Neural State Machine for 2D and 3D Visual Question Answering

by

Leon Kochiev

B.Sc., Moscow Institute of Physics and Technologies, 2019

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2021

Declaration of Committee

Name:	**Leon Kochiev**
Degree:	**Master of Science**
Thesis title:	**Neural State Machine for 2D and 3D Visual Question Answering**
Committee:	**Chair:** Greg Mori Professor, Computing Science
	Angel Xuan Chang Supervisor Assistant Professor, Computing Science
	Anoop Sarkar Committee Member Professor, Computing Science
	Frederick Popowich Examiner Professor, Computing Science

Abstract

This thesis focuses on the Visual Question Answering (VQA) task in 2D images and 3D environments using the Neural State Machine (NSM). The NSM is a state-of-the-art approach for VQA that simulates reasoning over scene-graphs. We re-implement and extend the NSM by adding a narrowing mechanism for localised attention, and by applying bilinear attention on scene graph representations of the input scene. We show that these extensions lead to improved performance on the VQA task in both the 2D and 3D domains. Prior work on VQA has focused on reasoning in the 2D image domain, and has not addressed how the VQA task can be formulated with 3D data. To address the latter domain, we create a 3D VQA dataset based on 3D reconstructions of real environments. Then, we compare the performance of the NSM with common approaches for 3D VQA in on a range of question types. We show that the NSM is competitive with other VQA methods in the 3D domain and our extensions also lead to improved VQA accuracy in the 3D domain.

Keywords: Visual Question Answering, 3D, Neural State Machine

Acknowledgements

In loving memory of my dear grandfather.

Table of Contents

List of Figures

Chapter 1

Introduction

Nowadays, personal assistants are strongly integrated in our daily lives, but for now their abilities are limited to search queries, note taking and interacting with applications. There is a limited ability to answer questions based on both visual and language input. Developing computational systems model that can analyze visual streams and provide informative answers would be a useful feature for smart security systems, assistants for people with disabilities and other applications.

The Visual Turing Test [20] was proposed as a test of an AI agent's ability to perceive and reason, and requires integration of visual and language capabilities. Motivated by this challenge, the Visual Question Answering (VQA) [8, 70] problem has been widely studied in the vision-and-language community. VQA tasks span a spectrum from visual reasoning in abstract 'blocks world' [31] settings to answering questions about complex real-world scenes while incorporating common-sense knowledge [70]. Despite the rapid progress in this area the technology cannot yet meet people's expectations of an intelligent agent. Looking at state-of-the-art results of models it is worth noting, that the existing models can perform simple synthetic visual question answering based on the CLEVR [31] dataset, but when models face real world data from the VQA [8] or GQA [29] datasets they tend to fail more frequently. This area offers much room for growth, an intuitive setting, various directions for further work, which makes it attractive for students and researchers.

Visual Question Answering has a variety of underlying aspects, and models that tackle it are required to have perceptual abilities such as distinguishing objects and understanding their attributes, relationship between objects as well as estimating the number of certain objects in the scene, comparing attributes, performing logical inference and making inferences based on general world knowledge. Various datasets ([31, 39, 8, 22]) were introduced to study how to address these key challenges. Designing a diverse and comprehensive set of questions can be as difficult as building a model for solving them.

The ideal VQA model has to consist of perfect components: feature extractors and feature combiners. The majority of modern approaches focus on the most studied part of the problem: recognising entities and patterns (feature extractors), while paying less

attention to the way of combining them to properly reason the answer (feature combiners). The simplest VQA model takes the input question and image and extracts language and visual features, which are concatenated and used to predict the answer. The Neural State Machine (NSM) [27] proposes an interesting and unique approach to scene understanding and cross-modal data alignment. The NSM operates on a scene-graph representation of the image, where objects are represented as nodes and relationships between the objects are represented as edges. Associated with each node and edges are continuous embedding vectors that capture the properties of the objects and relationship between objects. In addition, there is a probability weight for each node. The question text is converted into a sequence of instructions, each represented as an embedding vector as well. The NSM then takes the encoded question (as a sequence of instructions), and simulates the computation of a finite state machine by using a neural network to compute transition functions for distributing attention across different states. This generic representation of visual data using scene graphs allows us to apply the NSM approach to 3D visual data as well.

1.1 Thesis contribution

In this thesis, I investigate the ability of the NSM to answer questions over scene graphs for both standard 2D image based VQA and in 3D environments. NSM is a state-of-the-art model on the GQA [29] dataset. As there is no available codebase for the model from GQA, I started by re-implementing the NSM and evaluating the model on ground truth scene-graphs. For evaluation we use standard metrics such as accuracy as well as metrics introduced by Hudson and Manning [29] for GQA.

I considered two extensions to the NSM: adding a narrowing mechanism for localized attention and applying Bilinear Attention on scene graphs. The narrowing mechanism is motivated by considering how humans ask questions about objects in relation to other objects. Typically, when formulating questions, people will provide information to narrow the area of interest to be within a few relations of objects mentioned in the question. For instance, if someone is asking about a pizza on a plate, they may say 'What is on the plate?'. It is unlikely for them to say 'What is in the kitchen by the chair on the table and on the plate?'. Thus I propose to narrow the attention weights of the NSM to a restricted subgraph based on the question. The size of the subgraph is determined by from the number of words in the question. Using this narrowing mechanism, we are able to improve the performance of the Neural State Machine on the GQA dataset.

Another idea that sparked my interest was applying Bilinear Attention [36] on scene graphs. Bilinear attention has been shown to be a strong multimodal fusion method with good performance on VQA and thus it is more plausible for it to find connection between constrained representations of scene graphs and words. In this paper we compare the performance of this attention mechanism with the NSM one.

In addition to exploring these extensions to the NSM, I also investigated the application of the NSM to 3D VQA. Using ScanNet, we developed the first 3D Visual Question Answering dataset. It allowed us to conduct experiments using NSM in 3D environments and compare it against other VQA models.

1.2 Thesis organization

This thesis is structured as follows: Chapter 2 gives an introduction to Visual Question Answering and describes recent advances of datasets and models for VQA. Chapter 3 describes the Neural State Machine (NSM) and my efforts to re-implement it, as well as initial experimental results on GQA. Chapter 4 describes the proposed extensions to the NSM (attention localization and bilinear attention) and experiments comparing the extensions with the basic NSM. In Chapter 5, we present 3DVQA along with baseline models and experiments comparing the NSM on 3D VQA with a simpler fused model. Chapter 6 concludes the thesis and describes potential future work and limitations of our work.

Chapter 2

Visual Question Answering

This chapter introduces the problem of Visual Question Answering and provides a brief summary of the terminology, datasets, recent advances in the field and ways to measure performance of a VQA model.

In the VQA task, the system is given as input an image and a question related to it and needs to generate as output the answer to the question Figure 2.1. A natural model architecture for solving this problem arose: visual encoder, language encoder, fusion module to align visual and language features and answer predictor. The answer can either be free-form, multiple choice, or binary ('yes'/'no').

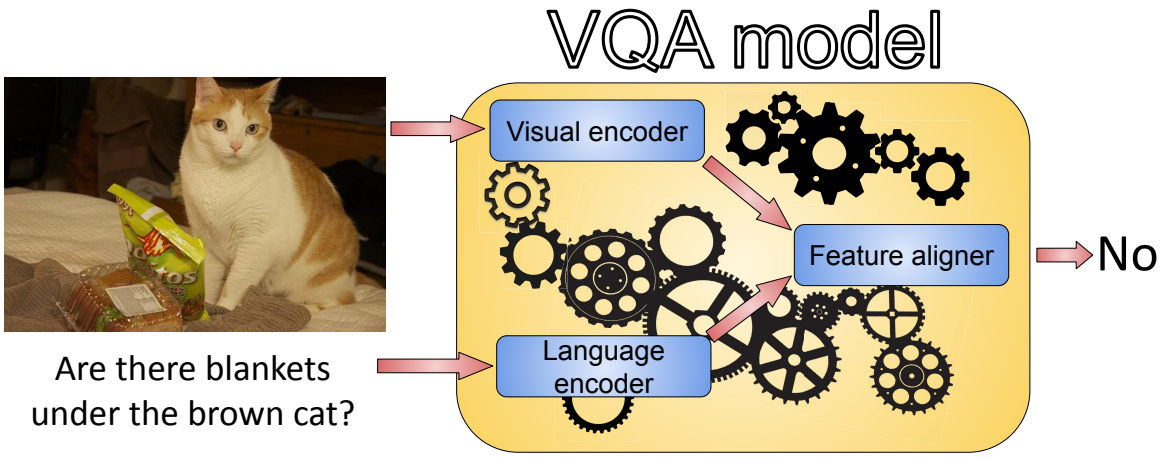


Figure 2.1: VQA system receives image and question pair, extracts features from both and combines them in the feature aligner (or fusion module) to produce the answer.

Research in VQA has focused on different aspects of the problem. Some researchers examine how to improve the backbones used for the language and visual encoders by using the latest neural architectures such as EfficientNet [60] for images and transformers for language [47, 59, 10]. A key challenge of visual question answering lies in improving the fusion module for aligning the visual and language features. One way to improve the fusion module is by using more sophisticated attention algorithms [36, 28, 46], while others explore

Name	Number of images	Number of questions	Images type	Questions type	Functional programs	Scene graphs
CLEVR [31]	100,000	999,968	Synthetic	Synthetic	✓	✓
VQAv2 [22]	204,721	1,105,904	Real	Real	✗	✗
Visual Genome [39]	101,174	1,773,258	Real	Real	✗	✓
GQA [29]	113,018	22,669,678	Real	Synthetic	✓	✓

Table 2.1: Comparison of important datasets in VQA.

new reasoning architectures that reason over objects and relations in the images. In this thesis, we study the Neural State Machine, introduced by Hudson and Manning [27].

The choice of visual representation can drastically influence the model performance. A common choice is to extract grid features from the image using CNN, while later systems used object detectors to identify objects. More recently, researchers have used a graph representation of the image, known as a scene-graph (SG) that represented objects as nodes, and relationship between objects as edges. More formally, given a set O of objects from the scene, set of their attributes A and set of relations R connecting objects, we define a graph $G = (O, R)$ - called *scene graph*. Each object o_i from O paired with a set of object attributes a_i from A. Typically, a set of objects and their attributes are generated using object detection models, and the relations are predicted with scene graph generation models.

2.1 Datasets

To investigate the different challenges of VQA, a variety of datasets was proposed in recent years ([29, 31, 8]). We provide a short overview of some important datasets commonly used for Visual Question Answering. They can be divided by how they were collected. It can be fully synthetic [31], partially synthetic [29] or consist fully of human created data [39]. To collect natural question and answers from humans, researchers typically turn to crowdworkers. While collected questions and answers are linguistically diverse and capture the type of questions and answers that people are likely to provide, there can be problems with these human created question and answers. For one, the collected data is often noisy and contain misspellings and other errors. In addition, it's difficult to control for diversity in vocabulary and comprehensiveness in the set of collected questions. It is also challenging to obtain the precise set of reasoning steps that were used to obtain the answer.

To allow for systematic control over questions and answers, researchers turn to generating questions and answers automatically [31, 29]. The questions and answers are typically generated using a functional program (FP) that can be run over a semantic representation (typically a scene-graph) of the image to obtain the answer. From the function program, a set of templates or probabilistic grammar is used to synthetically generate the questions. The functional program is typically provided with the dataset. It is provided as a textual

instruction describing a set of actions that need to be executed over an image's scene graph to find the correct answer. The set of basic actions varies between datasets. Generally it is an elementary operation of visual reasoning such as filtering by attribute, counting or comparing values. Often if the dataset is annotated with object bounding boxes the dataset will also include links between text and object bounding boxes for the objects mentioned in the question and answer.

2.1.1 Fully human annotated datasets

With the proposal of the Visual Turing Test[20], there were concurrent efforts to formalize and investigate the problem of VQA, with several different datasets including FM-IQA[18]), Toronto COCO-QA[57], DAQUAR [48], Visual7W[70], and the VQA dataset[8]. Due to its large size and a series of yearly benchmark challenges, the VQA dataset[8] became one of the most widely used datasets. The dataset was constructed by taking images from MS COCO [42] and abstract scenes datasets [71], and asking crowdworkers on Amazon Mechanical Turk (AMT) to provide questions and answers. For each question, they collected a set of 10 answers and chose the most frequent one as the correct answer. One issue with this initial VQA dataset (VQAv1) was that it was heavily biased. For instance, binary questions account for about 38% of all questions, with 59% of them having positive answer ('yes'). This is due to a human propensity to pose questions about objects that they see. For instance,a person would ask 'Is there a bird in the sky' when shown a image of an bird flying. This type of dataset bias allows a model that simply guess 'yes' for all questions to achieve 59% accuracy on binary questions, and 22% accuracy overall. In addition to the overall bias in answers, the data was also biased when conditioned on just the question. Kafle and Kanan [33] showed that with a language only model it was possible to achieve 49% on the dataset. Zhang et al. [69] attempted to balance the binary questions, and later Goyal et al. [22] introduced a balanced version of the dataset, VQAv2 that also address the issue with language priors. The balanced VQAv2 dataset was significantly more challenging than VQAv1, leading to drastic drops in performance for existing state-of-the-art models. For instance, the accuracy of the co-attention model from Lu et al. [46] dropped by 3%, and a bilinear pooling based model from Fukui et al. [17] dropped by more than 1% compared to an original accuracy of 66.7% on the unbalanced VQAv1. The VQA dataset further received more updates in Agrawal et al. [2] with newer splits and it is still used as a state-of-the-art benchmark for testing models.

Note that at around the same time that the VQA dataset was introduced, the Visual7w dataset [70] was also introduced. Compared to the VQA dataset Visual7w had more variability and included annotated regions of the images. In addition, the dataset was designed to be more balanced. Despite these strengths, the Visual7w dataset was less popular than the VQA dataset.

Visual Genome

Another key step in the development of VQA datasets is Visual Genome (VG) [39]. Visual Genome provided densely aligned language-to-image annotations on top of MS-COCO in the form of scene-graphs. Using crowdworkers, they collected bounding boxes for regions in each image along with descriptions for each region. Each description is then decomposed into objects (nouns), attributes (adjectives), and relations (verbs and prepositions) and linked to the image in a semi-automated fashion to create region graphs. For each region description, workers annotated the objects and linked them to a corresponding object bounding box in the image. The Stanford CoreNLP toolkit [49] was used to help automatically extract nouns as candidate objects to be identified. Attributes and relations are marked by the workers and associated with the appropriate objects. By combining the region graphs, a scene-graph containing all mentioned attributes, objects, and relationships is constructed for each image. For objects whose bounding boxes overlap significantly (>0.8 IoU), workers were asked to indicate whether the objects are the same so co-referent object nodes can be merged together. To normalize the terms, words are matched against WordNet [16] using heuristics favoring the most common sense, and checked by crowdworkers.

In addition to collecting the scene-graphs, Krishna et al. [39] also collected question and free-form answer pairs for both the entire image and selected regions (regions that are relatively large). To increase the overall difficulty of questions, this dataset did not include binary ('yes/no') questions. Overall Visual Genome has greater diversity of answers compared to other datasets. For instance, 100 most frequent answers of VG covers only 65% of the VG dataset while the top 100 answer for VQA covers 82% [34]. In addition only, 57% of VG answers are single words, compared to 88% in the VQA dataset. However, the quality of the VG scene-graphs were far from perfect. The scene graphs were noisy and incomplete, and since they were derived from free-form text, the set of object classes, attributes, and relations were overlapping and ambiguous (despite normalization efforts). There were also large biases in the types of relations that were included in the scene-graphs [62, 68]. Despite the richness of the VG VQA dataset, it did not achieve the popularity of the VQA dataset.

2.1.2 Synthetic datasets

CLEVR

Another impactful dataset is CLEVR [31], a fully synthetic diagnostic dataset designed for exploring VQA models that requires spatial reasoning. In the CLEVR dataset, rendered scenes composed of simple arrangements of 3D shapes with different colors, shapes, and materials. These functional programs are built using simple basic operations of visual reasoning, such as count or relate. Because the dataset was synthetically generated, it came with ground truth annotations for object position and attributes, and questions with executable programs. This dataset became a test bed for exploring different VQA models

for improved attention and visual reasoning ([5, 50, 54]). CLEVR was designed in order to challenge the model's visual reasoning simple shapes were used instead of complex objects to allow for easier object detection. In addition to the synthetically generated question and answers, Johnson et al. [32] also introduced CLEVR-Humans composed of questions and answers written by humans. To collect this data, crowdworkers on Amazon Mechanical Turk were primed with a question from CLEVR and restricted to answers in the CLEVR answer space. The CLEVR-Humans dataset still focused on visual reasoning but was more challenging as it provided more linguistic variability.

GQA

To address the limitations of the previous datasets, Hudson and Manning [29] introduced the GQA dataset, a dataset designed for visual reasoning over real images. The GQA dataset draws inspiration fromthe construction of CLEVR to generate questions and answers using a functional program. Instead of using synthetically generated scenes of simple shapes, GQA leverages the scene-graphs provided by Visual Genome from which the functional programs are created. Following the development of the VQA datasets, GQA also takes care to balance the dataset. Hudson and Manning [29] attempt to remedy many of the drawbacks of the Visual Genome dataset to produce the GQA dataset. They take images and scene graphs from VG, generating a completely new set of question and answer pairs with a functional program for each question-answer pair. Each answer has a full and short version, and a region to which the answer is related (represented as a bounding box). The VG scene graphs are normalized by mapping all words to 2,690 classes and then adding language variation for these classes. After that they trim inaccurate or unrealistic edges. As a final part of the normalization process, information about position of objects in the image and semantic properties (location, weather) are added. For question generation they use a template-based engine (which composes questions from a small set of sub-templates) and covers 524 (250 unique, 274 from the VQA dataset) patterns. To increase diversity they used synonyms for objects and attributes, as well as alternative expressions and optional phrases. Distractor objects and attributes are used to rewrite questions to form questions with negative answers. These negative questions were added with a careful sampling of distractors. Distractors were selected to avoid extremely unrealistic questions (*Is the girl eating a machine?*) and provide more plausible substitutes (*Is the girl eating meat?* when the girl is eating ice cream in the image). Almost the same strategy was chosen for attribute distractors, with the only difference being that they have to vary enough from ground truth data (pink vs violet). After generating the whole dataset they obtained a little over 22 million questions, from which they sub-sampled a balanced set of 1 million. This is needed to avoid most frequent questions which appear due to normal real world biases. One major drawback of the dataset which remains from the VG is noise in the scene graphs. Normalization of

the SGs partially solved the problem and improved the quality of it a bit, but they still require work.

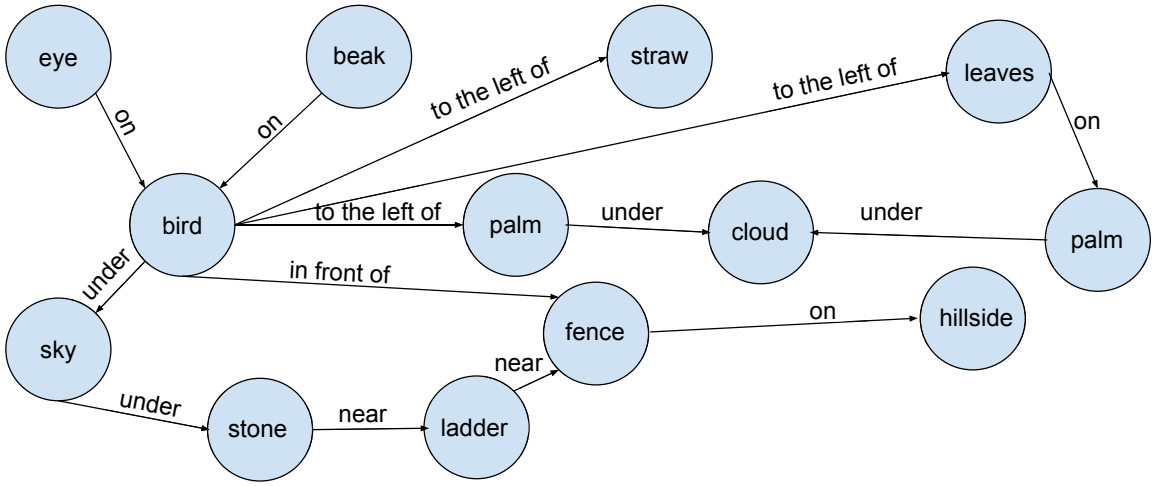
1. Q: *What is this bird called?;* A: *parrot;* Global: *bird;* Local: *11q-bird;* FP: *select: bird ->query: name [0]*

2. Q: *Do you see trains to the right of the fence?;* A: *No;* Global: *None;* Local: *13-fence-train;* FP: *select: fence -> relate: train,to the right of [0]->exist [1]*

3. Q: *What color does this parrot have?;* A: *white;* Global: *'color';* Local: *10q-parrot_color;* FP: *select: parrot -> query: color [0]*

4. Q: *Is that a parrot or an eagle?;* A: *parrot;* Global: *bird;* Local: *08oc-eagle_parrot;* FP: *select: this -> choose name: parrot/eagle [0]*

(a) Visual and language data example



(b) Scene graph example: nodes - objects, edges - relations between the objects

Figure 2.2: Sample from GQA dataset

2.2 Models

Since the introduction of the VQA task, a large number of VQA models was proposed. Despite the diversity of models, all models follow the same basic structure: vision feature extractor, language feature extractor and feature combiner. Covering all of them within this thesis would be impossible, so we chose to touch upon only those models that made the most impact from our very subjective point of view.

2.2.1 Standard VQA Model

The standard VQA model consists of a visual encoder, a language encoder, and a fusion module to align features from the two modalities to obtain a fused representation that is

used for determining the answer. The answer prediction module is typically a classifier that classifies over the possible answers (either a small number of multiple choice options or a larger space of all possible answers). In the case of multiple choice VQA, there is often an answer encoder that encodes the answer options, and the answer encoding are also fed into the fusion module. For longer free form answers, sometimes a sequence decoder is used to generate the answer. The performance of an VQA system is dependent on all of these components. Typical choices for the visual encoder is a backbone trained on ImageNet [15], such as Resnet-101 [23]. For encoding the question, word embeddings are used to encode words and then passed into a sequence encoder, such as an RNN. The simplest fusion module is to concatenate the embedded language and visual features and feed the concatenated vector into a multilayer perceptron (MLP) and use a cross-entropy loss to predict the correct answer. Early work in neural-based VQA explored these types of models [8, 48, 18, 57, 70]. Jabri et al. [30] showed that a simple model can outperform more complex models. Their model used ResNet-101 as visual feature encoder, and encoded question words with Word2Vec [52] and then used a bag-of-words model to average the embeddings to obtain the question features. The language and visual features were concatenated together and a MLP is used predict the answer. By also encoding the answer choices and supplying it to the MLP, this simple model was able to outperform more advanced fusion modules (such as Multimodal Compact Bilinear Pooling [17], see Section 2.2.3) on multiple choice settings for VQA and Visual7W.

2.2.2 Attention based models

From the beginning, attention based models have been found to be very useful for VQA [66]. Attention puts weights on parts of the input, and allows the model to focus on some areas more than others. In addition, attention weights can also shift over time. For instance, as the question is passed, attention weights over the image can shift to focus on more parts of the image that is more related to the words being processed. Various attention methods has been proposed including stacked attention [66] and hierarchical attention[46]. On the vision side, typically these attention are over image grid cells and are driven in a top-down manner by the task at hand. Another work that influenced the field and remained a standard point of comparison as well as being on top of the VQA leaderboard [1] for an extended time is Bottom-Up attention network [4]. For bottom-up attention, it utilizes an object detector (Faster R-CNN [58]) to identify objects and extract object-level visual features to be used as input into the VQA model. This is similar to bottom-up attention in humans where people identify objects without having a specific goal in mind. The question is encoded with word embeddings that are fed into a GRU [11]. For top-down attention, the encoded question is used to determine attention weights over the objects. An attention

[1] `https://eval.ai/web/challenges/challenge-page/1/leaderboard`

score is computed for each object by taking the concatenation of the visual features of each object and the question encoding, and passing it through a single-layer feedforward network. Once the attention weights are computed, an overall attended visual representation (weighted sum of the individual object visual features) is element-wise multiplied with the question encoding to obtain a fused representation which is used to predict the answer. This model mimics human attention, by combining bottom-up attention corresponding to when a human identifies objects in the image (how Faster R-CNN looks for objects) and top-down, goal driven attention when a human looks for specific object (how LSTM attends for language over visual features).

2.2.3 Multimodal fusion

There are many different ways to fuse information from the visual and language features. The simplest consist of taking a concatenation of the visual and language features and passing them through a MLP. Other alternatives include taking element-wise sum or product of the features. These simple fusion methods are constrained in how the elements from the two modalities can interact. To allow for more flexibility, Bilinear Pooling [44] was introduced. In Bilinear Pooling, the fused features is the result of a learnable weight matrix multiplied by the linearized outer product of the vision and language features. Bilinear Pooling allows for multiplicative interaction between all elements of the language and visual features. While more flexible, Bilinear Pooling suffers from having a larger number of parameters that are learned. For example, in bilinear pooling, if the input visual and language features are both of size 2048, then the learnable weight matrix would have more than 12.5 billion parameters. To reduce the number of learned parameters, Kim et al. [35] proposed low-rank bilinear pooling where a low rank factorization ofthe weight matrix is used. Fukui et al. [17] proposed idea of Multimodal Compact Bilinear (MCB) Pooling, an alternative way to reduce the number of learnable parameters through the use of kernels. This method utilizes idea of Gao et al. [19] applied to multimodal case. Specifically, in MCB, the input features are projected into a higher dimension space using the Count Sketch projection function [9], and the Fast-Fourier Transform (FFT) is used to efficiently convolve the count sketch vectors by transforming the convolution into an element-wise multiplication.

2.2.4 Reasoning Based Models

The introduction of CLEVR has inspired the development of VQA models focusing on visual reasoning. Many of these models make use of the idea of a semantic parser to parse the question into a program. Semantic parsers are commonly used in the NLP community for text-based QA. This approach was adapted to VQA in the a line of work on CLEVR [7, 6, 32, 26, 50]. Andreas et al. [7] proposed Neural Module Networks (NMN), modular composable functions that were built from neural networks. These module correspond to different visual and logical operations such as: find, transform, combine, describe, and measure. 'Find'

is an attentional operation on the image. It transforms the input image to an attention map, and can be used to attend to a region of the image matching either an object or attribute. 'Transform' modules it into a two-layer perceptron. The 'combine' module merges two attentions using pipeline of stack, convolution and ReLU operators. The 'describe' module maps attention combined with image on the label space. Measure takes attention and maps it on labels space. This building block is good for answering counting questions. With these building blocks, the problem of VQA decomposes into parsing the question into a the neural program composed of the neural modules and then running the program on the image. The neural program or layout specified how to arrange and connect the modules. In the original NMN, the layout was deterministically obtained from running the Stanford Lexicalized Parser [38] and converting the resulting constituency parses to dependency parses [51], and then applying another set of rules over the dependencies to obtain the layout. In followup work, Andreas et al. [6] trained a model to generate the layout directly from the question. Since the training objective was not directly differentiable, they used reinforcement learning to train the layout model. Later, Hu et al. [26] proposed an end-to-end network using NMN. Johnson et al. [32] proposed to learn the program using a sequence to sequence model.

An alternative line of work[28, 27] investigated neural network architectures for reasoning. Hudson and Manning [28] noted that work based on NMN relied on human specified set of operators and were hard to train. They proposed MAC (Memory, Attention, and Composition), a new neural architectural cell modeled after standard CPU architectures with a read/write/control units, each a small neural network. The cells connected together in a recurrent manner, similar to a RNN. The MAC model reached 98.9% accuracy on CLEVR and 81.5% accuracy on CLEVR-Human, outperforming other VQA methods at the time such as the NMN line of work [27, 6, 26, 32]. Hudson and Manning [27] then proposed the Neural State Machine, which simulates the computation of a finite state automaton. It operates on a probablistic scene-graph representation of the image and converts the question in to a series of instructions that is run over the graph. We describe the NSM in more details in Chapter 3.

One of the current best performing model on the CLEVR dataset is Neural Symbolic-VQA [67]. This method is so good at answering CLEVR that it even outperforms humans! For each image Mask R-CNN [24] segments every mask over each object of the scene along with its material, color, shape and size. Each of the produced outputs is paired with original images and fed to ResNet-34 [23] to extract the spatial pose and 3D coordinate. For translation from questions into programs Yi et al. [67] utilise encoder-decoder (the encoder is a bidirectional LSTM, the decoder is a unidirectional LSTM) model fused with attention over its both outputs. The next part of the model is the program executor, which is a collection of deterministic Python functions that operates over visual output guided by generated program. Training of the language parser is a two-step procedure. They first train it on a small

subset of programs with direct supervision and then train in a reinforced learning manner to fine tune it on the full set of programs. A similar idea is used by Neuro-Symbolic Concept Learner [50]. Surprisingly, this model performs slightly worse compared to the predecessor. It utilizes the same concept with one difference: instead of learning attributes and working within natural language it operates with a concept space. Another very minor difference is the Neuro-Symbolic Concept Learner used a GRU instead of a LSTM for processing the language.

2.3 Evaluation metrics

VQA is typically evaluated under had two settings: an open ended setting where the model has to generate a free-form string as the answer, and a multiple choice setting where the model selects the most likely answer. For both settings, accuracy is the most prevalent metric in Visual Question Answering.

2.3.1 Evaluating multiple choice answers

In the multiple choice setting, evaluation is simple and non-ambiguous. Models are easily evaluated using accuracy as there is no ambiguity as to whether an answer is correct or not.

$$accuracy = \frac{\text{number of correctly answered questions}}{\text{total number of questions}}$$

2.3.2 Evaluating free-form answers

On the other hand, evaluating free-form answers can be tricky. The problem of comparing a generated answer with reference answers is similar to determining if two phrases are paraphrases of each other, which remains a unsolved task in NLP. If exact string match is used, models that produce a free form answer that differs form the correct answer by even a single character would be considered wrong. It does not differentiate between roughly correct answers (‘kitten’ vs ‘cat’) and complete incorrect answers (‘kitchen‘ vs ‘cat’). For example, the model’s answer ‘grey’ to the question ‘what color is that door? ‘ when the correct answer is ‘gray’ will be incorrect for this metric same as ‘giraffe‘. For humans, it is obvious that these two cases are not equivalent and the first one should be penalized much less than the second. To handle these issues, researchers use one of the following metrics:

Comparison against reference answers

One way to evaluate free-form answers is to compare it against reference answers. When multiple answers are solicited, it is common to take the majority answer as the correct one. Antol et al. [8] proposed to consider an answer to be correct if it matched responses provided by at least k worker. The VQA dataset has 10 answers for each question and for

evaluation they defined the accuracy to be:

$$accuracy = \min\left(\frac{\text{number of people provided that answer}}{3}, 1\right)$$

This partially addresses the issue with using exact match for accuracy and allows for multiple possible answers to a question. However it does not completely solve the problem of answers that are paraphrases. This metric requires agreement between at least three people before an answer is considered to be correct. However, there can be many different answers that are equivalent. For instance, English has many words that have multiple synonyms. Specifically in the VQA dataset, 59% of 'why' questions has less then 3 identical answers from people, which makes it impossible to answer questions these perfectly. For 13% of 'yes/no' questions both 'yes' and 'no' are considered correct answer by this metric as each was provided by workers as an answer more than 3 times.

Wu-Palmer Similarity (WUPS) [64]. It measures the semantic difference between words based on their distance in WordNet [16]. WUPS assigns a number from 0 to 1 to pair of words by finding least common subsummer (LCS) of both words in the WordNet hierarchy. The score is computed based on depth of the two words (the predicted answer and the correct answer), and the depth of the LCS:

$$WUPS(word_1, word_2) = \frac{2 \times depth(LCS)}{depth(word_1) + depth(word_2)}$$

Note that this metric has several drawbacks. For one, it can only cover words that are present in WordNet. In addition, the WordNet hierarchy is known to have inconsistent depths based on topic. Along with it WUPS does not consider questions about very specific topics.

2.3.3 GQA metrics

For evaluating VQA models, accuracy is a good initial metric. However, it does not take into account whether some questions are more difficult to answer than others. For example, binary questions are obviously easier to answer than questions with a larger answer space. thus often results are broken down by question type. Some questions may require more reasoning steps then the others. One simple proxy for the number of reasoning steps is the question length and analyze performance by question length. However, in some datasets [31] question length is not a good measure of the number of reasoning steps since the question may contain extra information or unnecessary phrase. This is often done to train models that are more robust to irrelevant information in the question. Another issue with accuracy is that it does not provide any insight into why the model answered the way it did. For instance, it is possible for a model to ignore important keywords of the question but still produce the correct answer by chance [1]. Value of the most common questions answered

correctly should has less impact on the final metric value then the value of more distinct ones. To address these issues, Hudson and Manning [29] introduced the following evaluation metrics

Consistency. This metric measures consistency between responses. For example if a model answer with a 'yes' to the question 'Are there any armchairs in the room ?' then for the question 'How many armchairs are there in the room ?' the models should answer with such a number that is one or more. To compute this metric, a set of entailed questions with inferred answers need to be defined. Current, the GQA dataset is the only dataset with a set of entailed questions for computing this metric. For each correctly answered question, the metric measures the model's accuracy over the entailed set passing score for a single question. After that, all scores are averaged into the final result.

Validity and **Plausibility.** *Validity* checks whether the answer given is related to the question scope, e.g. giving an answer with a number to a counting question. *Plausibility* measures whether the answer given is a plausible. For example, answering with '50' to the question 'How many walls are there in the scene?' sounds implausible. For now, very few datasets have data available to conduct analysis based on this metrics. We added data in 3DVQA to introduce validity computation and planning to add plausibility. Both scores computed by dividing number of validly (plausibly) answered questions by total number of questions.

Distribution. The Distribution metrics measure how well the distribution of the model predictions matches the distribution of ground-truth answers. This metric shows, if the model correctly predicts only most common questions or less frequent ones. This metric is computed by applying Chi-Square statistic over predicted and truth distribution.

Grounding. Grounding allows us to evaluate whether a model is able to ground or attend to relevant regions in the image, or whether it relies on language priors. This metric is only applicable to attention based models. It checks whether the model attends to the correct region that is relevant to the answer. For a dataset to support computation of this metric, the dataset needs to provide additional information In addition to the standard (image, question, answer) triplet. This can be in the form of a dense alignment of text to regions as found in Visual Genome, or specification of relevant region for the answer. Both the GQA and our 3DVQA dataset provide information for evaluating this metric.

Chapter 3

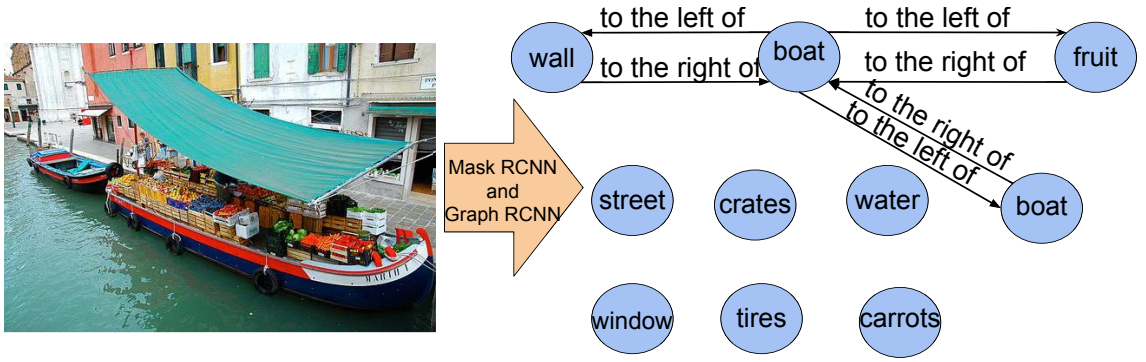
The Neural State Machine

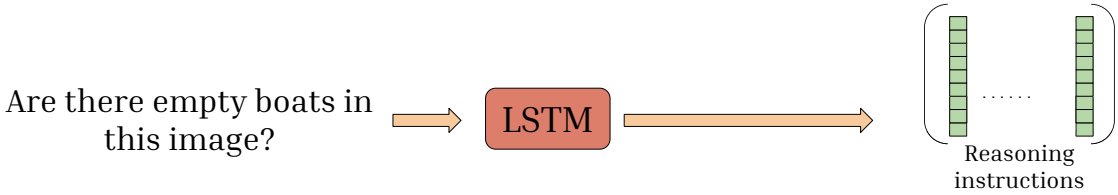
The Neural State Machine [27] (NSM) is a sequential VQA model. It reasons using language over a probabilistic scene graph by using a specific attention mechanism. The NSM is divided into two stages: modeling and inference. In the modeling stage, the image is transformed into a scene-graph, and the question is converted into a sequence of reasoning instructions. During the inference stage reasoning is performed by processing the language instructions and shifting attention over the scene graph.

3.1 NSM model

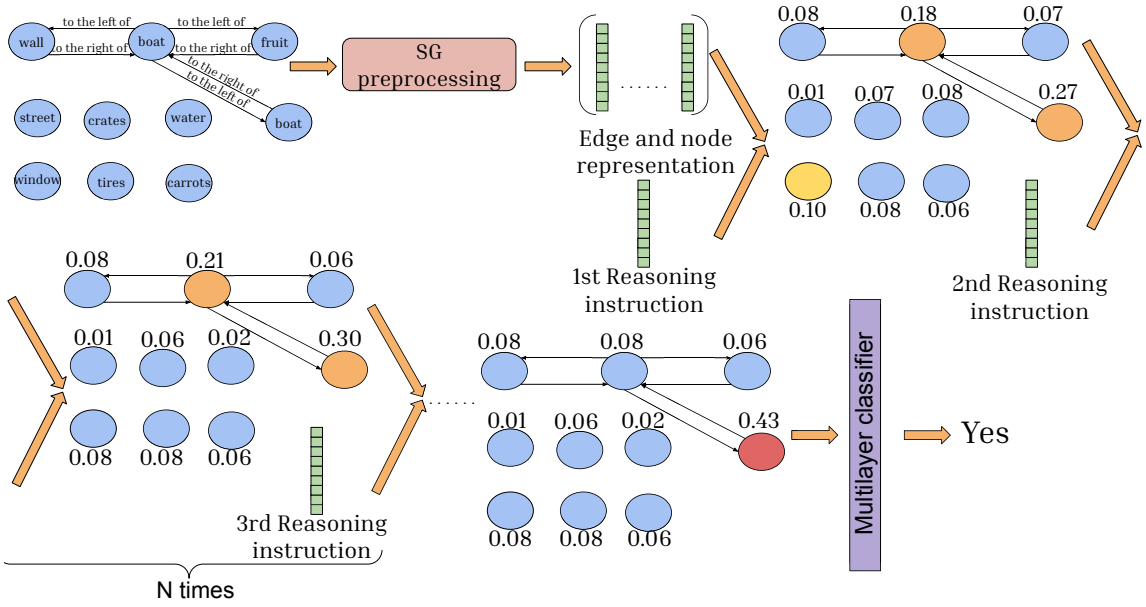
The NSM is a sequential model where on each step of computation it redistributes weights associated with the nodes of a probabilistic scene graph. The NSM is an extension of finite state machine where the computation is performed by a neural network and the alphabet, states, and edges have embeddings associated with them. The simulation of the NSM is similar to the computation of a finite automaton [25] over the scene graph, with only difference in an initial weight distribution. The NSM runs for a a constant number of steps for instead of reaching an end state. Mathematically, the NSM can be described as a set $(C, S, E, \{r_i\}_{i=0}^{N}, p_0, \delta)$ where:

- C is the model's alphabet represented as embeddings
- S is the collection of states (nodes of the scene graph)
- E is the collection of edges (edges of the scene graph)
- $\{r_i\}_{i=0}^{N}$ is the set of reasoning instructions
- p_0 is the initial probability distribution
- $\delta : p_i \times r_i \rightarrow p_{i+1}$ is the state transition function which redistribute weights between states at each step





(a) Modeling stage



(b) Inference stage

Figure 3.1: he modeling (a) and inference (b) stages of the NSM. During the modeling stage (top), the image is translated into a scene graph,and the question into reasoning instruction. During the inference stage (bottom), the reasoning instructions are used to re-distribute attention over the scene-graph nodes. At the end of the process, the final scene-graph representation is concatenated with the encoded question (last hidden state of the LSTM encoder), and passed into a multilayer classifier to obtain the final answer.

In the NSM model, the model's alphabet maps to a concept vocabulary *(C)*, initialized with GloVe vectors that will be used to capture and represents the semantic content of input images. The vocabulary is divided into three parts: set of object identities ($C_O = C_0$),

set of concepts for set attributes $A_C = \bigcup_{i=1}^{L} C_i$ (where L is the number of attribute types and C_i is a set of concepts for attribute a_i), and set of relationship concepts $C_R = C_{L+1}$. Similarly a set of embeddings *(D)* for properties is introduced.

3.1.1 Modeling stage

At the beginning of the modeling stage a probabilistic scene graph (PSG) is created. Unlike a regular scene graph (SG) which has only one prediction without probabilities, the nodes of a PSG, corresponding to objects in a image, is equipped with a set of label predictions with corresponding probabilities. Similarly, probabilities are kept for attributes and relationships.

To represent information on the scene graph, for each node $s \in S$, the NSM maintains a set of $L+1$ property variables $\{s^j\}_{j=0}^{L}$. Each property variable s^j is an embedding vector that is a weighted sum of concept embeddings:

$$s^j = \sum_{c_k \in C_j} P_j(k) c_k$$

where $c_k \in C_j$ denotes each embedded concept, and P_j refers to the probability distribution of a corresponding property. By maintaining a probability distribution over properties, the NSM can maintain information for multiple values of an attribute. For instance, if an object has two colors (e.g. a fabric with checkerboard pattern), then the NSM can distribute probability over the two colors. In contrast, a model that relies on one predicted attribute will only be able to capture a single color. Similarly the edge representation is defined as:

$$e' = \sum_{c_k \in C_{L+1}} P_{L+1}(k) c_k$$

for each edge $e \in E$.

To generate reasoning instructions from the language, the first step is to embed words in the instruction to the closest matching word in the model's . Vocabulary for unknown words that do not match any of the dictionary words closely, the default embedding c' is used. To find the closest word in the concept vocabulary for embedded word w_i, a distribution based on similarity is computed

$$P_i = \text{softmax}(w_i^T \mathbf{W} C)$$

where w_i is an embedded word and W is a weight matrix which is initialized to the identity matrix. Next we represent each word as a linear combination of words from the concept vocabulary and c' literal. Note that all structural and non-content words, such as 'the' and 'and', are treated as unknown words.

$$v_i = \sum_{c \in C \setminus c'} P_i(c) c + P_i(c') w_i$$

Once we have the embeddings for the question, they are passed through an encoder-decoder LSTM to obtain the reasoning instruction. Given a question of P normalized words $V^{P\times d}$ the encoder LSTM outputs its final state q which is used to represent the encoded question. Then a decoder LSTM outputs N+1 hidden states $\{h_i\}_{i=0}^{N}$ which are transformed into reasoning instruction

$$r_i = \text{softmax}(h_i V^T)V$$

The reasoning instructions are obtained by taking a normalised attention over question words and guides the state machine during inference.

3.1.2 Inference stage

During the inference stage, the NSM performs its sequential computations and outputs final scores for the answers. Based on reasoning instructions $\{r_i\}_{i=0}^{N}$, a neural module δ acts as the state transition function and shifts attention weights from p_i to p_{i+1}.

We first find the property type that is most relevant to the instruction r_i by computing:

$$R_i = \text{softmax}(r_i^T \circ D)$$

where $D \in R^{300\times(L+2)}$ is a set of property embeddings.

During the inference stage we determine the probability distribution over the states at each reasoning step. For that purpose, we sequentially compute the relevance score for nodes and edges:

$$\gamma_i(s) = \sigma\left(\sum_{j=0}^{L} R_i(j)(r_i \circ \mathbf{W_j} s^j)\right)$$

$$\gamma_i(e) = \sigma\left(r_i \circ \mathbf{W_{L+1}} e'\right)$$

where e' is an embedding for the edge type and $\mathbf{W_j} \in R^{300\times L+1}, \mathbf{W_{L+1}} \in R^{300\times 1}$ are learned parameters with 300 corresponding to the embedding size.

After obtaining the relevance scores we redistribute attention p_i from the current nodes $s \in S$ to its reachable neighbors:

$$p_{i+1}^s = \text{softmax}_{s\in S}(\mathbf{W_s}\gamma_i(s))$$

$$p_{i+1}^r = \text{softmax}_{s\in S}(\mathbf{W_r} \sum_{(s',s)\in E} p_i(s')\gamma_i((s',s)))$$

$$p_{i+1} = R_i(L+1)p_{i+1}^r + (1 - R_i(L+1))p_{i+1}^s$$

Here we obtain new probability distribution by averaging transition probability p_{i+1}^r and probability of state being addressed by the guidance signal p_{i+1}^s. To obtain the final encoded representation m for the scene graph, we average first by instruction type (R_N) and then

by attention over the states (p_N). This can be described by the following equation:

$$m = \sum_{s \in S} p_N(s) \left(\sum_{j=0}^{L} R_N(j) s^j \right)$$

Answer Prediction

To predict the answer, once the NSM is simulated for a fixed number of steps ($N = 8$), the final representation m is concatenated with the question vector q and is fed into a MLP classifier: 2-layer fully-connected layers with ReLU and dropout followed by softmax over the entire vocabulary.

The process described above allows us to simulate state machine guided by instructions derived from the questions. Given the image and question, the model infers a probabilistic scene graph over which the state machine runs. The language inputs are transformed into instructional guidance signals. All representations are determined using embeddings of the concept vocabulary, which adds alignment in two different streams of data. Then, the model starts its iterative pass, at each step it shifts along detected relationships - edges of scene graph and its attention between detected objects - nodes of scene graph. It allows determining image patterns which were addressed by a certain part of question and reasoning over SG to finally focus on the answer.

3.1.3 Scene Graph Generation

As input to the NSM, probabilistic scene graphs are first extracted from the images. The predicted scene-graphs were generated using a variant of MaskRCNN [24] to identify object masks and a graph R-CNN [65] is used to identify edges between the objects. For MaskRCNN, ResNet-101 [23] is used for feature extraction and FPN [43] for region proposal. Instead of just predicting the object class in a flat manner, the detection heads were updated to predict the object class and category using a hierarchical softmax, as well as the object attributes for each property type. This results in a probability distribution for the object properties. The scene graph structure is predicted following Yang et al. [65]. To reduce computational overhead and produce a sparser graph, edges are restricted to close by objects (objects with a distance that is less than 15% of the image dimensions in both axes). After obtaining the scene graph structure, graph attention networks [63] was used to predict the label of the edges and obtain probability distribution over the relation type for each edge. The object detector and graph generation model were both trained over the normalized VG scene-graphs provided by GQA.

3.2 GQA

We run experiments on the GQA dataset [29] on which the original NSM model was developed and evaluated. The GQA dataset is designed to be a balanced dataset constructed using scene-graph from Visual Genome for real-world images. It first normalizes the scene-graphs from VG and then construct synthetic questions and answers, ensuring diversity in the questions and answers.

We chose to evaluate our solution on GQA dataset due to its novelty and richness. The dataset is a recent general-purpose VQA dataset and it uses careful balancing to remove biases from the data. In addition, it is one of a few datasets with scene graphs which are required for our model.

3.2.1 Dataset Structure

The full GQA dataset consists of 113K images and 22M exhaustively generated questions. Answers exist in two forms: single word and a detailed response. Answers are associated with relevant regions in the image, which allows non standard VQA models to be trained. For each question a functional program is constructed. It describes reasoning steps to be performed which allows a tight control over answer distribution.

3.2.2 Dataset statistics

Hudson and Manning [29] used rejection sampling to take a subset of questions to balance the diversity of question and answer types. At the end of balance, the final dataset has 1.7 million questions. The dataset consist of five question groups: *verify, query, choose, logical, compare*. The most common category is *query*, which occupies more than a half of the dataset. The dataset was also balanced by number of semantic types. There are four types of them: object, attribute, category, relation and global. The dataset has distribution over the number of semantic steps needed to resolve the question. They include relating objects between each other, determining its color and any other interaction with the scene graph (see Figure 3.2 for distribution of the GQA dataset). The dataset is split into 70% train, 10% validation, 10% test and 10% challenge.

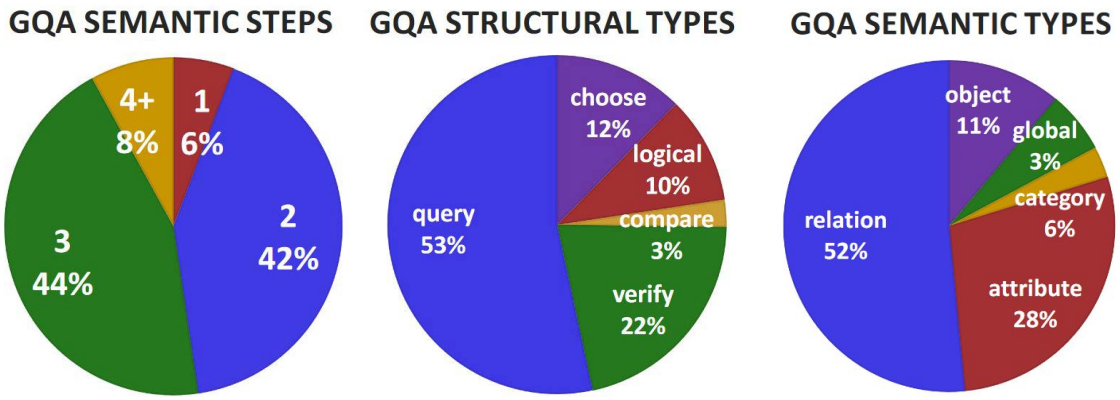


Figure 3.2: GQA distribution by semantic steps, structural types, semantic types. Figure reproduced from Hudson and Manning [29]

3.3 Re-implementation and Experiments

3.3.1 Implementation

Since the original NSM code was not released, I re-implemented the NSM using PyTorch following the details provided in the paper[1]. While I aimed to follow the design of the NSM as specified, there are certain details and components that were challenging to match. Hudson and Manning [27] did not provide any information about number of groups (L) in concept vocabulary, as well as the initialization for learned vector of <UNK> literal.

For my re-implementation of the NSM, I chose to use an enlarged concept vocabulary. Similarly as the original code the concept vocabulary was not released and I collected it based on the GQA dataset. By enlarging the concept vocabulary, it was no longer necessary to map to the closest word (except non-content words which are mapped into <UNK >). Another aspect of the original paper that was difficult to replicate was the extraction of predicted scene-graphs from the image. The authors did not provide the predicted scene graphs and the paper lacks details necessary for full re-implementation. I investigated using an existing scene-graph predictor [61] to generate the predicted scene-graphs, but found it was challenging to train and run due to memory limitations. In addition, for the full re-implementation of the scene-graph prediction model, it was necessary to update the model to predict attributes and determine appropriate hyperparameters. As scene-graph generation is not a core component of the main NSM model, I focused on running experiments with the ground-truth scene-graph instead of the predicted scene-graph. While the overall VQA system performance will improve with the use of state-of-the-art scene graph generator, it is

[1]At the beginning of the project I tried contacting the first author, Drew Hudson, for the code. While she indicated that she would like to release the code as soon as possible, the original implementation for the NSM is still unavailable as of the time of this thesis.

not a direct component of the NSM model. In this thesis, I aimed to improve performance of the NSM which is responsible for Visual Question Answering, not scene graph generation.

Implementation details

For the experiments we utilized the same parameters and initialisation for the network as in original paper. Namely number of reasoning steps is equal to 8, batch size to 32, number of epochs to 15, learning rate to 10^{-4} and its scheduling, GloVe vectors, output answer space.

For our version of NSM we used Adam optimizer, while original paper used SGD. Experiments with SGD gave me 1% less then ADAM in the same epoch. SGD was reaching peak accuracy around 5-7 epochs later than ADAM. This allowed us to reach the reported results on 10th epoch and the rest of the training does not increase the accuracy more than 0.004%. Also computation speed using ADAM was significantly higher.

For our experiments we chose GloVe vectors with length of 300 for initialisation of learned vectors. The answer space for the model final classifier is words from the GloVe dictionary. After experimenting with various hyperparameters, we determine that most beneficial number of reasoning steps is 8 for GQA dataset. We experimented with different learning rates, initialization of learned parameters, and number of reasoning steps. I followed the NSM paper and set the initial distribution to the uniform distribution over the nodes, or to $1/n$ for every node, where n is a number of nodes. I also experimented with sampling the weight of each node from a Gaussian distribution ($\mathcal{N}(1/n, 1/n^2)$) within the first quantile and then normalizing, but found that it hurt the model's performance. For reasoning steps we experimented with values between 4 and 10, the results are shown on Figure 3.3. Adjusting the learning rate from 0.01 to 0.003 gave significant improvement,with lowering the learning rate further resulting in worse results. Also, I noticed that there is no significant difference in the results obtained from models trained with learning rates within 0.003 ± 0.002.

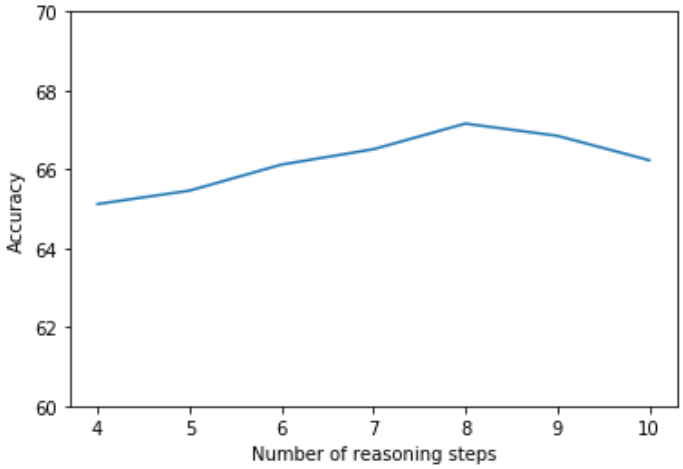


Figure 3.3: Accuracy of NSM depending on number of reasoning steps

Model	Scene Graphs	Binary	Open	Consistency	Validity	Plausibility	Distribution	Accuracy
NSM re-implemented	Predicted	85.39	53.08	96.14	97.58	86.23	3.19	67.16
NSM original	Ground truth	78.94	49.25	93.25	96.41	84.28	3.71	63.17

Table 3.1: Evaluation of NSM on the GQA validation split. Note that the two rows are not directly comparable since for the re-implemented model, we use ground-truth scene-graphs, while for the original results from the NSM paper is with predicted scene-graphs.

Models were trained on a workstation with a Core i9-9900K CPU and RTX 2080 Ti GPU. Training lasted 15 epochs with ADAM [37] optimizer. Initial learning rate of 0.003, decaying the learning rate by half every two epochs. Running time and memory usage of re-implemented model matches the same values of reported model, namely 30 hours and 11GB. The model is trained on train split, the reported accuracy is for test split of the data.

3.3.2 Experiments

Table 3.1 shows results of the results of the re-implemented NSM on ground truth scene-graph compared with the original NSM results with predicted scene-graphs as reported in the paper on the GQA dataset. The results indicate that my re-implementation of the NSM is able to achieve reasonable performance on GQA with ground-truth scene-graphs. Note that the results in Table 3.1 is not directly comparable due to the use of ground truth scene graphs for the re-implementation and the predicted scene-graphs for the original NSM.

From Table 3.1, we can see a clear advantage from using ground-truth scene-graphs. The results show that ground-truth scene-graphs enhance the model's ability to provide more plausible and valid answers (higher Plausibility and Validity), and correctly predict less frequent answers (lower Distribution). Comparing binary and open-ended questions performance, we see that binary questions benefit more from ground truth scene-graphs.

Chapter 4

NSM Extensions to 2D VQA

In this chapter, I describe two extensions that I make to the NSM model: attention localization and using bilinear attention networks (BAN [36]). Attention localization helps NSM to consolidate its reasoning on a very narrow area which prevents redistributing of model confidence on irrelevant data. Bilinear attention is a well known model for Visual Question Answering which shows strong reasoning skills. I decided to test its performance on scene graph reasoning by replacing the existing NSM inference stage with a BAN.

4.1 Attention localization

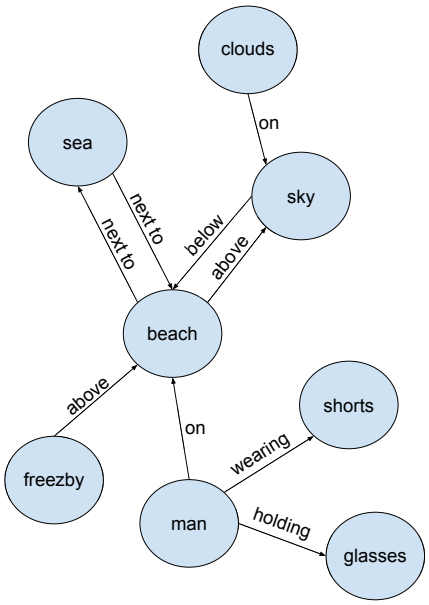
We make the observation that questions asked by people are usually answered with an object located on the image close to question subject, ignoring questions which do not have an object related answer like yes/no, left/right part of the image, etc. Consider the example question 'What is in the sky?' from the VQA dataset (see Figure 4.1 for image and scene graph). Since the original VQA dataset did not have scene graphs so we asked 10 volunteers to draw a scene graph for the given image. We consolidated the 10 scene graphs and selected the most common objects and relationships).

In this example, the question subject 'sky' is located one edge away from the answer 'clouds'. The edge corresponds to the only spatial relationship used in the question - the word 'on'. If we can correctly identify this central object ('sky') from the question in the scene-graph, and only focus on the narrow sub-graph of radius one around it, we can easily identify the answer ('cloud') as there are less nodes to distribute the probability over.

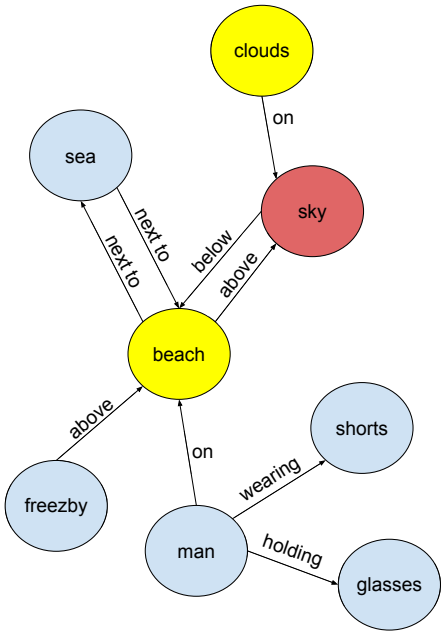
We formalize the idea by introducing the following terminology and notation:

- **Question radius** is the number of relationship words in the question. *(Q_R)*
- **Question center** is a node in the scene graph which refers to an object from question or to an object with the attribute mentioned in question. *(Q_C)*
- **Answer center** is a node in the scene graph which refers to the central object from the answer. *(A_C)*

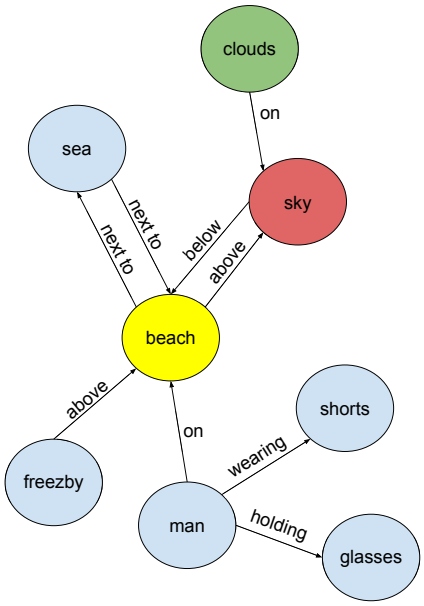
(a) VQA image for 'What is in the sky?'



(b) Constructed scene graph



(c) Red node - question center, yellow nodes - nodes of narrowed scene graph



(d) Red node - question center, green node - answer center

Figure 4.1: Scene-graphs illustrating the narrowing mechanism for attention localization. From the image (a), we construct a scene-graph (b). Then we predict the question center and question radius from the question, and use it to extract a sub-graph for the attention weights to distribute over (c). Finally, the answer is show in green (d).

- **Narrowed questions** is a set of questions with A_C within Q_R+1 distance from Q_C.
- **Accuracy narrowed** is an accuracy on questions with A_C within $Q_R + 1$ distance from Q_C.
- **Graph distance between A and B ($d(A, B)$)** is the length of the shortest path between node A and node B.

To implement the idea, we adjust the NSM modeling stage to compute Q_R and Q_C for each question. To obtain the question center, Q_C, we identify the first object in the sentence. Recall that in the original NSM modeling stage, each word was matched to an embedding in the concept vocabulary. Using this mapping, we know what words map to objects, attributes, and relations. Taking the first object, we find the corresponding node in the scene graph. If there are multiple matching nodes, we use the attributes and relations to avoid ambiguity. Since we have identified the relation words in the sentence, we can easily compute the total number of relations in the sentence. We use as the question radius (Q_R) the number of relationship words in the sentence plus one. We retain every vector (namely vectors e' and s^j) of data within $Q_R + 1$ graph distance from Q_C and mask out vectors beyond the $Q_R + 1$ graph distance. In our NSM with localized attention, the model will predict the question radius $Q_R + 1$, a question center Q_C. Using the predicted $Q_R + 1$ and Q_C, it will mask out (set to zero) attention weights for nodes and edges that are more than $Q_R + 1$ edges away from Q_C. In Figure 4.1c a narrowed graph is shown with yellow nodes. This redistribution will allow the model to concentrate more on the most probable nodes and ignore irrelevant objects in the image. In Figure 4.1d shows that the green node - answer center lies inside the narrowed graph. Note that this extension can be applied only to questions with answers that involve objects (i.e. the answer is an object, or is an attribute of an object, or a object relation). Both GQA and CLEVR datasets also contain questions for which this localization idea does not apply. Based on the question we determine its type and then determine if it might be useful for us to apply our idea to it. For this purposes I created a dictionary of keywords, which define the question type. Our dictionary consist of 8 words which cover all questions which prediction can be improved. Namely, these words are: 'what', 'which', 'who', 'on', 'how', 'what's', 'where', 'the'. For example 29% of all GQA questions starts with 'what' which is a part of our dictionary. Similarly for CLEVR dataset size of the dictionary is 10. For example, questions about the global position of an object in the image where the answer would be 'the left bottom corner' or 'the right middle part' of the image.

For deeper understanding of how this idea should work let's look at example on Figure 4.1. The question and answer pair is 'What is in the sky? Clouds', so in this case the question center (Q_C) would be sky and answer center (A_C) would be clouds. 'In' is a relationship word, so $Q_R = 1$ here, so from Figure 4.1c we get a subgraph which remain untouched. The full SG consists of 8 nodes which provides a 8×8 attention map in the

original NSM. In NSM with attention localization a meaningful part of the attention map will be squeezed to the size of 3×3, which allows the model to concentrate on the most plausible nodes. Formally one could think of applying function f on attribute embeddings:

$$f(attr) = \begin{cases} attr, & \text{if object related attribute within} \\ & Q_R \text{ from } Q_C \\ 0, & \text{otherwise} \end{cases}$$

And for each relationship from E:

$$f(E(m,n)) = \begin{cases} E(m,n), & \text{if } n \text{ within } Q_R \text{ from } Q_C \\ 0, & \text{otherwise} \end{cases}$$

To check the feasibility of this idea we computed statistics on both the GQA and CLEVR dataset (using predicted values of Q_C, Q_r and A_C over ground truth scene graphs provided along with the datasets. Similar to how we computed Q_C, we compute A_C by taking the provided answer and identifying nodes from the scene graph which correspond to it. In case if there are multiple nodes that match, we take the farthest (in terms of graph distance d) node from Q_C. We then grouped the data into questions with answers involving an object and those that did not (for these, the narrowing idea is not applicable). Of the questions with answers that involve an object, we checked whether there was 1) no path between Q_C and A_C, 2) the distance between the two centers is within the question radius (d(Q_C, A_C) $< Q_R$) and 3) the distance is greater than the question radius d(Q_C, A_C) $> Q_R$. We report the results of our analysis in Table 4.1. The percentage of questions that can be potentially improved is 48% and 34.3% for GQA and CLEVR datasets respectively. We believe that there are answers which are not within the distance or do not have a path in the scene graph between Q_C because the GQA scene graphs are incomplete and noisy. In contrast, for the synthetically constructed CLEVR, all answers involving objects are within the narrowed subgraph. On GQA, the basic version of NSM predicts only 72% of answers questions are within the radius and only 52% of them are correct. The gap indicates that there is space for potential improvement.

Figure 4.2 shows the distribution of distances between question centers and answer centers for questions with answers involving an object. Note that these questions correspond to the data samples that can be improved using the narrowing idea. To understand the coverage of the question radii we also plot the distribution of the full scene graph radii (see Figure 4.2c, 4.2d). By comparing these two sets of histograms, we see that for the full graphs the area where attention is being distributed is much larger than for the localized sub-graph. By comparing the graphs for the two datasets, we also see that GQA has larger variance of radii due to the fact the images were captured in the real world and annotated

	GQA		CLEVR	
Condition	Count	Percent	Count	Percent
no path	2192	1.7	0	0
within	58141	44.0	240321	34.3
not within	3825	2.9	0	0
not object related	67904	51.4	459668	65.6

Table 4.1: Statistics on whether the question/answer center have no path, are within the narrowed subgraph, not within, or the answer does not involve an object (in which case, the narrowed attention localization does not apply).

by humans, while CLEVR uses a deterministic algorithm for generating the images and questions-answer pairs.

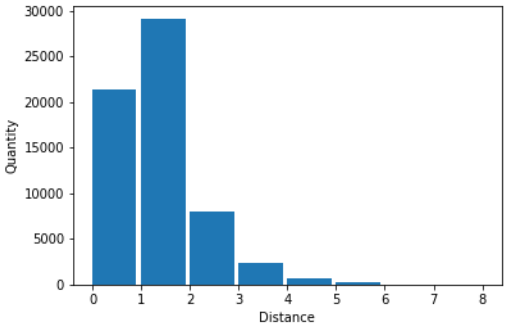
In the bottom part of Figure 4.2, we plot the distribution of $Q_R + 1$. Comparing Figure 4.2f,4.2e with 4.2b,4.2a an overlap between Q_R+1 and the actual radius can be noted. In this thesis, I determined the Q_R based on the number of relationship words in the question. It is also possible to use a model to learn to predict the Q_R (the localization radius). The overlap between $Q_R + 1$ and the actual radius indicates that there can be gains in performance by learning the localization radius Learning of the question radius might reduce the attention localisation area only in case of precise enough predictions. In order for this model to induce better localization, it should produce a smaller total sum of differences (between prediction and minimum possible distance between Q_C and A_C) compared to the current baseline (which deterministically assigns $Q_R + 1$ to each question).

4.2 Bilinear attention in scene graphs

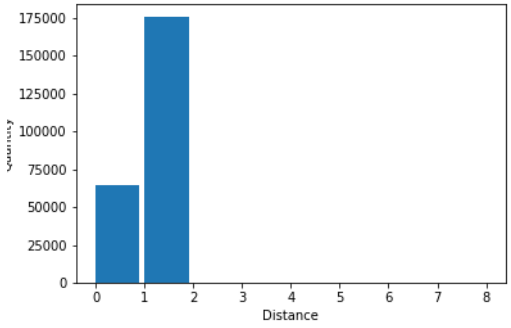
The other extension we explored was to use Bilinear attention [36] as the alignment method. In this case we retain the modeling stage of NSM, and use the same technique of language and image preprocessing as before. The inference stage is reworked so that it follows Kim et al. [36].

Recall that in the modelling stage we have transformed the extracted probabilistic scene graphs into set of vectors representing edges and nodes. Each word in the question is transformed into the closest concept embedding (GloVe is used for the embedding), and is passed into an encoder LSTM. The last hidden state of the encoder is passed into a decoder LSTM which is unrolled for N steps to convert the sequence into N hidden states. Then the hidden states are combined with the initial question representation to obtain N reasoning instructions. We take these reasoning instructions and the encoded probabilistic scene graph as inputs into the BAN model.

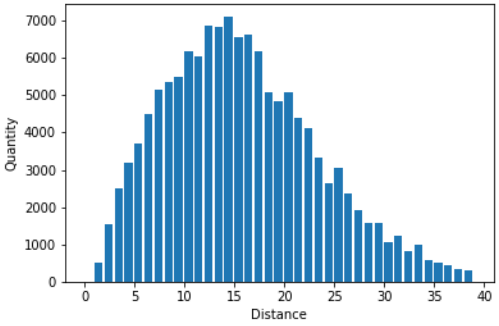
In the inference stage, we replace the shifting attention used to simulate the NSM with BAN model to fuse the two inputs. In the inference stage, we stack the N generated



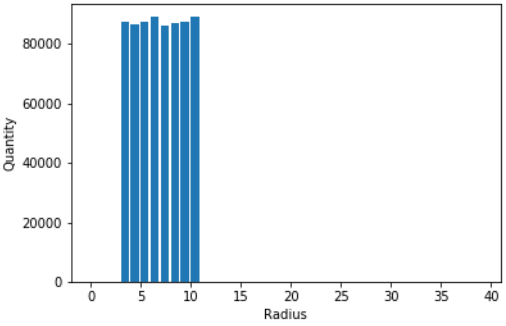
(a) Distance for GQA dataset



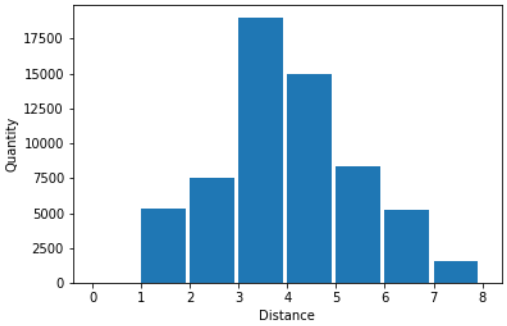
(b) Distance for CLEVR dataset



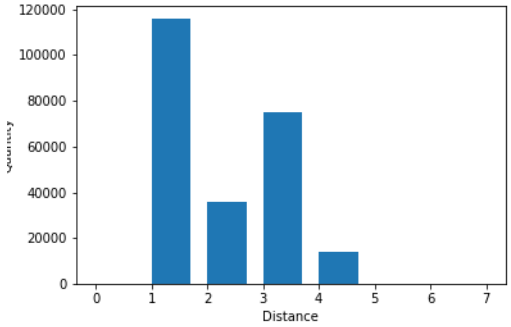
(c) Total graph radius for GQA dataset.



(d) Total graph radius for CLEVR dataset.



(e) Q_R+1 distribution for GQA dataset.



(f) Q_R+1 distribution for CLEVR dataset.

Figure 4.2: Distribution of distances between A_C and Q_C for questions which can be improved (top part). Distribution of Q_R+1 (middle part). Distribution of Q_R+1 (bottom part.

reasoning instructions into $\mathbb{Y} \in \mathbb{R}^{N \times 300}$, where 300 is an embedding dimension. For each node of scene graph we construct a representation $S_n^i = \sum_{j=1}^{L+1} s^j$ where $s^j \in \mathbb{R}^{300}$ for node n. This results in a set of $S^i{}_{i=0}^{M}$ where M is the number of nodes in scene graph. We obtain a vector output f by using a bilinear attention map $\mathbb{A} \in \mathbb{R}^{300 \times 300}$. The kth element of f (f_k)

is given by:

$$f_k = (X^T \boldsymbol{U})_k^T \mathbb{A} (\mathbb{Y}^T \boldsymbol{V})_k$$

where $X \in \mathbb{R}^{N \times 300}$ is a stacked node representations, $\boldsymbol{V} \in \mathbb{R}^{N \times K}, \boldsymbol{U} \in \mathbb{R}^{M \times K}$.

The bilinear attention map A is given by:

$$\mathbb{A} = \text{softmax}\left(\left((\mathbb{1} \cdot \boldsymbol{p}^T) \circ X^T \boldsymbol{U}\right) \boldsymbol{V}^T Y\right)$$

where $\mathbb{1}, \boldsymbol{p} \in \mathbb{R}^{300}$. $\circ$ operator denote the Hadamard product. Note that $\boldsymbol{p}$ is a new learnable parameter, in addition to the previously mentioned learnable parameters $\boldsymbol{V}, \boldsymbol{U}$. BAN can be used with multiple attention maps that are chained together. When using multiple attention maps all weights are shared between the different attention maps with the exception of p_g, for which is there a different set of weights for each attention map $\mathbb{A}_g$:

$$\mathbb{A}_g = \text{softmax}\left(\left((\mathbb{1} \cdot \boldsymbol{p}_g^T) \circ X^T \boldsymbol{U}\right) \boldsymbol{V}^T Y\right)$$

At the end after performing multi-channel alignment we sum over channel dimension, concatenate it with LSTM encoder output in order to give model like it is done in NSM. This vector is being fed into two-layer classifier over the whole vocabulary.

4.3 Results and Analysis

We implement the extensions on top of our re-implementation of the NSM and conduct experiments on GQA and CLEVR ground truth scene graphs comparing the two extensions with the basic NSM model.

Model	Accuracy	Accuracy narrowed	Accuracy non-narrowed
NSM basic	52.24%	48.81%	58.56%
with narrowing	53.23%	50.16%	59.28%
Bilinear attention on SG	51.12%	46.84%	57.32%

Table 4.2: Results on CLEVR dataset. Performance of different NSM modifications on full set of questions, on narrowed questions and non-narrowed questions.

We found our extensions to the NSM required significant increases to the training time and memory usage for both GQA and CLEVR. For NSM with narrowing, the memory usage increased by 20% (from 9 GB to 11 GB) compared to the original NSM model. For BAN, network on scene graph the memory usage increased by to 15% (from 9 GB to 10.35 GB). Training time of the models is 30 hours for basic NSM, 112 hours for enhanced NSM and 150 hours for NSM with BAN over SG. Both versions of NSM were written from scratch. For re-implementation of core functions of the BAN model we follow the original paper and source code [36]. Both versions of NSM were trained with the following settings: N = 8

(number of produced reasoning instructions), batch size = 32, learning rate = 10^{-4} and a learning rate schedule of decreasing learning rate every third epoch by 1/3 of its value. The output answer space is constrained to 400000 words which form GloVe vocabulary.

Note that the performance improvement is not restricted to the "narrowed question" for which the answer involves an object (47% of the data). Even for the "non-narrowed questions", the performance improvement is also 2%. Keeping in mind a through careful reader can note that this improvement does not fully describe the general performance improvement, because the specified subset consist of only 47% of original data. That means that questions which were not affected by the improvement also benefited from it. We believe the gain comes from the partitioning of the two groups, and the attention localization itself for the "narrowed" group. From Table 4.1, we also see that the BAN performs much worse than the original NSM. This indicates that the proposed simulation of the NSM was critical to its performance.

Figure 4.3 shows that with narrowing the model is able to achieve higher results on common question types from the narrowed question set. These results also show that verification and logical questions can also benefit from our improvement. And, as it was stated, the untouched question category - comparing does not benefit from the enhancement.

Considering that the difference between model performances for narrowed and non-narrowed versions are almost the same (4.01% and 4.54%) we can assume that the most impactful part of the enhanced model is data partition. That separation into two parts might give internal information to the model, that the answer for the non-narrowed question lies far from the question subject. Such a difference between two classes of questions (affected by the improvement vs not affected) mostly achieved by the imbalance between question groups. As it was described, binary questions and questions about global position of an object on the image can not be improved by the narrowing idea, which leads to that inequality between two classes.

Model	Accuracy	Accuracy narrowed	Accuracy non-narrowed
NSM basic	67.16%	63.15%	71.70%
NSM with narrowing	69.82%	65.28 %	73.50%
Bilinear attention on SG	61.22%	51.13%	67.97%

Table 4.3: Results on GQA dataset. Performance of different NSM modifications on full set of questions, on narrowed questions and non-narrowed questions.

Comparing the difference between the whole and narrowed set of questions for NSM-based models (4%) and the same value for the BAN model (11%) we see considerable difference between them. This indicates that the BAN model is less adapted for solving *query* and *choose* questions (see Table 4.4). These are two categories of questions that are most common in the narrowed questions set in (see Figure 4.3).

Model	Query	Choose	Verify	Compare	Logical
NSM basic	58.80%	65.41%	72.31%	59.55%	81.08%
NSM with narrowing	60.17%	67.00%	75.14%	59.29%	84.94%
Bilinear attention on SG	49.13%	61.73%	71.69%	58.20%	77.08%

Table 4.4: Accuracy on GQA for each question type..

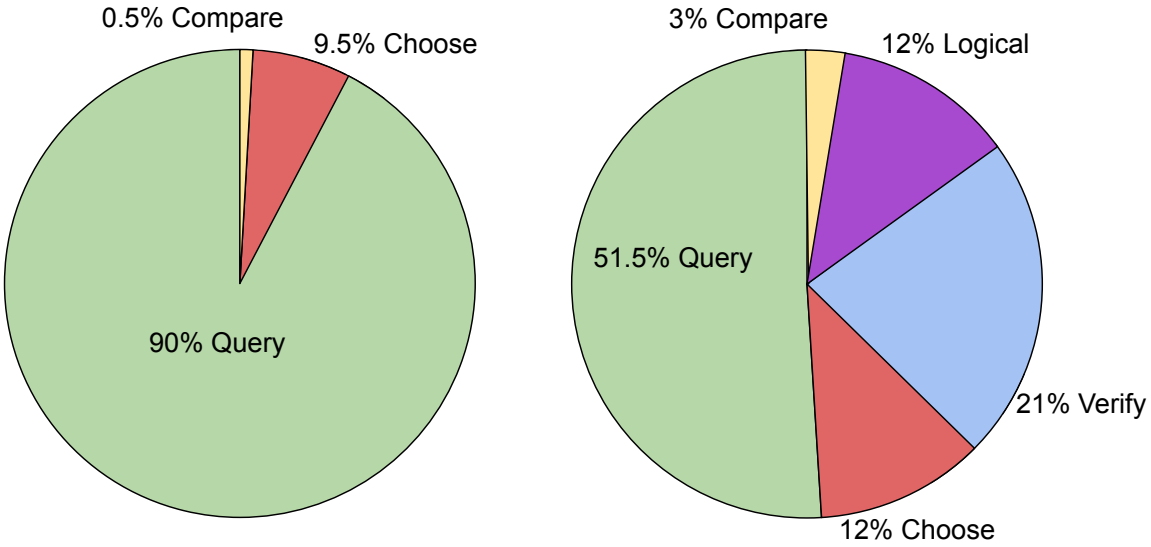


Figure 4.3: Question distribution for narrowed questions set (left) and full dataset (right).

To further show the effectiveness of our enhancement we compute the number of predictions within the narrowed subgraph (see Table 4.5). We see that compared with the basic NSM, narrowing produces a significantly higher number of predictions and correct answers within the narrowed subgraph.

Model	Predictions within	Correct within
NSM basic	19695	12749
NSM with narrowing	21228	14589
Bilinear attention on SG	17784	11203

Table 4.5: Number of predicted answers and correctly predicted answers on narrowed set of questions.

We provide a comparison of the models using metrics introduced by Hudson and Manning [29] for GQA: consistency, validity, plausibility, distribution. From the results in Table 4.6, we see that the performance of the NSM models is very similar, except on distribution where our NSM with narrowing surpasses the others. From the distribution score of NSM with narrowing we see that it predicts significantly less common answers. Also, from the same metric it can be seen that BAN on SG fails to predict more distinctive answers.

Validity scores are almost the same for all models which indicates that they rarely predict answers which are not connected to the question subject. Plausibility metric scores signify that all models produce similarly plausible results (i.e. equally often answer 'chicken' to the question 'What does the chicken eat?')

Model	Validity ↑	Plausibility ↑	Distribution ↓
NSM basic	93.20%	89.66%	108.01
NSM with narrowing	93.32%	89.90%	83.37
Bilinear attention on SG	93.59%	89.87%	145.67

Table 4.6: GQA metrics for implemented models.

For qualitative results, we show a couple examples of 'image, question, answer' triplets which were correctly answered by the narrowed model and wrong using the standard model in 4.7 and Section 4.3.

Figure 4.4: Image A

Figure 4.5: Image B

Image	Question	Prediction NSM original	Prediction NSM narrowed
A	Which color does the sofa have?	purple	brown
A	What vegetable is to the right of the tomato?	tomato	lettuce
A	What is the tomato in?	sandwich	salad
B	Which color is this fence?	orange	blue
B	What is the person to the right of the flag doing?	standing	skiing
B	Is the man to the right or to the left of the device the person is holding?	left	right

Table 4.7: Examples of correctly answered questions by narrowed NSM vs original NSM.

Chapter 5

Visual Questions Answering with Point Clouds

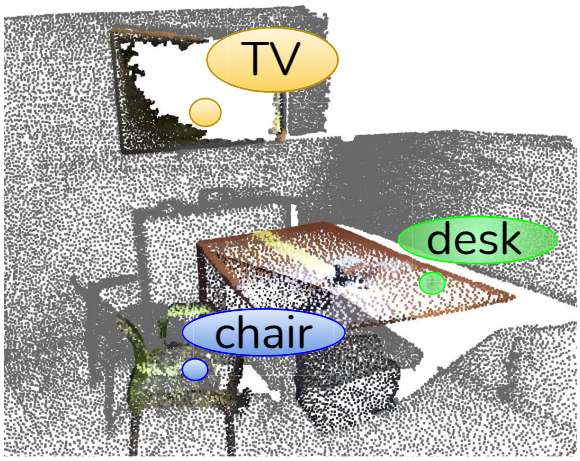
The material in this chapter is from work in collaboration with Yasaman Etesam and Angel Chang. Yasaman worked on developing the dataset, the baseline models and was responsible for running models (except NSM). Leon worked on models design (VoteNet, VoteNet+language, NSM), contributed to the dataset construction and visualization. Angel Chang is the principal investigator and provided guidance, consolidation of ideas and results, and help in writing. This thesis focuses on comparing the NSM with more standard VQA models extended to the 3D VQA setting.

All prior work in Visual Question Answering lies in the area of reasoning in 2D and can not provide expertise in system's ability to reason in 3D setting. There is work on VQA for 360° panoramas [12] but this setting is still subject to the limitations of visual reasoning only within a 2D image domain. Recently, embodied question answering (EQA) [14, 21] was proposed to investigate an agent's ability to answer questions in a 3D environment where it can move, act, and perceive. The EQA setting is quite complex as it couples navigation, interaction, perception, and reasoning. This makes it difficult to disentangle whether failure to provide correct answers is due to the inability to move or act correctly, or an inability to model and reason about the 3D environment. Moreover, dataset biases may mean the agent does not even need perception to provide a correct response [3]. In this part of the thesis, we investigate the ability of a model to answer questions given a 3D environment represented as a 3D point cloud. The 3D point cloud provides 3D structure information, and allows us to focus on spatial relations and size.

We created a 3D VQA dataset using 3D reconstructions of real environments from ScanNet [13], along with set of models aimed to solve it. To support generation of questions based on spatial relations, we construct a 3D scene graph for each ScanNet scene. Using the 3D scene graph, we create a set of synthetic questions and answers). We use synthetically generated questions to control the complexity of the questions and to control the aspects of vision-language reasoning we study. This paradigm follows prior work such as CLEVR [31]

and GQA [29] which also programmatically generate question-answer pairs. Compared with CLEVR [31], which consists of a few simple shapes (cube, sphere, and cylinder), our dataset consists of various real-world objects over 500 categories. Our aim is similar to GQA [29] (to provide a VQA dataset for investigating reasoning and compositionality in real-world scenarios), but we focus on 3D indoor environments instead of 2D internet images.

5.1 Dataset



Q: Is there any white banner on the tv?; Qtype: Binary; A: No; FB: <select>:banner - <filter color>:white [0] - <relate>:on [1] - <select>:tv - <existence> [3]

Q : How many chairs are there ?; Qtype: Count; A: 4; FB: <select>:chair - <count> [0]

Q: what is the color of the tv ? Qtype: Query attribute; A: grey; FB: <select>:tv - <measure color> [0]

Q: where is the grey tv ?; Qtype: Location; A: above the cabinet; FB: <select>:tv - <filter color>:grey [0] - <get location> [1]

(a) Visual and language data for part of the sample

From \ To	TV	Table	Chair
TV	-	smaller, skinnier	skinnier, same height
Table	larger, wider	-	larger, same height, wider, next to
Chair	wider, same height	smaller, same height, skinnier, next to	-

(b) Partial scene graph of the sample

Figure 5.1: Partial sample from 3DVQA dataset with 3 objects.

The 3D scenes are represented as a point cloud with a set of values assigned to each point, including: object id, class id, RGB values, XYZ coordinates. Each scene was equipped with a scene graph. Similarly to 2D case, the 3D scene graph consists of objects at the nodes, and relationships between the objects. Scene graphs were used to generate question answer pairs.

For synthetic dataset the question of balancing is acute. Thus similarly to [29] we decided to introduce a measure of question difficulty. We define it as the number of times we need to look at the scene graph to determine the correct answer. Having a difficulty measure we can carefully weight all classes with respect to difficulty of questions. Having all control levers we can study the complexity of different question groups and the effect of complexity level on the performance of different models.

Similarly to other VQA datasets in order to widen the range of problems which can be addressed with the dataset we added: functional programs and ids of answer objects. This will allow using our dataset for such problems as semantic parsing and VQA but with other forms of output.

5.1.1 Scene graph generation

We built scene graphs using annotated objects from ScanNet and their oriented bounding boxes (OBB). We extracted four attributes for each object: color, lightness, height, and size (volume).

To obtain the color name for an object, we classify the color of each object's point using a lookup table with predefined RGB colors. To account for the variation of human visual system sensitivity to different parts of the visible spectrum and the common use of gamma-corrected color values in cameras, we follow [41] and measure the distance between two RGB colors as:

$$D(RGB, rgb) = \sqrt{((R-r)\times 0.3)^2 + ((G-g)\times 0.59)^2 + ((B-b)\times 0.11)^2}$$

Using this distance, we match against a list of predefined colors commonly used in indoor scenes. We take the majority vote of the points to obtain the final color name.

We compute object lightness by converting the RGB color for each point to HSL (hue, saturation, lightness) and taking the mean of lightness (L) for all points of the object.

For size attributes (height, volume), we use the axes length of the OBBs to estimate the height (z) and size (volume as the product of the axes lengths). Such volume calculation provides coarse estimation of object size.

Since all introduced attributes (excluding color) are gradable we base a subset of questions on comparison of the properties. For each of the attributes we induce three levels of intensity (below average, within average, and above average). For each class of objects and each characteristic we compute mean and standard deviation within certain object class.

For characteristic to be within average it has to be within standard deviation from mean value, everything outside from that interval will be lower and higher than average. Thus we have 3 levels for each of 4 attributes for each object category which has more than one object instance in it. With respect to designated level we choose the appropriate adjective (short vs tall, small vs large) for describing the attribute.

We divided the set of possible relationships into two parts: spatial and comparative ones. Since we are operating in a view-agnostic manner, we include only view-independent spatial relationships: `on`, `under`, `above`, `support`, `next to`, and `between`. Where `between` relation is based on whether the bounding box of one object is in between the bounding box of two other objects. For comparative relations, we use the attributes to determine if two objects are: `same color`, `lighter`, `darker`, `same category`, `same volume`, `larger`, `smaller`, `same height`, `taller`, `shorter`, `same width`, `skinnier`, and `wider`.

Based on the objects from point clouds and retrieved information we build a scene graph for each scene.

5.1.2 Question and answer generation

The dataset consists of four types of questions using templates: *counting*, *query attribute*, *location*, and *binary (yes/no)*. Question type correspond to the type of the answer. Using the templates, we can generate questions of varying complexity with different lexical surface forms. We generate questions ranging from difficulty level 1 to 5.

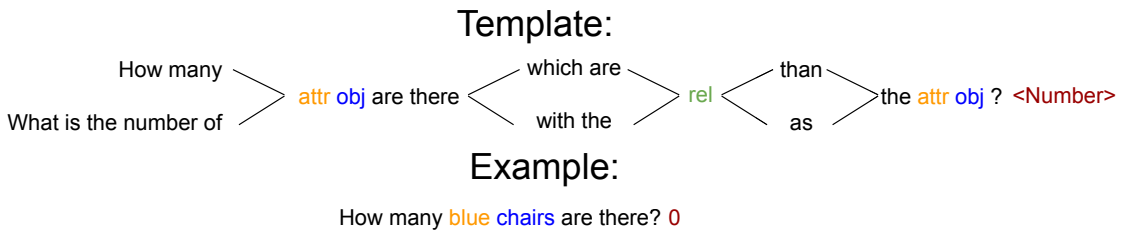


Figure 5.2: One question template with an example from 3DVQA dataset

Figure Figure 5.2 shows a template we use, as well as example question and answer pair.

Similarly as in the problem of 2D Visual Question Answering *binary (yes/no)* are the easiest questions which can be answered with only 'yes' and 'no'. Thus we decided to put fewer of them into final version of dataset. For them we generate questions that require checking the existence of a specified object, comparison of counts, and checking the relations. Answers to *location* questions are designed to be in the form of a short phrase, indicating the relative spatial location of the focus object relative to another object. This question category is the most challenging due to the biggest answer space. *Query attribute* questions ask about the color, height, or size of an object. *Counting* questions are designed to assess the ability

of a model to pick out and count the number of objects matching a set of attributes and relations.

By looking at the functional programs it can be noted, that overall sequence of actions which has to be performed to answer the question are quite similar. But as it will be covered further models fail to follow that chain.

To determine the answer, we traverse the scene graph to identify nodes in the graph matching the focus objects (category and attributes). If there are multiple objects that match the reference, we consider the question to be ambiguous and discard the question. We then consider the relationship between the matched nodes to determine the appropriate answer. For *query attribute*, *location*, and *check relation*, we will also ensure that the target object is non-ambiguous.

5.1.3 Dataset statistics and analysis

The dataset has more than 40M questions for 707 scenes, from which we sampled around 0.5M questions. A lot of questions address the same objects on the scene but have different answers (due to rich interior in the scenes) and a lot of binary questions are collected addressing a lot of similar attributes/objects. Thus we need to balance the dataset and our main target was to counterbalance all question categories. Since binary questions are easier, we aim for less binary questions than other question categories, resulting in a split of 16% binary questions and $27 - 30\%$ for the other question types.

To ensure that the generated questions and answers agree with human judgement, we sampled 153 questions from 4 scenes. Our generated answers matched at least one human response 84.31% of the time and matched both 66.01% of the time. Humans were able to answer *binary (yes/no)* and *counting* questions with high accuracy (91.89%). For *query attribute* questions, color was easier for humans than height/size questions. Answers to *location* questions had the most variation, and matched generated answers less frequently due to the free-form nature of the answer.

5.2 Approach

Along with the first 3D VQA dataset we provided a set of baselines and models. The problem of visual question answering in 3D space can be formulated as the following: given the question and a point cloud which represents the scene, the model has to give an answer to the question. Every point in the point cloud has six components: (x,y,z) position and (r,g,b) color values. In the original ScanNet reconstructions, the number of points for each scene can vary significantly, so we ollow prior work [55] and sample 40K for each scene. We start with a set of 'single data stream' baselines which only utilize either language, or visual data. and enrich it with unique pipelines we developed. We consider all answers which the model encountered during training as our answer space at other stages.

VoteNet

We use VoteNet [55], a 3D object detection network, as a backbone for a two of our models. Using VoteNet we obtain object-level features from a raw point cloud. This 3D object detection network takes the point cloud as an input and outputs object labels for each point. It operates over the 18 most common categories of the ScanNet dataset. It utilizes a unique voting algorithm, similar to Hough voting [40]. Firstly points are being passed into the PointNet++ [56] backbone which outputs M seed points equipped with feature vectors and (x,y,z) coordinates, each point has one vote. A shared voting module (which is a MLP) generates votes. Next, votes are clustered by simple sampling and are grouped (with radius r) by searching for the nearest one using Euclidean measure. Given a vote cluster $\mathbb{C} = \{w_i\}$ and its center w_j, where $w_i = [z_i, j_i]$ - concatenation of vote location and its feature, the vote location is being transformed into a central coordinate system with center at votes cluster: $z_i = (z_i - z_j)/r$. These transformed votes are then passed into PointNet-like module:

$$p(\mathbb{C}) = MLP\left(\max_{i=1...n}(MLP(z_i; h_i))\right)$$

The resulting vector is a proposal equipped with a bounding box, objectness score and semantic classification scores.

The original VoteNet mode color data, which is provided for each point. We modified the model to consider the RGB values of each point and at the inference time it will predict the RGB value of each point along with the object class. We compute the mean of RGB values for each object individually to obtain its unified color. We use this unified color represented as RGB values as ground truth data. Similar to the original model we pass coordinates with their colors into PointNet backbone and then into proposal module. This layer is the final destination of color features which are used to produce color proposals for each detected object.

5.2.1 Fused attention

In our fused attention model, we adapt the multimodal low-rank bilinear attention network (MLB) proposed by Kim et al. [35] to 3D VQA. The input is the final hidden state $h_n \in \mathbb{R}^{d_l}$ of the LSTM as the question encoding $\mathbf{q}$, and visual features $V \in \mathbb{R}^{K \times d_v}$ extracted by the 3D encoder (PointNet++ or VoteNet). For PointNet++ we use $K = N_2 = 1024$ and for VoteNet we use $K = N = 256$.

We then apply language-guided spatial attention on the 3D features V to obtain attention-weighted visual features. To compute the attention, we first project the visual V and language features h_n into a common space by passing each of them through fully connected layers f_v and f_q with $f_v(V) = \sigma(\mathbf{W}_v V^T)$, $f_l(\mathbf{q}) = \sigma(\mathbf{W}_l \mathbf{q})$ where $\mathbf{W}_v \in \mathbb{R}^{d \times d_v}$, $\mathbf{W}_v \in \mathbb{R}^{d \times d_l}$ and σ is a non-linear activation. We use $\sigma =$ ReLu as we experimentally found ReLu worked

better than tanh. We then compute attention over the K visual features v_i to obtain the attended visual features $v_{\text{att}} = \sum_{i=1}^{K} \alpha_i v_i$ with $\alpha = \text{softmax}(\mathbf{g})$ where $\mathbf{g}$ are the attention scores. The attention scores are computed by taking the low-rank bilinear approximation $\mathbf{g} = \mathbf{w}_a(f_v(V) \circ f_q(h_n \cdot \mathbb{1}^T))$ where $\circ$ is the Hadamard product (i.e. element-wise multiplication), $\mathbb{1} \in \mathbb{R}^K$ is a column vector of ones, and $\mathbf{w}_a \in \mathbb{R}^d$ are learned weights. Note that bias terms are omitted for simplicity. Finally, we concatenate the visual and language representations and feed them through our answer prediction module.

Answer prediction

We use a softmax classifier to predict the answers from the visual and textual representations. The answer module takes the concatenated visual and texual features and pass them through a simple classifier consisting of linear, ReLU, and Softmax layers. Our answer space consist of all answers encountered at the training stage, which means that the model potentially can not answer some questions. For our dataset the fraction of unanswerable questions is 8% and is mostly due to location questions. The metric that we use to measure error in this case is accuracy.

The input to this module is a concatenation of the encoded question and 3D features which is set as the initial hidden state of the decoder LSTM.

5.2.2 Baselines

Non-neural baselines

Firstly, we establish random and majority baselines. We consider four variations: fully random (Rand), random considering question type (Rand (Q-type)), full majority (Maj) and majority considering question type (Maj(Q-type)). Random baselines for all questions pick an absolutely random answer. Majority baseline chooses as an answer to all questions the answer that occur mostly frequently given across the whole dataset. Random considering question type picks an arbitrary answer only from the answers which occur for a given question type. A similar scheme is used for Maj(Q-type) with one answer chosen for questions with the same question type based on the most frequent answer for that question type.

Neural Baselines

In order to estimate competence of backbone modules and overall contents of the dataset, we experimented with language-only and vision-only models. In every model, we use LSTM in order to get features from language data.

Language models. We consider language-only baselines with a sequence model based on an LSTM.

In that case we use LSTM with a combination of GloVE [53] embeddings of size 300. We pad each question to the length of the longest question in the training set. In single-layer LSTM, we have a hidden state of size $d_l = 300$. To improve the language encoding, we apply one hop of self-attention [45] to get the attended representation Q. For a question of length n, let $H = (\mathbf{h}_1, \ldots, \mathbf{h}_n) \in \mathbb{R}^{n \times d_l}$ be the hidden states of the encoder stacked into a matrix H. We compute the attention weights as $A = \text{softmax}(W_a \tanh(W_h H^T))$, where $W_h \in \mathbb{R}^{d_a \times d_l}$ and $W_a \in \mathbb{R}^{r \times d_a}$ are learnable weights, and this softmax is performed across the second dimension. The final self-attended question representation is then $Q = AH$. We use one-hop self-attention (with $r = 1$) and set $d_a = d_l$.

Vision models. For vision-only baselines we use VoteNet.

2DVQA. We also consider a 2D VQA baseline where instead of using the point clouds, we use a 2D top-down rendering of the scene as input. We use ResNet-18 on the 2D top-down rendering to get the visual features. The concatenation of this vector and the language representation are passed through a classifier to predict the answer. Since we use the bird-eye-view (BEV) as input, we refer to the model as **LSTM+BEV**. The vision-only version of this model is referred to as **BEV** and consist of ResNet-18 in combination with standard classifier.

5.2.3 Implementation details

All our models are implemented in PyTorch. We use the official VoteNet implementation. We train our models on a workstation with a Core i9-9900K CPU and RTX 2080 Ti GPU. The stopping criterion was a change of less than 0.0001 in validation set accuracy between epochs. We used dropout on the last network layers. For the 3D fused model, we used ADAM [37] with an initial learning rate of 0.001, decaying the learning rate by half every two epochs, and a dropout rate of 0.7. We trained the network for up to 30 epochs, with the training stopping when the validation accuracy stabilizes (increase less than 0.0001), typically after 10 epochs. For the NSM, we trained using ADAM with a learning rate of 0.0003, decaying learning rate by half every epoch. We used a last layer dropout rate of 0.15 and trained up to 10 epochs.

5.3 Experiments

In our experiments, we follow the ScanNet v1 scene split, ensuring consistenccy with other papers. It contains 494 scenes for train, 71 for validation, and 142 scenes for the test. Table 5.1 compares the performance of different models using the accuracy, validity and distribution metrics. The random baselines show that it is challenging to randomly guess the answer. The majority baselines show that there is some bias in the dataset (e.g., most binary questions are answered by 'no'). Similarly the bias is reflected in the strong performance of the language-only baseline (LSTM). In contrast, the low performance of the 3D point-cloud

only baselines using VoteNet, shows that since there are many questions for each scene, the text of the question is necessary for determining the answer. The other cause for such low performance is due to the in noise in ScanNet scenes. The ScanNet scenes were initially reconstructed from scans of indoor environments. Due to the quality of the reconstructions, the scenes are noisy and incomplete. Models that combine information from the question and the visual modality, improve the performance only slightly over the LSTM only model. Incorporating VoteNet on top of the LSTM improves the performance by 0.9% while using the top-down view improves the performance by 0.5%.

There is no conceptual difference between 2D and 3D case for NSM as it uses scene graphs as input. One key difference between the two settings is that the scene-graphs in 3DVQA is more complete with many relations between objects, while the scene-graphs in GQA has only a partial set of relations between objects that were hard annotated.

	Val			Test		
Method	Acc↑	Val↑	Dist↓	Acc↑	Val↑	Dist↓
Rand	0.03	3.56	133.58	0.05	5.27	267.88
Rand(Q-type)	11.08	44.58	2141.49	9.92	46.78	5564.31
Maj	10.11	28.77	1094.32	9.13	26.79	1430.89
Maj(Q-type)	27.44	87.72	2082.14	25.04	86.44	2113.33
LSTM	42.02	98.91	1874.36	41.44	98.72	1652.45
BEV(2D)	9.21	28.88	163.33	8.72	28.84	314.92
VoteNet	11.47	30.79	174.44	9.62	29.59	381.47
LSTM + BEV(2D)	42.52	96.94	1324.47	40.75	97.56	609.86
LSTM + VN	**42.98**	**98.99**	542.28	**43.07**	**98.76**	608.99
NSM on GT SG	41.32	95.68	252.32	41.44	96.91	234.4
NSM narrowed on GT SG	41.72	95.93	303.81	41.87	96.36	289.46
NSM on predicted SG	37.12	96.04	275.48	37.24	95.79	312.60
NSM narrowed on predicted SG	39.06	95.81	243.30	39.15	95.42	286.95

Table 5.1: Performance of different models on 3DVQA-ScanNet. Accuracy, validity, distribution trends are most correlated across models.

We conduct experiments using the NSM with both ground truth and predicted scene-graphs. We construct the predicted scene-graphs using VoteNet trained on ScanNet v1 train split with ScanNetv2 annotations. Because VoteNet is typically trained and evaluated on the 18 most frequent classes of ScanNet, we subsample our dataset to restrict it to question-answer pairs mentioning only objects from these 18 classes.

To check the quality of the generated scene graphs we measure the performance of VoteNet predictor that was used to detect objects in the scene. Note that we adapted our VoteNet model to not only predict the object category, but also the object color by taking the mean of the RGB values of points belonging to the predicted object. To measure

Method	mAP@0.25	AR @0.25	mAP@0.5	AR @0.5	Color error
VoteNet re-implemented	0.50	0.62	0.31	0.42	0.019
VoteNet original [55]	0.52	0.65	0.33	0.45	N/A

Table 5.2: VoteNet performance on test split with color error. Error is averaged among R, G, B channels.

	Val							Test						
Method	Acc	loc	count	query	y/n	clr	h/s	Acc	loc	count	query	y/n	clr	height/size
LSTM	37.71	6.29	46.20	39.43	78.69	33.59	28.38/36.27	38.25	6.47	47.72	39.43	**74.49**	40.58	**38.54**/37.89
LSTM + BEV(2D)	38.34	6.71	47.69	35.45	75.25	35.93	32.24/37.70	36.53	4.90	45.26	37.86	73.55	43.10	29.62/34.63
LSTM + VN	39.12	10.82	46.04	36.70	74.46	33.59	39.38/41.19	38.88	**9.45**	48.68	39.18	72.02	40.58	34.26/**40.75**
NSM on GT SG	36.71	6.48	41.01	36.09	76.90	31.26	39.19/**44.26**	35.51	6.89	41.68	36.60	72.31	35.26	36.66/39.24
NSM narrowed on GT SG	42.25	9.52	49.89	42.11	**79.62**	35.43	**45.32**/38.67	40.13	9.07	45.33	39.56	70.40	46.40	38.22/40.15
NSM on predicted SG	34.13	5.97	40.68	36.12	76.11	30.79	38.41/43.06	33.50	6.71	42.01	36.25	71.64	34.29	36.02/38.57
NSM narrowed on predicted SG	**42.87**	**14.06**	**50.26**	44.01	75.68	**50.57**	41.15/36.85	**40.76**	8.96	**49.90**	**40.29**	73.22	**54.21**	**40.53**/39.08

Table 5.3: Performance of baselines and NSM models on 18 object category 3DVQA dataset.

the quality of color prediction we measure the average RGB color error of the predicted color from the ground truth color. The error is computed as an absolute value of difference between prediction and ground truth. Table 5.2 shows the performance of our re-trained VoteNet. Our retrained model has a slightly lower performance compared to the results reported in the original VoteNet paper.

In Table 5.3, we report the performance of the NSM models on the 18-object subset of question-answer pairs. Considering the fact that in 3DVQA the scene graphs are more complex with more spatial relations, we would expect bigger improvements. From Table 5.3 we see that that of narrowing works in all NSM applications and potentially can improve attention mechanisms which involve matrix as a learnable parameter.

In Table 5.3 we also split out height/size questions to a separate category to highlight significant overlap of our 3D models compared to LSTM and 2DVQA baselines. We hypothesize that height/size questions cannot be answered correctly without modeling the 3D extent of objects in the 3D scene. And from the results it can be observed that specifically designed 3D scene-aware models perform notably better (>10%) than the others for these questions.

Chapter 6

Conclusion

To conclude, in this thesis we studied the performance of the NSM model for VQA. We re-implemented the NSM and considered two extensions (attention localization and the use of BAN) on 2D VQA on the GQA dataset. We found that while BAN did not improve performance, the attention localization helped to improve performance. We also presented our work on 3D VQA where we constructed a synthetically generated set of questions and answers for ScanNet. Using this dataset, we conducted experiments and compared the performance of NSM against baselines and more standard VQA models. Again, we showed that the attention localization helped improve the performance over the basic NSM.

One key limitation of the NSM is its dependency on an external module to extract a probabilistic scene-graph from the image. Since a only a few datasets have scene-graphs, it makes it challenging to investigate the performance of NSM with ground-truth scene-graphs. The simulation of the NSM during the inference stage is also computationally and memory intensive. After adding our extensions, the inference time doubled. This make it impossible for the model to be used in real-time applications or mobile devices. While it is possible to use multiple GPUs or GPUs with more memory, the computational inefficiency of the NSM makes it less appealing for real-world use.

Our 3DVQA work is limited by its construction methodology. It is challenging to create a fully balanced subset considering all objects and all distributions. With real world scenes there will always exist natural biases: for instance, the majority of tables are brown and almost every pillow is on the bed or sofa. While we can attempt to correct for these biases, it is impossible to completely eliminate them nor is it entirely desirable. While it is important to have a balanced dataset for the purposes of investigating models that can perform compositional reasoning, it is also important to study whether models capture the common sense priors of the world. We believe there is a lot of potential future work in the area of 3DVQA.

In the short term, for 3DVQA we plan to add oriented bounding boxes. This would allow us to induce new relative spatial relationships such at `in front`, `to the left`, and `to the right`. Along with that would add choice and logical questions. This will increase

the diversity of questions we have. In the future, we hope the 3DVQA would become a basis for work by other researcher exploring VQA systems in 3D environments.

Bibliography

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Austin, Texas, November 2016. Association for Computational Linguistics.

[2] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980, 2018.

[3] Ankesh Anand, Eugene Belilovsky, Kyle Kastner, Hugo Larochelle, and Aaron Courville. Blindfold baselines for embodied qa. *arXiv preprint arXiv:1811.05013*, 2018.

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[5] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. corr abs/1511.02799 (2015). In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1181. URL `https://aclanthology.org/N16-1181`.

[7] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

[8] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[9] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[10] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.

[11] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL `https://aclanthology.org/W14-4012`.

[12] Shih-Han Chou, Wei-Lun Chao, Wei-Sheng Lai, Min Sun, and Ming-Hsuan Yang. Visual question answering on 360deg images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1607–1616, 2020.

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[14] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2054–2063, 2018.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[16] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[17] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2016.

[18] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and W. Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, 2015.

[19] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 317–326, 2016.

[20] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. Visual Turing test for computer vision systems. volume 112, pages 3618–3623. Proceedings of the National Academy of Sciences, 2015.

[21] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual question answering in interactive environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098, 2018.

[22] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[25] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

[26] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

[27] D. A. Hudson and Christopher D. Manning. Learning by abstraction: The neural state machine. In *NeurIPS*, 2019.

[28] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

[29] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

[30] Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer, 2016.

[31] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[32] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017.

[33] Kushal Kafle and Christopher Kanan. Answer-type prediction for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4976–4984, 2016.

[34] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017.

[35] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. In *International Conference on Learning Representations*, 2017.

[36] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear Attention Networks. In *Advances in Neural Information Processing Systems 31*, pages 1571–1581, 2018.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[38] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075150. URL `https://aclanthology.org/P03-1054`.

[39] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Bernstein Michael S., and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. volume 123, pages 32–73. Springer, 2017.

[40] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. 2(5):7, 2004.

[41] Ze-Nian Li, Mark S Drew, and Jiangchuan Liu. *Fundamentals of multimedia*. Springer, 2004.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[43] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[44] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[45] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=BJC_jUqxe`.

[46] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems*, 29:289–297, 2016.

[47] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

[48] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, pages 1–9, 2015.

[49] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[50] Jiayuan Mao, Chuang Gan, P. Kohli, J. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes words and sentences from natural supervision. In *International Conference on Learning Representations (ICLR)*, 2019.

[51] Marie-Catherine Marneffe, Bill MacCartney, and Christopher Manning. Generating typed dependency parses from phrase structure parses. volume 6, 01 2006.

[52] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL `http://arxiv.org/abs/1301.3781`.

[53] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[54] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[55] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[56] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[57] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. *Advances in neural information processing systems*, 28:2953–2961, 2015.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[59] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473, 2019.

[60] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[61] Kaihua Tang. A scene graph generation codebase in pytorch, 2020. `https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch`.

[62] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3716–3725, 2020.

[63] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`. accepted as poster.

[64] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. page 133–138, 1994.

[65] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[66] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016.

[67] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1039–1050, 2018.

[68] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5831–5840, 2018.

[69] P. Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5014–5022, 2016.

[70] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004, 2016.

[71] C Lawrence Zitnick and Devi Parikh. Bringing semantics into focus using visual abstraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3009–3016, 2013.