN: 2575-7075) doi: 10.1109/CVPR.2018.00009

Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D., & Franco, L. (2018, November). Forward Noise Adjustment Scheme for Data Augmentation. In *IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 728–734). doi: 10.1109/SSCI.2018.8628917

Munkhdalai, T., & Yu, H. (2017, July). Meta Networks. In *International Conference on Machine Learning* (pp. 2554–2563). PMLR. Retrieved from https://proceedings.mlr.press/v70/munkhdalai17a.html (ISSN: 2640-3498)

Neumann, M., Pinto, A. S., Zhai, X., & Houlsby, N. (2019). In-domain representation learning for remote sensing. *CoRR*, *abs/1911.06721*. Retrieved from http://arxiv.org/abs/1911.06721 (arXiv: 1911.06721)

Ngiam, J., Peng, D., Vasudevan, V., Kornblith, S., Le, Q. V., & Pang, R. (2018). Domain Adaptive Transfer Learning with Specialist Models. *arXiv:1811.07056 [cs.CV]*.

Nguyen, B. D., Do, T.-T., Nguyen, B. X., Do, T., Tjiputra, E., & Tran, Q. D. (2019). Overcoming data limitation in medical Visual Question Answering. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (pp. 522–530). Springer. doi: 10.1007/978-3-030-32251-9_57

Nichol, A., Achiam, J., & Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *CoRR*, *abs/1803.02999*. Retrieved from http://arxiv.org/abs/1803.02999 (arXiv: 1803.02999)

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., ... Gurevych, I. (2020, October). AdapterHub: A Framework for Adapting Transformers. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 46–54). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.emnlp-demos.7 doi: 10.18653/v1/2020.emnlp-demos.7

Platanios, E. A., Stretcu, O., Neubig, G., Poczos, B., & Mitchell, T. (2019, June). Competence-based Curriculum Learning for Neural Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 1162–1172). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N19-1119 doi: 10.18653/v1/N19-1119

Puigcerver, J., Riquelme, C., Mustafa, B., Renggli, C., Pinto, A. S., Gelly, S., ... Houlsby, N. (2021). Scalable Transfer Learning with Expert Models. In *International Conference on Learning Representations (ICLR)*.

Qin, Y., Du, J., Zhang, Y., & Lu, H. (2019). Look Back and Predict Forward in Image Captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)* (pp. 8367–8375). Retrieved 2022-02-23, from https://openaccess.thecvf.com/content_CVPR_2019/html/Qin_Look_Back_and_Predict_Forward_in_Image_Captioning_CVPR_2019_paper.html

Quattoni, A., Collins, M., & Darrell, T. (2008, June). Transfer learning for image classification with sparse prototype representations. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). (ISSN: 1063-6919) doi: 10.1109/CVPR.2008.4587637

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*. (arXiv: 2103.00020)

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Raghu, M., Zhang, C., Kleinberg, J., & Bengio, S. (2019). Transfusion: Understanding Transfer Learning for Medical Imaging. In *Advances in Neural Information Processing Systems* (Vol. 32). Curran Associates, Inc. Retrieved from https://papers.nips.cc/paper/2019/hash/

eb1e78328c46506b46a4ac4a1e378b91-Abstract.html

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007, June). Self-taught learning: transfer learning from unlabeled data. In *International conference on Machine learning (ICML)* (pp. 759–766). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1273496.1273592 doi: 10.1145/1273496.1273592

Ravi, S., & Larochelle, H. (2017). Optimization as a Model for Few-Shot Learning. In *International Conference on Learning Representations (ICLR)*.

Ray, A., Sikka, K., Divakaran, A., Lee, S., & Burachas, G. (2019, September). Sunny and Dark Outside?! Improving Answer Consistency in VQA through Entailed Question Generation. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, {EMNLP-IJCNLP}*. Retrieved from http://arxiv.org/abs/1909.04696 (arXiv: 1909.04696)

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Reed, C. J., Yue, X., Nrusimha, A., Ebrahimi, S., Vijaykumar, V., Mao, R., ... Darrell, T. (2021, March). Self-Supervised Pretraining Improves Self-Supervised Pretraining. *arXiv:2103.12718 [cs]*. Retrieved from http://arxiv.org/abs/2103.12718 (arXiv: 2103.12718)

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Hong Kong, China: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D19-1410 doi: 10.18653/v1/D19-1410

Ren, M., Kiros, R., & Zemel, R. (2015). Image Question Answering: A Visual Semantic Embedding Model and a New Dataset. In *Advances in Neural Information Processing*

Systems (NIPS).

Riviere, M., Joulin, A., Mazaré, P.-E., & Dupoux, E. (2020). Unsupervised pretraining transfers well across languages. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 7414–7418). IEEE.

Ruprecht, D., & Muller, H. (1995, March). Image warping with scattered data interpolation. In *IEEE Computer Graphics and Applications* (Vol. 15, pp. 37–43). doi: 10.1109/38.365004

Sachan, M., & Xing, E. (2016, August). Easy Questions First? A Case Study on Curriculum Learning for Question Answering. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 453–463). Berlin, Germany: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P16-1043 doi: 10.18653/v1/P16-1043

Saha, A., Aralikatte, R., Khapra, M. M., & Sankaranarayanan, K. (2018). DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension. In *Association for Computational Linguistics (ACL)*. Retrieved from http://arxiv.org/abs/1804.07927 (arXiv: 1804.07927)

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016, June). Meta-Learning with Memory-Augmented Neural Networks. In *International Conference on Machine Learning* (pp. 1842–1850). PMLR. Retrieved from https://proceedings.mlr.press/v48/santoro16.html (ISSN: 1938-7228)

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

Selvaraju, R. R., Tendulkar, P., Parikh, D., Horvitz, E., Ribeiro, M. T., Nushi, B., & Kamar, E. (2020). SQuINTing at VQA Models: Introspecting VQA Models With Sub-Questions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10003–10011). Retrieved from https://openaccess.thecvf.com/content_CVPR_2020/html/Selvaraju_SQuINTing_at_VQA_Models_Introspecting_VQA_Models_With_Sub-Questions_CVPR_2020_paper.html

Shah, M., Chen, X., Rohrbach, M., & Parikh, D. (2019). Cycle-Consistency for Robust Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition*. Retrieved from http://arxiv.org/abs/1902.05660 (arXiv: 1902.05660)

Shah, S., Mishra, A., Yadati, N., & Talukdar, P. P. (2019, July). KVQA: Knowledge-Aware Visual Question Answering. In *AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 8876–8884). Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/4915 (Number: 01) doi: 10.1609/aaai.v33i01.33018876

Sharma, H., Agrahari, M., Singh, S. K., Firoj, M., & Mishra, R. K. (2020, February). Image Captioning: A Comprehensive Survey. In *International Conference on Power Electronics IoT Applications in Renewable Energy and its Control (PARC)* (pp. 325–328). doi: 10.1109/PARC49193.2020.236619

Shorten, C., & Khoshgoftaar, T. M. (2019, July). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. Retrieved from https://doi.org/10.1186/s40537-019-0197-0 doi: 10.1186/s40537-019-0197-0

Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training Region-Based Object Detectors With Online Hard Example Mining. In *IEEE onference on Computer Vision and Pattern Recognition (CVPR)* (pp. 761–769).

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In Y. Bengio & Y. LeCun (Eds.), *International Conference on Learning Representations (ICLR)*. Retrieved from http://arxiv.org/abs/1409.1556

Spitkovsky, V. I., Alshawi, H., & Jurafsky, D. (2010, June). From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 751–759). Los Angeles, California: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N10-1116

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., & Dai, J. (2020). VL-BERT: Pre-training of Generic Visual-Linguistic Representations. In *International Conference on Learning Representations (ICLR)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=SygXPaEYvH

Su, Z., Zhu, C., Dong, Y., Cai, D., Chen, Y., & Li, J. (2018). Learning Visual Knowledge Memory Networks for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from https://arxiv.org/abs/1806.04860v1

Suhr, A., Lewis, M., Yeh, J., & Artzi, Y. (2017, July). A Corpus of Natural Language for Visual Reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 217–223). Vancouver, Canada: Association for Computational Linguistics. Retrieved 2022-02-19, from https://aclanthology.org/P17-2034 doi: 10.18653/v1/P17-2034

Sukhbaatar, S., Szlam, A. D., Weston, J., & Fergus, R. (2015). Weakly Supervised Memory Networks. *ArXiv*.

Sun, C., Baradel, F., Murphy, K., & Schmid, C. (2019). Contrastive Bidirectional Transformer for Temporal Representation Learning. *arXiv:1906.05743 [cs.LG]*.

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. *International Conference on Computer Vision (ICCV)*, *1*, 7463–7472. doi: 10.1109/ICCV.2019.00756

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P., & Hospedales, T. (2018, June). Learning to Compare: Relation Network for Few-Shot Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1199–1208). IEEE. doi: 10.1109/CVPR.2018.00131

Sussillo, D., & Abbott, L. F. (2015, February). Random Walk Initialization for Training Very

Deep Feedforward Networks. *arXiv:1412.6558 [cs, stat]*. Retrieved from http://arxiv.org/abs/1412.6558 (arXiv: 1412.6558)

Tan, H., & Bansal, M. (2019, November). LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 5100–5111). Hong Kong, China: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D19-1514 doi: 10.18653/v1/D19-1514

Tang, R., Ma, C., Zhang, W. E., Wu, Q., & Yang, X. (2020, July). Semantic Equivalent Adversarial Data Augmentation for Visual Question Answering. In *European Conference on Computer Vision (ECCV)*. Retrieved from http://arxiv.org/abs/2007.09592 (arXiv: 2007.09592)

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., & Fidler, S. (2016). MovieQA: Understanding Stories in Movies through Question-Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/CVPR.2016.501

Teney, D., & Hengel, A. (2018). Visual Question Answering as a Meta Learning Task. In *The European Conference on Computer Vision* (pp. pp. 219–235).

Teney, D., Liu, L., & Van Den Hengel, A. (2017, July). Graph-Structured Representations for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3233–3241). Honolulu, HI: IEEE. Retrieved from https://ieeexplore.ieee.org/document/8099827/ doi: 10.1109/CVPR.2017.344

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jegou, H. (2021, July). Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)* (pp. 10347–10357). PMLR. Retrieved 2022-02-23, from https://proceedings.mlr.press/v139/touvron21a.html

Tu, K., Meng, M., Lee, M. W., Choe, T. E., & Zhu, S. C. (2013). Joint Video and Text

Parsing for Understanding Events and Answering Queries. *CoRR*, *abs/1308.6628*. Retrieved from http://arxiv.org/abs/1308.6628 (arXiv: 1308.6628)

Turing, A. M. (1950). Computing Machinery and Intelligence. *Oxford University Press on behalf of the Mind Association*, *59*(236), 433–460.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A., Gouws, S., ... Uszkoreit, J. (2018, March). Tensor2Tensor for Neural Machine Translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)* (pp. 193–199). Boston, MA: Association for Machine Translation in the Americas. Retrieved 2022-02-26, from https://aclanthology.org/W18-1819

Vickers, P., Aletras, N., Monti, E., & Barrault, L. (2021, August). In Factuality: Efficient Integration of Relevant Facts for Visual Question Answering. In *59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 468–475). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.acl-short.60 doi: 10.18653/v1/2021.acl-short.60

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *IEEE conference on computer vision and pattern recognition* (pp. 3156–3164).

Wang, L., Bai, Z., Zhang, Y., & Lu, H. (2020, April). Show, Recall, and Tell: Image Captioning with Recall Mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)* (Vol. 34, pp. 12176–12183). Retrieved 2022-02-23, from https://ojs.aaai.org/index.php/AAAI/article/view/6898 doi: 10.1609/aaai.v34i07.6898

Wang, P., Wu, Q., Shen, C., Dick, A., & Hengel, A. v. d. (2017). Explicit Knowledge-based Reasoning for Visual Question Answering. In *International Joint Conference on Artificial Intelligence (IJCAI)*. Retrieved from https://www.ijcai.org/proceedings/2017/179

Wang, P., Wu, Q., Shen, C., Dick, A., & van den Hengel, A. (2018, October). FVQA: Fact-Based Visual Question Answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(10), 2413–2427. Retrieved from https://doi.org/10.1109/TPAMI.2017.2754246 doi: 10.1109/TPAMI.2017.2754246

Wang, S., Thompson, L., & Iyyer, M. (2021). Phrase-BERT: Improved Phrase Embeddings from BERT with an Application to Corpus Exploration. In *Empirical Methods in Natural Language Processing (EMNLP)*. Retrieved 2022-02-16, from http://arxiv.org/abs/2109.06304 (arXiv: 2109.06304)

Wang, S., Wang, R., Yao, Z., Shan, S., & Chen, X. (2020). Cross-modal scene graph matching for relationship-aware image-text retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1508–1517).

Wei, J., & Zou, K. (2019, August). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. *arXiv:1901.11196 [cs]*. Retrieved from http://arxiv.org/abs/1901.11196 (arXiv: 1901.11196)

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., ... Vajda, P. (2020, November). Visual Transformers: Token-based Image Representation and Processing for Computer Vision. *arXiv:2006.03677 [cs, eess]*. Retrieved 2022-02-26, from http://arxiv.org/abs/2006.03677 (arXiv: 2006.03677)

Wu, Q., Wang, P., Shen, C., Dick, A., & Hengel, A. V. D. (2016, June). Ask Me Anything: Free-Form Visual Question Answering Based on Knowledge from External Sources. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4622–4630). IEEE Computer Society. Retrieved from https://www.computer.org/csdl/proceedings-article/cvpr/2016/8851e622/12OmNvkplaP (ISSN: 1063-6919) doi: 10.1109/CVPR.2016.500

Wu, Y., Rabe, M., Li, W., Ba, J., Grosse, R., & Szegedy, C. (2021, January). LIME: Learning Inductive Bias for Primitives of Mathematical Reasoning. *arXiv:2101.06223 [cs]*. Retrieved from http://arxiv.org/abs/2101.06223 (arXiv: 2101.06223)

Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised Feature Learn-

ing via Non-Parametric Instance Discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3733–3742). Computer Vision Foundation / IEEE Computer Society. Retrieved from http://openaccess.thecvf.com/content_cvpr_2018/html/Wu_Unsupervised_Feature_Learning_CVPR_2018_paper.html doi: 10.1109/CVPR.2018.00393

Xiong, C., Merity, S., & Socher, R. (2016). Dynamic Memory Networks for Visual and Textual Question Answering. In *ICML* (pp. 2397–2406).

Xu, H., & Saenko, K. (2016, March). Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. In *European Conference on Computer Vision (ECCV)*. Retrieved from http://arxiv.org/abs/1511.05234

Yang, X., Tang, K., Zhang, H., & Cai, J. (2019). Auto-Encoding Scene Graphs for Image Captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)* (pp. 10685–10694). Retrieved 2022-02-23, from https://openaccess.thecvf.com/content_CVPR_2019/html/Yang_Auto-Encoding_Scene_Graphs_for_Image_Captioning_CVPR_2019_paper.html

Yao, T., Pan, Y., Li, Y., & Mei, T. (2018). Exploring Visual Relationship for Image Captioning. In *Proceedings of the European Conference on Computer Vision (ECCV 2018)* (pp. 684–699). Retrieved 2022-02-23, from https://openaccess.thecvf.com/content_ECCV_2018/html/Ting_Yao_Exploring_Visual_Relationship_ECCV_2018_paper.html

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. (2018). Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. *Advances in Neural Information Processing Systems*, *31*. Retrieved from https://proceedings.neurips.cc/paper/2018/hash/5e388103a391daabe3de1d76a6739ccd-Abstract.html

Yin, M., Tucker, G., Zhou, M., Levine, S., & Finn, C. (2020). Meta-Learning without

Memorization. In *International Conference on Learning Representations*. Retrieved from http://arxiv.org/abs/1912.03820 (arXiv: 1912.03820)

Yong Jae Lee, & Grauman, K. (2011, June). Learning the easy things first: Self-paced visual category discovery. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1721–1728). USA: IEEE Computer Society. Retrieved from https://doi.org/10.1109/CVPR.2011.5995523 doi: 10.1109/CVPR.2011.5995523

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (Vol. 27). Curran Associates, Inc. Retrieved from https://papers.nips.cc/paper/2014/hash/375c71349b295fbe2dcdca9206f20a06-Abstract.html

Yu, D., Fu, J., Mei, T., & Rui, Y. (2017, July). Multi-level Attention Networks for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4187–4195). Honolulu, HI, USA: IEEE. doi: 10.1109/CVPR.2017.446

Yu, F., Tang, J., Yin, W., Sun, Y., Tian, H., Wu, H., & Wang, H. (2021). ERNIE-ViL: Knowledge Enhanced Vision-Language Representations through Scene Graphs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event* (pp. 3208–3216). AAAI Press. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/16431

Yu, L., Lin, Z., Shen, X., Yang, J., Lu, X., Bansal, M., & Berg, T. L. (2018). Mattnet: Modular attention network for referring expression comprehension. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1307–1315).

Yu, L., Park, E., Berg, A. C., & Berg, T. L. (2015). Visual Madlibs: Fill in the blank Image Generation and Question Answering. In *International Conference on Computer Vision (ICCV)*. Retrieved from http://arxiv.org/abs/1506.00278 (arXiv: 1506.00278)

Zellers, R., Bisk, Y., Farhadi, A., & Choi, Y. (2019). From Recognition to Cognition: Visual Commonsense Reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from http://arxiv.org/abs/1811.10830 (arXiv: 1811.10830)

Zhang, C., Chao, W.-L., & Xuan, D. (2019, July). An Empirical Study on Leveraging Scene Graphs for Visual Question Answering. In *arXiv:1907.12133 [cs]*. (arXiv: 1907.12133)

Zhang, P., Goyal, Y., Summers-Stay, D., Batra, D., & Parikh, D. (2016, June). Yin and Yang: Balancing and Answering Binary Visual Questions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5014–5022). Las Vegas, NV, USA: IEEE. Retrieved from http://ieeexplore.ieee.org/document/7780911/ doi: 10.1109/CVPR.2016.542

Zhang, X., & LeCun, Y. (2015). Text Understanding from Scratch. *arXiv:1502.01710 [cs.LG]*.

Zhang, X., Wang, Z., Liu, D., & Ling, Q. (2019, May). DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2807–2811). (ISSN: 2379-190X) doi: 10.1109/ICASSP.2019.8683197

Zhou, T., & Bilmes, J. (2018). Minimax Curriculum Learning: Machine Teaching with Desirable Difficulties and Scheduled Diversity. In *International Conference on Learning Representations, (ICLR)*. Vancouver, BC, Canada.

Zhou, T., Wang, S., & Bilmes, J. (2020). Curriculum Learning by Dynamic Instance Hardness. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 8602–8613). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/file/62000dee5a05a6a71de3a6127a68778a-Paper.pdf

Zhou, T., Wang, S., & Bilmes, J. (2021, March). Curriculum Learning by Optimizing Learning Dynamics. In *International Conference on Artificial Intelligence and Statistics*

(pp. 433–441). PMLR. Retrieved from http://proceedings.mlr.press/v130/zhou21a.html

Zhu, Y., Groth, O., Bernstein, M., & Fei-Fei, L. (2016). Visual7w: Grounded question answering in images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4995–5004).

Zhuang, B., Wu, Q., Shen, C., Reid, I., & Hengel, A. V. D. (2018). ParalleL AttentioN: a unified framework for visual object discovery through dialogs and queries. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4252–4261). IEEE, Institute of Electrical and Electronics Engineers. doi: 10.1109/CVPR.2018.00447