Graph Attention Networks for Compositional Visual Question Answering

Alexander Mirrington

This thesis is presented as part of the requirements for the conferral of the degree:

Bachelor of Information Technology (Honours)

Supervisors:
Dr. Caren Han
Dr. Josiah Poon

The University of Sydney
School of Computer Science

August 15, 2021

Declaration

I, *Alexander Mirrington*, declare that this thesis is submitted in partial fulfilment of the requirements for the conferral of the degree *Bachelor of Information Technology (Honours)*, from the University of Sydney, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Alexander Mirrington

August 15, 2021

Abstract

In recent years, the task of visual question answering (VQA) has seen rapid growth as deep learning has seen more and more use in the computer vision (CV) and natural language processing (NLP) fields. The VQA task has two primary inputs: a visual representation, such as an image, and a question grounded in that representation. The large majority of VQA research to-date has focused on VQA as an end-to-end problem, aiming to answer questions about the contents of an image. More recently, the VQA community has seen a shift towards graph-based methods, whereby an image is represented as a collection of objects, attributes and relationships, called a *scene graph*. Instead of learning to answer questions about the image directly, recently proposed VQA models first convert the image to a scene graph, and then formulate an answer to the question using the scene graph as a visual representation. The primary issue with these methods is that they often use proprietary methods for generating scene graphs from images, making it extremely difficult to gauge what proportion of the VQA model's performance stems from the quality of the underlying scene graph generation process versus the reasoning capabilities of the question-answering model itself. In this dissertation, I focus solely on the latter portion of the VQA pipeline, using scene graphs from the GQA dataset [1] as a visual input to my VQA model instead of images. The backbone of my VQA model is the Graph Attention Network (GAT) [2], a highly capable graph representation learning model that learns contextual node embeddings from graph-structured data. Since GATs only operate on the nodes of a graph but scene graphs embed relationships between image objects as edges, I propose a novel method of representing scene graph edges as nodes that increases the expressivity of the learned node embeddings. Moreover, I prove that this graph transformation method is in the order of only $\min\{\rho, F\}$ times slower than performing GAT convolutions on scene graphs that don't embed relationships as nodes, where ρ is the ratio between the number of scene graph objects and the number of scene graph relationships, and F is the input dimension to GAT. For the GQA dataset, we have that $\rho \approx 3$, making this graph embedding method both efficient and expressive. I leverage the rich semantic embeddings learned by GAT for VQA by replacing the traditional visual knowledge-base of the Compositional Attention Network [3], a widely-accepted baseline for VQA performance. The resultant model is highly effective, outperforming other scene-graph oriented VQA baselines by over 9%, state-of-the-art image-based VQA models by over 14%, and achieving a higher top-1 accuracy than humans on the GQA dataset. I provide the full source code for my primary model architecture and various ablations online at `https://github.com/alexmirrington/gat-vqa`.

Acknowledgements

To my supervisors, Dr. Caren Han and Dr. Josiah Poon,

Thank you for the opportunity to work alongside you in the University of Sydney Natural Language Processing (NLP) research group. To Caren, I thank you for always making yourself available to answer my questions, reviewing my drafts and motivating me to keep developing new ideas. To Josiah, thank you for all your efforts in organising weekly research group meetings and providing wonderful opportunities to collaborate with other researchers despite the challenges of moving online this year.

To my parents,

Thank you for your endless support over the past four years, for providing me with the opportunity to study full-time and for helping me to keep pushing forward though the many challenges I faced this year.

To my girlfriend, Ayuka,

Thank you for standing by my side through the good and the bad, supporting me regardless of how my research was progressing throughout the year. Thank you for being the highlight of my day each and every day this year. I could not ask for a better friend and companion, and I am inexpressibly grateful to have you as a part of my life.

To my sisters,

Thank you for always being generally cheerful and wonderful people to be around. It's been a pleasure to see you both approach the challenges of 2020 with such enthusiasm and success.

To my friends, Sammy, Matt, Jessie and Ben,

Thank you for making the effort to stay in touch over what has been a difficult year. Though we haven't been able to meet in person for a while, I'm extremely grateful for our sporadic conversations over text and/or occasional video chats over a cup of coffee.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

To keep nomenclature consistent with that of the original publications, I do not adopt this nomenclature in Chapter 2, and instead introduce each symbol on a case-by-case basis. Consequently, the list of symbols below is used from Chapter 3 onwards.

Model Architecture

$H.$ An embedding matrix for some entity $\cdot$, *e.g.* a visual representation r, a question q or batches thereof.

$K.$ A knowledge-base for some entity $\cdot$, *e.g.* a visual representation r, a question q or batches thereof.

$d.$ A scalar used to denote the feature dimension of a vector, matrix or tensor $\cdot$.

W A trainable weight tensor.

b A trainable additive bias tensor.

Data

$S = \{((q_i, r_i), a_i)\}_{i=1}^{n}$ A visual question answering (VQA) dataset of n samples, each comprising a question q, a visual representation r in which the question is grounded, and an answer a.

q A visually-grounded question.

r A visual representation in which a question is grounded. Since different VQA datasets contain one or more visual representations for each dataset sample (*e.g.* an image, a scene graph, a set of spatial or object-based features, or a combination of these), I specify the kind of data r represents for a given context.

a A ground-truth answer to a visually-grounded question

$\hat{a}$ A predicted answer to a visually-grounded question.

$Q = \{q \mid ((q, s), y) \in S\}$ The collection of all questions in a dataset S.

$Q_{[i:i+k]} = \{q_j \mid ((q_j, s_j), y_j) \in S\}_{j=i}^{\min(i+k,n)}$ A batch of questions of size $\min(k, n-i)$ from a dataset S of size n, starting at index i.

$R = \{r \mid ((q, r), y) \in S\}$ The collection of all visual representations in a dataset S.

$R_{[i:i+k]} = \{r_j \mid ((q_j, r_j), y_j) \in S\}_{j=i}^{\min(i+k,n)}$ A batch of visual representations of size $\min(k, n-i)$ from a dataset S of size n, starting at index i. For readability, I refer to the size of the batch dimension of tensors as k even though smaller batches of size $\min(k, n-i)$ may occur.

Graphs

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ A directed graph.

$\mathcal{V} = \{v_i \mid i \in \mathbb{Z}^+\}$ A set of vertices of a directed graph

$\mathcal{E} = \{(v_i, v_j) \mid i \in \mathbb{Z}^+, j \in \mathbb{Z}^+\}$ A set of edges of a directed graph.

v A vertex of a graph $\mathcal{G}$.

$\tilde{A} = A + I_{|\mathcal{V}|}$ An adjacency matrix with added self-loops.

$A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ An adjacency matrix given a graph $\mathcal{G}$ with vertices $\mathcal{V}$.

$D \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ A degree matrix with diagonal elements $D_{ii} = \sum_j A_{ij}$

General

I_n An identity matrix of size $n \times n$.

$x_1 \parallel x_2$ The concatenation of two vectors a and b.

$[x_1, ..., x_p]$ The concatenation of d tensors of the same size along a new first dimension. *e.g.* If $x_i \in \mathbb{R}^q$, then $[x_1, ..., x_p] \in \mathbb{R}^{p \times q}$

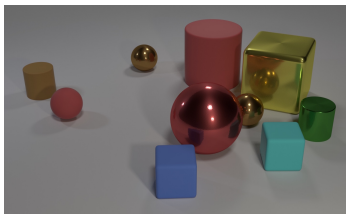
Chapter 1

Introduction

We live in an exciting technological era. In the last decade, we have witnessed the emergence of smart-home devices that harness the power of complex natural language understanding, text-to-speech and speech-to-text models to assist millions of people every day. The collaboration of hardware and software engineers has fueled the rapid evolution of the deep learning field, leading us into a world where semi-autonomous vehicles are becoming commonplace and face detection models are being used to unlock mobile devices. Given the wide adoption of deep learning (DL) techniques in industry for natural language processing (NLP) and computer vision (CV) tasks, we have seen a recent shift in the research community towards audio-visual and visio-linguistic tasks that require the learning of complex interactions between multiple input data modes. As a research community, we are excited by the technical challenge that these new tasks present, in addition to the extensive practical applications of multimodal reasoning tasks; for example, well-designed image captioning and visual question answering models could aid the visually impaired in consuming and understanding visual information in a natural manner when combined with existing speech-to-text and text-to-speech systems. Other multimodal tasks like text-to-image generation could be leveraged by law enforcement teams to build realistic composites of people or places relevant to an investigation, or by creatives as an external source of visual inspiration.

In this dissertation, I focus solely on visual question answering (VQA). More specifically, I delve into how we can leverage graphical representations of both visual and textual data to aid reasoning models in their decision-making processes. At its core, the VQA problem takes two inputs, an image and a question pertaining to one or more objects, relationships and/or concepts presented in the image. Given these inputs, we aim to answer the question, as illustrated in Figure 1.1. As humans, we implicitly solve the VQA problem every day when making decisions grounded on visual inputs. When crossing the road, we ask the implicit question *'Is it safe to cross the road?'*, formulate an answer based on visual signals, and then act upon our decision. The holy grail of VQA is to be able to perform such decision-making for all feasible combinations of visual inputs and questions. Naturally, the true distribution of these input combinations is unknown, motivating the first major hurdle

(a) Is there something to cut the vegetables with? *no*



(b) How many objects are either small cylinders or red things? *5*

(c) What kind of furniture is to the right of the chair? *sofa*

Figure 1.1: Example image, question and answer triples from the VQAv2 [4], CLEVR [5] and GQA [1] datasets, figures 1.1a, 1.1b and 1.1c respectively.

in VQA research: *How do we design a dataset that effectively captures subtle relationships between questions and images as presented in real-world situations where VQA would prove useful?*

As evident in Figure 1.1, this question is still open to interpretation; the VQAv2 dataset contains real-world images and free-form questions often requiring conceptual reasoning, whilst CLEVR leverages generated images alongside questions requiring more compositional reasoning, targeting logical reasoning operations like counting and boolean arithmetic. The GQA dataset combines the approaches of VQAv2 and CLEVR datasets by combining the complexity of real-world images with the difficulty of questions requiring multiple discrete reasoning steps. I explore the methods used to create these three datasets in more detail in Section 2.1.

Assuming we do have an ideal dataset that effectively models real-world VQA problems, we need some way of combining both visual and textual data in a way that enables a model to perform the complex reasoning required for answer formulation.

A wide variety of VQA models have been proposed for this task over the last few years, with each approaching the task from a different angle. Early VQA methods relied on convolutional neural network (CNN) features such as those obtained from ResNet [6] or object-based features such as those obtained from Faster R-CNN [7]. In addition, they leveraged recurrent models such as Long Short Term Memory (LSTM) [8] networks or Gated Recurrent Units (GRU) [9] as question embeddings. Many researchers proposed different ways to fuse these image and question features, with the three main categories being attention-based methods, fusion-based methods and graph-based methods.

Attention-based methods started out as simple pointwise multiplications [10], or question-guided image attention mechanisms [11], before progressing to self-attention [3], transformer-style attention [12], co-attention [13], [14] and bilinear attention [15] mechanisms.

Where attention mechanisms like co-attention and bilinear attention aim to high-

light which parts of the image are important to the question and vice-versa, fusion methods aim to project the question and image domains onto a single semantic space. Simple fusion methods such as concatenation appear everywhere in the VQA field, however dedicated fusion methods like MUTAN [16] and BLOCK [17] aim to approximate more effective but expensive bilinear fusion methods by using various tensor decompositions.

With the inclusion of scene graphs in various VQA datasets, VQA research has more recently shifted more towards graphical methods, which leverage Graph Convolutional Networks (GCN) [18] or Graph Attention Networks (GAT) [2] for capturing contextual relationships between image object, attribute and relationship information. Methods like Relation Aware Graph Attention networks (ReGAT) [19] and Dual-Channel GCN (DC-GCN) [20] have proven effective on popular datasets like VQAv2 [4] and VQA-CP [21]. With an increase in graph-based VQA models has come an increase in scene-graph generation methods, which focus on generating scene graphs from images. However, as mentioned briefly in the abstract, many graph-based VQA models use arbitrary scene graph generation methods in addition to experimenting with new question answering architectures, making it difficult to determine whether performance increases are due to improved scene graph generation methods or improved visual reasoning models.

For this reason, I focus solely on the the visual question answering task in this dissertation, in an effort to demonstrate the effectiveness of graph-based methods like GATs for visual reasoning. To build my VQA model, I rely heavily on datasets like GQA [3], which come with pre-annotated scene graphs, and draw inspiration from existing attention-based and graph-based architectures. Consequently, I devote a large portion of Chapter 2 to exploring various VQA datasets and prevalent graph-based VQA methods to better understand why scene graphs in combination with graph-based models perform so well for VQA tasks.

1.1 Contributions

My primary contributions to the VQA field are three-fold:

1. I propose a scene graph transformation method that allows object-object relationships to be embedded as nodes alongside object and attribute embeddings. This allows node-based graph convolutions to operate on the modified scene graph, capturing contextual object, attribute and relationship information. Moreover, I provide a theoretical upper bound on the time complexity of applying GAT convolutions on the modified scene graph; given an original scene graph r containing r_o objects, r_a attributes and r_r relations, we define $\rho = \frac{r_r}{r_o}$, and F to be the input dimension of a given GAT layer. Given that my graph transformation method yields a new graph $\mathcal{G}_r$, I prove that performing a GAT convolution on $\mathcal{G}_r$ is in the order of $\min\{\rho, F\}$ times slower than performing a GAT convolution on r. For the GQA, $\rho \approx 3$, meaning that performing GAT convolutions on $\mathcal{G}_r$ is both feasible and effective, as it allows traditional node-based convolutions to operate on what would usually be an edge embedding.

2. I present a new, simple and effective VQA model that combines my novel graph transformation method, a graph attention network, a question processing module such as a BiLSTM, and a compositional attention network. When evaluated on the GQA dataset, it outperforms existing scene-graph oriented models on the top-1 accuracy measure by over 9%, as well as current state-of-the-art image-based ensemble methods[1] by over 14%, and outperforms even human accuracy by 1%, binary question type accuracy by almost 0.4%, and open-type accuracy by 1.7%.
3. In the spirit of reproducibility, I provide the full source code for my primary model architecture and various ablations online at `https://github.com/alexmirrington/gat-vqa`.

1.2 Thesis Structure

In the introduction to this dissertation, I provide an overview of the VQA problem and its uses, before introducing a variety of VQA datasets and methods to provide the reader with an understanding of the breadth of the field.

In Chapter 2, I delve further into the design decisions behind various VQA datasets, focusing on how these datasets are created at such a large scale whilst still remaining relatively unbiased in their question and answer distributions, as well as how different authors approach the concept of scene graphs for VQA. Next, I expand on the VQA model architectures mentioned in the discussion that inspire or form part of my model architecture, before finalising the literature review with a summary of various evaluation metrics for different VQA tasks.

In Chapter 3, I introduce my model architecture, describing its four main components: the question embedding and processing module, the scene graph embedding and processing module, the reasoning module, and the output module. I also analyse the complexity of my scene graph embedding process in this chapter, and summarise the primary motivations behind the inclusion of each model component.

In Chapter 4, I introduce the GQA dataset in more detail, before providing a brief overview of the baseline VQA methods that I compare my model's performance against. Finally, I provide a detailed list of my model's parameter configuration, including its dataset vocabulary sizes, training schedule and optimisation method.

In Chapter 5, I report and comment on the performance of my model compared to baseline models for a variety of metrics. Moreover, I provide details about the ablation studies I performed, presenting empirical evidence to support my model architecture design decisions. Finally, I provide a brief overview of the parameter optimisation process I used to obtain the set of final model parameters introduced in Chapter 4.

In Chapter 6, I analyse how my model arrives at an answer given a scene graph and

[1] According to the GQA leaderboard [22]

a question, delving into the attention maps used to support the reasoning processes of the scene graph processing module and reasoning modules.

Finally, in Chapter 7, I provide closing remarks on the primary observations of each chapter in this dissertation, and briefly explore potential directions for future research.

Chapter 2

Literature Review

2.1 Visual Question Answering Datasets

In this section, I address the first major hurdle for the Visual Question Answering(VQA) task as introduced in Chapter 1: *How do we design a dataset that effectively captures subtle relationships between questions and images as presented in real-world situations where VQA would prove useful?* Effectively answering this question requires answering multiple related sub-questions:

1. What kind of images represent the situations in which we would use VQA models, and how do we gather these types of images?
2. What kind of questions would humans ask about these images and how can they be collected?
3. How do we ensure that the answer distribution for the dataset reflects the types of answers we see to questions in the real world whilst not over- or under-representing certain answer types?

I address the first two of these questions in Subsection 2.1.1, outlining the wide variety of approaches to image and question collection and/or generation employed by a variety of datasets. In Subsection 2.1.2, I emphasise the importance of the final question, describing strategies used by dataset creators to ensure question and answer distributions accurately represent our observations of the real world, and explain how statistical imbalances in question and answer distributions can lead to an overly-optimistic evaluations of VQA model performance.

2.1.1 Collecting Questions and Images for VQA Datasets

Naturally, the first decision to make when designing a VQA dataset is to determine the type of questions and images to use and how to gather them. As illustrated in Figure 1.1 in the Introduction, different datasets take different approaches to image and question collection, depending on the type of visual reasoning skills they wish to test. For example, the VQAv1 [10] and VQAv2 [4] datasets contain photographs of

many real-world scenarios in which VQA may prove useful for helping the visually-impaired, including busy urban areas and rooms within people's homes. Other datasets adopt a different approach, using images of scenes assembled from 2D clip art objects [10], [23], shapes [24], or assorted 3D solids with various colours [5]. These datasets provide a standard benchmark for assessing the reasoning capabilities of different VQA models, and are thus not viable training signals for real-world VQA applications. Similarly, many datasets collect questions for each image via crowdsourcing platforms like Amazon Mechanical Turk (AMT) in order to simulate the types of questions that humans wish to ask in the real world. Others like CLEVR [5] and GQA [1] generate questions using a combination of templates and scene graph information, intentionally avoiding the inherent ambiguity of questions created by humans and instead focusing on how well VQA models are able to interpret and understand complex questions syntax.

I summarise the most common VQA datasets along with their image sources, question creation methods and other features in Table 2.1, before delving into the question and image collection methods and motivations behind the most influential VQA datasets.

Dataset	Year	Images	Question-Image Pairs	Image Source	Question Creation	Answer Type	Additional Data	Evaluation Metrics
DAQUAR [25]	2014	1,449	12,468	NYU-Depth V2 [26]	Both	Multi-label (894)	Human segmentation, image depth values, and object labels from NYU Depth V2.	Accuracy, WUPS
Visual Madlibs [27]	2015	10,738	360,001	COCO	Human	Fill-in-the-blank, Multi-choice (4)	COCO annotations	Accuracy, BLEU
COCO-QA[28]	2015	123,287	117,684	COCO	Automatic	Single-concept	COCO annotations	Accuracy, BLEU
VQAv1 [10]	2015	204K	614K	COCO	Human	Single-concept, Multi-choice (18)	COCO image captions	Accuracy, Consensus Accuracy
Abstract Scenes	2015	50K	150K	2D clip art	Human	Single-concept, Multi-choice (18)	Image captions	Accuracy, Consensus Accuracy
Changing Priors (CP) [21]	2018	≈204K	≈370K	COCO	Human	Single-concept	See VQAv1	Consensus Accuracy
Compositional VQA (C-VQA) [29]	2017	204K	369K	COCO	Human	Single-concept	See VQAv1	Consensus Accuracy
VQAv2 [4]	2017	204K	1.1M	COCO	Human	Single-concept	COCO image captions, complementary image pairs	Accuracy, Consensus Accuracy
Balanced Binary Abstract Scenes [30]	2016	31K	33K	2D clip art	Human	Multi-choice (2)	Image captions	Accuracy
Changing Priors (CP) [21]	2018	≈204K	≈658K	COCO	Human	Single-concept	See VQAv2	Consensus Accuracy

Dataset	Year	Images	Question-Image Pairs	Image Source	Question Creation	Answer Type	Additional Data	Evaluation Metrics
Visual Genome [31]	2016	108K	1.7M	COCO, YFCC100M [32]	Human	Single-concept	COCO annotations, region descriptions, scene graphs	Accuracy
Visual7W [33]	2016	47K	327K	COCO	Human	Single-concept, Multi-choice (4)	Groundings for each object in each question-answer pair.	Accuracy
TDIUC [34]	2017	167K	1.6M	COCO, Visual Genome	Both	Single-concept	See Visual Genome.	Per-question-type accuracy, regular & normalised arithmetic & harmonic mean accuracy
CLEVR [5]	2017	100K	999K	3D renderings	Automatic	Single-concept	Functional programs, scene graphs	Accuracy
CoGenT-A & B	2017	100K	999K	3D renderings	Automatic	Single-concept	Functional programs, scene graphs	Accuracy
Humans	2017	32K	32K	CLEVR	Human	Single-concept	See CLEVR.	Accuracy

Dataset	Year	Images	Question-Image Pairs	Image Source	Question Creation	Answer Type	Additional Data	Evaluation Metrics
GQA [1]	2019	113K	22.6M	COCO, Visual Genome	Both	Single-concept, open-ended	Scene graphs, functional programs, full-sentence answers, valid and plausible answer sets for each question, entailed questions, faster R-CNN [7] features, ResNet-101 [6] features, bounding boxes.	Accuracy, Consistency, Validity, Plausibility, Distribution, Grounding

Table 2.1: A comparison of relevant features of the most popular VQA datasets. Dataset variations are listed in regular font below their bolded counterparts. For clarity, I make the distinction between single-concept and open-ended VQA tasks: what many datasets call an *open-ended* VQA task is a task requiring a short 1-3 word (*single-concept*) answer. Most often, these types of tasks are approached as a classification problem across the entire answer vocabulary of the dataset. I reserve the *open-ended* answer type for VQA tasks that require full-sentence answers.

VQAv1

Given the rise in popularity of VQA tasks inspired by DAQUAR, Antol, Agrawal, Lu, *et al.* recognised a need for a new, sufficiently large dataset that contained free-form, open-ended questions that still allowed results to be quantified; this need was realised by two datasets: an open-ended dataset derived from COCO [35], as well as a smaller synthetic dataset containing abstract scenes assembled from 2D clip-art objects [23]. These datasets are referred to as VQAv1 and Abstract Scenes (listed underneath VQAv1) in Table 2.1 respectively. The key contributions of these datasets are two-fold:

1. The real-world image dataset provided the first feasible dataset for benchmarking VQA models, thanks to its large size compared to existing VQA datasets like DAQUAR and COCO-QA. Whilst smaller, the synthetic abstract scenes dataset allowed researchers to investigate the underlying theory behind visual reasoning without focusing on the difficult task of extracting features from images.

2. Both datasets employed a new consensus accuracy metric that rewards models for feasible answers based answers given by 10 human volunteers. Whilst not a perfect metric due to the small number of collected human answers, the authors correctly emphasise that metrics such as BLEU and ROUGE from other NLP tasks are not suited to the dataset, since the answers to questions are only ever single concepts, spanning 1-3 words. As a result, this consensus accuracy has been widely adopted as a standard metric for the evaluation of VQA systems. I refer the reader to Subsection 2.3.2 for details on the consensus accuracy metric.

Despite the success and contributions of the VQAv1 dataset, it suffers from large statistical biases in its answer set distribution. As an example, [4] points out that a VQA model that answers *yes* to any question beginning with *Do you see a ...* in the VQAv1 dataset achieves a consensus accuracy of 87% on those questions. This is largely due to the question generation process and general human disposition; humans are much more likely to ask the question *Do you see a dog?* if there is actually a dog in the image compared to if there is not. These skewed answer distributions encourage models to rely on question language priors and answer distribution priors to formulate answers without properly incorporating image data into their reasoning processes. I delve further into the impact of language priors in Subsection 2.1.2, focusing on dataset balancing methods developed by other dataset creators to help capture the observations we see in the real world more accurately.

CLEVR

Motivated by the prevalance of skewed answer and question distributions in existing datasets, Johnson, Hariharan, Maaten, *et al.* built the CLEVR dataset to help diagnose common issues and limitations with existing VQA models. They took an entirely different approach to question and image collection, automating both processes to give complete control over question, image and answer distributions.

The key to automating question and image generation is the scene graph, a representation of an image's objects, their attributes and their relationships to other objects. Each object is represented as a node in the scene graph and is annotated with zero or more attributes, and inter-object relationships are represented as graph edges. Scene graphs were made popular in the VQA field by the Visual Genome dataset, but had been previously used for image retrieval tasks [36]. I refer the reader to Subsection 3.3.1 for a formal description of a scene graph and its components.

The creation of each question-image pair in the CLEVR dataset begins with a randomly generated scene graph containing objects *e.g. cube, sphere, cylinder*, attributes *e.g. colour, material and size* and inter-object relationships *e.g. left, right, in front, behind, same attribute*. Scene graph objects and their associated attributes are placed on a 2D plane, positioned according to the relationships encoded by the scene graph, and then rendered using Blender's Python API [37]. Questions are drawn from a collection of around 360 templates, each of which is associated with a functional program. Choosing elements from the scene graph to populate the template and functional program results in a fully-formed question and functional program describing the reasoning processes required to answer the question for that scene graph. For example, the question template *How many* {*colour*} *things are there?* is associated with a functional program `count(filter_colour({colour}, scene))`. To obtain an answer to a question-image pair, the associated functional program is evaluated on the scene graph to yield an object, attribute or relation.

The main issue with the CLEVR dataset as a whole is its limited vocabulary. This leads to reasonably simple images that provide easily distinguishable visual signals based on object and attribute type, as well as an extremely limited answer vocabulary when compared to other VQA datasets. This is clearly seen in the extremely low error rates of VQA models on the dataset; not long after the creation of CLEVR, FiLM [38] obtained an overall accuracy of 97.7. This was swiftly beaten by Hudson and Manning's Compositional Attention Network [3], which achieved an overall accuracy of 98.9 in 2018.

GQA

Inspired by the complex compositional structure of questions in the CLEVR dataset and motivated by the lack of question annotations and suboptimal question and answer distributions in existing datasets, Hudson and Manning developed the GQA dataset. GQA is the largest VQA dataset to date, containing over 22 million questions in entirety, and over 1.7 million questions after applying balancing techniques to reduce biases in its question type and answer distributions. Moreover, it is accompanied by a myriad of additional annotations, ranging from scene graphs, functional programs and full-sentence answers to Faster R-CNN object features and ResNet-101 spatial features.

At a glance, the GQA dataset construction process comprises four key stages:

1. Scene graph preprocessing

2. Question creation
3. Entailment relation discovery
4. Dataset balancing

Due to the complexity of the GQA dataset construction process, I provide a brief overview of these four key steps below. I refer the avid reader to the original GQA publication [1] for unabridged details on the construction of the GQA dataset.

Scene graph preprocessing: Similarly to CLEVR, scene graphs are the starting point for the creation of the GQA dataset. Instead of randomly creating scene graphs and generating images from these graphs, the GQA dataset draws 113K images and scene graphs from the Visual Genome (VG) dataset. Like CLEVR scene graphs, these graphs contain objects, attributes and relations. In addition to spatial relations like *left, right, etc.* as seen in the CLEVR dataset, the VG scene graphs contain verbal *e.g. using, holding, wearing* and comparative *e.g. larger than, shorter than, higher than* relationships. Hierarchical classification techniques and word embedding distances are used to consolidate the natural language used across the VG scene graphs, giving a final vocabulary of 1740 objects, 620 attributes and 330 relations across all dataset splits. After vocabulary creation, the authors remove incorrect and/or unnatural *e.g. (man, in, shirt)* relations between objects using both manual and automatic methods. Finally, additional spatial relationships are added to the scene graph based on object bounding box information, and global properties about the scene are stored alongside each scene graph.

Question creation: As seen in previous works, the GQA dataset uses a set of 524 question templates as the heart of its question generation process, 274 of which were created by creating templates from existing VQAv1 questions. In addition to substituting objects, attributes and relations into question templates, entire sub-templates can be substituted to create complex, compositional questions of varying lengths. For example, the question *"Are there napkins under the utensil to the left of the rice?"* may have been generated from the template *Are there* {*object*} {*preposition*} *the* {*object*}*?*, where the first object placeholder was replaced with *napkins*, but the second object placeholder was replaced with a sub-template *the* {*object*} {*relation*} *the* {*object*}*?*. Moreover, some question templates support random omission or substitution of phrases or expressions. To avoid answer distribution imbalance for logical questions requiring object/attribute existence, decoy objects or attributes are sometimes substituted into question templates. For example, in the VQAv1 dataset, we saw that questions of the form *Do you see a* {*object*}*...* are correctly answered with *yes* 87% of the time. Substituting the object reference with a decoy object that is not a part of the scene graph, 50% of the time provides control over the answer distribution for those types of questions, as well as over object and attribute occurrence and co-occurrence probabilities.

Entailment relation discovery: Prior to question generation, each question template is associated with a functional program that describes the reasoning operations required to answer the question. My substituting objects, attributes, relations and sub-programs during the question generation phase, a functional program for each

question-image pair is created. By categorising each of these functional programs, the authors determine which types of questions are answerable given the answer to another question on the same image. For example, if we know the answer to the question *What colour is the apple?*, then we can guarantee we know the answer to the question *Is the apple green?*. These entailment relations enable the GQA consistency metric as described in Subsection 2.3.3, as well as fine-grained analysis of the types of questions that VQA models find trivial or difficult.

Dataset balancing: Hudson and Manning take a different approach to answer distribution balancing, first smoothing the answer distribution based on global answer types *e.g. colour, material* and then by local answer types which include information about the question subject *e.g. colour-apple* for the question *What colour is the apple?*. Global smoothing penalises VQA models that choose the most common answer for certain question structures, and local smoothing makes it more difficult to guess answers based on the subject of a question. This smoothing is performed by reweighting each global or local conditional distribution so that more frequent answers become less frequent and vice versa, whilst maintaining the frequency-ranked order of answers for each bin. Most importantly, this process is tunable, allowing fine control over the balance between uniformity of answer distributions and closeness to real-world distributions. As a result, at the time of writing, I consider Hudson and Manning's implementation of answer distribution balancing as the most appropriate answer to my 3rd question for effective VQA dataset creation, as posed at the start Section 2.1.

2.1.2 Language Priors and Answer Distributions in VQA Datasets

As briefly mentioned in the previous Subsection, answer distribution and question type distribution have a significant impact on model evaluation. Skewed question and answer distributions are clearly seen in early VQA datasets, but as the first widely accepted VQA dataset, the VQAv1 dataset brought these issues to the forefront of the VQA field. Many prominent authors [1], [4], [5], [21], [30], [39], [40] have noted that VQA models will exploit statistical priors present in textual information and often neglect visual information if given the opportunity. To encourage the creation of VQA models that learn useful visual reasoning skills, VQA dataset authors aim to create datasets that contain minimal evaluation-hindering biases, whilst staying true to observations that we see in real-world situations.

As an example of this trade-off, consider the question *What colour is the apple?*. A dataset where the answer to this question is *red* 80% of the time and *green* 15% of the time and *yellow* 5% of the time could be considered biased, since a VQA model could simply identify the type of question and blindly answer *red* to achieve an accuracy of 80% on these questions. However, balancing the dataset's answer distribution is not as simple as making the answer distribution uniform, since the real world has its own priors. In the real world, the distribution of red, green and yellow apples is dependent on apple variety, ripeness and growing conditions, and is likely not uniform.

In the remainder of this Subsection, I will explore a variety of techniques used by different dataset creators, highlighting their novelties and downfalls.

VQAv2

Since its creation, many researchers have created variations and derivatives of the VQAv1 dataset, creating different types of questions [29], alternative train and test splits with differing distributions [21] and smoothing out its skewed question and answer distributions [4], [30]. The VQAv1 dataset creators highlight that VQA models can perform surprisingly well without any access to visual information; even a simple "per-question type prior" model achieves 71.03% and 35.77% on binary and numeric questions respectively, and a Long Short-Term Memory (LSTM) [8] recurrent network with only question word embeddings performs similarly at 78.20% and 35.68% [10]. Two primary studies aim to solve these issues: the first [30] focuses on the imbalance in the answer distribution of binary questions in the VQAv1 abstract scenes dataset, and the second [4] focuses on all question types in both the primary VQAv1 and VQAv1 abstract scenes datasets. The results of these two studies form the dataset now known as VQAv2, a popular benchmark for VQA systems on natural questions.

The first study is motivated by the large imbalance in the answer distribution of binary questions, noting that *yes* is the answer to 68% of all binary questions in the abstract scenes dataset. They propose a solution that allows the answer distribution of binary questions in the existing abstract scenes dataset to be balanced: for each question-image pair, they task humans with the creating a new scene using existing clip-art objects [23] that is similar to the existing image, but had a complementary answer to the same question, as shown in Figure 2.1 below.

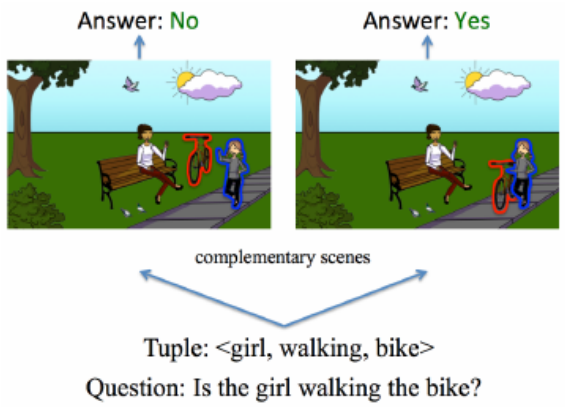


Figure 2.1: Zhang, Goyal, Summers-Stay, *et al.* [30] demonstrate their answer distribution balancing technique. Given the scene on the left and the question *Is the girl walking the bike*, workers were tasked with creating a scene that differs from the scene on the left but has the opposite answer to the question, as illustrated by the scene and answer on the right.

Though novel, this approach does not extend well to real-world image datasets, due to an inability to generate complementary images that are indistinguishable from photographs. Moreover, the approach is only applicable to binary questions,

which comprise almost 41% of the VQAv1 abstract scenes dataset [10]. Despite this method's success in reducing the proportion of unbalanced question-image pairs by almost 72%, around 20.48% of question-image pairs in the balanced dataset were left unbalanced. This was either because a complementary scene could not be created due to limitations of the clip-art library (5.93%) *e.g.* a question such as *'Is it raining?'* cannot be negated effectively, since the clip-art object bank does not contain a rain graphic, or due to disagreement between the answers collected by workers for a given scene (14.55%)

For the rest of the questions in the VQAv2 dataset, Goyal, Khot, Summers-Stay, *et al.* use a similar approach, but instead of asking AMT workers to create a new image, they provide them with 24 nearest-neighbour images to choose from such that the answer to the new image is not the answer to the existing image for the same question. Answers to these new question-image pairs were collected in a similar fashion to the VQAv1 dataset. As a result, the number of question-image pairs in the VQAv2 dataset is almost double that of the VQAv1 dataset, and the frequency-weighted average entropy of answer distributions increased by 56% due to the balancing process.

GQA

Hudson and Manning approach the task of dataset balancing from a different angle, aiming to mitigate biases in the answer distributions of the GQA dataset whilst maintaining some degree of representation of real-world priors. For brevity, I refer the reader to Subsection 2.1.1 for details on the balancing process used in the GQA dataset, as it is built-in to the dataset creation process.

2.2 Visual Question Answering Methods

Given the widespread adoption of scene graph annotations in VQA datasets [1], [5], [31], [33] it makes sense to leverage scene graph information for training VQA models. In previous years, many researchers have avoided using scene graphs as supervisory data, as they serve as an almost perfect visual signal. Additionally, scene graphs could not always be relied on as a supervisory data source; the scene graphs in the Visual Genome dataset were meticulously annotated by over 33,000 AMT workers [31], and the scene graphs in GQA required a large amount of additional pruning and modification to fit them to a standardised vocabulary of objects, relations and attributes. However, more recently, researchers have been using these datasets to develop scene graph generation methods [41]–[44]. As these scene graph generation methods become more accurate in their relationship existence and annotation methods, generated graphs as an image embedding for VQA tasks will become more and more effective.

Consequently, the majority of the VQA methods that I cover in this literature review are graph-based, and leverage scene graphs in their reasoning processes.

2.2.1 Graph Methods

Originally proposed for node classification tasks, both the Graph Convolutional Network (GCN) [18] and Graph Attention network (GAT) [2] have gained the attention of computer vision researchers in the last three years, particularly for image-to-scene-graph [43] and text-to-scene-graph [45] generation, and, more recently, visual question answering [19], [20]. Prior works have demonstrated the effectiveness of attention-based graph convolutions for processing scene graph information in a variety of different contexts, with Relation-aware Graph Attention networks (ReGAT) [19] achieving state-of-the-art results on the VQAv2 and VQA-CPv2 datasets in 2019, and aligned Dual-Channel GCN (DC-GCN) [20] achieving comparable or greater performance than ReGAT on various VQAv2 subtasks in 2020. However, both of these models do not evaluate the effectiveness of GATs for scene graph processing using scene graph information provided in datasets like GQA and CLEVR. This makes it difficult to determine how much of the reported performance gain over non-scene-graph-based models can be accredited to the GAT architecture, and how much is due to the inclusion of generated scene graph information. In the remainder of this subsection, I outline how GCN and GAT architectures can be leveraged for VQA tasks, before giving a brief overview and analysis of the DC-GCN and ReGAT VQA architectures. I refer the reader to the List of Symbols for graph-specific nomenclature used in this subsection.

Graph Convolutional Networks (GCN)

Originally proposed for graph-based semi-supervised classification tasks, the Graph Convolutional Network (GCN) extends the core concepts of Convolutional Neural Networks (CNNs) to sparse graphs; instead of performing convolutions on regions of pixels, GCNs convolve features from connected nodes in graphs.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an associated matrix of node features for the lth GCN layer $H^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_l}$, a GCN has the following propagation rule:

$$H^{(l+1)} = f\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2.1}$$

where $\tilde{A}$ is the adjacency matrix of $\mathcal{G}$ with added self-loops, $\tilde{D}$ is the degree matrix associated with $\tilde{A}$, $W^{(l) \in \mathbb{R}^{d_l \times d_{(l+1)}}}$ is a set of learnable weights for the lth layer of the GCN and f is some non-linear activation function.

The beauty of the GCN convolution lies in the $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ portion of Equation 2.1; by multiplying $\tilde{A}$ in this manner, the ith node feature in $H^{(l+1)}$ becomes a linear combination of the ith element of $H^{(l)}W^{(l)}$ and the elements of $H^{(l)}W^{(l)}$ corresponding to its neighbours, where the weights of the neighbours are equal and smaller if the ith node has many neighbours. Unfortunately, the assignment of equal weights to neighbouring nodes makes GCN convolutions ineffective for embedding relationship data in scene graphs for VQA, as some relationships are more important than others. For example, given a scene graph containing a man and a woman standing next to each other and a small cloud in the background, the relationship between

the man and the woman is more important in a VQA context than the relationship between the man and the cloud or the woman and the cloud, as the people are the most salient aspects of the image.

Graph Attention Networks (GAT)

Originally proposed for inductive and transductive node-classification tasks, the Graph Attention network (GAT) addresses the issues of the GCN propagation rule by allowing nodes in the neighbourhood of the propagation target to be assigned different learned weights.

Adopting similar notation as the GCN propagation rule, the GAT propagation rule is defined as:

$$H_i^{(l+1)} = f\left(\alpha_{ii} W^{(l)} H_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W^{(l)} H_j^{(l)}\right) \tag{2.2}$$

where α_{ij} is the attention coefficient for nodes i and j, calculated using some attention function a and normalised over the neighbourhood $\mathcal{N}(i)$ of i as follows:

$$\alpha_{ij} = \frac{\exp\left(a(W^{(l)} H_i^{(l)}, W^{(l)} H_j^{(l)})\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(a(W^{(l)} H_i^{(l)}, W^{(l)} H_k^{(l)})\right)} \tag{2.3}$$

Any attention function can be used for a, however the authors use a modified additive attention mechanism $a(W^{(l)} H_i^{(l)}, W^{(l)} H_j^{(l)}) = \text{LeakyReLU}(w^\top [W^{(l)} H_i^{(l)} \parallel W^{(l)} H_j^{(l)}])$, distinguished by a learnable weight vector w. Moreover, separate GAT convolutions can be computed concurrently, each acting as a single head in a multi-head attention mechanism [12], then combined either by concatenation or mean aggregation. In general, a GAT layer with multiple heads helps stabilise training and allows each attention head to focus on different types of relationships between nodes as needed, making them highly effective for visual reasoning tasks.

Aligned Dual Channel Graph Convolutional Networks (DC-GCN)

Noting that many current graph-based VQA methods focus only on capturing relationships between objects in images, and process textual information using traditional RNN-based approaches, Huang, Wei, Cai, *et al.* propose the use of GATs for processing both image features and question words. The DC-GCN model comprises three main components: an image GAT, a question GAT and an attention-based alignment module.

The image GAT operates over a generated scene graph, with each node represented by a 2048-dimensional Faster-RCNN [7] object feature and edges determined by a bounding box intersection-over-union threshold. The question GAT operates over a separate graph, with each node represented by a 300-dimensional GloVe vector [46]

and edges determined using the dependency tree output of a Stanford dependency parser [47].

The attention-based alignment module aims to reconcile the image GAT and question GAT features, determining which image features are relevant to the question. The first part of this process is a scaled dot product self-attention computation [12] on question GAT features, used to highlight the important question words $\tilde{H}_q$. Secondly, a scaled dot product attention computation on image GAT features using question self-attention results as a query extracts important visual features $\tilde{H}_v$. Finally, the attended question and image features are concatenated and passed through a single linear layer to yield an answer distribution.

Whilst one of the first methods to leverage graph structures for question embeddings, the DC-GCN model has two major issues. The first is that the scene graph generation method captures only naive spatial relations, completely neglecting verbal, comparative and other relationship types and consequently limiting its ability to answer certain types of questions. Secondly, the size of the attention alignment module seems insufficient for multi-modal fusion and reasoning tasks when compared to other complex multi-modal reasoning models like Bilinear Attention Networks (BAN) [15] and Compositional Attention Networks [3].

Relation-aware Graph Attention Networks (ReGAT)

Instead of using separate GATs for question and image features, Li, Gan, Cheng, *et al.* use multiple GATs for different types of relations, and enrich GAT node feature inputs with question information in an effort to promote the learning of question-adaptive relationship embeddings.

The ReGAT model uses three GATs in total, one for semantic relations *e.g. wearing, holding*, one for spatial relations *e.g. inside, to the left of*, and one for implicitly learned relations. All three GATs operate on the same 36 node embeddings, where each node corresponds to an object in the image and is represented by the concatenation of its corresponding Faster R-CNN feature vector $v_i \in \mathbb{R}^{2048}$ and a common question representation $q \in R^{1024}$, the self-attended hidden states of a bidirectional Gated Recurrent Unit (GRU) [9]. Edge existence and labelling in the spatial and semantic graphs are framed as classification problems, using bounding box features and Faster R-CNN features as inputs respectively. Each unique edge label is associated with a bias term, which is used in tandem with concatenated visual and question features to compute the attention coefficients for each spatial and semantic GAT layer.[1]

The implicit GAT operates on a complete graph, computing edge attentions using a combination of positional bounding box encodings and dot-product attentions between node features. Even though many of these attention coefficients are zero, the dynamic computation of attention weights for all edges undermines the computational benefit of graph attention networks on sparse graphs.

[1] I refer the reader to the original publication [19] for specifics on ReGAT attention coefficient computations.

Unlike DC-GCN, ReGAT uses existing VQA methods as its fusion module instead of a custom alignment module. The authors demonstrate its ability to provide additional information for VQA tasks, reporting improvements of 1.5-3% compared to baseline performance for three popular VQA model [11], [15], [16] on the VQAv2 validation set.

2.2.2 Attention Methods

Compositional Attention Networks

Motivated by the lack of interpretable models capable of performing complex reasoning tasks, Hudson and Manning created a recurrent attention-based model capable of performing discrete reasoning steps, where each recurrent cell is responsible for performing a specific reasoning operation. Whilst originally developed for VQA tasks, the compositional attention network can be extended to any task that requires performing reasoning operations on an input, visual or otherwise. This input is known as the knowledge-base, denoted KB in Figure 2.2. In the context of VQA, the knowledge-base can be any visual signal such as a raw image, or more commonly, CNN and/or Faster R-CNN features.

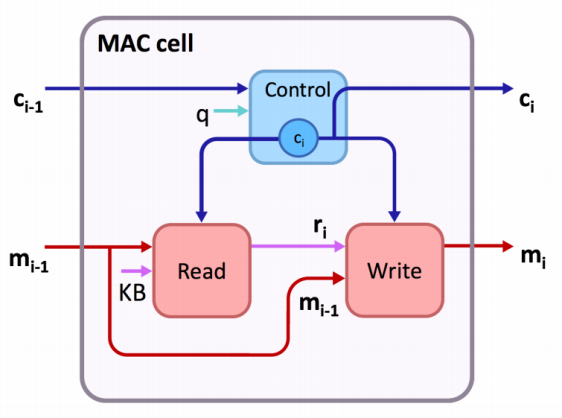


Figure 2.2: A single recurrent cell from a compositional attention network, as illustrated by Hudson and Manning. [3]

At each time step i, the control module determines which part of the question q to attend, represented by the control vector c_i. By accounting for the previous control vector c_{i-1} in this process, the control module can determine which parts of the question are relevant based on what it considered relevant in the previous reasoning step.

The read module is responsible for determining which parts of the knowledge-base are relevant to the current reasoning step, based on the relevant parts of the question contained in c_i and result of the previous reasoning step, stored in the memory vector m_{i-1}. This retrieved information r_i is passed to the write module, which stores the result of the current reasoning step in the cell's memory vector m_i. This result is determined from the result of the previous reasoning step m_{i-1} and the retrieved information r_i, and may be the same as the previous reasoning step in the event

that there are no more reasoning operations to perform (determined by c_i), or may be an entirely new reasoning result.

Hudson and Manning substantiated their intuition behind the compositional attention network, demonstrating its effectiveness via empirical results and attention visualisations on the CLEVR dataset, achieving a new top accuracy of 98.9% in 2018. They also showcased its reasoning capabilities on the GQA dataset, achieving an accuracy of 54.06% and sealing its status as a solid baseline model for compositional reasoning tasks.

In addition to outperforming all existing state-of-the-art models on the full CLEVR dataset, compositional attention networks also perform well on subsets of CLEVR, with an accuracy of up to 35% higher than other state-of-the-art models when trained on a 10% subset of the CLEVR training data.

The combination of interpretability, data efficiency and wide acceptance as a GQA baseline motivate my use of the compositional attention network as the reasoning component in my own VQA model, which I describe in detail in Chapter 3.

2.3 Visual Question Answering Metrics

In their seminal VQA paper, Malinowski and Fritz [25] note that traditional accuracy measures fail to capture partial correctness of answers; given a reference answer 'carton', the traditional accuracy metric would give zero weight to the answer 'box', despite the semantic similarity between the ground-truth and predicted answers. In general, larger answer vocabularies typically lead to blurrier semantic boundaries between answer classes when compared to datasets with a smaller answer vocabulary like CIFAR [48]. Since datasets that frame the VQA problem as a classification task typically have a large answer vocabulary containing objects, numbers, colours and other concepts, it follows that an effective VQA metric should reward answers that are partially correct or share semantics with the reference answer.

Metric design becomes even more important when considering full-sentence answers instead of single-word answers; evaluation of full-sentence answers requires metrics that are robust under semantic and syntactic variations between predicted and reference answers. A variety of natural language generation (NLG) metrics [49]–[52] have been applied to image captioning tasks [52], [53], and are consequently applicable to open-ended VQA tasks requiring full-sentence answers. Unfortunately, none of these metrics were designed explicitly for VQA tasks, and must be considered in addition to class-based metrics to gain useful insights into model performance.

In the following subsections, I explore a variety of metrics for both class-based and free-form VQA tasks as outlined in Table 2.2, focusing on how they reward partially correct answers.

Metric	Suitability			Common Uses
	Multi-class	Multi-label	Open-ended	
Accuracy	✓	✓		Multiple
WUPS	✓	✓		Multiple
Consensus Accuracy	✓	✓		VQA
Consistency	✓	✓		VQA
Validity	✓			VQA
Plausibility	✓			VQA
Grounding	✓	✓	✓	VQA
Distribution	✓			VQA
BLEU [49]			✓	Machine translation
ROUGE [50]			✓	Text summarisation
METEOR [51]			✓	Machine translation
CIDEr [52]			✓	Image captioning

Table 2.2: A comparison of metrics and their suitability for various VQA tasks.

2.3.1 Semantic Similarity Measures

Wu-Palmer Similarity

Originally coined the "conceptual similarity measure" [54], the now-labelled Wu-Palmer (WUP) similarity is used to measure the semantic similarity of two concepts based on their positions in a taxonomy tree such as WordNet [55]. Formally, given two concepts A and B in a taxonomy tree T, we define the Wu-Palmer similarity of A and B as

$$WUP(A, B) = \frac{2N_C}{N_A + N_B + 2N_C} \tag{2.4}$$

where N_A and N_B are the number of edges from A and B to their lowest common ancestor C respectively, and N_C is the depth of C in T, as illustrated in Figure 2.3.

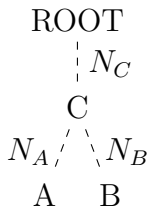


Figure 2.3: Given two concepts A and B, C denotes the "least common super-concept" of A and B. (Figure adapted from [54])

Given this definition, it naturally follows that two equal concepts occupy the same position in the taxonomy tree and thus have a similarity of 1, whilst two concepts whose lowest common ancestor is the root of the tree have a similarity of 0.

Metric	Advantages	Disadvantages
Accuracy	- Simple and interpretable. - Suitable for problems with a small answer set cardinality.	- Penalises slightly incorrect answers harshly. - Requires an exact set of answers for multi-label tasks.
WUPS	- Rewards incorrect but semantically similar answers.	- Rewards answers that share a concept class *e.g. 'colour'* with the true answer but have opposite meanings e.g. *'white'* vs *'black'*.
Consensus Accuracy	- Simple and interpretable. - Aligns with human answers.	- Multiple correct reference answers may be contradictory. - Not all questions have 3 matching reference answers - Reference answers are expensive to collect.
Consistency	- Ensures models avoid contradictory answers to related questions.	- Requires a set of entailed questions for each question.
Validity	- Measures a model's ability to respond to various question types.	- Requires a set of valid answers for each question type.
Plausibility	- Ensures models avoid nonsensical answers.	- Requires scene graphs for all dataset samples. - Can be imprecise for sparsely occurring objects.
Grounding	- Ensures models use image data to answer questions instead of exploiting language priors.	- Only applicable to attention-based models. - Requires knowledge about which image regions are relevant to each dataset sample.
Distribution	- Ensures models respond well to all question subject-type groups	- Cannot be compared across datasets with different distributions.
BLEU	- n-grams capture positional relationships between words. - Penalises answers that are too short or too long.	- Requires multiple reference answers for good performance. - Performs better for entire corpora compared to individual sentences due to the corpus-level brevity penalty, an issue for VQA. - A zero score for any n-gram component gives a zero BLEU score.

Table 2.3: A comparison of metrics and their advantages and disadvantages for various VQA tasks.

Whilst WUP similarity can capture similarities between a target and predicted concept, Malinowski and Fritz extend WUP similarity to sets of concepts, proposing the WUP set score for evaluation of multi-label classification tasks:

$$\text{WUPS}(A, B) = \frac{1}{N}\sum_{i=1}^{N} \min\{\prod_{a \in A_i} \max_{b \in B_i} \text{WUP}(a, b), \prod_{b \in B_i} \max_{a \in A_i} \text{WUP}(a, b)\} \quad (2.5)$$

To apply this metric to VQA tasks, we let A and B be a collection of sets of predicted and reference answers respectively, each of size N. Noting that $WUP(A_i, B_i) \in [0, 1]$ for each $A_i \in A$ and $B_i \in B$, the following useful properties of the metric become evident:

1. $WUPS(A, B) \in [0, 1]$, with $WUPS(A, B) = 1$ when $A = B$.
2. Naive models which overestimate the number of target answers will be penalised according to how dissimilar the additional provided answers are to their closest target answer.
3. Models which underestimate the number of target answers *i.e.* $|A_i| < |B_i|$ will be penalised, with a harsher penalty if the concepts in the target set are dissimilar.

The authors note that WUP yields large values for related but not interchangeable concepts, noting that $WUP(\text{carton}, \text{box}) = 0.94$ where $WUP(\text{stove}, \text{fire extinguisher}) = 0.82$. To encourage precise answers and avoid accepting related but inaccurate answers, the authors down-weight $WUP(a, b)$ by a ratio $r = 0.1$ whenever it is less than some threshold $t = 0.9$.

While this down-weighting alleviates this issue for questions that query objects in an image, answers to questions about object attributes and colours are often poorly evaluated; Kafle and Kanan note that even after down-weighting WUP scores, $WUPS(\text{white}, \text{black}) = 0.91$. Whilst the metric correctly rewards the answerer for correctly identifying that the answer to the question is a colour, a score of 0.91 seems too high, highlighting the non-linearity of the metric that makes it more difficult to interpret at a glance unlike other common metrics like accuracy.

By nature, the metric also operates on single WordNet concepts, which often coincide with single words. To evaluate full-sentence or even multi-word answers with WUPS, we must treat sentences as a set of distinct words, disregarding essential syntactic and contextual information. This makes WUPS a poor candidate for open-ended VQA evaluation when compared to other NLG metrics such as BLEU, ROUGE, METEOR and CIDEr, as briefly discussed in Subsection 2.3.4 and summarised in Table 2.3.

2.3.2 Consensus Measures

VQA Consensus Accuracy

Introduced by Antol, Agrawal, Lu, *et al.* in 2015, the VQA consensus accuracy metric is the most common metric used to benchmark VQA models, due to the widespread adoption of the VQA dataset and its variants. To reward partially correct answers, VQA consensus accuracy compares a candidate answer to multiple reference answers, provided by humans via crowdsourcing platforms like Amazon Mechanical Turk (AMT) in the case of the VQA dataset. The VQA consensus accuracy metric is defined as

$$\textsc{Accuracy} = \min\left(\frac{n}{3}, 1\right) \tag{2.6}$$

where n is the number of answers in the set of reference answers that match a given candidate answer, *i.e.* an answer is considered entirely correct if it matches 3 or more of the 10 reference answers for the corresponding question and image.

At a surface level, the metric is interpretable and can reward multiple correct and/or partially correct answers effectively. However, the latter can become a disadvantage depending on the quality of the reference answers. In their analysis of the VQAv1 dataset, Kafle and Kanan point out that 16.7% of questions have less than 3 matching answers in their group of 10 reference answers, capping the performance of a perfect VQA model. A large contributor to this statistic is the presence of multi-word answers in the dataset; 89.32% of answers in the VQAv1 dataset are single-word answers, leaving ample room for disagreement amongst reference answers.

Moreover, supporting multiple correct answers is unfavourable if the correct answers are contradictory. Again, Kafle and Kanan identify that 13% of VQA v1 yes/no questions contain both 'yes' and 'no' 3 or more times in their group of 10 reference answers, making both options entirely correct.

DAQUAR Consensus Measures

Motivated by the relatively poor human baseline performance (Accuracy: 50.20%, WUPS@0.9: 50.82%) on the DAQUAR dataset, Malinowski, Rohrbach, and Fritz extended the dataset by collecting an average of 5 reference answers for each image-question pair and developing two consensus metrics based on their previous WUPS measure:

$$\text{WUPS}_{\text{mean}}(A, B) = \frac{1}{NK}\sum_{i=1}^{N}\sum_{i=1}^{K}\min\{\prod_{a\in A_i}\max_{b\in B_i}\text{WUP}(a,b), \prod_{b\in B_i}\max_{a\in A_i}\text{WUP}(a,b)\} \tag{2.7}$$

$$\mathrm{WUPS}_{\min}(A,B) = \frac{1}{N}\sum_{i=1}^{N}\max_{k=1}^{K}\left(\min\{\prod_{a\in A_i}\max_{b\in B_i}\mathrm{WUP}(a,b), \prod_{b\in B_i}\max_{a\in A_i}\mathrm{WUP}(a,b)\}\right) \tag{2.8}$$

where N, A and B are as defined in Equation 2.5, and K is the number of reference answers for the i-th dataset sample.

The mean consensus metric rewards candidate answers that coincide with more frequent reference answers, whilst the minimum consensus metric rewards candidate answers according to their similarity to the closest reference answer.

2.3.3 Metrics for Visual Reasoning

As discussed earlier, the standard accuracy metric penalises partially correct answers as harshly as incorrect answers. When using accuracy alone, it is impossible to distinguish between a model that provides reasonable but incorrect answers and one that provides nonsensical answers. Moreover, the standard accuracy metric gives no insights into how a VQA model arrived at its answer to a given question and image.

To combat these issues, Hudson and Manning developed five new metrics in tandem with their GQA dataset [1] to evaluate the internal reasoning processes of VQA models: Consistency, Validity, Plausibility, Grounding and Distribution.

Consistency

The consistency metric is motivated by the following scenario:

Say we have two questions q_0 and q_1 that ask about properties of the same image, *e.g.* q_0 = *'What colour is the apple?'* and q_1 = *'Is the apple red?'*

Since both questions are asking about the same apple, a consistent model is one that does not contradict itself in answering the two questions *e.g.* a_0 = *'red'* and a_1 = *'yes'* are consistent answers, where a_0 = *'red'* and a_1 = *'no'* are not.

To calculate consistency, each question-answer pair q, a in the GQA dataset is annotated with a set of entailed questions E_q. For the example above, q_1 is an entailed question of q_0, since knowing the answer to q_0 infers the answer to q_1. Given a set of questions Q that a model answered correctly and a set of entailed questions for each $q \in Q$ denoted $\mathcal{E} = \{E_q \mid q \in Q\}$, the consistency metric is calculated according to Algorithm 2.1.

Unfortunately, the consistency metric does not account for inaccurately answered questions. There is certainly an argument to be made that including incorrect answers in the consistency calculation would skew the consistency metric in favour of more accurate models. Moreover, including these incorrect answers could give large consistency scores for poorer models that exploit statistical priors in the dataset's answer distribution. Nonetheless, knowing consistency amongst incorrectly answered

Algorithm 2.1: Consistency metric algorithm

function $Q, \mathcal{E}$
  $s_c \leftarrow 0$
  foreach $\{q_i \in Q \mid E_{q_i} \neq \emptyset\}$ **do**
    $s_{q_i} \leftarrow 0$
    foreach $q_e \in E_{q_i}$ **do**
      if $q_e \in Q$ **then** $s_{q_i} \leftarrow s_{q_i} + 1$;
    end
    $s_c \leftarrow s_c + \frac{s_{q_i}}{|E_{q_i}|}$
  end
  return $\frac{s_c}{|\{q_i \in Q | E_{q_i} \neq \emptyset\}|}$
end

questions gives information about the model's overall reasoning skills and tendency to exploit statistical biases in certain question types. Hence, for error analysis purposes, it would be more suitable to compute consistency over incorrectly answered questions instead of computing consistency over all questions.

Validity

The validity metric is the most lenient of the GQA metrics, and simply captures whether a candidate answer corresponds with the type of answer the question demands. For example, valid answers for questions starting with *What colour* are colours like *red* or *blue*, where answers like *yes* or *three* would be invalid.

To calculate validity, the GQA evaluation script [57] contains a set of valid answers V_q for every question in the dataset. For each question $q \in Q$ and its corresponding candidate answer $\hat{a}_q$, the validity metric is simply:

$$\textsc{Validity} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} 1, & \text{if } \hat{a}_q \in V_q \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

Plausibility

Where the validity metric aims to measure whether answers to questions are non-degenerate, plausibility measures whether the candidate answer is rational by checking for co-occurrence of the answer and the question's subject in the scene graphs for the rest of the dataset.

In practice, plausibility is computed in the same way as validity, using a pre-computed set of plausible answers P_q for each question in place of the set of valid answers V_q used in Equation 2.9.

The method used to compute P_q for each question depends on the question's type; for the GQA dataset, the scene graph contains attributes, objects and relations, meaning that plausibility needs to be computed differently depending on whether

the question's answer is an object, relation or attribute. Moreover, plausibility also needs to be computed for other types of questions whose answers are not in the scene graph, such as binary and counting type questions.

For the sake of brevity, I will focus on two different question types to give an intuition about how sets of plausible answers are determined.

For questions that ask about an attribute of an object o, an attribute is considered plausible if it co-occurs with o at least once in the scene graphs for other images in the dataset. For example, *red* would be a plausible answer to the question *What colour is the apple?* if the dataset contains at least one image of a red apple, where *blue* would be implausible.

For questions that ask about a target object t given a subject s and relation r *e.g. What is the man holding?*, t is considered plausible if there exists a relation s, r, t in the scene graph for any image in the dataset.

Grounding

The grounding measure is unlike other metrics in that it does not compare the quality of answers given by a VQA model. Instead, it focuses on model interpretability, ensuring models use relevant visual information to formulate answers. For example, consider a model that can answer the majority of questions about the colour of an apple correctly based purely on the fact that the most common answer to such questions is red; such a model does not utilise any visual information in its reasoning process. Conversely, a model that answers the same proportion of questions correctly by identifying the location and colour of apples in an image is a visually *grounded* model, and exhibits the type of behaviour that we strive for as a research community.

At its core, the grounding metric takes as input a set of sets of bounding boxes $\hat{\mathcal{B}} = \{\hat{B}_{(q_i,r_i)} \mid ((q_i, r_i), a_i) \in S\}$ for each question-image pair (q_i, r_i) in a VQA dataset $S = \{((q_i, r_i), a_i)\}_{i=1}^{n}$. Each bounding box in $\hat{B}_{(q_i,r_i)}$ corresponds to a region in the image r_i that the the model attended to when answering the associated question q_i. Moreover, each bounding box $\hat{b} \in \hat{B}_{(q_i,r_i)}$ has an associated attention value $\alpha_{\hat{b}}$, indicating the importance assigned to that image region by the model during the question-answering process. These predicted bounding boxes and attention values are compared to a set of sets of reference bounding boxes $\mathcal{B} = \{B_{(q_i,r_i)} \mid ((q_i, r_i), a_i) \in S\}$, where each box in $B_{(q_i,r_i)}$ indicates the visual regions in r_i that are essential for answering question q_i.

The grounding metric for any (q_i, r_i) pair in a dataset is defined as the total precision-based overlap between each set of predicted and reference regions $\hat{B}_{(q_i,r_i)}$ and $B_{(q_i,r_i)}$ respectively. For an entire dataset S, per-sample grounding scores are averaged to give the final grounding metric, as outlined in Algorithm 2.2.

Unfortunately, any VQA model that does not use attention-based mechanisms to attend to regions or objects in an image cannot be evaluated using the grounding metric. This limitation is becoming more important, as more recent models have

Algorithm 2.2: Grounding metric algorithm

function $S, \hat{\mathcal{B}}, \mathcal{B}$
  $s_g \leftarrow 0$
  foreach $((q_i, r_i), a_i) \in S$ **do**
    $B \leftarrow B_{(q_i,r_i)} \in \mathcal{B}$
    $\hat{B} \leftarrow \hat{B}_{(q_i,r_i)} \in \hat{\mathcal{B}}$
    foreach $b \in B$ **do**
      foreach $\hat{b} \in \hat{B}$ **do**
        $\alpha_{\hat{b}} \leftarrow$ Attention coefficient for $\hat{b}$
        $s_g \leftarrow s_g + \alpha_{\hat{b}} \times \frac{\text{IntersectionArea}(\hat{b},b)}{\text{Area}(\hat{b})}$
      end
    end
  end
  $s_g \leftarrow \frac{s_g}{n}$
  return s_g
end

adopted graph-based representations of visual inputs instead of spatial or object-based image features. For this reason, I do not report grounding metric results for my model in Chapter 5.

Distribution

Given the importance of dataset balancing and the influence of language priors on model performance as discussed in Section 2.1, the distribution metric compares the distributions of reference and candidate answer sets. This provides a way of measuring whether a VQA model exploits statistical priors and focuses on more common answers or whether it can predict less-frequent answers based on its knowledge of the question and image.

The similarity score of the candidate and reference distributions for each question type is computed using the Pearson Chi-squared statistic [58], and the distribution metric is a weighted average of these scores, with the weights being the number of questions of that type. I refer the reader to [1] for detailed analysis of question type distributions in the GQA dataset.

2.3.4 Metrics for Open-ended VQA

Given the challenge of multi-class and multi-label VQA tasks, there has been little research into open-ended VQA, even though some datasets contain full-sentence answers for each question. In addition, research into open-ended VQA has been stunted by the lack of a widely accepted metric for the task; metrics such as BLEU, ROUGE, METEOR and CIDEr were originally developed for other tasks as shown in Table 2.2, and thus have various disadvantages when used in VQA contexts.

Since I focus on VQA as a multi-class classification problem in this dissertation, I

keep this section brief, delving into the characteristics of BLEU to exemplify the types of issues that open-ended VQA evaluation poses. I choose to elaborate on BLEU instead of metrics with more robust characteristics like ROUGE, METEOR and CIDEr since BLEU was used for evaluation of early open-ended VQA datasets such as Visual Madlibs [27]. Moreover, since BLEU suffers from multiple weaknesses [51] in the context of machine translation, it serves as a good example to illustrate common points of consideration for more advanced metrics. For the keen reader, I outline some of the key shortcomings of the BLEU metric that metrics like ROUGE, METEOR and CIDEr address at the end of this subsection.

BLEU

Originally developed for automatic evaluation of machine translation systems, the Bi-Lingual Evaluation Understudy (BLEU) metric was created in an effort to reduce dependence on manual evaluation methods and thus promote more rapid prototyping of ideas. The three keys to its success in the machine translation field are its efficiency, language independence and correlation with human evaluation results.

In the context of machine translation, the core component of the BLEU metric compares groups of n adjacent words (n-grams) in a candidate translation to n-grams in a group of reference translations using a modified precision measure. This comparison is best described with an example:

Candidate	**the the** the the the the the
Reference 1	**the** cat is on **the** mat
Reference 2	there is a cat on the mat

Table 2.4: An example given by Papineni, Roukos, Ward, *et al.* [49] of a poor candidate translation that performs well (7/7) on the standard unigram precision measure, but poorly (2/7) on the modified unigram precision measure used by BLEU.

As illustrated in Table 2.4, the standard n-gram precision measure simply counts how many n-grams in the candidate translation appear in any of the reference translations, allowing poor translations that repeat common n-grams to perform well. To combat this, the BLEU metric uses a modified n-gram precision measure, where it calculates a clipped number of n-gram matches between the candidate and reference translations: in the example in Table 2.4, the unclipped candidate count of the unigram *the* is 7, but the clipped count is 2, since the maximum number of occurrences of the unigram *the* in any reference translation is 2.

Papineni, Roukos, Ward, *et al.* [49] extend their modified n-gram precision measure to multiple sentences, however I omit this section for the sake of clarity since open-ended VQA answers are almost always phrased as single sentences.

Whilst these modified n-gram precisions are useful for gaining an idea of whether a candidate translation is good or bad, the BLEU metric accounts for the fact that different n-gram precisions capture different aspects of a translation. For example, unigram precision is useful for measuring whether a candidate translation uses

the same words as a reference translation, but cannot account for the *fluency* of a translation like bigram or trigram precisions can. To address this, the BLEU metric combines multiple modified n-gram precision scores according to Equation 2.10.

$$\text{BLEU} = P_B \cdot \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right) \tag{2.10}$$

In Equation 2.10, we see that the BLEU metric is defined as some brevity penalty P_B multiplied by the geometric mean of individual n-gram precisions p_n. This brevity penalty is essential for ensuring candidate translations are not too short; a candidate translation can achieve a modified precision score of 1 if it contains a single n-gram and that n-gram is in one of the reference answers. The brevity penalty achieves this by comparing the length of the candidate translation c to the length of the reference translation closest in length to c, denoted r, taking a value < 1 whenever $c < r$. Moreover, the penalty decays exponentially towards zero as $c \to 0$, as seen in Equation 2.11.

$$P_B = \begin{cases} 1, & \text{if } c > r \\ e^{\frac{1-r}{c}}, & \text{if } c \leq r \end{cases} \tag{2.11}$$

Naturally, the BLEU metric can be used for any natural language evaluation task where we need to compare a candidate sentence to a reference sentence. This includes open-ended VQA evaluation, where we compare sentence-based candidate answers to one or more reference answers.

Analysing the design decisions of the BLEU metric, we see that it handles a variety of issues that apply to open-ended VQA evaluation. Firstly, it encourages responses of appropriate length; the brevity penalty ensures that candidate responses are not too short, and the modified n-gram precision measure inherently penalises responses that include superfluous n-grams not found in reference responses. This ensures that candidate responses not only get the question correct, but provide an answer in the context of a sentence. Secondly, by including n-gram precision measures for multiple values of n, it ensures candidate answers are correct and fluent in their vocabulary usage.

Unfortunately, there are numerous disadvantages to using the BLEU metric for open-ended VQA evaluation, some of which have been addressed by one or more of ROUGE, METEOR and CIDEr. Firstly, VQA datasets rarely contain multiple open-ended answers. For example, the GQA dataset [1] comes with a single open-ended answer per question. This means that a candidate answer that is correct but differs in phrasing to the reference answer may achieve a low BLEU score. This occurs because BLEU was originally designed for large corpora with multiple reference translations sourced from different people to promote answer variety. Even the more recent CIDEr metric is consensus based, requiring multiple image captions to evaluate the quality of a candidate description. To effectively utilise metrics like

BLEU and CIDEr for open-ended VQA tasks, it is essential to have multiple candidate answers for each question in the dataset. Secondly, the BLEU metric does not account for synonyms or morphological variations of words. If the reference answer refers to a *carton* but the predicted answer uses the word *box*, or *cartons*, we observe a decrease in n-gram precision despite the semantic or morphological similarities between words used by the candidate and reference answers. This issue is addressed in the METEOR and CIDEr metrics, making them more robust to candidate answer phrasing variations. Moreover, we see that open-ended VQA answers often have a large overlap with their corresponding question. For example, given the question *What colour is the apple?* and a reference answer *The apple is red.*, a candidate answer *The apple is.* would achieve a modified unigram, bigram and trigram precision of 1 despite drawing all of its words from the question and not providing a concrete answer. This emphasises the importance of using regular multi-class VQA metrics like accuracy, validity and plausibility in tandem with NLG metrics, as a high BLEU, ROUGE, METEOR or CIDEr score may not necessarily correspond to a correctly answered question. This makes it difficult to compare the performance of VQA models since there is no one-size-fits-all metric for open-ended VQA evaluation.

Chapter 3

Methodology

3.1 Architecture Overview

In this section, I propose a new model for single-concept visual question answering tasks, treating the task as a multi-class classification problem. The proposed model comprises four main components, as shown in Figure 3.1 and outlined briefly below. In the following subsections, I explain the motivations behind each architecture component, along with detailed descriptions of their composition. I refer the reader to the List of Symbols for descriptions of common nomenclature used in this chapter.

Question embedding and processing module: In classic Natural Language Processing (NLP) fashion, the question embedding component is responsible for the preprocessing and vectorisation of raw question data for use by the rest of the model. The question processing module has two main roles: the first is to learn a sequence of word embeddings that capture the relevant features of each question word in the context of the question as a whole. Secondly, the question module learns a question-level embedding that contains information about the entire question.

Scene graph embedding and processing module: The scene graph embedding model is responsible for the preprocessing and vectorisation of scene graph data. In this dissertation, I use the scene graphs provided as part of the GQA dataset to evaluate the effectiveness of various scene graph modules under controlled conditions. This component can be readily extended to utilise the output of existing scene graph generation models [19], [43], yielding a true end-to-end visual question answering (VQA) model. The scene graph processing module then learns a dense knowledge-base that captures the interactions between scene graph object, relation and attribute embeddings. The reasoning module draws information from this knowledge-base when answering questions.

Reasoning module: Given the output features of the scene graph and question processing modules, the reasoning module is in charge of determining which visual features are relevant to the question and vice-versa. The reasoning capabilities of many existing VQA models could be leveraged here; I justify my reasoning module

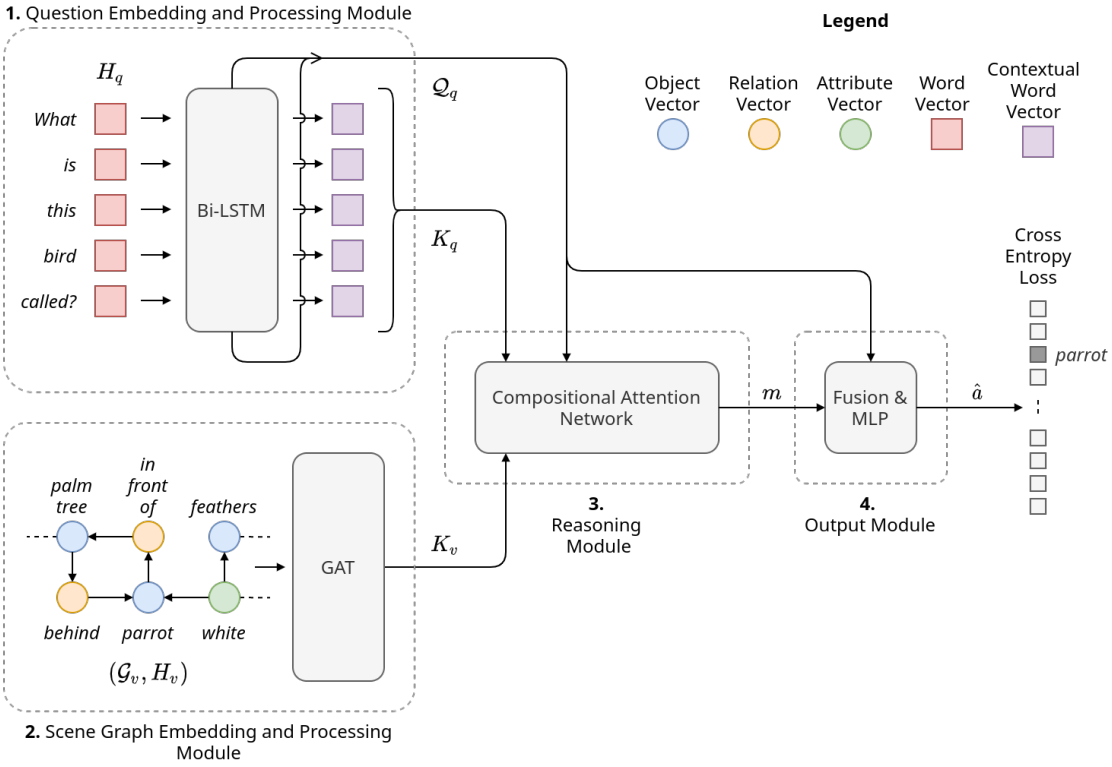


Figure 3.1: A high-level overview of my proposed visual-question-answering model. Both questions and scene graphs are embedded as tensors, then contextual features are obtained by the question and scene graph processing modules. Finally, the reasoning and output modules formulate an answer to the question.

of choice and describe its implementation in detail in Section 3.4.

Output module: The output module is the simplest part of the architecture, and is responsible for transforming the output of the reasoning module into a concrete answer to the input question.

3.2 Question Embedding and Module

Preprocessing and Embedding

The question embedding component of the architecture is responsible for two major tasks: preprocessing and vectorisation. The preprocessing step is performed prior to the training of the model, where vectorisation of preprocessed question words occurs at runtime.

I use a neural preprocessing pipeline for each raw question string in the dataset, comprising tokenisation, part-of-speech (POS) tagging, lemmatisation and dependency parsing steps.[1] For reproducibility, I use the Stanza NLP package [59] for all steps of the pipeline.

After tokenisation, each question q is represented as a sequence of tokens $(t_1, ..., t_{l_q})$, where l_q is the number of tokens in the question. For each question q, each token t is converted to an index to enable vector embedding lookups during training. A mapping of all question tokens to their respective indices is kept, with indices being shared between the training and validation sets. Reference answers are converted to indices in a similar fashion, however they are not tokenised first: multi-word answers like *parking meter* or *cutting board* are treated as a single concept, identically to single-word answers like *bed*. This ensures multi-word answers can be predicted just as easily as single-word answers and allows the output module to treat the question-answering task as a multi-class classification problem across the set of all candidate answers.

At runtime, each question token is converted to a $d_{H_q} = 300$ dimensional GloVe vector [46], then token embeddings for a given question are stacked together, yielding a matrix of question features $H_q \in \mathbb{R}^{l_q \times d_{H_q}}$ where l_q is the number of tokens in the question. For a batch of questions $Q_{[i:i+k]}$ the corresponding features $\{H_{q_i}, ..., H_{q_{i+k-1}}\}$ are padded with zeros along the first dimension and then stacked to yield a batch of question features $H_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{H_q}}$, where $l = \max_{i \leq j < i+k} l_{q_j}$, the number of tokens in the longest question in the batch.[2]

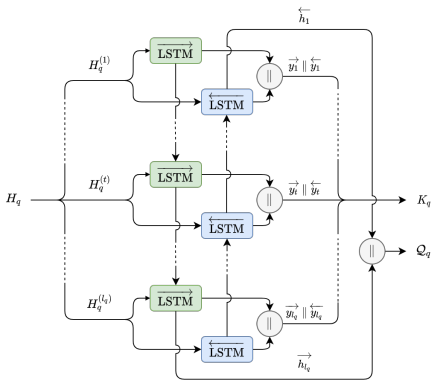
[1] For this architecture, only the tokenisation step is required since the question module uses word embeddings but not syntactic dependency information. Graph-based question modules use syntactic dependencies for graph construction, as described in Subsection 5.2.1.

[2] In practice, $H_{Q_{[i:i+k]}}$ is constructed and batched similarly to $H_{R_{[i:i+k]}}$ as described in Subsection 3.3.2 to allow the training of graph-based question modules, but I avoid graph-based notation here for simplicity.

Question Processing Module

Given a batch of question features $H_{Q_{[i:i+k]}}$ from the question embedding component, the question module is responsible for learning a holistic representation of each question in the batch, as well as capturing the syntactic and contextual dependencies between words in each question. More formally, the question module learns a d_{K_q} dimensional vector for each of the k questions in the batch, denoted $\mathcal{Q}_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times d_{K_q}}$, as well as a knowledge base of contextual question words $K_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{K_q}}$.

A variety of architectures have been used to obtain these representations in VQA contexts, the most common being recurrent models like LSTMs [8] or GRUs [9], used by [3], [13], [14], [38] and [11], [15], [19], [60]–[62] respectively. More recently, others have explored graph-based question processing architectures like GCNs and GATs for text classification [63], [64] and now VQA [20]. After trying a variety of question modules, my initial experiments showed more promising results with a traditional bi-directional LSTM, depicted in Figure 3.2. I compare and analyse the performance of various question modules in Subsection 5.2.1.



$$\overrightarrow{y_t} = \overrightarrow{\text{LSTM}}(H_q^{(t)}) \in \mathbb{R}^{\frac{d_{K_q}}{2}}, 1 \leq t \leq l_q \tag{3.1}$$

$$\overleftarrow{y_t} = \overleftarrow{\text{LSTM}}(H_q^{(t)}) \in \mathbb{R}^{\frac{d_{K_q}}{2}}, 1 \leq t \leq l_q \tag{3.2}$$

$$K_q = \left[\overrightarrow{y_0} \parallel \overleftarrow{y_0}, ..., \overrightarrow{y_{l_q}} \parallel \overleftarrow{y_{l_q}}\right] \in \mathbb{R}^{l_q \times d_{K_q}} \tag{3.3}$$

$$\mathcal{Q}_q = \overrightarrow{h_{l_q}} \parallel \overleftarrow{h_1} \in \mathbb{R}^{d_{K_q}} \tag{3.4}$$

Figure 3.2: An overview of my model's question processing module architecture. For each question q, the question module takes a matrix of question embeddings H_q as an input, and produces a knowledge-base of contextual word information K_q as well as a question embedding vector $\mathcal{Q}_q$.

To obtain $K_{Q_{[i:i+k]}}$ and $\mathcal{Q}_{Q_{[i:i+k]}}$, batched question embeddings $H_{Q_{[i:i+k]}}$ are passed through a single-layer bidirectional LSTM, with each direction having a hidden layer dimension of $\frac{d_{K_q}}{2} = 128$. I refer the reader to Section 5.3 for details on how these parameters were determined. For each question $q \in Q_{[i:i+k]}$, the contextual knowledge-base vectors for each word are the concatenated outputs of the forward and backward LSTMs $\overrightarrow{y_t}$ and $\overleftarrow{y_t}$ at each time step t respectively. In Equation 3.1 and Equation 3.2, I represent the tth word in q as $H_q^{(t)}$, which is used to derive $\overrightarrow{y_t}$ and $\overleftarrow{y_t}$ for each t. Since LSTM weights are shared between time steps, the tth step of the forward LSTM is able to capture contextual information from the previous $t-1$ steps, provided t is not excessively large.[3] The same can be said for the backward LSTM, except it captures contextual information starting from the end of the question, as shown in Figure 3.2. After obtaining $\overrightarrow{y_t}$ and $\overleftarrow{y_t}$ for

[3]In the GQA and CLEVR datasets, the longest questions are in the order of 25-30 words.

each $1 \leq t \leq l_q$, the contextual knowledge-base K_q is built according to Equation 3.3. Similarly to [3], the holistic representation $\mathcal{Q}_q$ of a question q is obtained by concatenating the final hidden states of the forward and backward LSTM passes, since they contain contextual information for all question words in each direction. More formally, the respective forward and backward final hidden states $\overrightarrow{h_{l_q}}$ and $\overleftarrow{h_1}$ are combined according to Equation 3.4 to yield $\mathcal{Q}_q$.

Of course, all questions in the batch are processed simultaneously to yield the entire contextual knowledge base $K_{Q_{[i:i+k]}} = [K_{q_i}, ..., K_{q_{i+k-1}}]$ and question representations $\mathcal{Q}_{Q_{[i:i+k]}} = [\mathcal{Q}_{q_i}, ..., \mathcal{Q}_{q_{i+k-1}}]$. Both $K_{Q_{[i:i+k]}}$ and $\mathcal{Q}_{Q_{[i:i+k]}}$ are inputs to the reasoning module, as described in Section 3.4.

3.3 Scene Graph Embedding and Module

3.3.1 Motivation

The primary goal of the scene graph embedding process is to build a directed graph $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$ and an associated node feature matrix $H_r \in \mathbb{R}^{|\mathcal{V}_r| \times d_{H_r}}$ from a visual representation r. These node features are essential information for the scene graph module, a three-layer Graph Attention Network (GAT) [2] which utilises node-based graph convolutions to capture dependencies and relationships between nodes in the graph. In this dissertation, I intentionally use the scene graphs provided as part of the GQA dataset as a visual input to evaluate the effectiveness of different scene graph modules, as discussed in Section 5.2. Since my primary focus is on scene graphs as a visual input signal, I will consider all general visual representations r to be scene graphs for the remainder of this section.

Both the GQA and CLEVR datasets contain scene graph information for each image in the training and validation sets. Although different datasets store scene graph information differently, the three basic components of a scene graph are objects, relationships and attributes. For example, the CLEVR dataset contains three object types (cube, sphere, cylinder), twelve attribute types covering size (small, large), material (metal, rubber) and colour (red, green, blue, yellow, purple, cyan, brown, grey), and four spatial relationships (left, right, in front, behind). The GQA dataset is more complex, containing, a total of 1703 unique objects, 310 unique relations, and 617 unique attributes across the combined train and validation sets. Adopting common notation from [1], [19], [43], each directed relationship between two objects in a scene graph can be described as a triplet `<subject, relation, object>`, *e.g.* `<parrot, behind, fence>`[4]. For scene graphs containing attributes, we can use a pair `<object, attribute>` to denote attribute associations, *e.g.* `<parrot, white>`. More formally, we can represent a raw scene graph r like so:

[4]Although the `subject` and `object` are both represented by objects in a scene graph, this notation disambiguates the direction of the relationship by drawing upon well-defined grammatical terms, *e.g.* `<parrot, behind, fence>` corresponds to the clause: *The parrot is behind the fence.*

r	A scene graph, $(r_o, r_r, r_a, r_{(s,r,o)}, r_{(o,a)})$
r_o	The sequence of all objects in r, not necessarily unique.
r_r	The set of all unique relations in r
r_a	The set of all unique attributes in r
$r_{(s,r,o)}$	The sequence of (subject, relation, object) triples in r, $((x, y, z) \mid x \in r_o, y \in r_r, z \in r_o)$
$r_{(o,a)}$	The sequence of (object, attribute) pairs in r, $((x, y) \mid x \in r_o, y \in r_a)$

Note that I represent r_o as a sequence and not a set since some scene graphs r may contain multiple instances of the same type of object, *e.g.* an image of a city sidewalk may contain multiple people, each of which have their own attributes and relationships with other objects in the scene.

There are multiple points to consider when converting a general scene graph representation r to its directed graph $\mathcal{G}_r$ and associated node features H_r; my scene graph embedding choice has two main motivations:

1. The scene graph embedding needs to provide adequate information to answer a variety of questions, regardless of whether they ask about an object *e.g. What type of bird is that?*, an attribute *e.g. What colour is the bird?* or a relationship between two objects *e.g. Is the bird behind the fence?*.

2. The goal of the scene graph module is to update the initial object, attribute and relation embeddings created by the scene graph embedding stage to include dependency information. These dependencies are readily captured by existing node-based graph convolutions. However, to utilise these methods, all features that may be relevant to question answering must be represented as node features.

The first point insinuates that objects, attributes and relations should be embedded similarly in the scene graph to aid the reasoning module in its decisions; if we embedded objects differently to relations in the scene graph, then the reasoning module has to learn to respond differently to questions that ask about objects versus those that ask about relationships between objects. Conversely, by embedding relationships, attributes and objects in a similar manner, I hypothesise that the reasoning module is more likely to transfer reasoning skills between similarly-posed but differently-targeted questions.

The second motivation poses an issue: for a scene graph r, we have $|r_o|$ objects, $|r_{(s,r,o)}|$ relations and $|r_a|$ attributes, where all $|r_o|$ objects and $|r_a|$ attributes are represented as nodes in the graph. If we were to embed every relationship between two objects as a node instead of an edge, we would increase the upper bound of nodes in the graph from $O(|r_o| + |r_a|)$ to $O(|r_o| + |r_a| + |r_{(s,r,o)}|) = O(|r_o|^2 + |r_a|)$ nodes,

increasing the computational complexity of any parts of the model that operate on the graph's nodes. Conversely, omitting relational data from node embeddings would reduce time complexity, but may hinder the model's ability to answer questions requiring logical reasoning about relationships between objects. These types of questions account for 438,964 of 943,000 (46.55%) and 61,634 of 132,062 (46.67%) of the training and validation sets respectively in the GQA dataset, motivating the need to capture the type of relationship *e.g. below* or *to the left of* using vector embeddings instead of via edge existence alone. In Subsection 3.3.2, I describe my scene graph construction algorithm, detailing how I capture relationship information as nodes in the graph. In Subsection 3.3.3, I analyse the time complexity of applying node-based convolutions to the resultant scene graph.

3.3.2 Technical details

Preprocessing and Embedding

The preprocessing of objects, relations and attributes is similar to the way answers were processed in the question embedding embedding step; each unique object, attribute or relation is converted to a unique index regardless of whether it is a single-word concept like *parrot* or a multi-word concept like *to the left of*. To simplify graph construction and embedding lookups, I use a combined index set for objects, relations and attributes, computed across the combined train and validation sets S. All unique objects, relations and attributes are converted to indices, with objects, relations and attributes corresponding to indices in the ranges $[0, |S_o|)$, $[|S_o|, |S_o| + |S_r|)$ and $[|S_o| + |S_r|, |S_o| + |S_r| + |S_a|)$ respectively, where $S_o = \bigcup_{r \in R} \{x \mid x \in r_o\}$ is the set of unique scene graph objects in S, $S_r = \bigcup_{r \in R} r_r$ is the set of unique scene graph relations in S and $S_a = \bigcup_{r \in R} r_a$ is the set of unique scene graph attributes in S. By computing the sets of unique objects, attributes and relations separately, I ensure that words which may be used as an object or an attribute *e.g. glass* are given an two indices, one in the object index subset and one in the attribute index subset. Consequently, the scene graph module is able to learn different embeddings for the same word based on the contexts in which they occur in a scene graph.

At runtime, each object, relation and attribute index is associated with a $d_{H_r} = 300$ dimensional GloVe vector. For multi-word objects, relations and attributes, I average the GloVe embeddings for each word to avoid out-of-vocabulary (OOV) issues. All vectors are then stacked together to produce an embedding matrix of scene graph node features H_S spanning the object, relation and attribute vocabulary of the combined train and validation datasets S, *i.e.* $H_S \in \mathbb{R}^{(|S_o|+|S_r|+|S_a|) \times d_{H_r}}$. The first $|S_o|$ rows correspond to object embeddings, the next $|S_r|$ to relations, and the last $|S_a|$ to attributes. These global node features H_S are shared by all scene graphs, and fine-tuned via backpropagation during training. Even if a graph $\mathcal{G}_r$ contains multiple nodes with the same associated index but different neighbourhoods in $\mathcal{G}_r$ *e.g. two palm trees*, gradient signals are propagated back to the corresponding row in H_S for both nodes. This fine-tuning enables the model to update the original GloVe vectors with contextual information regardless of a given question and thus capture co-occurrence information between objects, attributes and relations. Moreover,

embedding fine-tuning is particularly important for multi-word relations like *to the right of* and *to the left of*, where the averaged GloVe vectors of individual words may not capture the meaning of the relation effectively.

After building the embedding matrix H_S, I create an augmented directed graph $\mathcal{G}_r$ from r according to Algorithm 3.1. As the graph is built, I store the corresponding object, relation or attribute index for each vertex. At runtime, the node embedding matrix $H_r \in \mathbb{R}^{|\mathcal{V}_r| \times d_{H_r}}$ associated with $\mathcal{G}_r$ is built by selecting all rows in H_S indexed by each vertex in $\mathcal{V}_r$

Algorithm 3.1: Scene graph construction algorithm

function $r_o, r_r, r_a, r_{(s,r,o)}, r_{(o,a)}$
  $\mathcal{V}_{\text{obj}}, \mathcal{V}_{\text{rel}}, \mathcal{V}_{\text{attr}} \leftarrow \emptyset, \emptyset, \emptyset$
  $\mathcal{E}_{(\text{obj,rel})}, \mathcal{E}_{(\text{rel,obj})}, \mathcal{E}_{(\text{obj,obj})}, \mathcal{E}_{(\text{attr,obj})} \leftarrow \emptyset, \emptyset, \emptyset, \emptyset$
  foreach $x \in r_o$ **do**
    Construct a vertex v_x corresponding to object index x
    $\mathcal{V}_{\text{obj}} \leftarrow \mathcal{V}_{\text{obj}} \cup \{v_x\}$
  end
  foreach $(x, y, z) \in r_{(s,r,o)}$ **do**
    Construct a vertex v_y corresponding to relation index y
    $\mathcal{V}_{\text{rel}} \leftarrow \mathcal{V}_{\text{rel}} \cup \{v_y\}$
    Look up vertices v_x, v_zin $\mathcal{V}_{\text{obj}}$ corresponding to indices x and y respectively.
    $\mathcal{E}_{(\text{obj,rel})} \leftarrow \mathcal{E}_{(\text{obj,rel})} \cup \{(v_x, v_y)\}$
    $\mathcal{E}_{(\text{rel,obj})} \leftarrow \mathcal{E}_{(\text{rel,obj})} \cup \{(v_y, v_z)\}$
    $\mathcal{E}_{(\text{obj,obj})} \leftarrow \mathcal{E}_{(\text{obj,obj})} \cup \{(v_x, v_z)\}$
  end
  foreach $x \in r_a$ **do**
    Construct a vertex v_x corresponding to attribute index x
    $\mathcal{V}_{\text{attr}} \leftarrow \mathcal{V}_{\text{attr}} \cup \{v_x\}$
  end
  foreach $(x, y) \in r_{(o,a)}$ **do**
    Look up vertex $v_x \in \mathcal{V}_{\text{obj}}$ corresponding to object index x.
    Look up vertex $v_y \in \mathcal{V}_{\text{attr}}$ corresponding to attribute index y.
    $\mathcal{E}_{(\text{attr,obj})} \leftarrow \mathcal{E}_{(\text{attr,obj})} \cup \{(v_y, v_x)\}$
  end
  $\mathcal{V}_r \leftarrow \mathcal{V}_{\text{obj}} \sqcup \mathcal{V}_{\text{rel}} \sqcup \mathcal{V}_{\text{attr}}$
  $\mathcal{E}_r \leftarrow \mathcal{E}_{(\text{obj,rel})} \sqcup \mathcal{E}_{(\text{rel,obj})} \sqcup \mathcal{E}_{(\text{obj,obj})} \sqcup \mathcal{E}_{(\text{attr,obj})}$
  return $(\mathcal{V}_r, \mathcal{E}_r)$
end

For clarity, I represent $\mathcal{V}_r$ and $\mathcal{V}_e$ as unions of disjoint subsets in Algorithm 3.1, where each subset of vertices and edges is constructed slightly differently. To gain a better understanding of the different types of vertices and edges in $\mathcal{G}_r$ and the motivations behind their inclusion in the graph, I explore each of these subsets separately, giving some examples from Figure 3.3.

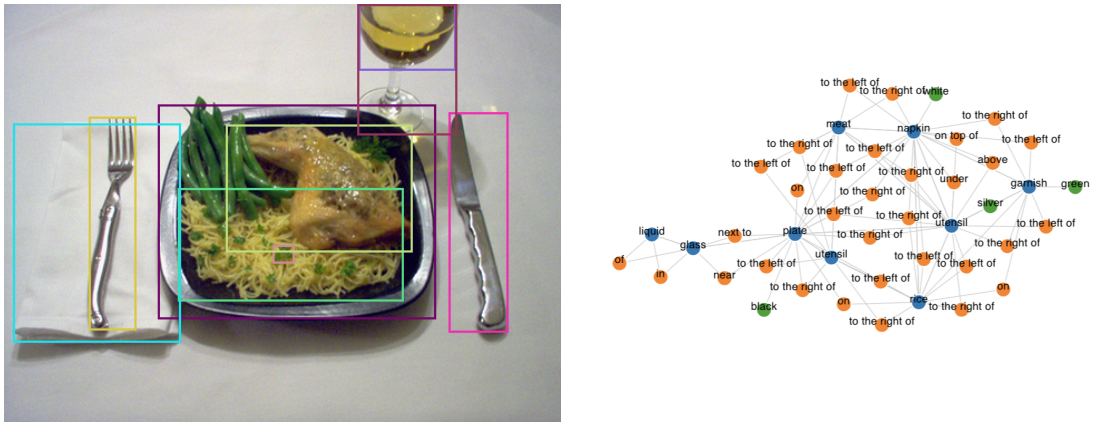


Figure 3.3: A sample from the GQA validation set and its scene graph $\mathcal{G}_r$, constructed according to Algorithm 3.1. Object, relation and attribute nodes are coloured blue, orange and green respectively.

$\mathcal{V}_{\text{obj}}$ contains a node for each object in r_o. Since r_o is a sequence and not a set, $\mathcal{V}_v$ may contain multiple nodes corresponding to objects that have the same type/index but refer to different entities in r. We see this in Figure 3.3, where there are two object nodes labelled *utensil*, one corresponding to a fork and the other to a knife. Moreover, I use the sequence r_o for object node creation over the objects in $r_{(s,r,o)}$ or $r_{(o,a)}$, since r_o may contain objects with no associated relations or attributes. As a result, $\mathcal{G}_r$ often contains multiple connected components.

$\mathcal{V}_{\text{rel}}$ contains a node for each relation in $r_{s,r,o}$. Again, since $r_{s,r,o}$ is a sequence and not a set, $\mathcal{V}_v$ may contain multiple nodes corresponding to relations with the type/index but with differing source and target entities in r. This is clear in Figure 3.3, where we see many instances of common relations like *to the left of* and *to the right of*.

$\mathcal{V}_{\text{attr}}$ contains a node for each attribute in r_a. Since r_a is a set, only a single node is created for each type of attribute. The construction of $\mathcal{E}_{\text{attr,obj}}$ ensures that attribute nodes can only ever have outgoing edges, meaning their embeddings are never updated by graph convolutions in the scene graph module. As a result, there is no need to create multiple nodes for attributes with the same name/index but different target objects.

$\mathcal{E}_{(\text{obj,rel})}$ only contains edges from objects to relations (blue to orange in Figure 3.3), allowing the scene graph module to capture subject-relation interactions. Conversely, $\mathcal{E}_{(\text{rel,obj})}$ contains edges from relations to objects (orange to blue), allowing the scene graph module to learn object representations that include information about incoming relations, as well as the subject of those relations via second-order message passing. $\mathcal{E}_{(\text{obj,obj})}$ contains edges from objects to objects (blue to blue) whenever there is a relation between the two objects. This allows direct message passing between object nodes, which may prove useful in cases where we care less about relationship type and more about whether two objects are related or not *e.g.* for logical or counting-based questions. Lastly, $\mathcal{E}_{(\text{attr,obj})}$ $\mathcal{E}_r$ contains an edge from each attribute to its associated object (green to blue). These edges encourage the

fortification of object representations with attribute information, allowing objects of the same type to be distinguished according to their individual features.

Scene Graph Processing Module

The primary goal of the scene graph processing module is to learn a knowledge-base of scene graph features that captures dependencies between scene graph objects, relationships and attributes. More formally, for each batch of scene graphs $\mathcal{G}_{R_{[i:i+k]}}$ and associated node embeddings $H_{R_{[i:i+k]}} \in \mathbb{R}^{|\mathcal{V}_{R_{[i:i+k]}}| \times d_{H_r}}$ where $|\mathcal{V}_{R_{[i:i+k]}}|$ is the total number of nodes across all graphs in the batch, the scene graph module builds a dependency-aware knowledge-base of visual features $K_{R_{[i:i+k]}} \in \mathbb{R}^{k \times m \times d_{K_r}}$, where $m = \max_{\mathcal{G}_r \in \mathcal{G}_{R_{[i:i+k]}}} |\mathcal{V}_r|$ is the number of nodes in the largest graph in the batch.

The batching procedure for scene graphs and their associated node embeddings is more involved than that of the question processing module, as it needs to allow each graph in the batch to draw its node embeddings from $H_{R_{[i:i+k]}}$, whilst ensuring GAT convolutions on each graph in $\mathcal{G}_{R_{[i:i+k]}}$ don't affect other graphs, all the while supporting sparse tensor operations.

To handle the creation of a batch of scene graphs $\mathcal{G}_{R_{[i:i+k]}}$, I leverage the default batching procedure of PyTorch Geometric (PyG) [65], a GPU-accelerated library for geometric deep learning built on top of PyTorch [66]. The primary idea behind the batching procedure is to construct a block-diagonal matrix $\mathcal{A}$ from the adjacency matrices of each graph $\mathcal{G}_r$ in the batch $\mathcal{G}_{R_{[i:i+k]}}$ and concatenate each node embedding matrix H_r to build a single matrix of node embeddings $\mathcal{H}$ for the entire batch:

$$\mathcal{A} = \begin{bmatrix} A_{r_i} & & \\ & \ddots & \\ & & A_{r_{i+k-1}} \end{bmatrix} \qquad \mathcal{H} = \begin{bmatrix} H_{r_i} \\ \vdots \\ H_{r_{i+k-1}} \end{bmatrix}$$

This allows GAT convolutions to be performed on a batch of graphs as if it were a single large graph with multiple connected components. Since PyG stores adjacency matrices as co-ordinate lists (COO), assembling adjacency matrices as described above simply requires concatenating the co-ordinate lists together, meaning no additional space is used for adjacency matrix storage.

In addition to implementing efficient batching methods for sparse graph data, PyG also contains a variety of node-based convolutional layers including GCNs and GATs, optimised for computation over sparse graphs. Consequently, I use PyG's sparse GAT convolution layer as the core of my scene graph module, as it supports the aforementioned batching procedure by default. Each convolution layer uses the default GAT layer-wise propagation rule defined in Equation 2.2 and attention computation defined in Equation 2.3 with a negative LeakyReLU slope of 0.2. A total of three layers are used, each using $n_h = 4$ attention heads. The weights of the

first layer transform node features from dimension d_{H_r} to d_{K_r}, and subsequent layers have weights of size $d_{K_r} \times d_{K_r}$. For all layers, each head attends to $\frac{d_{K_r}}{n_h}$ input features, and then the results of each head are concatenated as in the original paper [2]. Interestingly, my initial experiments showed that using non-linearities such as LeakyReLU between GAT layers has a negative impact on performance, so I omit the non-linear activation function f shown in Equation 2.2. I report quantitative results to support this decision in Section 5.2.2.

After the batch of graphs is passed through the GAT, we are left with contextual node features for each graph in the batch, stored in a batched matrix $K'_{R_{[i:i+k]}} \in \mathbb{R}^{|\mathcal{V}_{R_{[i:i+k]}}| \times d_{K_r}}$. Whilst the scene graph module is aware of which node features belong to which graph, the reasoning module does not. To address this, I convert $K'_{R_{[i:i+k]}}$ to a dense tensor $K_{R_{[i:i+k]}} \in \mathbb{R}^{k \times m \times d_{K_r}}$, where $m = \max_{\mathcal{G}_r \in \mathcal{G}_{R_{[i:i+k]}}} |\mathcal{V}_r|$ is the number of nodes in the largest graph in the batch. This allows the reasoning module to perform reasoning operations on contextual node features for each scene graph and image using traditional batching methods.

3.3.3 Complexity Analysis

In this subsection, I analyse the time complexity of applying node-based convolutions to two types of scene graphs: the first is a slight modification of the raw scene graph r, where all $|r_o|$ objects and $|r_a|$ attributes are represented as nodes in r, but relations between objects are represented as edges. The second is $\mathcal{G}_r$, the output of Algorithm 3.1 given a raw scene graph r.

Most importantly, I prove that the worst-case computational complexity of propagating $\mathcal{G}_r$ through a single-head GAT convolutional layer is in the order of $\min\{\rho, F\}$ times more expensive than propagating r through the same layer, where F is the layer's input feature dimension and $\rho = \frac{|r_{(s,r,o)}|}{|r_o|}$ is the ratio between the number of relation edges in r and the number of objects r_o.

Although I only use train and validation scene graphs from the GQA dataset, this result provides a useful guarantee for any models that use generated scene graphs. Any scene graph generation process requires determining relationships between pairs of objects and then pruning these candidate edges until the resultant graph is adequately sparse. Since ρ is the ratio between the number of relation edges and the number of objects in r, it is a direct measure of relationship edge sparsity. Hence if we limit relationship edge sparsity in the scene graph generation process to some constant, we can directly control the trade-off between the worst-case computational complexity of GAT convolutional layers and the amount of relationship data that we provide to a model.

According to the analysis of Veličković, Cucurull, Casanova, *et al.*, the time complexity of a single GAT attention head mapping features from dimension F to F' is $O(|\mathcal{V}|FF' + |\mathcal{E}|F')$. A scene graph r by its default definition where we encode objects and their attributes as nodes and relations as edges has $|r_o| + |r_a|$ nodes and $|r_{(s,r,o)}| + |r_{(o,a)}|$ edges. The proposed scene graph construction algorithm represents every relation in $r_{(s,r,o)}$ as a node connected to its subject and

object object nodes. A skip-connection between the subject and object nodes is also added, meaning the resultant graph $\mathcal{G}_r$ has $|r_o| + |r_a| + |r_{(s,r,o)}|$ nodes and $3|r_{(s,r,o)}| + |r_{(o,a)}| = O(|r_{(s,r,o)}| + |r_{(o,a)}|)$ edges.

Since the number of edges in r and $\mathcal{G}_r$ are comparable, we are interested in the conditions in which the additional $|r_{(s,r,o)}|$ nodes in $\mathcal{G}_r$ decrease GAT performance. Let's assume that $|r_{(s,r,o)}| = \rho|r_o|$, for some $0 < \rho \leq \frac{|r_o|-1}{2}$. I choose $\frac{|r_o|-1}{2}$ as an upper bound for ρ, since in the worst case, the subgraph on the nodes represented by r_o is complete, meaning the the number of relations between objects is $|r_{(s,r,o)}| = \binom{|r_o|}{2} = |r_o|\frac{(|r_o|-1)}{2}$.

Given the above assumption, it follows that the number of nodes in $\mathcal{G}_r$ is $(1+\rho)|r_o|+|r_a|$ compared to the $|r_o| + |r_a|$ in nodes in r.

In this case, we have that a GAT layer on r has complexity $O_r((|r_o| + |r_a|)FF' + (\rho|r_o| + |r_{(o,a)}|)F')$ where a GAT layer on $\mathcal{G}_r$ has complexity $O_{\mathcal{G}_r}(((1 + \rho)|r_o| + |r_a|)FF' + (\rho|r_o| + |r_{(o,a)}|)F')$. Since $(1 + \rho)|r_o| > |r_o|$ for all valid values of ρ, a GAT layer on $\mathcal{G}_r$ has the same worst-case complexity as a GAT layer on r when $(1 + \rho)|r_o|FF'$ is not the fastest growing term in $O_{\mathcal{G}_r}$, *i.e.* $(1 + \rho)|r_o|FF' \leq |r_a|FF'$ or $(1 + \rho)|r_o|FF' \leq \rho|r_o|F'$ or $(1 + \rho)|r_o|FF' \leq |r_{(o,a)}|F'$. Noting that $F \geq 1$, with some basic algebra we see this condition applies iff $(1+\rho)|r_o| \leq \max\{|r_a|, \frac{|r_{(o,a)}|}{F}\}$. For the GQA dataset, we see that the number of attributes is usually far less than the number of objects, and even though each attribute usually has around 1-3 associated objects, many objects do not have associated at all. This makes it reasonable to assume for the GQA dataset that $|r_o| > |r_a|$ and $|r_o| > \frac{|r_{(o,a)}|}{F}$. As a result, this condition rarely holds true.

Under these assumptions, we have that a GAT layer on $\mathcal{G}_r$ has complexity $O_{\mathcal{G}_r}((1 + \rho)|r_o|FF') = O_{\mathcal{G}_r}(|r_o|FF'+\rho|r_o|FF')$, and a GAT layer on r has complexity $O_r(|r_o|FF'+\rho|r_o|F')$. We have two separate cases here, depending on the value of ρ:

1. If $0 < \rho \leq 1$, we have that $|r_o|FF' \geq \rho|r_o|FF'$, so $O_{\mathcal{G}_r}(|r_o|FF' + \rho|r_o|FF') = O_{\mathcal{G}_r}(|r_o|FF')$. Moreover, we have that $|r_o|FF' \geq \rho|r_o|F'$ since $F \geq 1$, so $O_r(|r_o|FF' + \rho|r_o|F') = O_r(|r_o|FF')$. Thus, in the worst case, a GAT layer on $\mathcal{G}_r$ has the same complexity as a GAT layer on r. This fits with our definition of ρ; if $0 < \rho \leq 1$, then $|r_{(s,r,o)}| \leq |r_o|$ *i.e.* the number of relations is less than the number of total objects, so worst-case performance for both $\mathcal{G}_r$ and r is determined by the number of object nodes.

2. If $1 < \rho \leq \frac{|r_o|-1}{2}$, then $|r_o|FF' < \rho|r_o|FF'$ and hence $O_{\mathcal{G}_r}((1 + \rho)|r_o|FF') = O_{\mathcal{G}_r}(\rho|r_o|FF')$. If $\rho \leq F$, then we have $O_r(|r_o|FF' + \rho|r_o|F') = O_r(|r_o|FF')$, implying that a GAT layer computation on $\mathcal{G}_r$ is in the order of ρ times slower than a GAT layer computation on r. Alternatively, if $\rho > F$, then $O_r(|r_o|FF'+\rho|r_o|F') = O_r(\rho|r_o|F'))$, making GAT layer computations on $\mathcal{G}_r$ in the order of F times slower than a GAT layer computations on r.

3.4 Reasoning Module

3.4.1 Motivation

Given the goal of the question module is to learn a dense contextual and syntactic representation of the question, and the goal of the scene graph module is to capture dependency information between objects, attributes and relations in the scene, we still need to combine and reason about this extracted information. All VQA models have to handle this multi-modal fusion and reasoning in some capacity: some adopt fusion-only approaches, whilst others learn self-attention or bidirectional attention weights between question and image modalities, and others leverage the strengths of both methods. As discussed in Subsection 2.2.1, the authors of Relation-Aware Graph Attention Networks (ReGAT) use existing VQA models [11], [15], [16] for the multi-modal fusion and reasoning component of their model. I adopt a similar approach for my model, leveraging the reasoning capabilities of the Compositional Attention Network [3], a recurrent model that decomposes complex question-answering problems into discrete reasoning steps, each of which is performed by a single cell in the network. I refer the reader to Subsection 2.2.2 for details on the internal structure of compositional attention networks.

I chose the compositional attention network as my core reasoning module for three primary reasons. It is data-efficient and thus consequently reasonably quick to train, and is well-established as a baseline model on the GQA dataset. Moreover, the compositional attention network is readily extended to any task that requires performing reasoning steps on an input signal, making it a prime candidate for replacing traditional CNN or Faster-RCNN [7] features with scene graph embeddings. Moreover, I postulate that choosing an input signal that is semantically similar to the question signal will allow the compositional attention network to focus more on reasoning computations than multi-modal fusion operations. In addition, better semantic alignment between the control, retrieved information and memory vectors in the network may allow for more effective reasoning computations. It is this reasoning that motivates my choice to use GloVe vectors [46] as the initial embeddings for both scene graph and question vocabulary, as well as my decision to use the compositional attention network as my reasoning module instead of fusion based VQA methods like Bilinear Attention Networks (BAN) [15].

3.4.2 Technical Details

The reasoning module takes three inputs, each of which replace one of the compositional attention network's inputs:

1. A holistic representation of a batch of questions $\mathcal{Q}_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times d_{K_q}}$, from the question processing module.
2. A knowledge base of contextual question words $K_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{K_q}}$, from the question processing module.
3. A knowledge-base of contextual visual features $K_{R_{[i:i+k]}} \in \mathbb{R}^{k \times m \times d_{K_r}}$, the output of the scene graph processing module.

Adopting notation from the original paper, I note that the question knowledge-base $K_q \in \mathbb{R}^{l_q \times d_{K_q}}$ and holistic question representation $\mathcal{Q}_q$ serve as direct replacements for the compositional attention network's contextual word $cw_1, ..., cw_s$ and question representation q inputs respectively.

Where the compositional attention network as implemented for the GQA dataset takes a knowledge-base of o d_{K_o}-dimensional Faster-RCNN object features $K_o \in \mathbb{R}^{o \times d_{K_o}}$ for each image, I provide it with a knowledge-base of contextual scene graph features $K_r \in \mathbb{R}^{m_r \times d_{K_r}}$. As a result, the compositional attention network is able to attend to contextual embeddings of scene graph objects, relations and attributes instead of region-based image features.

The official source code for the compositional attention network [67] was implemented in TensorFlow [68], and is thus incompatible with the PyG. In order to leverage the computational benefits of sparse tensor operations implemented in PyG, I use a PyTorch re-implementation [69] of the compositional attention network, which has been trained to 98.6% on the CLEVR dataset [70], just 0.3% shy of the official result reported by Hudson and Manning. Notably, this re-implementation accounts for minor discrepancies between the model reported in the original compositional attention networks paper and the model implemented in the official source code.

3.5 Output Module

Responsible for determining a final answer to the question, the output module takes the final memory state of the compositional attention network m_{-1} and the batch of question representations provided by the question processing module, $\mathcal{Q}_{Q_{[i:i+k]}}$. The inclusion of the question in the output module is common practice amongst many VQA methods [3], [19], [20], as it allows the output module to incorporate question information into the final answer. Inclusion of question information also serves as a fallback for any questions that ask about information not present in the visual signal, allowing the model to derive an answer based purely on information present in the question.

My output module implementation is the same as the one implemented in the official compositional attention network source code. Since Hudson and Manning's description of the compositional attention network's output module is somewhat sparse in their original paper, I provide a more detailed description of the output module in Figure 3.4.

Notable inclusions over the output module presented in the original publication are the two dropout layers D_0 and D_1, as well as an additional question projection parameterised by W_p and b_p prior to the first dropout layer.

Whilst it is common practice to use a binary cross-entropy (BCE) loss for VQA tasks [60] on datasets like VQAv2 [4] that contain multiple potentially different reference answers for each question, I use a regular multi-class cross-entropy loss since the GQA dataset only contains a single reference answer for each question. This approach is both simple to implement and provides a probability distribution

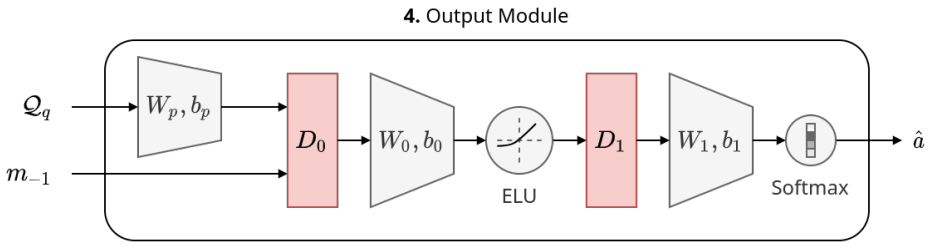


Figure 3.4: An overview of the output module, as implemented in the official compositional attention network source code.

over the answer vocabulary of the dataset.

Chapter 4

Evaluation

In this chapter, I introduce the setup I used for evaluating my visual question answering (VQA) model architecture. I describe important details about the GQA dataset [1], analysing its composition and highlighting important points to consider when evaluating model performance. Next, I introduce a variety of baseline and state-of-the-art VQA models, summarising their primary features. I present results for these VQA architectures in Section 5.1. Finally, I provide a detailed description of the hyperparameters of each of the components in my VQA architecture to aid the reproduction of my results.

4.1 Dataset

Whilst there are plenty of available datasets for visual question answering, I focus on just one for performance evaluation in this dissertation. I use the GQA dataset to evaluate the effectiveness of my VQA model architecture for three primary reasons. Firstly, it the only dataset that supports additional visual reasoning metrics such as validity, plausibility and distribution, as detailed in Subsection 2.3.3. Secondly, the GQA dataset is large in sample and vocabulary size and contains a balanced subset which deters models from exploiting statistical priors in question types and answer distributions, as described in Subsection 2.1.2. This makes it a suitable benchmark for evaluating the reasoning capabilities of VQA models when combined with the compositional nature of its questions. Finally, the GQA dataset contains scene graph annotations for its training and validation sets. Whilst other datasets contain scene graph annotations, these scene graphs are either too limited in their object, attribute and relationship vocabulary (*e.g.* CLEVR [5] with 3 unique objects, 5 unique relations and 12 unique attributes), or their object, attribute and relationship vocabulary is extremely large (*e.g.* Visual Genome (VG) [31] with 75,729 unique objects, 40,480 unique relationships and 40,513 unique attributes). Instead of manually normalising the vocabulary of VG scene graphs and using VG to evaluate my model architecture, it is wiser to leverage GQA scene graphs, which are derived from VG scene graphs and comprise an already normalised vocabulary of 1740 unique objects, 620 unique attributes and 330 unique relations.

In total, the GQA dataset comprises 22M question-image pairs, with 113M unique images. These question-image pairs are distributed across the training, validation, testing, development testing (test-dev) and challenge splits, ensuring that all question-image pairs with the same image are included in the same split. In total, the full training set contains around 77.1% of all question-image pairs, the validation set contains 10.8%, the test set 7.2%, test-dev set 0.9% and challenge set 3.8%. The balanced portion of the GQA is much smaller, as many questions were omitted to smooth question type and answer distributions as detailed in Subsection 2.1.2. As a result, the balanced training, validation, test, test-dev and challenge sets contain 943,000, 132,062, 95,336, 12,578 and 50,726 question-image pairs respectively. Most importantly, the GQA dataset contains publicly-available scene graph annotations for the full train and validation sets, but not for the test, test-dev or challenge sets. Consequently, I train models on the balanced training set and report results on the GQA validation set instead of the test or test-dev sets. For ablation studies in Section 5.2, I train models on the balanced training set, tune the my model architecture on the first half of the balanced validation, and report results on the second half of the balanced validation set.

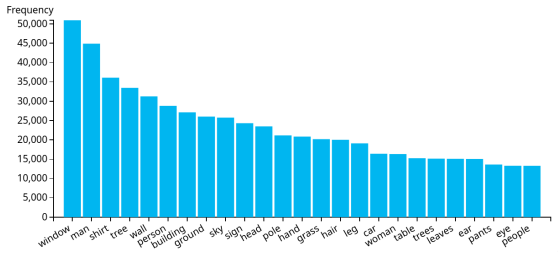
The full GQA dataset has a total vocabulary size of 3,097 words, and an answer vocabulary of 1,878. The scene graphs (including those not available to the public) span a total of 1740 unique objects, 620 unique attributes and 330 unique relations. On average, the scene graph for each image contains 16-17 different objects, possibly with the same label. Additionally, each object is is related to an average of 3 other objects, and is associated with an average of 0.5 attributes. In reality, we typically see multiple attributes for more salient objects in the image, and none for less prominent objects.

Moreover, more salient objects tend to be more common in GQA scene graphs overall, since the original VG scene graphs were annotated by humans, whose eyes are naturally drawn towards more salient objects in images. As illustrated in Figure 4.1, large objects like *window, man, tree, building, etc.* tend to be more common than less salient objects or parts of objects like *ear* and *eye*.

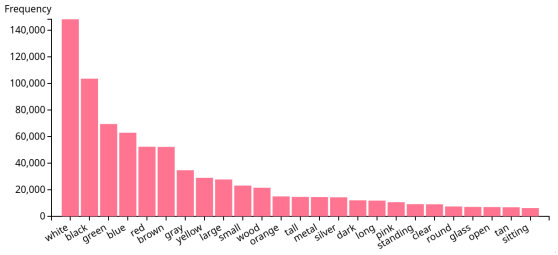
The majority of the most common attributes found in GQA scene graphs are colours. Some size-related attributes such as *large, small, tall* and *long* appear frequently as well, in addition to material-related attributes like *wood* and *glass*.

The most common scene graph relations in the GQA dataset are *to the left of* and *to the right of*, and have been excluded from the list of top relations to avoid dwarfing other relation types. Spatial relations like *on, near, in, in front of, behind, next to, etc.* are the most common, but semantic relations containing verbs like *holding, sitting, standing, carrying, riding* also comprise a large percentage of total relation types.

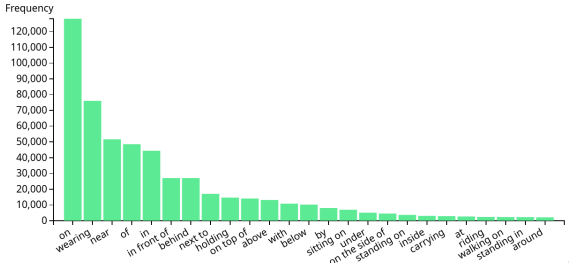
Despite the large variation in object, attribute and relation frequencies between different types displayed in Figure 4.1, it is essential to note that the GQA dataset's balancing process accounts for both the question type and the subject of the question as described in Section 2.1. For example, in the full GQA dataset, it might be reasonable to assume that questions about the material of a *table* will have the



(a) Top 25 GQA objects.



(b) Top 25 GQA attributes.



(c) Top 25 GQA relations, excluding left and right.

Figure 4.1: An overview of the objects, attributes and relations present in the GQA scene graphs, sorted by frequency. Figures are sourced from the GQA dataset website [71].

answer *wood* more often than less common materials like *metal* or *glass*. To create the balanced GQA subset, a subset of all questions about the material of a *table* are chosen so that less common materials are more common in the answer distribution. As a result, even though attributes like *metal* or *glass* are less common than *wood* in GQA scene graphs as a whole, the questions that ask about these attributes are more balanced in their answer distributions. This encourages VQA models to learn to identify and reason about less common scene graph objects, attributes and relations, and is one of the primary reasons why I choose to train my VQA model on the balanced portion of the GQA dataset instead of the full, unbalanced dataset.

To aid the evaluation of a VQA model's reasoning capabilities on different types of questions, every question in the GQA dataset is labelled as either *open* or *binary*, and is annotated with a semantic and a structural type. There are five structural types (query, compare, choose, logical and verify), and five semantic types (global, object, attribute, relation and category). I provide some example questions with various structural and semantic type combinations in Table 4.1.

As seen in Table 4.1, binary question types can always be answered with either *yes* or *no*, where *open* question types require answering with one of the 1,878 concepts in the GQA answer vocabulary. Naturally, I would expect that most models would perform better on binary questions than open questions.

Inspecting structural types, we see that the *query* and *choose* structural types only

Type	Structural	Semantic	Example Question
Open	Query	Global	How is the weather in the image?
Open	Query	Attribute	What color is the apple?
Open	Query	Relation	What is the small girl wearing?
Open	Query	Category	What kind of fruit is on the table?
Open	Choose	Attribute	Is the apple green or red?
Open	Choose	Relation	Is the cat to the left or to the right of the flower?
Open	Choose	Relation	What is the boy eating, an apple or a slice of pizza?
Open	Choose	Category	What kind of fruit is it, an apple or a banana?
Open	Compare	Object	What is common to the shirt and the flower?
Binary	Verify	Global	Is it cloudy today?
Binary	Verify	Object	Is there an apple in the picture?
Binary	Verify	Attribute	Is the apple red?
Binary	Verify	Relation	Is she wearing a blue dress?
Binary	Logical	Object	Do you see either an apple or a banana there?
Binary	Logical	Attribute	Is the apple red and shiny?
Binary	Logical	Object, Attribute	Do you see both green apples and bananas there?
Binary	Compare	Object	Does the shirt and the flower have the same color?

Table 4.1: A selection of questions from the GQA dataset and their corresponding semantic and structural types, adapted from the GQA supplementary material [72].

apply to *open* questions, as they require answering with an object, relation or attribute from the image, rather than with *yes* or *no*. The *compare* structural type is applicable to both binary and open questions, depending on question phrasing; open comparisons usually require reasoning about what is the same or different between two objects, where binary comparisons only require identifying if a certain property is the same or different. Finally, the *verify* and *logical* structural types apply to binary questions, as they aim to determine whether a specific combination of object, relationships and/or attributes exists.

The semantic type of a question is related to its subject; questions belonging to the *global* semantic type ask about properties of the image as a whole instead of individual objects, attributes and relations. Conversely, questions belonging to the *object*, *attribute* or *relation* semantic type naturally query about object, attributes and/or relations in the scene. Interestingly, the *category* semantic type is only applicable to *open*-type questions, as they require the a VQA model to identify that an object is part of a certain category, *e.g.* that an *apple* is a type of fruit. Converting the question *What kind of fruit is on the table?* to a binary question would result in a question like *Is the apple on the table a kind of fruit?*, which doesn't require a visual signal to answer effectively.

I delve into the semantic and structural question type distributions for the portion of the GQA dataset used for result collection in Section 5.2, as it is important to consider when comparing model performance on different types of questions.

4.2 Baselines

To gain a better understanding of how well my model architecture performs on the GQA dataset, I compare it to a variety of current baseline and state-of-the-art methods in Chapter 5. To understand why some models perform better than others, I introduce these baseline VQA models in this section, summarising their key components and highlighting the type of visual data that they use for the VQA task.

LXMERT [73]: Motivated by the success of transformer-based models [12] for Natural Language Processing (NLP) tasks, Tan and Bansal built LXMERT for learning general multi-modal representations. In the context of VQA, the LXMERT model uses two self-attention transformer encoders and one cross-modality transformer encoder. The first self-attention encoder is responsible for attending to Faster R-CNN [7] and bounding box features, and the second second is a language encoder, which attends to word features. The outputs of these encoders are fed to the cross-modality encoder, which employs a bi-directional attention flow between the visual and textual encodings to align the two modalities, followed by separate self-attention layers to allow the model to attend to any newly-discovered important features after multi-modal alignment.

Bilinear Attention Networks (BAN) [15]: In the context of VQA, co-attention methods capture which parts of the question are important given the visual signal as a whole, and vice versa. Bilinear attention methods capture which parts of the question are important given each feature of the visual signal and vice-versa, making them much more expressive, but computationally expensive. Kim, Jun, and Zhang build upon low-rank bilinear modelling techniques [74], [75] to approximate bilinear interactions between textual and visual input signals, using learned bilinear attention maps to obtain dense joint embeddings of textual and visual data.

To show the effectiveness of these joint representations, the authors collect multiple bilinear attention maps *(glimpses)* and combine their respective joint features using residual learning techniques [76]. In their final model, the authors use a matrix of Faster R-CNN object features as a visual input signal, and Gated Recurrent Unit (GRU) [9] outputs to represent the question. This model achieved state-of-the-art results on the VQAv2 dataset in 2018.

Graph Reasoning Networks (GRN) [77]: GRNs improve on the BAN architecture by capturing relationships between BAN's default joint question-image representations and previously captured joint representations. Given a visual representation and a question representation, GRNs first compute a vector of joint question-image features, as in BAN. These multi-modal features are then projected back on to the question to yield an output representation for the layer, in an effort to determine which joint question-object representations are relevant to the original question. The authors repeat this process multiple times in an effort to promote multi-step reasoning; with each repetition, instead of using the original question representation for computing joint representations, subsequent joint features are computed using the previous layer's output.

Compositional Attention Networks (MAC) [3]: As introduced in Subsection 2.2.2, the compositional attention network is a widely accepted VQA baseline for the GQA dataset. Its recurrent cell design promotes decomposing a question into discrete reasoning steps, where two values are passed from cell to cell: The first is a control vector, responsible for determining which part of the question to attend to for a given reasoning step. The second is a memory vector, which contains the intermediate result of the current reasoning process. As originally implemented for the GQA dataset, the compositional attention network relies on a BiLSTM-encoded question representation and a knowledge-base of Faster R-CNN [7] object features as its visual signal, however its knowledge-base is easily replaced with scene graph features as discussed in Section 5.1.

Mutual and Self Attention (MSA) [78]: The three primary inputs to the MSA model are a question representation q, a visual representation v and a collection of semantic relationships r. From these inputs, the MSA module computes eight attention distributions which it fuses together, four from q and r, and four from q and v. Taking q and r as an example, the MSA module computes the following:

- The attention distribution over q, guided by a fused representation of q and r, called *mutual attention*. This captures which parts of the question are important to the combined question and relationship features.
- The attention distribution over r, guided by a fused representation of q and r. This captures which relationship features are important to the combined question and relationship embeddings.
- The attention distribution over q, guided by self-attended question features $\psi_{q,q}$. This emphasises the most important parts of the question.
- The attention distribution over r, guided by a linear transformation of $\psi_{r,q}$, where $\psi_{r,q}$ is a multi-head attention distribution over r, guided by q. This emphasises relationship features that are deemed important to answering the question.

The authors achieve promising results with this setup using various combinations of features for r and v, with their best model representing using Faster R-CNN and bounding box features in its visual signal v and word embeddings for subject, object and relationship labels as the foundation of its semantic relationship signal r. Moreover, the authors evaluate their model using pre-annotated scene graph features from the GQA dataset instead of generated features.

4.3 Implementation Details

In this section, I provide relevant details about the implementation of my architecture, from its vocabulary construction through to its parameter configuration and optimisation method. I refer the reader to Chapter 3 for model-specific nomenclature used in this section.

I build a question vocabulary of 2,912 case-sensitive words, and a scene graph vo-

cabulary of $|S_o| = 1703$ unique objects, $|S_a| = 617$ unique attributes and $|S_r| = 310$ unique relations using the combined train and validation questions and scene graph annotations respectively. Moreover, the output module of my architecture predicts an answer from one of 1,842 possible answers in the combined balanced train and validation sets, slightly less than the 1,878 answers in the full GQA set.

In both the question and scene graph embedding modules, I use $d_{H_q} = d_{H_r} = 300$-dimensional GloVe vectors [46] to embed all words. In the event where a question token or scene graph object, attribute or relation comprises multiple words *e.g. to the right of*, I represent that token as an average of the GloVe embeddings of its individual words.

The question processing module is a single-layer bi-directional LSTM [8] with input dimension $d_{H_q} = 300$. Each direction of the BiLSTM has a hidden dimension of $\frac{d_{K_q}}{2}$, where $d_{K_q} = 256$ is the dimension of each token in the question knowledge-base K_q. To avoid processing the vectors used for padding and consequently ruining question embeddings, I use PyTorch's inbuilt `pack_squence` function. [66]

The scene graph module is a three-layer graph attention network (GAT), with all attention computations as described in the original GAT publication [2]. The first layer has an imput dimension of $d_{H_r} = 300$ and an output dimension of $d_{K_r} = 256$, and subsequent layers have input and output dimensions of $d_{K_r} = 256$. Each layer has $n_h = 4$ attention heads, the outputs of which are concatenated after each layer to maintain a constant layer size. No non-linear activation function is used to transform the outputs of each layer.

The compositional attention network used for the reasoning module has a length of 4, and a hidden dimension of 512 as described in the original paper [3]. As in the original implementation, I use ELU activations over ReLU in the control, read, write units, and all dropout layer probabilities are set to 0.15.

The output module is implemented according to the original implementation of the compositional attention network, again using ELU over ReLU for activation functions and dropout probabilities of 0.15. Naturally, the last linear layer of the output module has a dimension of 1,842, the same as the answer vocabulary size for the combined balanced GQA train and validation sets.

For optimisation, I use a standard categorical cross-entropy loss with an Adam optimiser [79]. The optimiser uses standard exponential decay rates of $\beta_1 = 0.9, \beta_2 = 0.999$ and an epsilon of $\varepsilon = 10^{-8}$. I clip all gradients with a ℓ_2 norm greater than 8, as used by Hudson and Manning for their compositional attention network. All models are trained for 32 epochs with a batch size of 16 at a constant learning rate of 4.7611×10^{-5} (*5 s.f.*).

Chapter 5

Results and Discussion

In this chapter, I evaluate the performance of my architecture on the GQA dataset using a combination of traditional accuracy measures and GQA-specific metrics. First, I compare my architecture to the collection of baseline VQA methods introduced in the previous chapter. Secondly, I present results for variations of my model, replacing or entirely removing its question and/or scene graph processing modules to justify my architecture design. Finally, I delve into the parameter configuration of my model, exploring how parameters such as learning rate, reasoning module length and scene graph module size affect the performance of my model and the methods I used to find my final parameter configuration.

5.1 Performance Evaluation

As demonstrated in Table 5.1, my model outperforms all baseline models by a significant margin across all accuracy metrics, and performs on par or slightly better than reported models for GQA validity, plausibility and distribution. Moreover, it outperforms human baseline performance for all accuracy measures.

For fairness of comparison, I split baseline model performance into two sections; all models in the upper section use only Faster R-CNN and/or bounding box information during evaluation time, where all models in the lower section have access to GQA scene graph object, attribute and relationship data.

Stepping through each of the baselines in the upper section of Table 5.1, it becomes evident how important the type of visual signal is for VQA models; we see that my model outperforms all four baselines that only have access to object and bounding box features by anywhere from 30.5% in the case of LXMERT [73] to 24.37% in the case of MSA [78] when comparing overall accuracy. Since I use pre-annotated scene graphs from the GQA dataset as my model's visual signal, it makes sense that my model outperforms existing baselines that don't have access to this information. Moreover, we see that whilst my model outperforms MSA by 24.37% in overall accuracy, the difference between the two models closes to only 9.24% for binary-type questions, but grows to almost 50% for open-type questions. Moreover, we see

Model	GQA					
	Accuracy	Binary	Open	Validity	Plausibility	Distribution
Human [1]	89.30	91.20	87.40	98.90	97.20	-
LXMERT [73], [80]	59.80	-	-	-	-	-
BAN-4 [15], [77]	61.95	-	-	-	-	-
GRN [77]	64.22	-	-	-	-	-
MSA[78]	65.93	82.35	49.27	94.98	91.57	4.88
MSA† [78]	68.71	71.84	68.71	94.94	92.99	7.29
MAC† [3]	73.64	81.86	65.94	**95.38**	92.46	2.05
MSA‡ [78]	81.15	85.06	77.48	**95.34**	94.26	1.08
Our Model†	**90.30**	**91.59**	**89.10**	**95.35**	**94.48**	**0.29**

Table 5.1: A comparison of the performance of various models on the GQA validation set. Human performance is based on majority vote of 5 human responses for 4000 random GQA questions. Models marked with a † have access to pre-annotated GQA scene graph objects, attributes and relations at inference time, and models marked with a ‡ use Faster R-CNN [7] features and/or bounding box information in addition to scene graph information. Where two citations are provided, the first corresponds to the original paper and the second corresponds to the source of the validation set results.

that my model has a much lower distribution than MSA, meaning it relies less on statistical priors in the dataset when forming its answers, and more on its visual input. Again, this is easily explained by the advantage of pre-annotated scene graph information. Interestingly, the difference in performance between my model and MSA on the validity and plausibility metrics is much less pronounced, at only 0.37% and 2.91% respectively. This is due to two primary reasons:

- Validity and plausibility are much more forgiving metrics than accuracy. Validity is the most forgiving, since a model only has to provide an answer of the correct type *e.g. provide a colour when the question about the colour of an object*. Plausibility is less forgiving than validity but still more forgiving than accuracy, since the provided answer has to make sense given the context of the question and image. As a result, the difference between validity for the two models is the smallest, followed by plausibility and then finally accuracy.
- Providing a valid or plausible answer does not always require visual information. We see this phenomenon in the results on the GQA test set in the original GQA publication [1], where a language only LSTM model provides valid answers between 0.37% and 0.21% more often than the CNN + LSTM, BottomUp [11] and MAC [3] baselines, and provides plausible answers between 2.73% and 3.05% more often. Consequently, despite having a much weaker visual signal than my model, the MSA model in the upper section of Table 5.1 still performs comparably on the validity and plausibility metrics.

These observations extend to models in the lower section of Table 5.1, which perform almost identically to my model on the validity metric, and between 0.2-2.0% worse

than my model for answer plausibility.

Moving our focus to the lower section of Table 5.1, we see that the baseline models perform much better than those in the upper section, as they also have GQA scene graph information at their disposal. Importantly, we see that my model is still over 9.1% more accurate than MSA despite not using any Faster R-CNN object features. When we remove Faster R-CNN features from MSA, we see a drop of almost 13%, making my model almost 22.6% more accurate than MSA given the same underlying information. Moreover, a MAC network that uses the same question encoding method as my model and uses GloVe embeddings for scene graph object, attributes and relations is still outperformed by my model by almost 16.7%, highlighting the importance of the scene graph embedding and processing module in my final model.

Examining accuracy for binary and open question types, we see that while all three models that use scene graph information have a significantly lower (between 3.1-7.5%) open-type accuracy compared to their binary-type accuracy, my model answers binary and open questions almost equally well, with only a 2.49% difference between the two.

Based on the results in Table 5.1, I believe the primary contributor to my model's excellent performance is its scene graph embedding and processing method. All three models in the lower half of the table use semantic word embeddings to represent scene graph objects, relations and attributes, and use attention-based reasoning methods. Consequently, the main point of differentiation between my model and the others is the use of graph attention networks (GATs) to process scene graph information. I confirm this hypothesis in Subsection 5.2.2, where I implement multiple scene graph processing module alternatives and demonstrate the effectiveness of GATs for learning semantically-rich scene graph embeddings.

5.2 Ablation Studies

To gain a more complete understanding of how the components of my model contribute to its performance, I explore how removing or replacing certain parts of my model change its behaviour. More specifically, I report how changes to the question processing module and scene graph processing module affect the performance of my model, investigating performance at a macro level using the metrics seen in the previous section, as well as at a micro level, examining model accuracy as a function of question length, question complexity and question structural and semantic types.

For all ablation studies in this chapter, I train models on the balanced GQA training set and split the balanced GQA validation set in half, reserving the first half for parameter optimisation and the second half for testing. Consequently, all results reported in this section are computed on the second half of the balanced GQA validation set, instead of the full validation set as in Table 5.1, to avoid giving an unfair advantage to my best model, which was tuned using the first half of the balanced validation set.

Before delving into ablation study results, it is important to understand the underlying distributions of the test set used to compute various metrics. As demonstrated in Table 5.2 and Table 5.3, the second half of the GQA validation set has a very similar semantic and structural type distribution to the rest of the balanced GQA dataset, as reported in [72].

Examining the distribution of semantic question types, we see that 46.6% of questions query about the relationships between objects, 31.8% ask about attributes, 11.8% about objects, 6.6% about object categories and 3.1% about global properties. Regarding structural types, 51.7% of questions are open-type questions that query about scene properties in general, 20.7% of questions require verifying some property about the scene, 12.3% require logical reasoning operations, 12.2% require choosing between a set of options provided in the question and 3.1% require comparing the properties of two or more objects.

Semantic Type	**Count**	**Proportion**
Relation	30,742	46.6%
Attribute	21,019	31.8%
Object	7,812	11.8%
Category	4,387	6.6%
Global	2,071	3.1%

Table 5.2: Distribution of semantic question types across the second half of the balanced GQA validation set.

Structural Type	**Count**	**Proportion**
Query	34,171	51.7%
Verify	13,688	20.7%
Logical	8,103	12.3%
Choose	8,050	12.2%
Compare	2,019	3.1%

Table 5.3: Distribution of structural question types across the second half of the balanced GQA validation set.

Figure 5.1 illustrates the compositional nature of the questions in the GQA dataset, with the majority of questions requiring 2-3 discrete reasoning steps to arrive at an answer. We also see that most questions contain around 5 to 11 words, however there are still a large number of longer questions that a good VQA model needs to be able to interpret.

5.2.1 Question Module Ablations

In this Subsection, I explore three alternative question processing modules, namely a convolutional neural network (CNN) inspired by work on sentence classification in

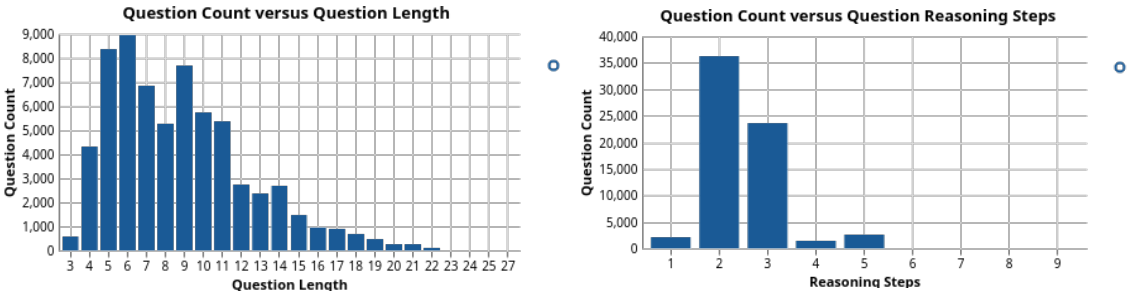


Figure 5.1: Distribution of questions across the second half of the balanced GQA validation set for both question length and reasoning step count.

the NLP field [81], a graph convolutional network (GCN) that uses synactic dependencies as edges, and a graph attention network (GAT) that operates in a similar fashion to the GCN but learn individual weights for each syntactic dependency. Moreover, I investigate the effects of removing the question processing module entirely, relying purely on GloVe embeddings as a question input signal. Before delving into results, I provide a brief overview of how each of these ablations fit in to the rest of my model architecture, adopting notation from Chapter 3.

No question processing module: For this ablation, I remove the scene graph module entirely, so the knowledge-base of contextual question words $K_{Q_{[i:i+k]}}$ is simply equal to the embedding feature matrix $H_{Q_{[i:i+k]}}$. To obtain holistic question representations $\mathcal{Q}_{Q_{[i:i+k]}}$, I simply average the embeddings of word features across the question length dimension of $H_{Q_{[i:i+k]}}$.

CNN: Modeled on the architecture presented in [81], the CNN question processing module takes a batch of raw question embeddings $H_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{H_q}}$ as its input and applies multiple 2D convolutions of varying receptive fields in parallel to learn a contextual representation of each question. I use a total of four Conv2D layers, each with 128 output channels and kernel sizes of $(3 \times d_{H_q})$, $(5 \times d_{H_q})$, $(7 \times d_{H_q})$ and $(9 \times d_{H_q})$. Appropriate padding is applied for each Conv2D layer so that convolved features for each layer are always of size $l \times 128$. The convolved features for each layer are concatenated together to yield $K_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times 512}$, the knowledge-base of contextual words used as an input to the reasoning module. After obtaining convolved features for each question in the batch, I take the maximum value of each output channel across the question length dimension to obtain a single vector $\mathcal{Q}_q$ of dimension 512, representing the question as a whole. When processed as a batch, this yields a matrix of question representations $\mathcal{Q}_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times 512}$.

GCN & GAT: As mentioned in Section 3.2, I retrieve syntactic dependencies for each question using the Stanza NLP package [59] during the question preprocessing stage. These syntactic dependencies form the edges of a question graph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q)$, and the words of the question form the nodes. The type of syntactic dependency between individual words is ignored, as GCNs and GATs perform convolutions on nodes and not edges. Moreover, I ensure that $\mathcal{G}_q$ is undirected, allowing messages to propagate through the dependency tree in both directions. To use this graph with

GCN or GAT models, we need a node embedding matrix $H_q \in \mathbb{R}^{l_q \times d_{H_q}}$ for each question q in the batch. These embeddings are obtained using the same question embedding technique described in Section 3.2, but are batched differently. Instead of creating a dense batch of question embeddings $H_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{H_q}}$, I use the default PyTorch Geometric (PyG) [65] batching procedure, as described in subsubsection 3.3.2. Consequently, the output of the question embedding module is a batch of graphs $\mathcal{G}_{Q_{[i:i+k]}}$ and associated batch of node embeddings, ready for input straight into a PyG GAT or GCN. Both the GAT and GCN question processing modules comprise 3 layers, with the first having an input size $d_{H_q} = 300$ and the rest having input and output sizes of $d_{K_q} = 256$. The GAT has the same head count and structure as the scene graph GAT module. For both the GAT and GCN, the final layer's output is converted to a dense batch of features and used as the question knowledge-base $K_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times l \times d_{K_q}}$. To obtain a batch of holistic question features, I use a global mean pooling operation on the final layer's output. These pooled features are converted to a dense batch of question embeddings $\mathcal{Q}_{Q_{[i:i+k]}} \in \mathbb{R}^{k \times d_{K_q}}$. Both $K_{Q_{[i:i+k]}}$ and $\mathcal{Q}_{Q_{[i:i+k]}}$ are then passed to the reasoning module as usual.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Binary	Open	Validity	Plausibility	Distribution
None	GAT	86.52	85.78	87.21	95.10	93.85	0.26
CNN	GAT	87.35	87.87	86.86	95.24	94.22	0.29
GCN	GAT	86.02	86.77	85.32	95.05	93.92	0.34
GAT	GAT	89.43	**91.88**	87.16	95.26	94.19	0.27
BiLSTM	GAT	**90.45**	91.73	**89.26**	**95.34**	**94.48**	**0.19**

Table 5.4: An overview of the impact of different question processing modules on overall model performance.

As seen in Table 5.4, the BiLSTM question processing module outperforms all other question processing module types for total accuracy, open-type accuracy, validity, plausibility and distribution, and falls just shy (0.15%) of the GAT question processing module on the binary accuracy metric. This is a testament to the expressive power of recurrent models for short sentences; the BiLSTM can capture relationships between all words in a given question regardless of its syntax, as it process each word in the question sequentially. In comparison, the effectiveness of the GAT and GCN question modules is highly dependent on the number of convolutional layers that they contain. Since each GAT or GCN layer can only propagate information to immediate neighbours of each node, the number of GAT or GCN layers must be equal or greater than the depth of the question's syntactic dependency tree to ensure that information can be shared between each word pair in the question. Whilst the CNN question module is capable of capturing contextual information about the question as a whole, it makes the assumption that related words occur near each other in the sentence, which is not always the case, especially for longer sentences. Since the CNN module was originally used for sentence classification [81], I expect it learns useful holistic question representations $\mathcal{Q}$, but struggles to capture relevant contextual information for individual words. This also explains why the CNN

module performs slightly (0.83%) better than the model variant with no question module at all, which simply takes the average of all embeddings for each word in the question as its holistic question representation $\mathcal{Q}$.

In addition to the success of the BiLSTM question module, we also see that the relative ranking of the question modules is similar across all metrics, with BiLSTM performing the best, followed by the GAT module, then CNN, None and finally GCN. Whlst the GCN question module outperforms the model with no question processing module on the binary accuracy metric, it struggles to answer open-type questions. This is likely because it always propagates features between syntactically-connected question words, regardless of whether these syntactical relationships are important to the question or not. As a result, the expressive power of the GloVe embeddings initially used to embed each word is lost due to unwanted propagations between question words. On the other hand, the GAT question processing module outperforms both the GCN module and the bare GloVe embedding variation across all metrics, due to its ability to learn attention weights between question words. This helps it to learn better contextual word-level embeddings, which are essential for obtaining useful control vectors in the reasoning module.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Query	Compare	Choose	Logical	Verify
None	GAT	86.52	87.21	80.44	78.89	93.77	85.88
CNN	GAT	87.35	86.86	67.66	85.99	92.57	89.17
GCN	GAT	86.02	85.32	73.60	84.96	92.85	86.17
GAT	GAT	89.43	87.16	**93.02**	86.92	96.43	91.93
BiLSTM	GAT	**90.45**	**89.26**	69.69	**90.87**	**97.45**	**92.10**

Table 5.5: An overview of the impact of different question processing modules on model accuracy for questions of differing structural types.

Moving our focus to Table 5.5, we see that the performance of each question module variation on query-type questions coincides with their open-type question accuracy. Moreover, the performance of each question module on verify-type questions is within 1.5% of their total binary-type question accuracy. Consequently, the BiLSTM question module outperforms other question processing module types for most structural types, with the exception of *compare* type questions, where the GAT question module outperforms other question types by anywhere from 12.58% to 25.36%. Whilst it may be tempting to attribute the large decrease in performance for most question modules on compare-type questions to the fact that it is the least common structural question type at only 3.1% of samples, we cannot ignore that the GAT question module performs so well. Referring back to Table 4.1 in the previous chapter, we see that comparison questions can be either binary or open, and most often ask about common or different properties of two objects. After inspecting the control unit attention weights for the GAT question module and the BiLSTM question module, we see that the GAT question module attends to entire phrases of the question like *have the same colour*, where the BiLSTM question module attends to

words like *do* or *is*, indicating it is relying on question priors information to formulate its answers. I provide an example of this in Appendix A, Section A.1, showing the regions of the question and scene graph that the model attends to.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Global	Object	Attribute	Relation	Category
None	GAT	86.52	**83.97**	94.75	87.35	83.67	89.01
CNN	GAT	87.35	83.78	93.63	85.37	87.37	87.14
GCN	GAT	86.02	83.39	93.36	83.65	86.32	83.43
GAT	GAT	89.43	83.82	97.24	**89.74**	87.88	87.62
BiLSTM	GAT	**90.45**	83.92	**97.39**	88.39	**90.51**	**90.65**

Table 5.6: An overview of the impact of different question processing modules on model accuracy for questions of differing semantic types.

Examining Table 5.6, we see that all question modules perform comparably for questions with global targets. This is primarily due to the fact that there is no global scene information included in the visual signal, so it is difficult to answer questions about global scene properties regardless of the quality of the question representation. This also explains why the global accuracy is much lower than the overall accuracy for all question module types.

For questions that query about objects and/or attributes, we see that the GAT and BiLSTM question modules perform the best, as they are able to learn good contextual representations of individual question words, unlike CNN or GCN which are better at embedding the question representation as a whole. Additionally, we see a difference of around 2.6% between object-type accuracy of the BiLSTM question module and the model with no question module at all, indicating that some object-type questions require contextual question information, for example, they query about a *red apple* as opposed to just an *apple*.

For relationship-based questions, we observe again that the BiLSTM performs the best, outperforming the GAT and CNN and GCN question modules by 2.63%, 3.14% and 4.19% respectively. This is primarily due to its ability to capture sequences of words where other question modules can't. This is essential for relations in the GQA dataset, as the two most common relationships are *to the left of* and *to the right of*, which the question module needs to be able to capture as single concepts. Furthermore, when we don't use any question module at all, we suffer a 6.84% performance drop when compared to the BiLSTM question module, since the raw GloVe embeddings don't contain any contextual information.

Finally, for category-based questions, we see that GCN, CNN and GAT question modules all perform worse than not using a question module at all by anywhere from 1.39% in the case of GAT through to 5.58% in the case of the GCN question module. In general, category-type questions require associating a category with an object, *e.g. fruit* with *banana*. As a result, it is essential that the category is represented adequately and not clouded by other parts of the question so the model

can learn to search for related concepts in the scene graph.

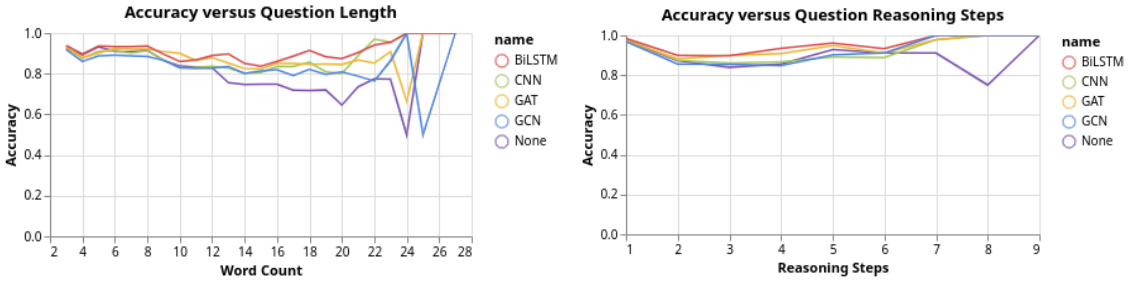


Figure 5.2: Per-question-length and per-reasoning-step accuracy for various question processing module types. The high performance of the right-hand tails in both cases are due to the small number of questions with a large number of words or reasoning steps, as illustrated in Figure 5.1.

In Figure 5.2, we see that both GAT and BiLSTM question modules perform similarly as the number of reasoning steps required by the question increases, clearly outperforming alternative question modules. In addition, we see that as question length increases, question modules that use mean or max pooling to obtain a holistic representation tend to decrease in performance faster than the BiLSTM question module, which uses its concatenated forward and backward hidden layers as a question representation. When using no question module at all, performance decreases drastically as question length increases, since the mean of the question words use to represent the question becomes a less reliable embedding of the question as a whole.

5.2.2 Scene Graph Module Ablations

In this subsection, I investigate two alternative scene graph processing modules, namely a BiLSTM and a graph convolutional network (GCN). In addition, I inspect how removing the scene graph processing module affects overall performance, comparing results for these different architectures across a variety of metrics. Again, I provide a brief overview of how each of these ablations interact with rest of my model architecture, adopting relevant notation from Chapter 3.

No scene graph processing module: For this ablation, I remove the scene graph processing module entirely. After constructing a scene graph $\mathcal{G}_{R_{[i:i+k]}}$ and associated node feature matrix $H_{R_{[i:i+k]}}$ in the scene graph embedding module, the node feature tensor is passed directly to the reasoning module. That is, I use the node feature tensor $H_{R_{[i:i+k]}}$ as the knowledge-base of scene graph objects, attributes and relations $K_{R_{[i:i+k]}}$, completely disregarding any edge information on $\mathcal{G}_{R_{[i:i+k]}}$. Investigating the effect of edge information removal is essential for understanding how much of my model performance stems from the embedding of scene graph object, attribute and relation features versus how much is due to the contextual graph information learned by GAT layers.

BiLSTM: The BiLSTM scene graph processing module aims to capture contextual

relationships between scene graph object, attribute and relation triples, without any structured graph-based information. Instead of building a matrix of node embeddings containing scene graph objects, attributes and relations, like in the regular scene graph embedding module, I create an input feature matrix $H_r \in \mathbb{R}^{m \times d_{H_r}}$ for each scene graph r as follows: For each of the m_r objects in the scene graph r, the ith row of H_r is the concatenation of three vectors u_i, v_i and w_i, where u_i the ith object's GloVe embedding, v_i is the mean of the GloVe embeddings of the ith object's incoming and outgoing relations, and w_i is the mean of the GloVe embeddings of all attributes associated with object i. In the event that an object contains no incoming or outgoing relations, or no attributes, I set v_i or w_i to a 300-dimensional zero vector respectively. All feature matrices are stacked to form a single tensor $H_{R_{[i:i+k]}} = [H_{r_i}, ..., H_{r_{i+k-1}}]$, which is in turn fed to a single-layer BiLSTM. Since each direction of the BiLSTM has a hidden size of 256, the concatenated last layer outputs of each direction form 512-dimensional vector for each of the m_r $d_{H_r} = 900$ dimensional input vectors, denoted K_r. These matrices are stacked to yield a knowledge-base of scene graph features for all graphs in the batch, denoted $K_{R_{[i:i+k]}} \in \mathbb{R}^{k \times m \times d_{K_r}}$, where $d_{K_r} = 512$ and m is the maximum number in any single scene graph in the batch. $K_{R_{[i:i+k]}} \in \mathbb{R}^{k \times m \times d_{K_r}}$ is then passed to the reasoning module as usual.

GCN: The GCN scene graph processing module is identical in terms of inputs and outputs as the GAT scene graph processing module, except it uses GCN convolution operations instead of GAT convolutions. Since GCNs have no attention heads, each GCN layer operates on all features of the input instead of a quarter of the input each in the case of GAT with four attention heads.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Binary	Open	Validity	Plausibility	Distribution
BiLSTM	None	73.63	81.84	65.98	**95.38**	92.46	1.09
BiLSTM	BiLSTM	83.35	85.81	81.06	95.27	93.73	0.31
BiLSTM	GCN	69.44	78.14	61.34	95.13	91.95	1.11
BiLSTM	GAT	**90.45**	**91.73**	**89.26**	**95.34**	**94.48**	**0.19**

Table 5.7: An overview of the impact of different scene graph processing modules on overall model performance.

The importance of an effective scene graph processing module is clearly outlined in Table 5.7, where we see just how successful the GAT architecture is at capturing contextual scene graph information. The GAT scene graph processing module outperforms all other processing modules on all three accuracy measures, as well as for plausibility and distribution. We see similar comparable results for validity regardless of the scene graph module type, for the same reasons introduced in Section 5.1.

We see improvements almost 17% on overall accuracy, almost 10% on binary-type accuracy and over 23% on open-type by processing scene graph features with a

GAT prior to the reasoning module, and increase the plausibility of answers by just over 2%. Without the GAT to process scene graph edges, the reasoning module simply receives a list of all objects, attributes and relations in the scene and is not provided with any context about which attributes and relationships are associated with which objects. Consequently, these large performance gains are solely due to GAT's ability to implicitly capture scene graph edge information and embed the interactions between object, attribute and relation nodes prior to performing reasoning operations.

The BiLSTM scene graph processing module achieves the second-highest performance across all metrics, falling just over 7% shy of the GAT processing module on the overall accuracy metric, but ourperforming both the GCN scene graph module and no scene graph module model variants. This is primarily a result of explicitly concatenating an object's relation and attribute information with the embedding of the object itself, as this allows the reasoning module to answer questions about the attributes and relationships of a particular object. Naturally, this method of expressing object, attribute and relationship data is far from optimal, but it performs surprisingly well despite its naivety.

Finally, we see that the GCN scene graph processing module performs extremely poorly. Not using a scene graph processing module at all results in an almost 4.2% overall increase in accuracy and a 0.5% increase in answer plausibility. These results show that even though the GCN knows which node embeddings are related, it is ineffective at capturing those relationships as node features due to its inability to attend to some relationships more than others.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Query	Compare	Choose	Logical	Verify
BiLSTM	None	73.63	65.98	69.44	72.84	92.85	82.44
BiLSTM	BiLSTM	83.35	81.06	**92.22**	73.98	95.79	85.91
BiLSTM	GCN	69.44	61.34	66.37	71.04	85.96	79.43
BiLSTM	GAT	**90.45**	**89.26**	69.69	**90.87**	**97.45**	**92.10**

Table 5.8: An overview of the impact of different scene graph processing modules on model accuracy for questions of differing structural types.

Progressing to Table 5.8, we see that the relative rankings of each of the scene graph modules for query, choose, logical and verify question types are the same as for overall accuracy, with GAT outperforming BiLSTM, and GCN performing worse than raw GloVe embeddings. We see that the query-type accuracy for all four modules is identical to the open-type accuracy presented in Table 5.7, and the verify-type accuracy is similar to the binary-type accuracy. We see a 5.6% performance increase on choose-type questions when using a GAT to process node features instead of using the features directly, but more interestingly, we see a minimal increase in performance by using the BiLSTM scene graph processing module over raw node features despite the BiLSTM module having access to the concatenated object, attribute and relation features that caused a 9.7% increase on overall accuracy. This

likely occurs because the concatenated features only capture first-order interactions between scene graph objects, relations and attributes; given the relationship *(man, to the left of, dog)*, the BiLSTM scene graph processing module knows that the man is to the left of something, and the dog has something to the left of it, but there is no way of knowing that the man is related to the dog. Since most choose-type questions require choosing between two objects based on attributes or relationships, we see that the BiLSTM module only sees a slight 1.1% performance gain over not using a scene graph processing module at all.

For compare-type questions, the GAT and GCN scene graph modules perform poorly at 69.69% and 66.37% respectively, where the BiLSTM scene graph module performs outstandingly well. Since the BiLSTM module uses concatenated object, attribute and relation features, compare-type questions about attributes like *Are the shirt and the flower the same colour?* are much easier to answer, since the reasoning module only has to identify the relevant objects by their embeddings and then compare the attribute portion of the concatenated vector. For other scene graph processing types, the reasoning module has to first locate the objects, and then use precious reasoning steps to query their colours before comparing them to retrieve an answer.

Question Module	**Scene Graph Module**	**GQA**					
		Accuracy	Global	Object	Attribute	Relation	Category
BiLSTM	None	73.63	**85.18**	95.15	64.46	72.38	82.54
BiLSTM	BiLSTM	83.35	82.86	95.60	80.88	81.02	89.92
BiLSTM	GCN	69.44	70.45	87.54	67.22	66.25	69.77
BiLSTM	GAT	**90.45**	83.92	**97.39**	**88.39**	**90.51**	**90.65**

Table 5.9: An overview of the impact of different scene graph processing modules on model accuracy for questions of differing semantic types.

In Table 5.9, we see that the GAT scene graph processing module outperforms alternative scene graph processing methods for all question semantic types except questions about global scene information. Interestingly, when using no scene graph processing module, we observe a 1.26% increase in global question-type accuracy over the GAT scene graph module. This indicates that raw GloVe embeddings provide a useful signal for establishing global scene properties such as weather and location; for example, a scene that contains embeddings for *sky*, *cloud* and *umbrella* objects is likely outdoors.

For questions about objects, we see that the GAT scene graph module outperforms the GCN module by almost 10% indicating the importance of attention weights in establishing which object, attribute and relationship interactions are important. Moreover, the GAT module outperforms the BiLSTM module and raw GloVe features by almost 1.7% and 2.2% respectively, indicating the benefits of enriching object features with attribute and relationship information. We see that the GCN module performs poorly again, at 21.17% and 24.26% worse than the GAT module for questions about attributes and relations respectively. As expected, the absence of a scene graph processing module makes it difficult to associate relation or attribute

features with an object, explaining the poor attribute-type and relation-type accuracies when using only GloVe embeddings. As discussed previously, the BiLSTM scene graph module is able to capture first order interactions between objects and their attributes and/or relations, but struggles to capture second-order interactions, explaining the 7.51% and 9.49% performance decrease for questions about attributes and relations when compared to the GAT module.

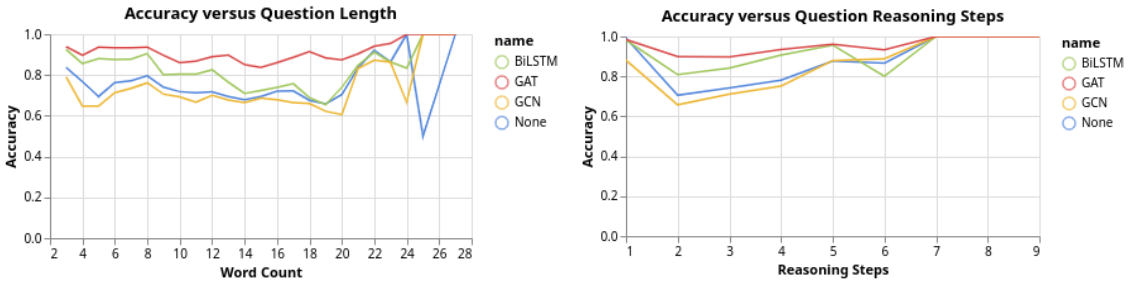


Figure 5.3: Per-question-length and per-reasoning-step accuracy for various scene graph processing module types. The high performance of the right-hand tails in both cases are due to the small number of questions with a large number of words or reasoning steps, as illustrated in Figure 5.1.

Figure 5.3 demonstrates the response of different scene graph processing modules to question length and complexity. As expected, the GAT module outperforms the other modules, and maintains reasonable performance as question length and reasoning step count increases. Moreover, we see that the GCN consistently decreases the performance of the model as a whole, regardless of question length or complexity. The BiLSTM module performs well for smaller question lengths and smaller reasoning step counts, but its performance rapidly decreases as the question word count increases. This is primarily due to its inability to capture interactions between objects as previously discussed, as these interactions are more common in longer questions.

5.3 Hyperparameter Optimisation

In this Section, I briefly introduce the parameter optimisation process that I used to arrive at the final set of model parameters described in Subsection 4.3. Prior to parameter optimisation, I performed extensive initial tests; when combined with the results reported in this dissertation, a total of 160 days of GPU computation time was used for training and evaluating models.

Due to the large number of parameters in my model, I used a bayesian optimisation process, using the hyperband [82] early stopping method. All experiments were tracked using *Weights & Biases* [83], so I used the *Weights & Biases* implementation of bayesian optimisation, using hyperband with $\eta = 2$ and the first cutoff iteration set to 3. Consequently, trained models were automatically stopped if they performed worse than half of the existing trained models.

The effects of the Hyperband method are clearly seen in Figure 5.5, where only a

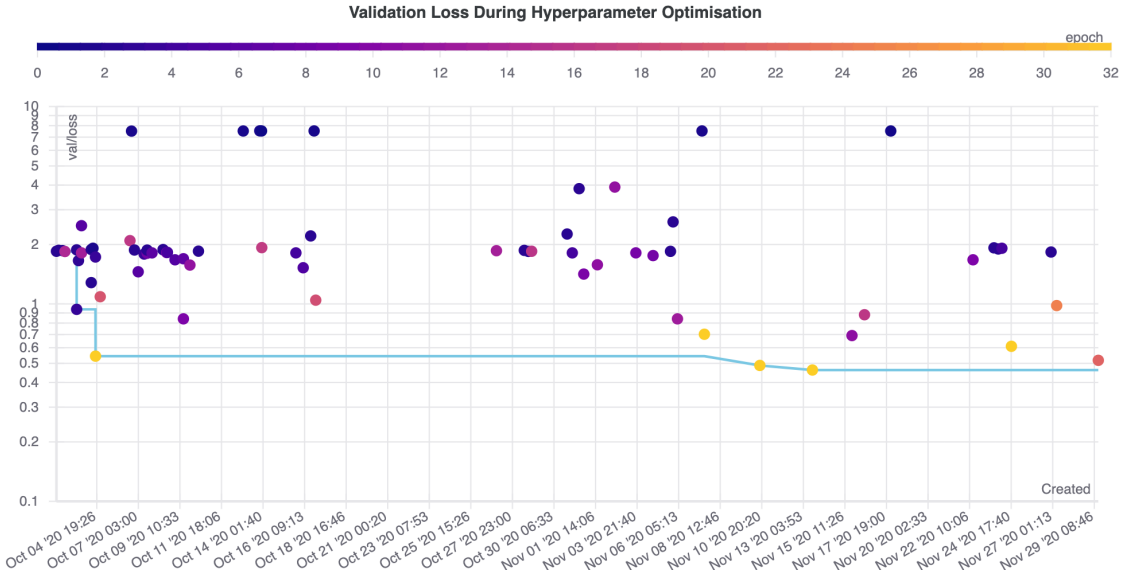
Figure 5.4: Hyperparameter importance with respect to validation loss. Correlation is the linear correlation between each parameter and the validation loss, between -1 and 1. Parameter importance a value between 0 and 1, estimated by *Weights & Biases* [83] by training a random forest classifier using configuration parameters as features and the validation loss as the target prediction.

few models are trained to completion and less promising models are stopped earlier to encourage exploration of the hyperparameter search space.

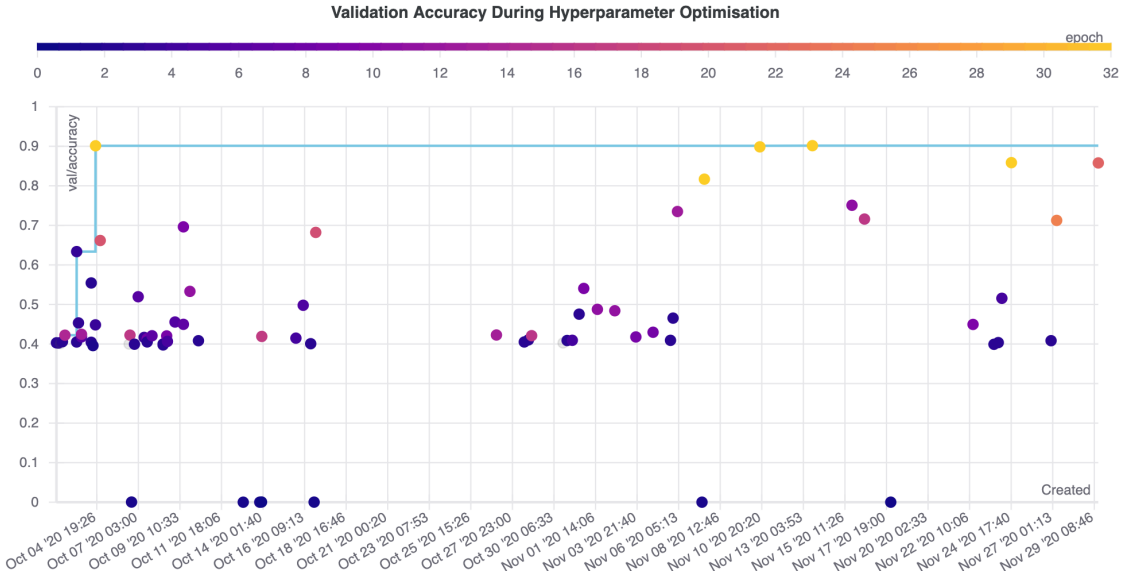
I optimised a total of 10 different parameters during the parameter optimisation process, as illustrated in 5.4; As expected, the learning process was extremely sensitive to the learning rate, due to the large amount of attention computations in the model. In general, I found that more attention heads for the scene graph module helped stabilise the training, as illstrated by its negative correlation with validation loss as illustrated in Figure 5.4.

For more details on the hyperparameter optimisation process, I refer the reader to my model's source code. [1]

[1] `https://github.com/alexmirrington/gat-vqa`



(a) Validation loss throughout the hyperparameter optimisation process.



(b) Validation accuracy throughout the hyperparameter optimisation process. Greyed out points correspond to models with a loss greater than 10.

Figure 5.5: A summary of validation loss and accuracy throughout the hyperparameter optimisation process. Each point represents a trained model, and its colour indicates how many epochs the model was trained for.

Chapter 6

Analysis and Discussion

In this chapter, I illuminate how my model arrives at an answer for a given question and scene graph. First, I illustrate how the scene graph processing module propagates data between object, attribute and relation nodes in the scene graph. Next I highlight the which question words the reasoning module attends to at each recurrent time-step, illustrating how complex questions are decomposed into smaller, discrete reasoning steps. Finally, I demonstrate which scene graph features the reasoning module attends to at each time-step, demonstrating that my model effectively extracts relevant scene graph information at each reasoning step. In addition, I present a variety of samples from the GQA dataset that my model failed to answer correctly as an error analysis exercise, with the goal of examining its failure modes and highlighting areas for future research.

6.1 Correctly Predicted Samples

6.1.1 Logical Questions

As discussed in the previous chapter, my model architecture performs particularly well for questions that involve logical reasoning, achieving a logical question-type accuracy of 97.45%, compared to an overall accuracy of 90.45%. In this subsection, I delve into how my model answers these logical questions effectively, using the question-image pair in Figure 6.1 as an example.

The example in Figure 6.1 is a canonical example of the type of logical reasoning required my many questions in the GQA dataset. To answer the question correctly, the model ideally requires four reasoning steps: one reasoning step is required for locating the dog in the image, one step should determine whether the dog is small, one should determine if the dog is brown, and the last reasoning step should combine the second and third reasoning steps to determine if the dog is both small and brown.

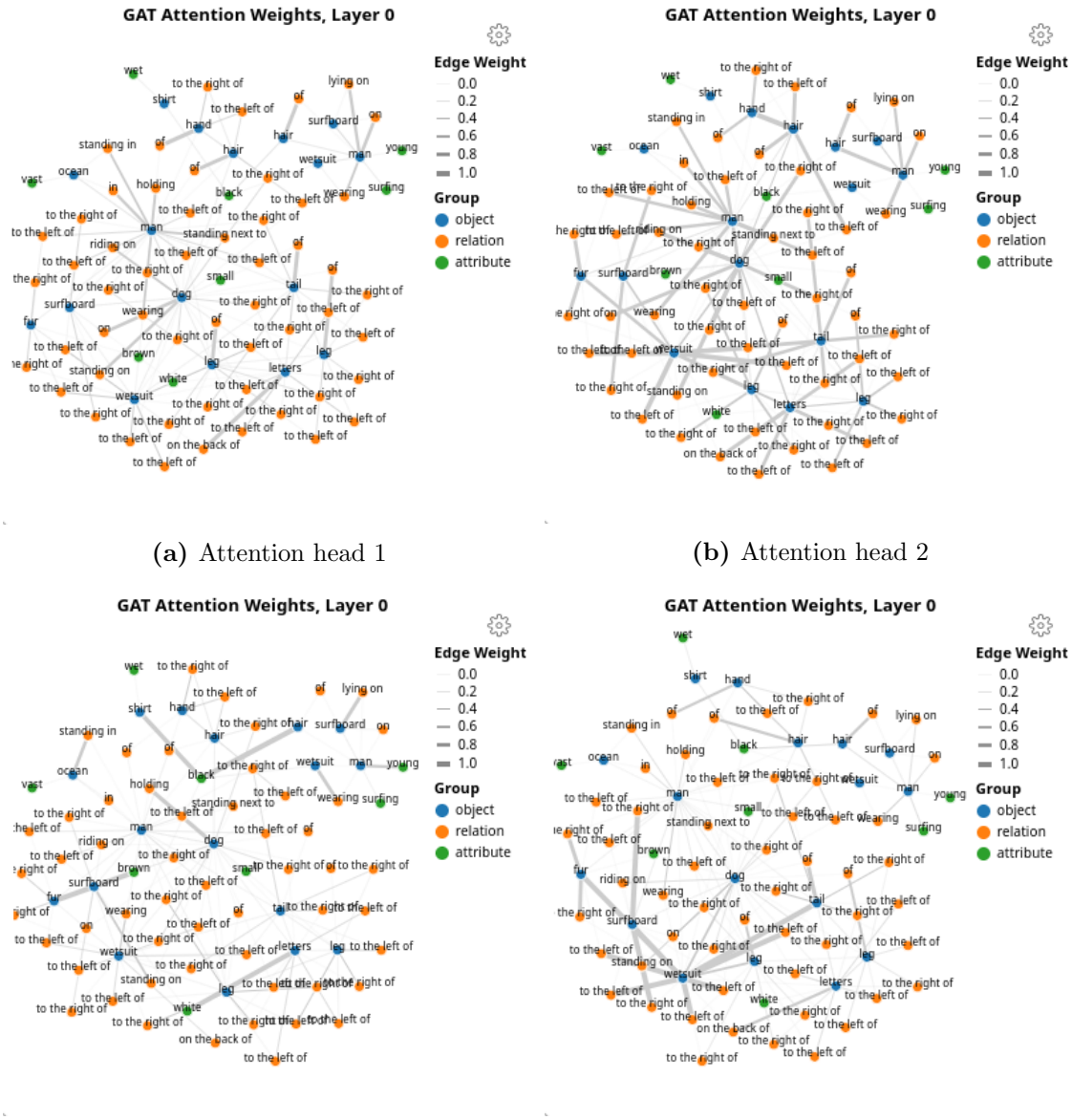
For this first example, I delve into how the learned GAT attention weights for each of the three scene graph processing GAT layers, for all four attention heads. The learned weights for the first, second and third GAT layers are visualised in Figure

Figure 6.1: A GQA sample that requires logical reasoning to answer correctly.
Q: *Does the dog look small and brown?* **A:** *Yes*

6.2, Figure 6.3 and Figure 6.4 respectively. In Figure 6.2, we see that the first attention head focuses on the outgoing edges for the primary objects in the scene, *e.g. man, man* and *dog*. The second attention head also focuses on edges that point from object nodes to relation nodes, attending to a wider range of objects than the first attention head. The third attention head focuses on attribute-to-object edges, and enriches object embeddings with useful attribute information *e.g. hair* $\rightarrow$ *black hair* and *fur* $\rightarrow$ *brown fur*. The fourth attention head performs a similar task to the first, updating relation node embeddings with object information. After propagating the entire scene graph through the first GAT layer, the new node features now capture combinations of attributes and objects, as well as combinations of objects and relations.

The second GAT layer takes the features learned by the first layer, and captures second order interactions. The first and second attention heads attend to a combination of object-object edges, object-relation edges and attribute-object edges, as illustrated in Figure 6.3. Since the source nodes for most of the attended edges were not target nodes in many of the first layer attention heads, most of these interactions still capture first-order relationships between nodes in the graph. The third and fourth attention heads in the second GAT layer capture primarily object-object and object-relation edges, however some of their attended edges contain objects like *surfboard* and *hair*, which were enriched with attribute information by the first GAT layer. Consequently, propagating object information from the *surfboard* or *hair* nodes to other relation nodes results in the capturing of second-order interactions, like *black hair of*. It is clear to see by now that the multiple heads across the two GAT layers are working together to capture complex semantic features.

Continuing on, we see that the last GAT layer begins to capture relation-object edges, propagating information from relation nodes back to object nodes, as demonstrated in Figure 6.4. At this point, many relation nodes already store second-order



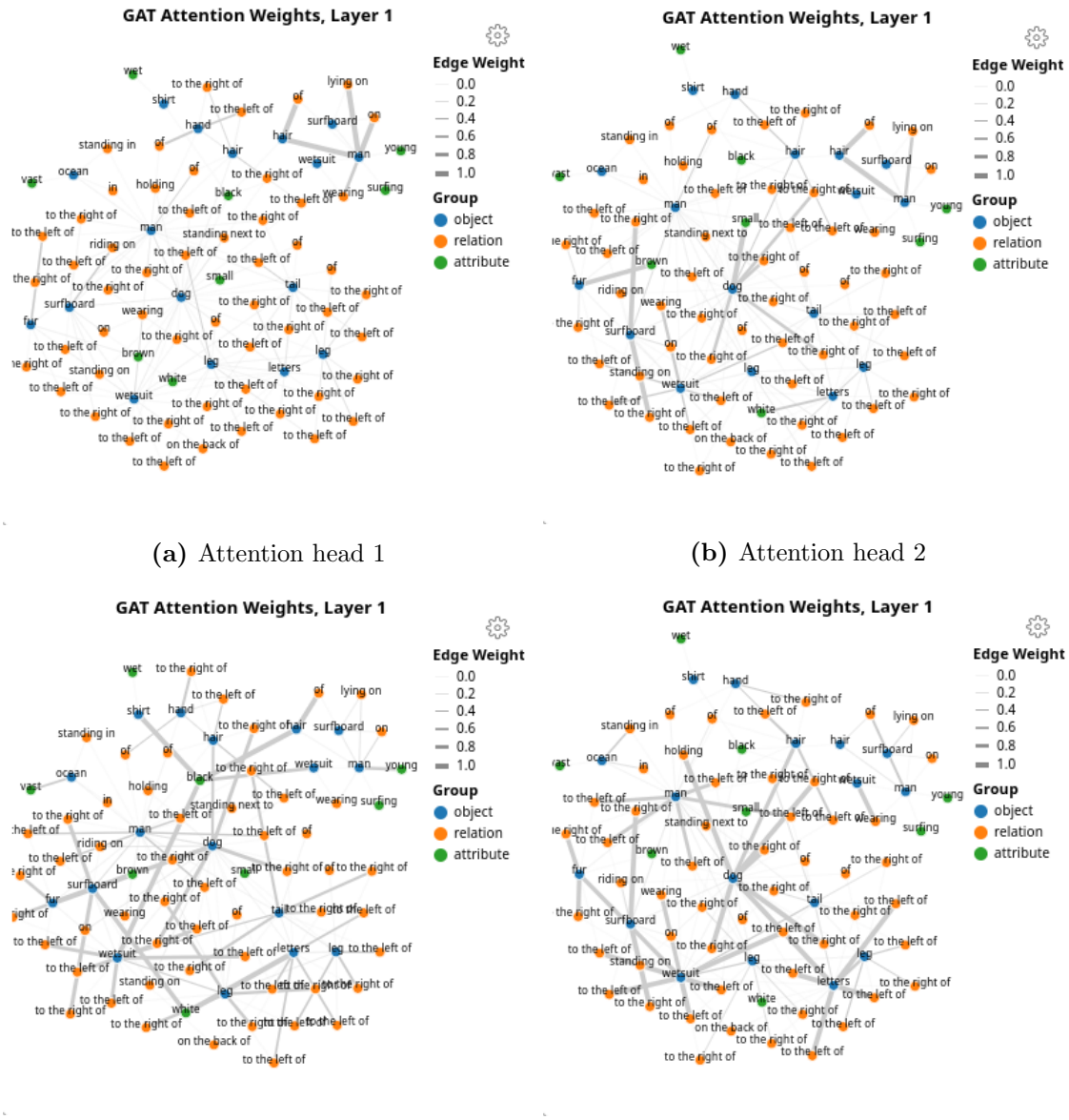
(a) Attention head 1

(b) Attention head 2

(c) Attention head 3

(d) Attention head 4

Figure 6.2: Scene graph processing module attention weights for the first GAT layer.



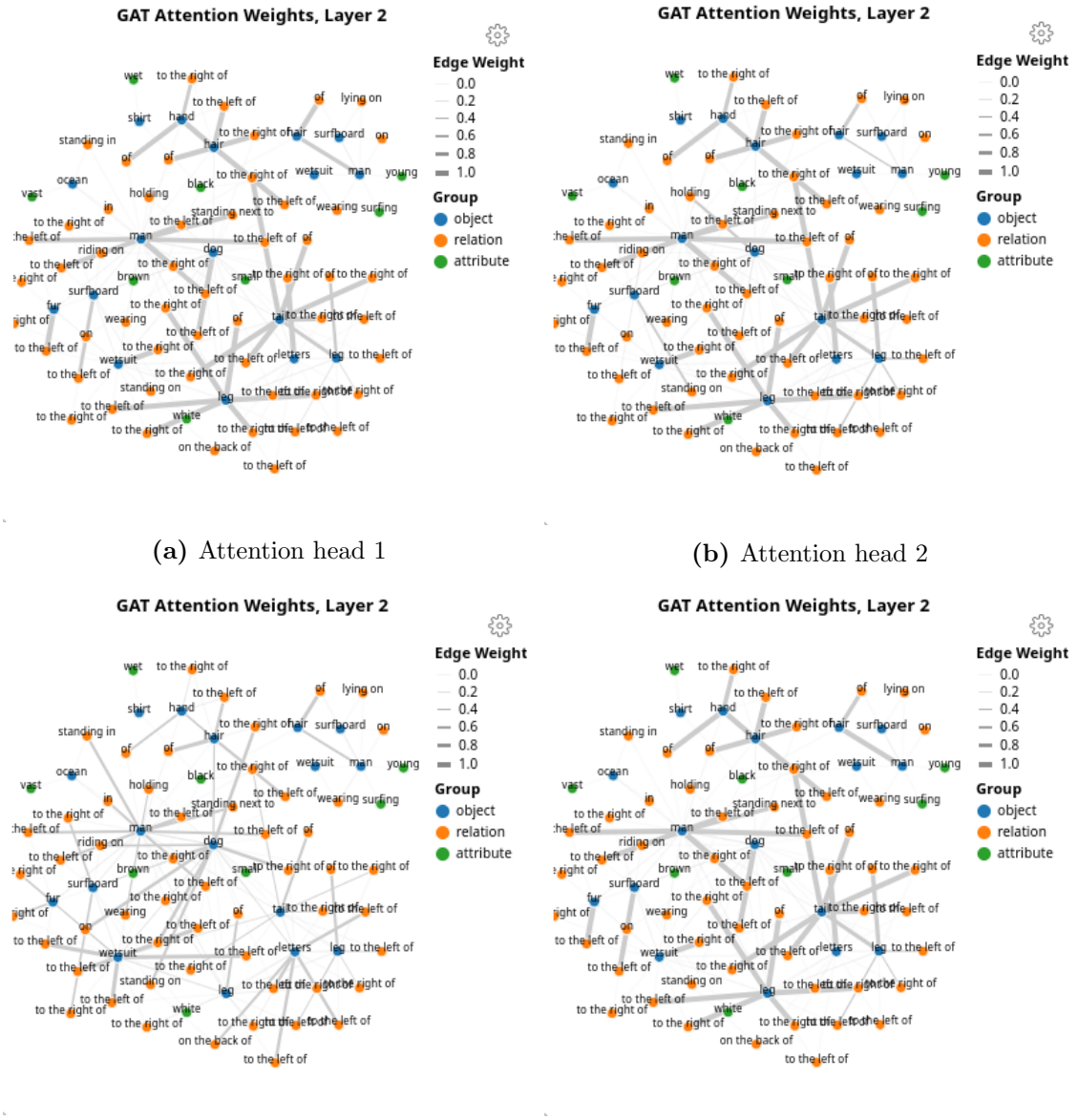
(a) Attention head 1

(b) Attention head 2

(c) Attention head 3

(d) Attention head 4

Figure 6.3: Scene graph processing module attention weights for the second GAT layer.



(a) Attention head 1

(b) Attention head 2

(c) Attention head 3

(d) Attention head 4

Figure 6.4: Scene graph processing module attention weights for the final GAT layer.

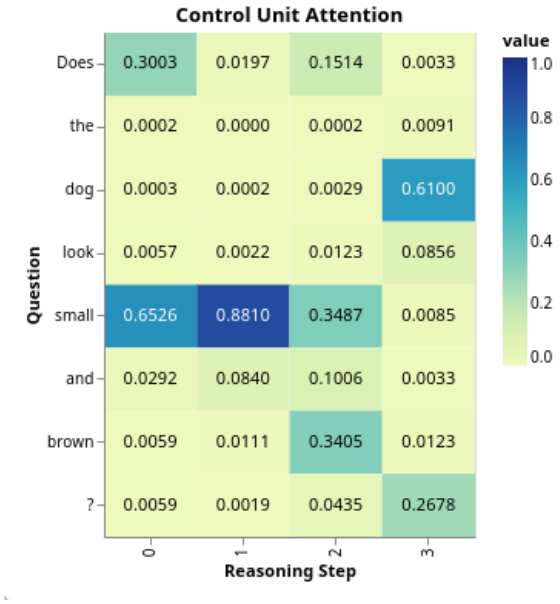


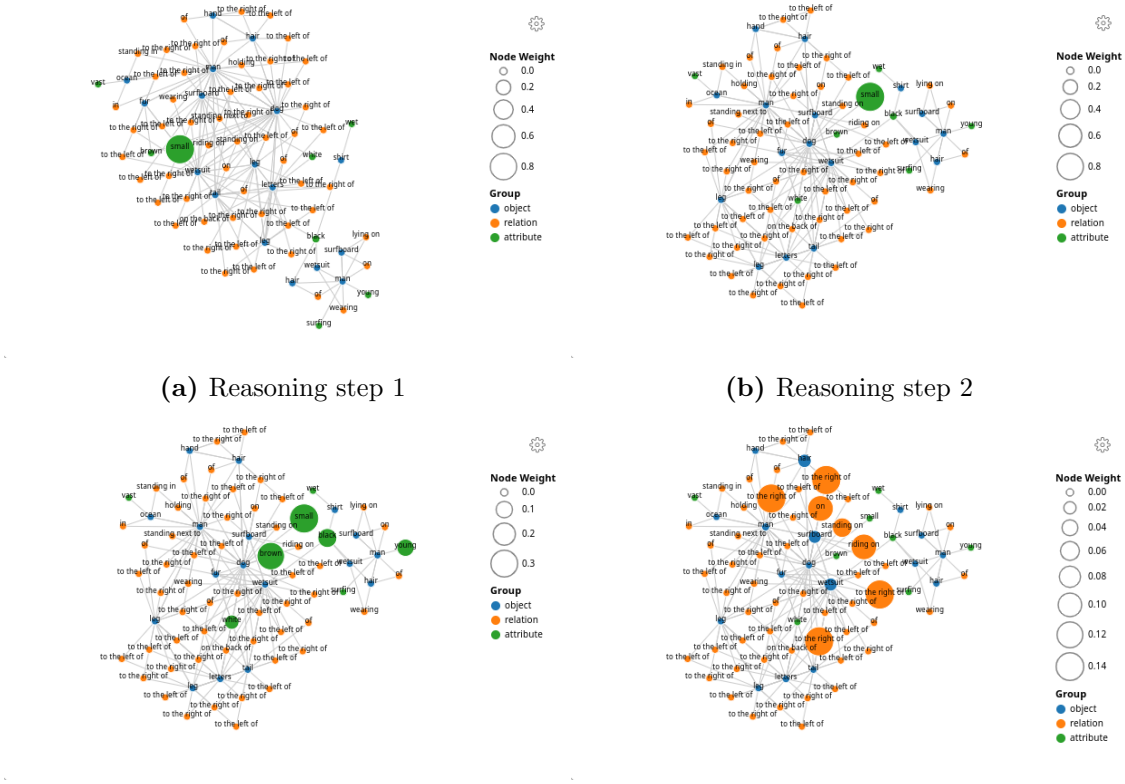
Figure 6.5: Question attention weights, extracted from the control unit of the reasoning module.

graph information like *black hair to the right of*, so fusing this data with existing object features leads to third-order interactions that capture entire relationships between two objects, with optional attribute information. For example, it is not unreasonable to expect that the node corresponding to the man on the right now represents the entire concept *black hair of man*.

After the last GAT layer we are left with a graph containing a large variety of semantic concepts that may be useful for answering any given question about the scene; all that needs to be done now is to determine which of these concepts are useful to the question at hand. This required both knowing about which question words are the most relevant for retrieving scene graph information, and knowing which nodes in the scene graph to attend to. These two tasks are handled by the reasoning module's control unit and read unit respectively, both of which are components of Hudson and Manning's compositional attention network. Figure 6.5 highlights the question words that the reasoning module attends to with each recurrent time-step. Ideally, the attended words for each time-step will correspond to a discrete reasoning step, such as focusing on an object or attribute in the scene graph or performing some form of logical operation. Figure **??** reveals the nodes that the reasoning module's read unit attend to at each recurrent time-step. If the model effectively uses the GAT outputs, we should see that the attended nodes correspond with the attended word features at each time step.

As seen in Figure 6.5, the reasoning module attends to the question word *small* for the first two reasoning steps, and then shares its attention between the words *small* and *brown*. These results coincide with the ideal logical reasoning processes that I introduced at the start of the example. In the last reasoning step, the control unit attends to the word *dog*, indicating it needs to locate the dog in the scene

graph and verify whether it has a similar embedding to the two attributes *small* and *brown*.



(a) Reasoning step 1 **(b)** Reasoning step 2

(c) Reasoning step 3 **(d)** Reasoning step 4

Figure 6.6: Scene graph attention weight for each reasoning step.

Most importantly, the question words that the control unit attends to are also attended to in the scene graph, indicating that the model is grounding its reasoning processes in the scene graph. In Figure 6.6, we see that the read unit of the reasoning module attends to the attribute *small* in the scene graph, indicating that the control unit is performing as intended, guiding the retrieval of scene graph features. In the third reasoning step, we see the read unit focuses on the two attributes *small* and *brown*, coinciding with the third reasoning step of the control unit. Finally, we see the read unit attend to a variety of relation nodes in the graph. Whilst this seems unintuitive at first, it is essential to remember that the nodes in the graph are complex combinations of nearby node features; upon further inspection, we see that all of the relation nodes that are attended to in the final reasoning step are connected to the *dog* node, indicating that they were provided with useful information about the *dog* by GAT convolutions. The model then passes all of this information to the write cell and finally the output module, where it predicts the answer *yes*.

6.2 Incorrectly Predicted Samples

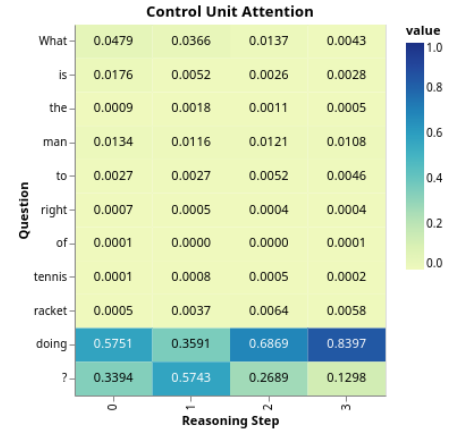
In this section, I provide some examples that illustrate the types of questions that my model fails to answer in an effort to understand why some questions are harder

than others. For brevity, I exclude the GAT attention maps that I provided in Subsection 6.1.1, and focus on the attention distributions learned by the control and read units of the reasoning module. In general, we can assume that the input to the reasoning module will be a contextual knowledge-base of scene graph features, that can capture up to third-order interactions between scene graph objects, attributes and relations.

6.2.1 Questions with Multiple Feasible Answers

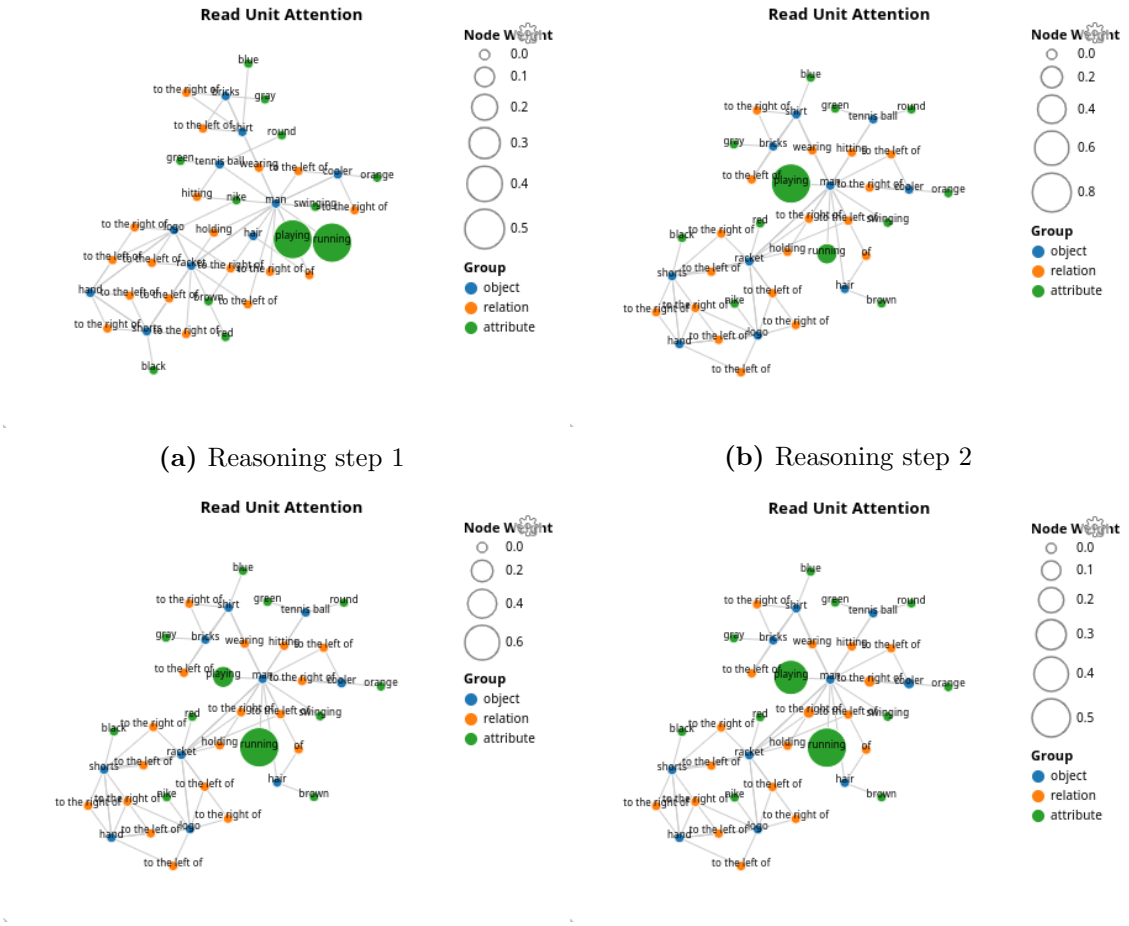
There are a variety of questions in the GQA dataset that have multiple feasible answers, but only one answer is considered truly correct. Such samples are difficult even for humans to answer correctly, as in many cases, both answers are equally valid and plausible. Consider the example in Figure 6.7a, where my model answered *running* instead of *playing*, despite both being perfectly acceptable answers to human eyes.

(a) A GQA sample that has multiple feasible answers.
Q: *What is the man to the right of the tennis racket doing?* **A:** *playing*



(b) Question attention weights, extracted from the control unit of the reasoning module.

In Figure 6.7b, we see that the reasoning module attends to the same part of the question for all reasoning steps, honing in on the fact that the question is asking for a verbal attribute. Investigating the attention maps of the reasoning module's read unit for each reasoning step, we see that the model alternates between playing and running as its two answers. In the final reasoning step, we see that it attends to *running* slightly more than *playing*, and thus answers the question incorrectly.



(a) Reasoning step 1 **(b)** Reasoning step 2

(c) Reasoning step 3 **(d)** Reasoning step 4

Figure 6.8: Scene graph attention weight for each reasoning step.

Chapter 7

Conclusion

To conclude, I review the primary outcomes of each chapter in this dissertation, summarising how they relate to the visual question answering (VQA) research community, from the literature that inspired my model architecture through to final results and contributions made to the field, and potential directions for future research.

In Chapter 2, I dissected the VQA problem into three main areas of interest, namely dataset collection, model design and metric creation. I explored methods developed by various VQA dataset creators to collect and/or generate questions and images in large quantities, whilst mitigating the problematic language biases that make effective evaluation of VQA models difficult. I investigated the promising results of recent graph-based models for graph representation learning and VQA, motivated by the lack of current research regarding VQA on pre-annotated scene graphs. Finally, I summarised the types of metrics used for the evaluation of both multiple-choice and open-ended VQA tasks.

Armed with this knowledge, I introduced the methodology behind my VQA model in Chapter 3, dividing it into four main components. Together, the question processing module, scene graph processing module, reasoning module and output module form my VQA architecture. I demonstrated the efficiency of my proposed scene graph transformation method, describing it in terms of ρ, the ratio of relations to objects in a scene graph. This analysis makes my VQA architecture extremely portable, as it can draw its scene graph information from any source, whether it be pre-annotated scene graphs like those reported in in this dissertation, or generated scene graphs. Moreover, when using generated scene graphs, as a visual input signal, we have complete control over ρ, allowing my model to be tuned for either performance or efficiency.

In Chapter 4, I introduced relevant details about the GQA dataset and the baseline VQA models used for performance evaluation. To promote a fair comparison between my model and baseline models, I split the models into two groups: those that have access to pre-annotated GQA scene graphs at evaluation time, and those that don't. I compared the performance of my architecture against these baselines

in Chapter 5, where it outperformed all other models across all reported metrics. In this comparison, I also demonstrated the importance of a reliable semantic visual signal for VQA tasks; both my own model and the Mutual and Self-Attention (MSA) model outperformed models LXMERT [44], BAN [15] and GRN [77] by over 26.0% and 16.9% respectively, since the other models rely upon traditional Faster R-CNN [7] and object bounding box features. I expect that as scene graph generation methods improve in the coming years, this gap will tighten, as generated scene graphs will be able to capture similar amounts of semantic information as pre-annotated scene graphs. In addition, I performed extensive ablation studies to demonstrate the effectiveness of Graph Attention Networks (GAT) for capturing semantic scene graph features. Finally, I provided a brief overview of the parameter optimisation process that I used to tune my model: over 160 days of GPU computation were used for both reported and non-reported models, and the parameter optimisation process alone required 55 days of GPU computation, indicating the large amount of effort that was required to develop my final model.

Finally, I analysed the reasoning processes used by my model to arrive at an answer to a question and a scene graph, for both correct examples and incorrect examples. Most importantly, we saw that the control and read units of my model's reasoning module attended to semantically similar concepts at each reasoning step, indicating my model's ability to interpret, understand and operate on question and scene graph information.

7.1 Future Work

As briefly mentioned earlier, the primary area for research stemming from results reported in this dissertation is scene graph generation. Whilst we have seen useful scene graph generation models emerge in the last few years [43], [44], they still perform relatively poorly when evaluated using standard top-k recall measures. The results presented in this dissertation are representative of the kind of performance we could expect from relatively simple graph-based VQA models given quality scene graph information. The model I present is interpretable, efficient and accurate, but still requires an accurate scene graph generation method to be properly used in production contexts.

Although my scene graph transformation and embedding method has a formal guarantee on its worst case complexity if we know ρ, I suspect that there is still ample room for optimisations on top of my proposed transformation method. More specifically, I propose that further research into sparse graph-based models that can handle both node and edge convolutions will greatly increase the quality of the scene graph embeddings demonstrated in my model. As it stands, the GAT weights in my model's scene graph processing module often incorporate object information into the relation nodes and vice-versa. I posit that adopting an approach that separates edge and node convolutions but still allows node and edge embeddings to interact would improve upon my model's interpretability and may result in increased performance.

Bibliography

[1] D. A. Hudson and C. D. Manning, "Gqa: A new dataset for real-world visual reasoning and compositional question answering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6700–6709.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[3] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning," *CoRR*, vol. abs/1803.03067, 2018. arXiv: 1803.03067. [Online]. Available: http://arxiv.org/abs/1803.03067.

[4] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6904–6913.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179.

[10] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[11] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. 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*, 2018, pp. 6077–6086.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[13] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," *Advances in neural information processing systems*, vol. 29, pp. 289–297, 2016.

[14] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, "Deep modular co-attention networks for visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 6281–6290.

[15] J.-H. Kim, J. Jun, and B.-T. Zhang, "Bilinear attention networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1564–1574, 2018.

[16] H. Ben-Younes, R. Cadene, M. Cord, and N. Thome, "Mutan: Multimodal tucker fusion for visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2612–2620.

[17] H. Ben-Younes, R. Cadene, N. Thome, and M. Cord, "Block: Bilinear superdiagonal fusion for visual question answering and visual relationship detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8102–8109.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] L. Li, Z. Gan, Y. Cheng, and J. Liu, "Relation-aware graph attention network for visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 313–10 322.

[20] Q. Huang, J. Wei, Y. Cai, C. Zheng, J. Chen, H.-f. Leung, and Q. Li, "Aligned dual channel graph convolutional network for visual question answering," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7166–7176.

[21] A. Agrawal, D. Batra, D. Parikh, and A. 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 (CVPR)*, Jun. 2018, pp. 4971–4980.

[22] D. Hudson and C. D. Manning. (2020). Gqa real-world visual reasoning challenge, [Online]. Available: `https://eval.ai/web/challenges/challenge-page/225/leaderboard/733` (visited on 12/28/2020).

[23] C. L. Zitnick and D. Parikh, "Bringing semantics into focus using visual abstraction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013.

[24] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[25] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014,

pp. 1682–1690. [Online]. Available: `http://papers.nips.cc/paper/5411-a-multi-world-approach-to-question-answering-about-real-world-scenes-based-on-uncertain-input.pdf`.

[26] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*, Springer, 2012, pp. 746–760.

[27] L. Yu, E. Park, A. C. Berg, and T. L. Berg, "Visual madlibs: Fill in the blank description generation and question answering," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2461–2469.

[28] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2953–2961. [Online]. Available: `http://papers.nips.cc/paper/5640-exploring-models-and-data-for-image-question-answering.pdf`.

[29] A. Agrawal, A. Kembhavi, D. Batra, and D. Parikh, "C-vqa: A compositional split of the visual question answering (vqa) v1. 0 dataset," *arXiv preprint arXiv:1704.08243*, 2017.

[30] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and yang: Balancing and answering binary visual questions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5014–5022.

[31] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations.(article)," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017, ISSN: 0920-5691.

[32] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research," *Commun. ACM*, vol. 59, no. 2, pp. 64–73, 2016, ISSN: 0001-0782. [Online]. Available: `https://doi.org/10.1145/2812802`.

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

[34] K. Kafle and C. Kanan, "An analysis of visual question answering algorithms," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1965–1973.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[36] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3668–3678.

[37] B. O. Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: `http://www.blender.org`.

[38] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," *arXiv preprint arXiv:1709.07871*, 2017.

[39] A. Agrawal, D. Batra, and D. Parikh, "Analyzing the behavior of visual question answering models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1955–1960.

[40] K. Kafle and C. Kanan, "Visual question answering: Datasets, algorithms, and future challenges," *Computer Vision and Image Understanding*, vol. 163, pp. 3–20, 2017.

[41] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5410–5419.

[42] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1261–1270.

[43] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 670–685.

[44] K. Tang, H. Zhang, B. Wu, W. Luo, and W. Liu, "Learning to compose dynamic tree structures for visual contexts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6619–6628.

[45] C. Han, S. Long, S. Luo, K. Wang, and J. Poon, "VICTR: Visual information captured text representation for text-to-vision multimodal tasks," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3107–3117. [Online]. Available: `https://www.aclweb.org/anthology/2020.coling-main.277`.

[46] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[47] M.-C. De Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal stanford dependencies: A cross-linguistic typology.," in *LREC*, vol. 14, 2014, pp. 4585–4592.

[48] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[49] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[50] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[51] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl*

workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.

[52] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[53] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[54] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '94, Las Cruces, New Mexico: Association for Computational Linguistics, 1994, pp. 133–138. [Online]. Available: `https://doi.org/10.3115/981732.981751`.

[55] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[56] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1–9.

[57] D. A. Hudson and C. D. Manning. (2019). Gqa evaluation script, [Online]. Available: `https://cs.stanford.edu/people/dorarad/gqa/evaluate.html` (visited on 11/13/2020).

[58] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[59] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020. [Online]. Available: `https://nlp.stanford.edu/pubs/qi2020stanza.pdf`.

[60] D. Teney, P. Anderson, X. He, and A. Van Den Hengel, "Tips and tricks for visual question answering: Learnings from the 2017 challenge," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4223–4232.

[61] F. Liu, J. Liu, Z. Fang, R. Hong, and H. Lu, "Densely connected attention flow for visual question answering.," in *IJCAI*, 2019, pp. 869–875.

[62] P. Gao, Z. Jiang, H. You, P. Lu, S. C. Hoi, X. Wang, and H. Li, "Dynamic fusion with intra-and inter-modality attention flow for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6639–6648.

[63] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7370–7377.

[64] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification.," in *AAAI*, 2020, pp. 8409–8416.

[65] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8026–8037. [Online]. Available: `http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[67] D. Hudson and C. D. Manning. (2019). Compositional attention networks for real-world reasoning, [Online]. Available: `https://github.com/stanfordnlp/mac-network` (visited on 12/29/2020).

[68] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[69] C. Eyzaguirre. (2019). Mac-network-pytorch, [Online]. Available: `https://github.com/ceyzaguirre4/mac-network-pytorch` (visited on 12/29/2020).

[70] C. Eyzaguirre and A. Soto, "Differentiable adaptive computation time for visual reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 12 817–12 825.

[71] D. Hudson and C. D. Manning. (2019). Gqa: Visual reasoning in the real world, [Online]. Available: `https://cs.stanford.edu/people/dorarad/gqa/vis.html` (visited on 12/31/2020).

[72] D. A. Hudson and C. D. Manning, "GQA: a new dataset for compositional question answering over real-world images," *CoRR*, vol. abs/1902.09506, 2019. arXiv: `1902.09506`. [Online]. Available: `http://arxiv.org/abs/1902.09506`.

[73] H. Tan and M. Bansal, "LXMERT: Learning cross-modality encoder representations from transformers," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5100–5111. [Online]. Available: `https://www.aclweb.org/anthology/D19-1514`.

[74] L. Wolf, H. Jhuang, and T. Hazan, "Modeling appearances with low-rank svm," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–6.

[75] H. Pirsiavash, D. Ramanan, and C. Fowlkes, "Bilinear classifiers for visual recognition," *Advances in neural information processing systems*, vol. 22, pp. 1482–1490, 2009.

[76] J.-H. Kim, S.-W. Lee, D. Kwak, M.-O. Heo, J. Kim, J.-W. Ha, and B.-T. Zhang, "Multimodal residual learning for visual qa," *Advances in neural information processing systems*, vol. 29, pp. 361–369, 2016.

[77] D. Guo, C. Xu, and D. Tao, "Graph reasoning networks for visual question answering," *CoRR*, vol. abs/1907.09815v1, 2019. arXiv: `1907.09815`. [Online]. Available: `https://arxiv.org/abs/1907.09815v1`.

[78] M. Farazi, S. Khan, and N. Barnes, *Attention guided semantic relationship parsing for visual question answering*, 2020. arXiv: `2010.01725 [cs.CV]`.

[79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[80] H. Tan and M. Bansal. (2020). Lxmert: Learning cross-modality encoder representations from transformers, [Online]. Available: `https://github.com/airsplay/lxmert` (visited on 12/28/2020).

[81] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: `https://www.aclweb.org/anthology/D14-1181`.

[82] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[83] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: `https://www.wandb.com/`.

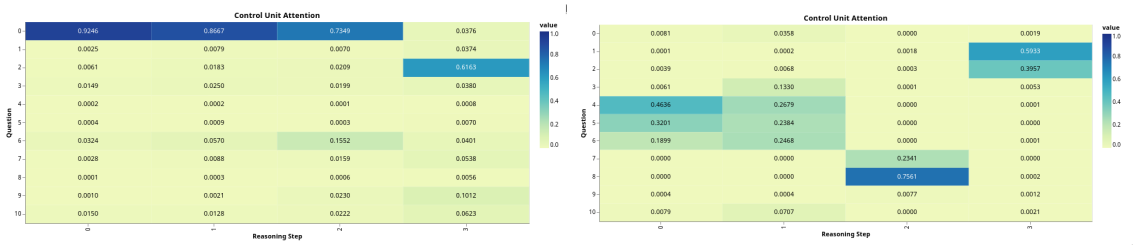
Appendix A

Ablation Study Visualisations

A.1 Question Processing Module Ablations

As described in Subsection 5.2.1, the BiLSTM question module performed poorly on compare-type questions compared to the GAT question module. In Figure A.1, we see an example image with a corresponding binary comparison question. For this sample, the model that used the GAT question module answered the question correctly, where the one that used the BiLSTM question module answered incorrectly. In Figure A.2, we see that the attended question features for the GAT question module are much more appropriate than those for the BiLSTM question module; the GAT question module attends to phrases like *have the same color* in the second reasoning step, *as the pants* in the third reasoning step and *the skis* in the final reasoning step. Conversely, the BiLSTM question module attends to the words *do* in reasoning steps 1, 2 and 3, and attends to *skis* in the final reasoning step. The attended question features for the BiLSTM and GAT question modules are reflected in the read unit attentions shown in Figure A.3 and Figure A.4 respectively.

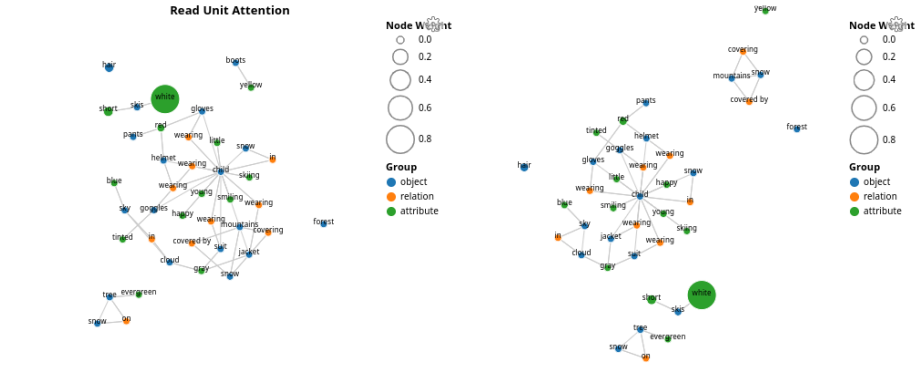
Figure A.1: *Do the skis have the same color as the pants ?* **No**



(a) BiLSTM question module attention map.

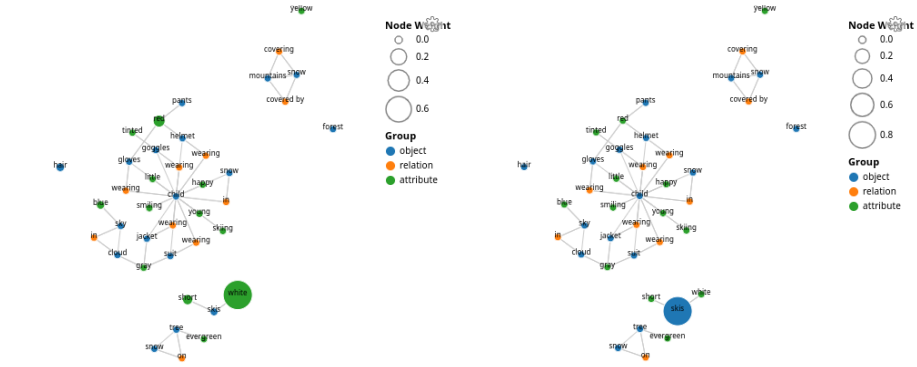
(b) GAT question module attention map.

Figure A.2: Control unit attention maps for a BiLSTM question module and a GAT question module given the question *Do the skis have the same color as the pants ?*



(a) Reasoning step 1

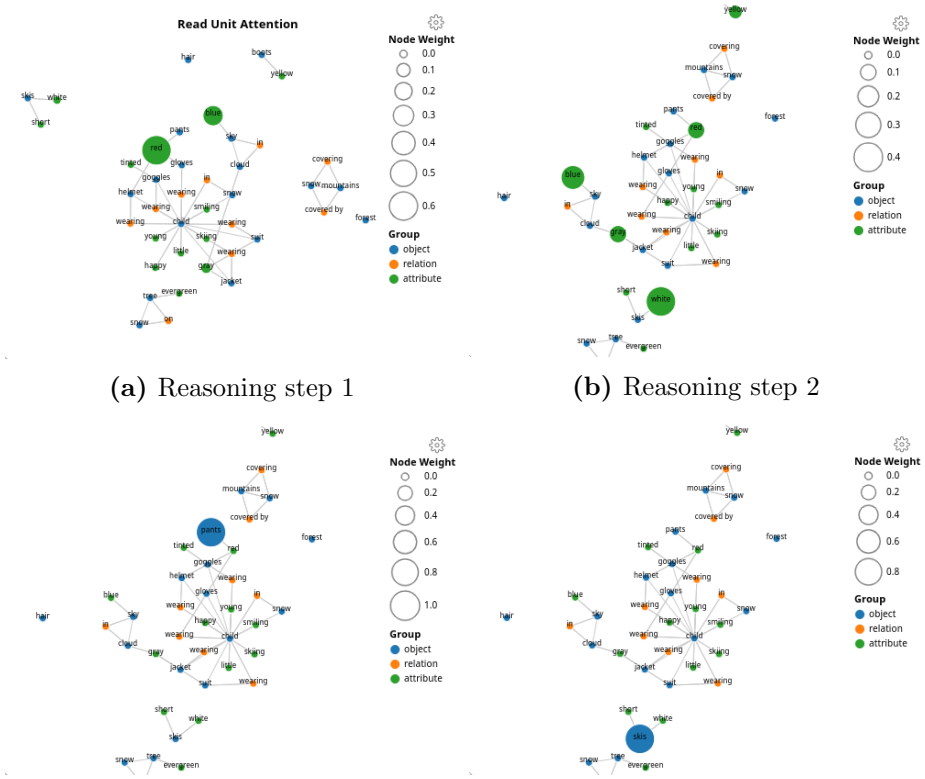
(b) Reasoning step 2



(c) Reasoning step 3

(d) Reasoning step 4

Figure A.3: Read unit attention maps at each reasoning step for the model that uses a BiLSTM question module given the question *Do the skis have the same color as the pants ?*. Across all reasoning steps, the main information extracted from the scene graph is the colour of the skis, which is not enough information to answer the question correctly.



(a) Reasoning step 1 **(b)** Reasoning step 2

(c) Reasoning step 3 **(d)** Reasoning step 4

Figure A.4: Read unit attention maps at each reasoning step for the model that uses a GAT question module given the question *Do the skis have the same color as the pants ?* In the first and second reasoning steps, the model attends to relevant colour attributes in the scene graph, before attending to the pants in the third reasoning step and finally the skis.