UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
Electronics and Computer Science

Multimodal Learning and Reasoning for Visual Question Answering

by

Vasin Srisupavanich

Supervisor: Dr Srinandan Dasmahapatra
Second Examiner: Professor Eric Rogers

A dissertation submitted in partial fulfilment of the degree of MSc Artificial Intelligence

September 2020

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
Electronics and Computer Science

Multimodal Learning and Reasoning for Visual Question Answering

by Vasin Srisupavanich

Current deep learning systems are very successful in sensory perception and pattern recognition (e.g. object detection and speech recognition). However, they often struggle in tasks with a compositional nature which require more deliberate thinking and multi-step reasoning. In this work, we study how we can improve the reasoning capability of artificial neural networks in the context of visual question answering (VQA) and visual reasoning. These two tasks are challenging multimodal research problem, which requires fine-grained understanding of both image and question, and typically demands the ability to perform multi-step inferences. As part of this work, we develop a new VQA model based on the novel Transformer architecture, utilising both self-attention and co-attention mechanism for dealing with multimodal input. Experimental results demonstrate that Transformer based model can be effective in a visual reasoning task, as our model achieves strong results on both CLEVR and GQA dataset with 98.3% and 56.28% accuracy respectively. Our analysis shows that the model learns to look around the image and iteratively focus on parts of the image that are relevant to finding an answer, indicating that it is capable of performing multi-step reasoning.

Acknowledgements

I would like to thank my supervisor, Dr Srinandan Dasmahapatra, for all the support throughout this project, and for giving me the chance to pursue the topic of my interest.

I also would like to thank British Council and University of Southampton for providing me with the scholarship and the opportunity to pursue a master degree in the UK.

Finally, I want to thank my family and friends for their love, support and encouragement throughout my study.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

https://github.com/MILVLG/openvqa
https://github.com/facebookresearch/mmf

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decade, **deep learning** systems have enjoyed tremendous success in the area of computer vision and natural language processing (NLP). From image recognition, object detection, to machine translation, deep learning has completely changed the state of Artificial Intelligence (AI). However, in recent years, it has become apparent that there are several limitations in the current deep learning models [1, 2]. In particular, they struggle in tasks with a compositional and structured nature which require more deliberate thinking and multi-step reasoning [3, 4]. Many argued that current deep neural networks are just a large correlation engine, which only captures statistical patterns between input and output to make prediction rather than the true underlying reasoning processes.

To improve the capabilities of the current deep learning systems, many datasets and benchmarks have been developed. One such task which has become very popular is **Visual Question Answering** (VQA). It is one of the main testbeds for pushing state of the art in vision and language research. Specifically, VQA is the task of answering question about an image. It is considered to be an "AI-complete" task, as it demands fine-grained understanding of images as well as the ability to understand questions and provide answers in natural language. The task requires a rich set of abilities, such as object recognition, activity recognition, commonsense understanding and relation extraction, spanning both the visual and linguistic domains.

Another line of research which is closely related to VQA is **Visual Reasoning**. Visual reasoning can be considered a sub-task of VQA, with a focus on evaluating the reasoning capabilities of deep learning models. Several datasets such as CLEVR [5] and GQA [6] have emerged to specifically test various reasoning skills, including spatial understanding and higher-level skills such as counting, performing logical inference and making comparisons. The questions in these datasets tend to be highly compositional and requires multi-step inferences. This is in contrast with the typical VQA datasets where questions can be quite straightforward and may contain real-world bias. The differences between the two tasks are demonstrated in Figure 1.1. The objective of visual reasoning task is to discourage the model

from developing a cheating process by memorising statistical patterns or bias in the dataset. Consequently, the model needs to learn the underlying reasoning process in order to perform well.

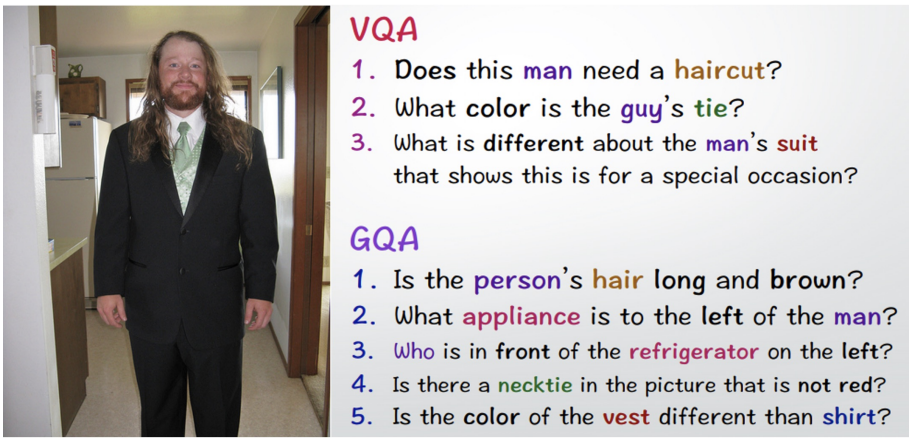


FIGURE 1.1: Differences between typical questions in VQA and visual reasoning dataset [6]. The questions in a visual reasoning dataset, such as GQA tend to be longer and more compositional, while the questions in typical VQA dataset can be more ambiguous and subjective.

1.1 Project Aims and Objectives

The aim of this project is to develop a new VQA model utilising Transformer's self-attention mechanism [7] and study how to adapt the Transformer architecture for a multimodal task, such as visual reasoning. This is inspired by the recent success of deep learning models based on Transformer architecture, such as BERT [8] and GPT [9]. These models have shown astonishing results in various NLP tasks, such as question answering, machine translation and text generation, and seem to be capable of reasoning to a certain extent. In addition, they have been shown to learn the underlying linguistic structures in natural language [10]. The hypothesis of using a Transformer architecture in a visual reasoning task is that by stacking multiple Transformer's self-attention layers, the model would learn to focus on different parts of the image, identify most important words in the question, and then come up with an answer.

The initial objective of this project is to study various types of deep learning models, and how they are applied in VQA and visual reasoning task. Afterwards, the next objective is to explore recent approaches which extend the reasoning capabilities of artificial neural networks, and investigate how they can be effective on a visual reasoning task. In particular, this project will focus on examining attention mechanism and its various types, including visual attention and self-attention. Additionally, several deep learning models based on Transformer architecture will be analysed to understand its inner working.

1.2 Contributions

In this paper, we present a new VQA model based on Transformer architecture which achieves strong results in two visual reasoning datasets. We also demonstrate one effective way to adapt the Transformer model to solve a multimodal task, such as visual question answering. In addition, we provide extensive analysis of the model, which clearly shows that the model is capable of performing multi-step reasoning.

1.3 Structure and Overview

This dissertation is composed of six chapters and has the following structure:

- **Chapter 2** presents an overview of the most important background knowledge of deep learning, as well as the building blocks of a VQA system, including Long short-term memory (LSTM), Convolutional Neural Network (CNN) and attention mechanism.
- **Chapter 3** reviews the most relevant works related to this dissertation, and explains various existing VQA models, from strong baseline to current state of the art models.
- **Chapter 4** describes the proposed model in detail, including the motivation behind each design decision.
- **Chapter 5** discusses the experiment performed, details of the datasets used, and results and analysis of the models.
- **Chapter 6** gives the final discussions and concluding remarks, along with suggestions for future work in this area.

Chapter 2

Background

In this chapter, we will review several important concepts which are essential to understand this work. We will start by explaining the task of visual question answering in detail, then describe convolutional neural network and recurrent neural network, which are the main components used to process the image and question in a VQA system. We will also explain the core concepts behind attention mechanism, including visual attention and self-attention. Finally, we will discuss about Transformer architecture, which is the backbone of our proposed model.

2.1 The Visual Question Answering Task

Visual Question Answering (VQA) is an emerging interdisciplinary research problem at the intersection of computer vision, natural language processing and knowledge reasoning. It has become one of the most compelling research topics as it is considered "AI-complete" [11], which is the problem of trying to make a machine as intelligent as human. The task of VQA is to find an answer to a question based on the content of an image (for example, see Figure 2.1). The questions can be in free-from and open-ended, and typically require the ability to solve several sub-problems from the computer vision domain as illustrate in Table 2.1.

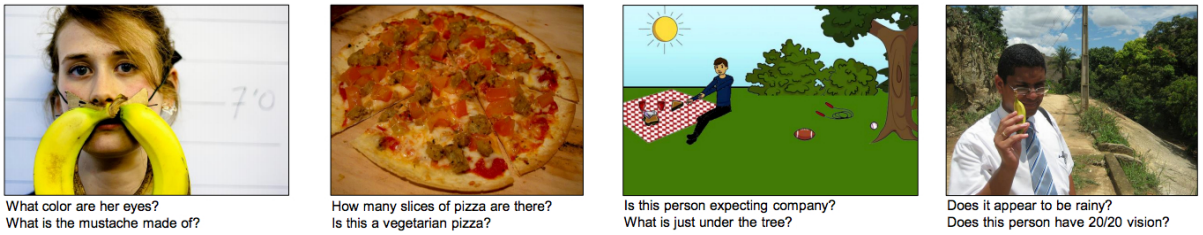


FIGURE 2.1: Example of VQA task [12]

Generally, a VQA task is framed as a classification problem where the list of all possible answers in the dataset are taken as the targets. This is possible since most of the answers are

Computer Vision Task	Representative VQA Questions
Object recognition	What is in the image?
Object detection	Are there any dogs in the picture?
Attribute classification	What color is the umbrella?
Scene classification	Is it raining?
Counting	How many people are there in the image?
Activity recognition	Is the child crying?
Spatial relationships among objects	What is between cat and sofa?
Commonsense reasoning	Does this person have 20/20 vision?
Knowledge-based reasoning	Is this a vegetarian pizza?

TABLE 2.1: Computer vision sub-tasks required to be solved in VQA [13]

concise and typically contain no longer than three words. Most recent successful VQA systems are based on an end-to-end deep neural network, and the general architecture of this method is shown in Figure 2.2. This general framework can be summarised into three stages as follows:

1. **Feature Extractions**. In a feature extraction stage, the image features are extracted using a convolutional neural network (CNN), while the questions are typically extracted using a recurrent neural network, such as LSTM or GRU. In addition, most VQA model employs pre-trained CNNs (e.g. ResNet [14], VGG [15]) to obtain better visual features.

2. **Multi-modal Fusion**. There are a variety of methods to combine the visual and textual features, ranging from vector concatenation, element-wise addition, element-wise multiplication, as well as more advanced methods, such as attention mechanism.

3. **Answer Prediction**. The features obtained from a multimodal fusion stage are put through a classifier layer to obtain the distribution over candidate answers.

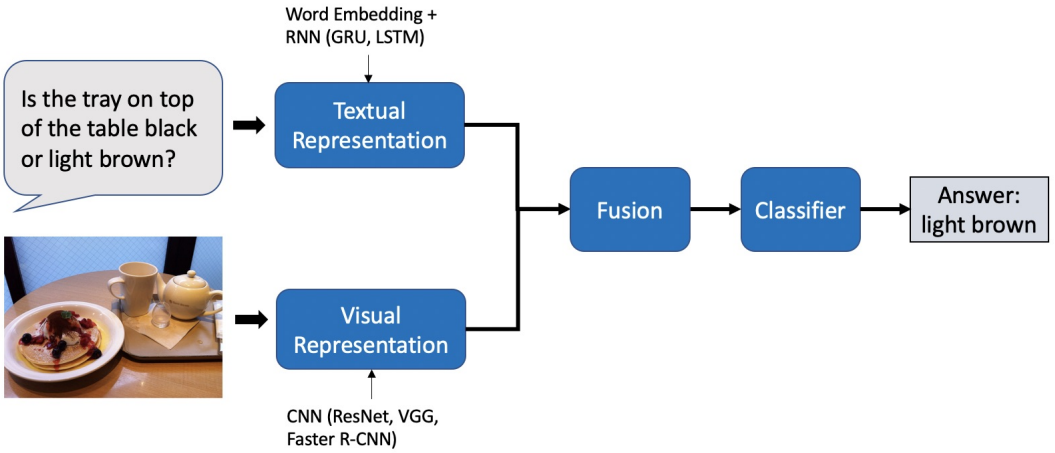


FIGURE 2.2: Typical VQA framework

2.2 Convolutional Neural Network

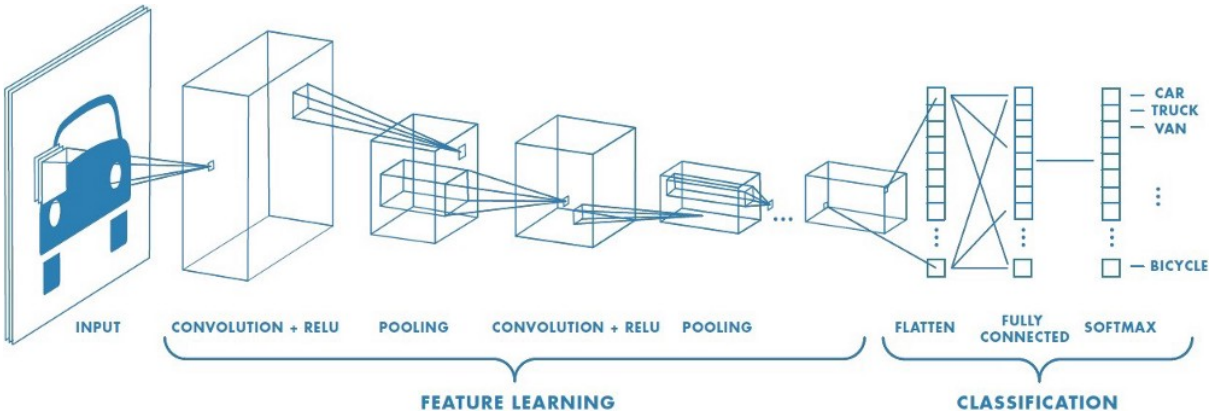


FIGURE 2.3: Architecture of Convolutional Neural Networks [16]

Convolutional neural network (CNN) is a class of deep neural networks, which is mainly used for analysing visual input. The main properties of CNN are the use local connectivity between layers and parameters sharing. CNN were inspired by biological processes of visual cortex, where the connectivity between neurons in each layer is restricted to local regions (receptive field). This enables the model to exploit spatial information of the input. In addition, the learned parameters, called "filters", are applied across the entire image, which means that all the neurons in the layer will respond to the same feature within their receptive field. Thus, the CNN is considered translation invariant to the input image. In contrast with traditional MLP, neurons in CNN are arranged in three dimensions: width, height and depth (see Figure 2.3), in order to maintain the spatial structure of the input.

As shown in Figure 2.3, the architecture of CNN contains multiple layers of convolution and pooling operation. These layers are summarised as follow:

1. **Convolution layer.** In the convolutional layer, the input features are convolved with the learnable parameters, often called kernels or filters (see Figure 2.4). The kernels consist of a defined number of NxN grids of weights with a bias. The convolution operation ($*$) between input I and kernel K is performed as follow:

$$(K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.1}$$

   The output of this operation is a feature map which becomes the input to the next layer. The number of feature maps after a convolutional layer depends on the number of kernels in the respective layer. As a result, the network learns filters that activate when detecting certain features at some spatial position in the input. Moreover, it is often useful to choose multiple kernels in a convolutional layer, as this will enable the model to extract different features from the same input.

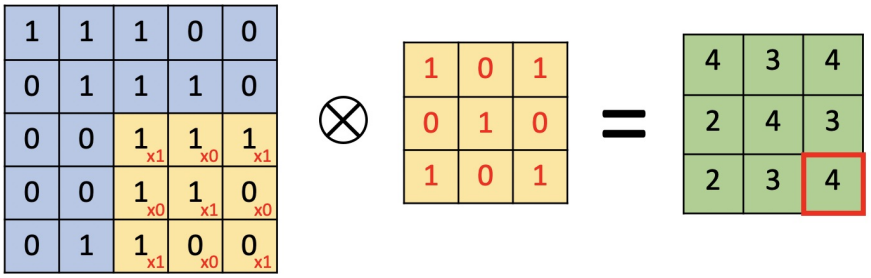


FIGURE 2.4: Convolution Operation

2. **Pooling layer.** Pooling operation is performed to reduce the spatial dimension (width, height) of the input features. This is useful for extracting dominant features, which are rotational and positional invariant. There are two types of pooling operation: max pooling and average pooling. As the name suggests, max pooling returns the maximum value from portion of the input covered by the kernel, while average pooling returns the average value from portion of the input covered by the kernel. Figure 2.5 demonstrates the process of max pooling.

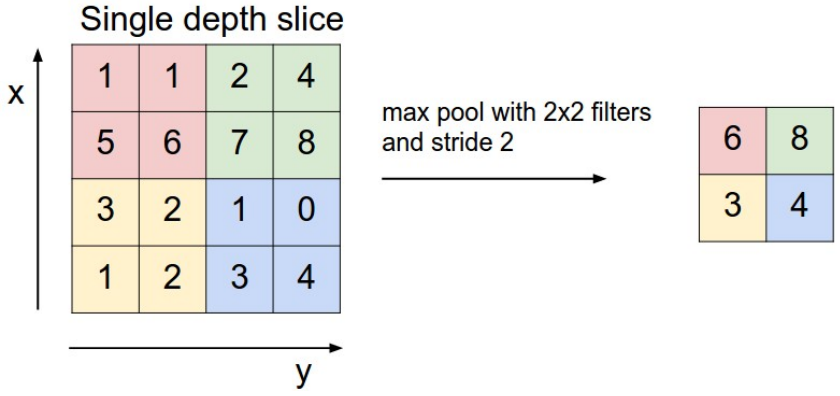


FIGURE 2.5: Max pooling Operation [16]

The ideas behind CNN are that by stacking multiple convolution layers, the model would learn primitive features, such as edges or corner, in the earlier layers, and more higher-level features, such as objects in the layers near the end. Compare to a traditional MLP, CNN is able to capture the spatial and temporal dependencies of the input. These properties lead to remarkable successes in a variety of computer vision tasks, such as image classification and object detection.

In a VQA system, CNN is essential for processing the input image. Most recent VQA systems usually employ state-of-the-art CNN architectures such as **ResNet** [14] and **VGG** [15] pre-trained on a large-scale dataset, such as ImageNet [17]. These networks, which have already learned useful representations of the image, can be greatly beneficial for a VQA task. Typically, the features extracted from the penultimate layer in the CNN, which maintain spatial information, are used to represent image features in a VQA system.

2.3 Recurrent Neural Network

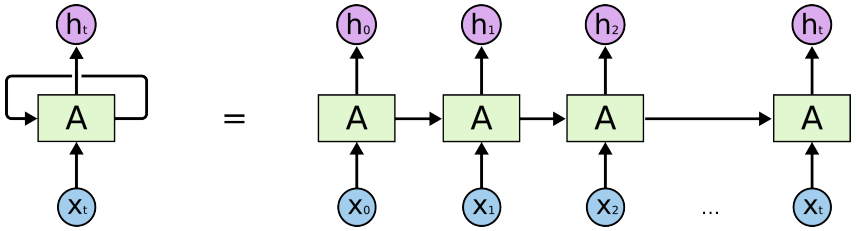


FIGURE 2.6: An unrolled recurrent neural network [18]

Recurrent Neural Network (RNN) is a type of artificial neural network used for processing sequential data, such as text and audio. They are a neural network with loops (see Figure 2.6), which allow the network to take in a variable sequence of inputs. Unrolling the loop of RNN (see Figure 2.6 (right)), it can be seen that RNN is like a traditional neural network with share weight along each time step. In addition, RNN contains a distributed hidden state (memory unit), which allow the model to store information of the past. Concretely, a vanilla RNN are defined as follows.

For each timestep t, given the input $\boldsymbol{x_t}$, the hidden state $\boldsymbol{h}_t$ and the output $\boldsymbol{y}_t$ are expressed as follows:

$$\boldsymbol{h}_t = \sigma_h(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{b}_{ih} + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hh}) \tag{2.2}$$

$$\boldsymbol{y}_t = \sigma_y(\boldsymbol{W}_y\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{2.3}$$

where $\boldsymbol{W}_{ih}, \boldsymbol{W}_{hh}, \boldsymbol{W}_{yy}$ are learnable weight that are shared across each time step, and σ_h, σ_y are the activation function.

From equation (2.2), the hidden state is calculated by element-wised addition between the projected input and the projected hidden state of the last timestep, follow by an activation function (usually a Tanh function). The interpretation is that the new hidden state (memory) is the aggregation of the new information with its previous memory, while the output is just a projected hidden state at time t.

One of the drawbacks of a vanilla RNN is that they are not very good at capturing long term dependencies in the sequence. For instance, when an RNN process a very long sentence, words that appear in the beginning of the sequence could become meaningless to the words at the end of that sequence. This is due to the problem of vanishing and exploding gradient where multiplicative gradient becomes exponentially decreasing/increasing with respect to the number of layers during training.

2.3.1 Long Short-Term Memory

Long Short-Term Memory network (LSTM) [19] is a special kind of RNN, which was designed to solve the long-term dependency problem. The core ideas behind LSTM are the

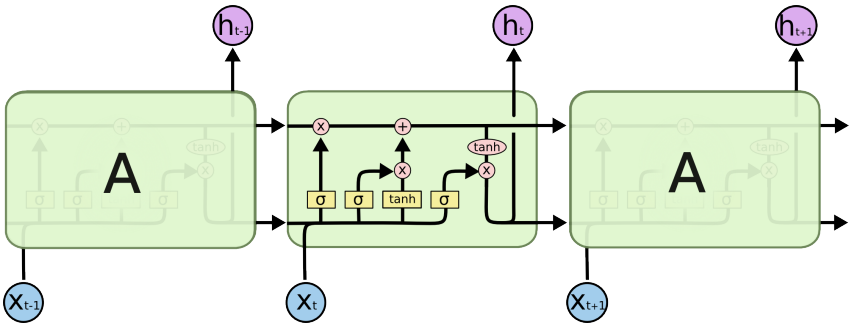


FIGURE 2.7: An LSTM cell [18]

introduction of cell state (long term memory), along with the gated units which can selectively add or remove information. As shown in Figure 2.7, an LSTM unit contains four interacting layers (shown in yellow), which are summarised as follows:

1. **Forgetting irrelevant information.** The first step of an LSTM is to decide which information in the cell state can be removed. This is responsible by the "forget gate layer" It looks at both the previous output and the new input, then outputs a number between 0 and 1. A 0 means that the corresponding element in the cell state is not relevant anymore and should be removed. This can be expressed as:

$$f_t. = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.4}$$

   where σ represents sigmoid function and $[\cdot,\cdot]$ represents vector concatenation.

2. **Storing new information.** The next step is to decide which of the new value should be updated to the cell state. This step is performed by the "input gate layer", and can be expressed as follows.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.5}$$

   Then a new candidate value of cell state $\hat{C}_t$, is created based on the new input and previous output.

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_C) \tag{2.6}$$

3. **Selectively update cell state.** The cell state is updated based on the outputs from the input gate (i_t), forget gate (f_t), and the new candidate value of cell state ($\hat{C}_t$):

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.7}$$

   where $*$ represents elementwise multiplication.

4. **Output.** Finally, the output (h_t) will be based on the filtered version of the cell state. The output gate (o_t) is responsible for deciding which part of the cell state is the relevant information.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.8}$$

$$h_t = o_t * tanh(C_t) \tag{2.9}$$

LSTM have gained a lot of successes and became the standard in various sequence processing task, including machine translation, speech and handwriting recognition. In addition, they are the main component used to process the textual input in a VQA system. Most recent VQA models either use the final output (the output feature for the last word) from LSTM to represent the whole question, or the output from each timestep to represent each word in the question.

2.4 Attention Mechanism

Attention mechanism was first introduced in 2014 in the context of machine translation [20] and became one of the most prominent tools in deep learning. This idea is loosely motivated by how human biological system works. In a human visual system, attention allows us to focus on specific region with high resolution, while ignoring other irrelevant information, making unfocused region blurry. Similarly, in machine translation, attention model enables the machine to focus on specific words at a time rather than the full sentence.

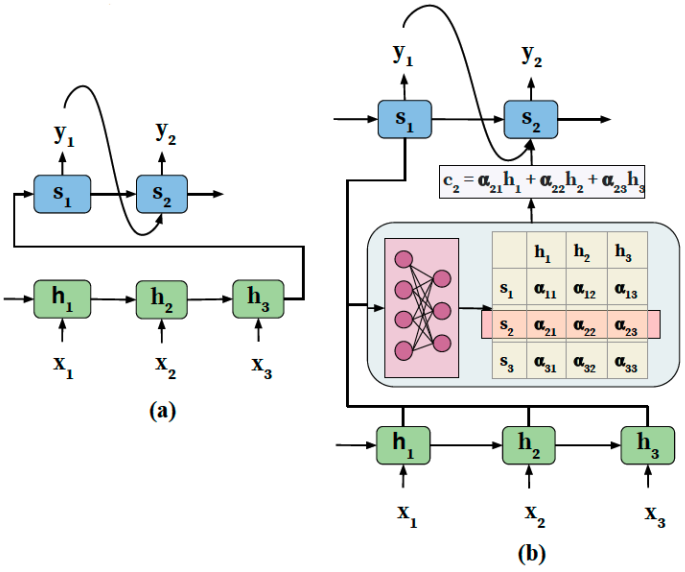


FIGURE 2.8: NMT architecture (a) traditional (b) with attention model [21]

In a neural machine translation (NMT) model, the architecture consists of an encoder-decoder (sequence to sequence) structure. An encoder, usually an RNN, learns to encode a source sentence into a fixed length vector. Then the decoder network output the encoded vector into another language. Figure 2.8(a) shows the traditional encoder-decoder architecture. The apparent problem of this architecture is that the model tends to forget relevant information in a very long sentence. With the addition of attention model (Figure 2.8(b)), this problem is mitigated, as the decoder can learn to attend to different parts of the source sentence.

There are several types of attention mechanism in the literature, however we will focus only on soft attention, which is the specific type of attention relevant to this dissertation. The underlying mechanism of soft attention will be explained in detail in the following section.

2.4.1 Visual Attention

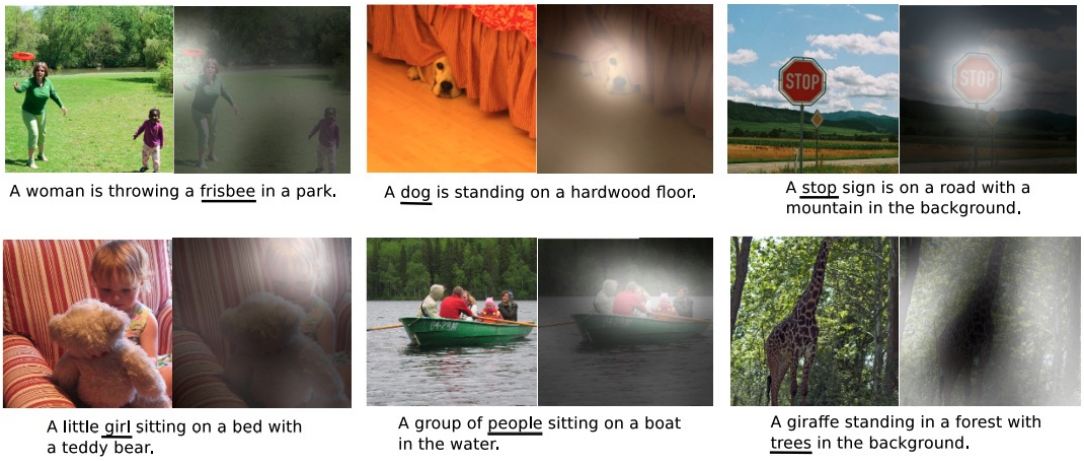


FIGURE 2.9: Visualisation of the attended region conditioned on the corresponding word [22]

In computer vision field, attention model was first applied to generate caption from images [22]. In this task, CNN is used as an encoder to extract features from raw images. Then an LSTM is used to generate the words, conditioned on the attention weights. Figure 2.9 demonstrates how the model learns to attend to specific part of the image with the corresponding word. As reported in the paper, incorporating attention mechanism has resulted in higher quality captions, and led to state-of-the-art results in various image captioning benchmarks.

In a soft attention model, the aim is to compute the context vector $\hat{z}_i$ which represents the weighted average of the input features x_i, for $i = 1...L$ corresponding to image features in different location. The input features in this case are extracted from a lower layer CNN, which maintains spatial information. For each location i, the model computes a weight α_i that represents the importance of feature vector x_i. First, the attention score e_{ti} is computed by an attention model f_{att} conditioned on the previous hidden state of the LSTM h_{t-1}. The attention model can be any function that can compute the similarity between vectors, such as dot product or MLP. Then the attention weight α_i is calculated by taking a softmax function, which ensures that the weights are sum to one. This process can be represented as follows.

$$e_{ti} = f_{att}(x_i, h_{t-1}) \tag{2.10}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{2.11}$$

Finally, context vector $\hat{z}_t$ is obtained by computing weighted average of the feature vectors x_i.

$$\hat{z}_t = \sum_{i}^{L} \alpha_i x_i \tag{2.12}$$

Visual attention has also been used extensively in many recent VQA systems. In the context of a VQA task, the attention score is calculated based on the image features conditioned on the question features (e.g. the final hidden state of an LSTM). The baseline VQA model using CNN for extracting image, LSTM for extracting question, and visual attention will be discussed in detail in the next chapter.

2.5 Transformer

Transformer is a novel neural network architecture proposed in the paper "Attention is all you need" in 2017 [7]. Since its introduction, Transformer has completely changed the field of NLP and how we work with sequential data. Initially, Transformer was designed for a sequence to sequence task (e.g. machine translation), where the input sequence is transformed into another sequence. However, unlike RNN, Transformer is entirely built on attention mechanism to draw global dependencies between input and output, without the use of recurrent architecture. In this method, sentences are processed as a whole, rather than word by word in an RNN. This characteristic allows greater parallelisation, since there is no dependency between each token in the sequence. The advantages over traditional RNNs are the ability to capture long-term dependencies, as well as significantly shorter training time.

As shown in Figure 2.10, the Transformer has an encoder-decoder architecture. Both encoder and decoder are composed of modules that can be stacked on top of each other (shown as Nx in the figure). The main components of Transformer consist of positional encoding, multi-head attention and feed-forward layer. Since Transformer contains no recurrent units and process sequence as a whole, positional encoding is added to the input vector to help the model determines the position of each word. In the original paper, sine and cosine functions were used as the positional encoding method.

The encoder and decoder part of the Transformer are conceptually very similar except the part where the decoder attends to the final vectors of the encoder. This is similar to the attention mechanism explained in the section 2.4. Thus, we will be focusing on discussing the encoder part of the Transformer, particularly multi-head attention module, since they are more relevant to the understanding of this project. The underlying mechanism of multi-head attention and its interpretation is discussed in detail in the following section.

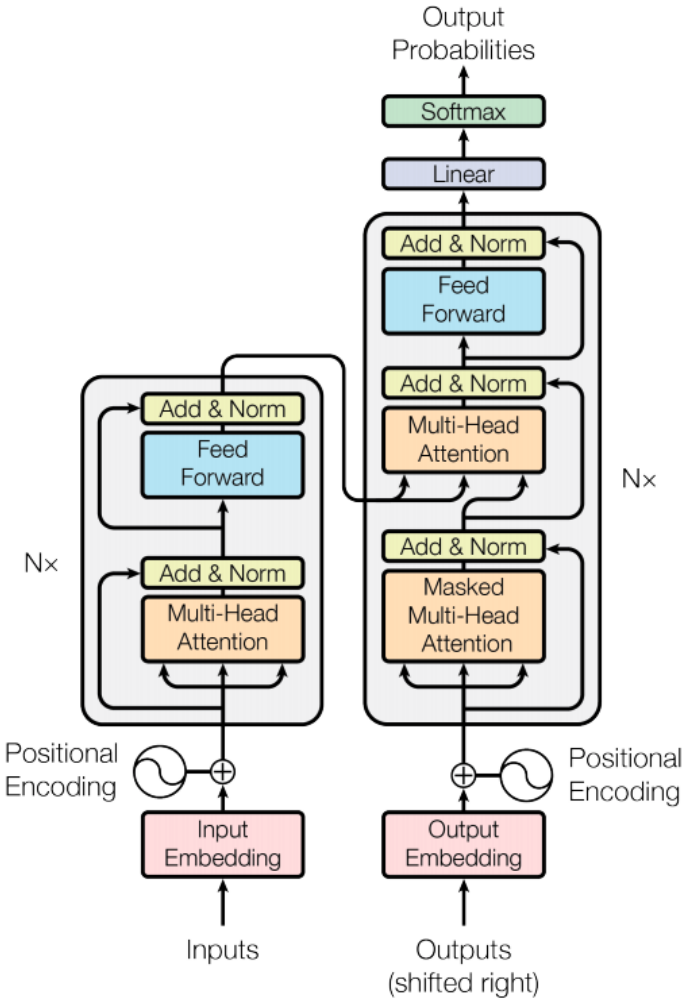


FIGURE 2.10: Overview of Transformer architecture [7]

2.5.1 Self-attention

According to the paper, "self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence" [7]. This is the core component of Transformer, and what makes the model works so well in sequence processing task. Figure 2.11 illustrates how an input sequence learns to attend to each element of its own sequence. With self-attention, the model is able to identify what the term "it" refers to. Essentially, self-attention allows the model to look at other words in the input sequence in order to get a better understanding of a word in the sequence.

Self-attention in transformer relies on the *scaled dot-product attention* to compute the attention score (similarity score) between each word and every other word. This whole process can be summarised as follows. First, the input sequences are projected into three matrices: Query (Q), Key (K), and Value (V), by multiply the input with the weight matrices which are learned in the training process. Then dot product between query and key (QK^T) is performed to discover the attention score between each input element and all other elements in the input

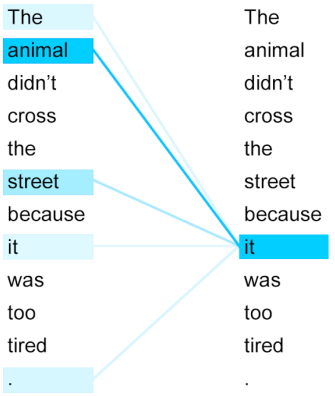


FIGURE 2.11: Visualisation of self-attention in textual input [23]

sequence. The attention scores are divided by the square root value of the key dimension $(\sqrt{d_k})$ to stabilize gradients during training, and passed through a softmax function, which ensures that the weights are sum to 1. Finally, the outputs are represented as the weighted sum of the value vector (V).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.13}$$

Rather than only computing the attention once, transformer performs scaled dot-product attention multiple times in parallel, which the paper called **"multi-head attention"**. The output from different attention heads are simply concatenated and transformed into the expected dimensions, as shown in Figure 2.12 (right). According to the paper, "Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions". As a result, individual attention head learns to perform different task, such as identifying syntactic, and semantic structure of sentences.

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}(head_1, ..., head_h)W^O \\ \text{where } head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^W) \end{aligned} \tag{2.14}$$

Where the projection matrices have the following dimensions: $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

In a high-level view, self-attention enables the model to look around the sequence, and aggregate relevant information which is helpful for the task at hand. Moreover, by stacking multiple layers of self-attention, the model would learn to capture more complex relationship between tokens in the sequence (by performing multiple attention glimpses over its sequence).

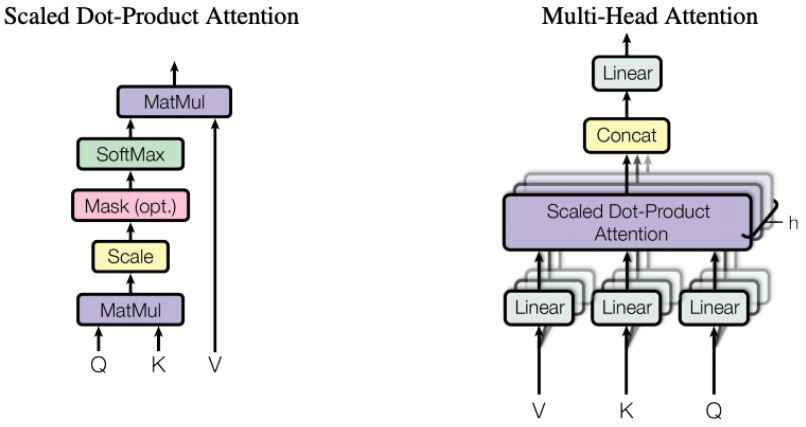


FIGURE 2.12: Multi-head self-attention mechanism [7]

2.6 BERT

The success of Transformer architecture has led to the development of many large pre-trained language representation models, such as BERT [8]. BERT stands for Bidirectional Encoder Representations from Transformers. It is based purely on the encoder part of the Transformer, which utilises self-attention to learns contextual relations between words in a text. As reported in the paper, the bidirectional training of Transformer helps facilitate deeper understanding of language context. This is in contrast with existing pre-trained language models, which only look at text sequence either from left-to-right or combined left-to-right and right-to-left training. As a result, the pre-trained BERT model can be fine-tuned to create state-of-the-art model on a wide range of NLP tasks, such as question answering and text classification, with just one additional output layer.

As BERT is applicable to a variety of downstream tasks, the input representation is required to be flexible. It needs be able to represent a single sentence as well as a pair of sentences. This is achieved by adding special input tokens: [CLS], [SEP], [MASK], and introducing another embedding to distinguish one sentence from another. As shown in Figure 2.13, a sentence embedding indicating Sentence A or Sentence B is added to each token. This can be thought as learned embedding with a size of 2. In addition, the [CLS] token is inserted at the beginning of the whole sequence, and the [SEP] token is inserted at the end of each sentence. The [CLS] token serves as a token for classification task, which is used to represent the whole sequence, while the [SEP] token is used to separate multiple sentences.

There are two stages of training in the BERT framework: pre-training and fine-tuning (see Figure 2.14).

1. **Pre-training**. BERT is pre-trained using a combination of **masked language modeling** (Masked LM) objective and **next sentence prediction** (NSP) on a large corpus comprising of Toronto Book Corpus and English Wikipedia.

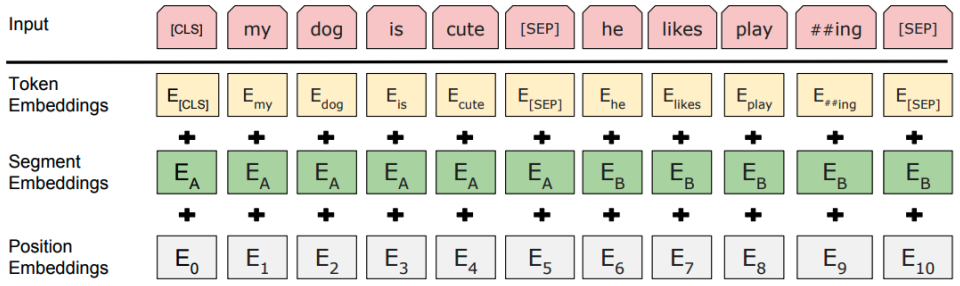


FIGURE 2.13: BERT input representation [8]

Task 1: In masked LM, 15% of the words in each sequence are replaced with a [MASK] token. Then the objective is to predict the original value of the masked word based on the context (non-masked words in the sequence).

Task 2: In NSP task, the model receives a pair of sentences as input, and the objective is to predict whether the second sentence is the actual subsequent sentence in the original document. This task allows the model to learn sentence relationships, which is beneficial for many downstream NLP tasks, such as question answering and natural language inference.

2. **Fine-tuning**. BERT can be used in a variety of language tasks, simply by attaching a classification layer on top of the core model. In the case of text classification task, such as sentiment analysis, the final hidden state of the [CLS] token is used as a pooled representation of the input sequence (see Figure 2.14 (right)). This pooled output is fed to the classification layer to obtain the prediction. During fine-tuning, the BERT model is initialised with the pre-trained parameters, then all of the parameters are fine-tuned in an end-to-end manner.

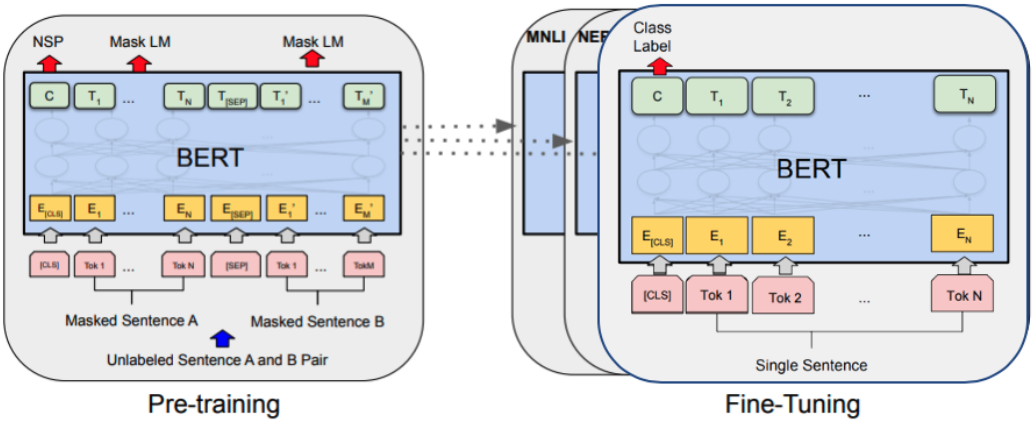


FIGURE 2.14: Overall pre-training and fine-tuning process in BERT [8]

Chapter 3

Related Works

In this chapter, we will review the most relevant works related to this dissertation. We will start by explaining the baseline VQA models, which will help with the understanding of more recent methods. Then we will study current state-of-the-art models in both VQA and visual reasoning task. After that, we will discuss about a large scale pre-trained vision and language model based on BERT architecture, which is the latest development in vision and language research. Finally, we will briefly mention some other interesting, but less related, works proposed in this area.

3.1 Baseline Models

The most common approach for a VQA task is based on visual attention mechanism. In the literature, many models based on this method have been proposed [24, 25, 26, 27], however the overall architecture of these models are quite similar to each other. In this section, we will be focusing on two of the most popular models namely - Stacked Attention Networks [24] and Bottom-Up and Top-Down Attention for VQA [26]. While the architecture of these models are relatively simple, they have been shown to achieve good results, and can be considered a strong baseline for this project.

3.1.1 Stacked Attention Networks

Stacked Attention Networks (SAN) [24] is a VQA model designed to perform multi-step reasoning by stacking multiple layers of attention mechanism. The aim is that by performing multiple visual attention, the model can look at the image multiple times to locate the relevant visual regions and to infer the answer iteratively. The overall architecture of SAN is shown in Figure 3.1. The model consists of three main components: CNN based image model, LSTM based question model, and stacked attention model:

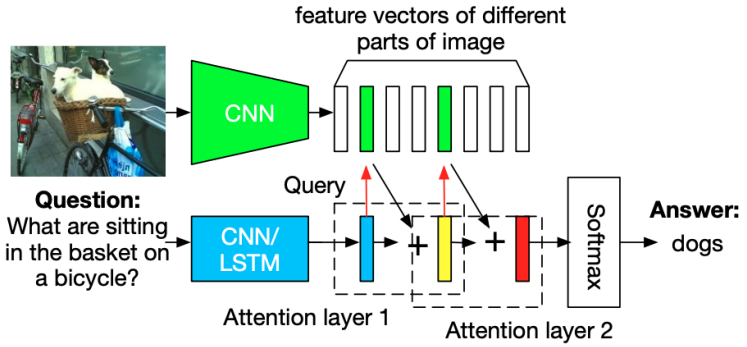


FIGURE 3.1: Overall architecture of Stacked Attention Networks [24]

1. **Image model.** In SAN, image features are extracted using a pre-trained VGG [15] from raw images. First, the input images (I) are re-scaled to 448×448 pixels. Then the features from the last pooling layer, which retains spatial information are used. As shown in Figure 3.2(B), the obtained image features have the dimension of $512 \times 14 \times 14$ ($depth \times width \times height$). Thus, there are 196 ($14 \times 14$) regions of the image, where each region corresponds to a 32×32 pixel regions of the input image. Finally, the image features f_I are projected to the same dimension with question features. This can be summarised as follows:

$$f_I = \text{CNN}_{vgg}(I) \tag{3.1}$$

$$v_I = \tanh(W_I f_I + b_I) \tag{3.2}$$

where $v_I \in \mathbb{R}^{d \times m}$ is the matrix of image representation with d dimension and m number of regions.

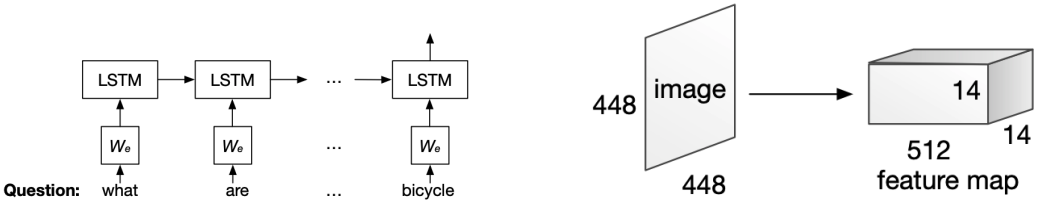
2. **Question model.** Given the question $q = [q_1, ..., q_T]$, with each word encoded as a one hot vector, the words are first embedded into vector space by multiply with an embedding matrix (W_e). Then the embedding vector of words are fed through an LSTM for every timestep. Finally, the last hidden state is taken as the representation vector of the question $v_Q \in \mathbb{R}^d$ (see Figure 3.2(A)).

$$x_t = W_e q_t, t \in \{1, 2, ..., T\} \tag{3.3}$$

$$h_t = \text{LSTM}(x_t), t \in \{1, 2, ..., T\} \tag{3.4}$$

$$v_Q = h_T \tag{3.5}$$

3. **Stacked attention model.** The primary concepts of stacked attention model are similar to visual attention explained in section 2.4.1. Given the image feature v_I, and the question feature v_Q, multiple (K) attention layers are used to select the regions in the image that are relevant to the potential answers. This can also be considered as the process to filter the noises from regions that are irrelevant. In SAN, attention



(A) LSTM based question model (B) CNN based image model

FIGURE 3.2: SAN input features extractor [24]

distribution over region of image is calculated by using a single layer neural network, follow by a softmax function:

$$\begin{aligned} h_A &= \tanh(W_{I,A} v_I \oplus (W_{Q,A} v_Q + b_A)) \\ p_I &= \text{softmax}(W_p h_A + b_P) \end{aligned} \tag{3.6}$$

where $\oplus$ denotes the addition between a matrix and a vector. Based on the attention distribution, the context vector ($\hat{v}_i$) can be calculated by weighted sum of image vector. This context vector represents the new image features where the noise from irrelevant regions are filtered. Then $\hat{v}_I$ is combined with the question vector v_Q to form a new query vector u. u can be regarded as a refined query, since it encodes both question information and relevant visual information. This can be expressed as follows.

$$\hat{v}_I = \sum_i p_i v_i \tag{3.7}$$

$$u = \hat{v}_I + v_Q \tag{3.8}$$

This process is repeated K times, then the final u^K is used to find the answer.

$$p_{ans} = \text{softmax}(W_u u^K + b_u) \tag{3.9}$$

In the paper, the number of visual attention layer K is set to 2, which is shown to be effective in a variety of VQA datasets. Figure 3.3 shows that SAN is able to progressively locate relevant region of the image, when trying to answer the question "What are sitting in the basket on a bicycle?".

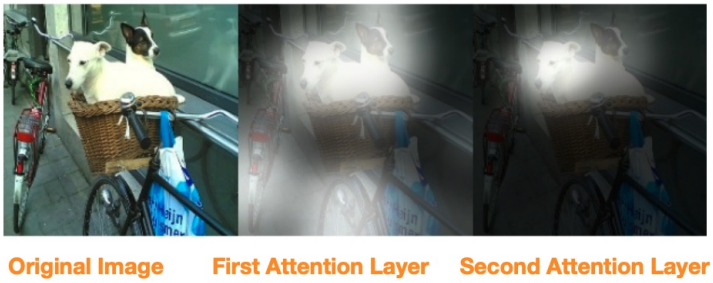


FIGURE 3.3: Visualisation of learned attention map of SAN [24]

3.1.2 Bottom-Up and Top-Down Attention for VQA

Bottom-Up and Top-Down Attention for VQA (BUTD) [26] is a model inspired by human visual system. In the paper, they adopted the terminology from neuroscience field, where **top-down attention** refers to the focused and volitional attention that is determined by the current task, and **bottom-up attention** refers to automatic attention that is arose by salient stimuli. Visual attention used in most VQA models (discussed in the previous section) are of top-down variety. The main difference between BUTD and the previous model (SAN) is the use of bottom-up attention. As bottom-up attention, BUTD uses image regions proposed by an object detector model (Faster R-CNN [28]) to represent image features. Figure 3.4 shows the difference between input image features between BUTD and other VQA models that use grid-based feature map from CNN. Consequently, bottom-up mechanism enables attention to be calculated at the level of objects and other salient regions. Experiments reported in the paper shows that BUTD was effective in both image captioning and VQA task and was the model that won the VQA challenge in 2017.

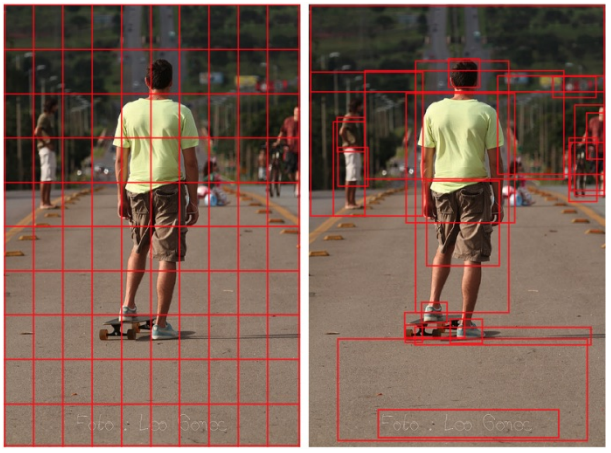


FIGURE 3.4: Grid based features vs Object based features [26]

3.2 Visual Reasoning Models

Despite achieving strong results in various VQA datasets, attention-based models discussed in the previous section are not very competitive in visual reasoning datasets which requires more sophisticate natural language understanding and reasoning. In this section, we will look into two models which are designed specifically to solve a visual reasoning task.

3.2.1 MAC Networks

Memory, Attention, and Composition (MAC) network is an end-to-end differentiable neural network model that is designed to facilitate explicit and expressive reasoning [29]. It is based

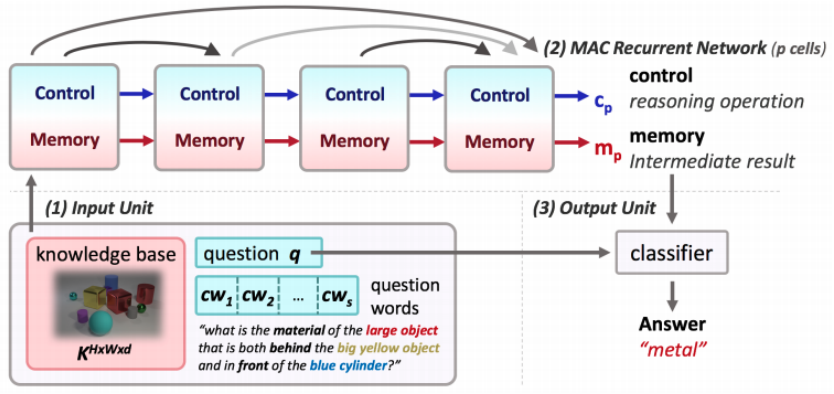


FIGURE 3.5: Overview of MAC Networks [29]

on the use of attention mechanism, internal memory, and recurrent structure. Similar to RNN, the model consists of multiple MAC cells chained together, with each cell contains two hidden states: control and memory (see Figure 3.5). In contrast to the traditional neural networks, MAC network has clear separation between control and memory. In each MAC cell, the control component is responsible for identifying the reasoning operation, while the memory component is responsible for storing the intermediate result. Consequently, this structural prior of the MAC network encourages the model to perform iterative reasoning process by decomposing the problem into a sequence of operation, rather than to approximate direct transformation between the input and output.

As shown in Figure 3.5, the MAC networks comprise of three components:

1. **Input unit.** The input unit is similar to other VQA models where raw images and questions are transformed into vector representation. In this model, LSTM and ResNet101 are used to extract the input features.

2. **Recurrent MAC cells.** The MAC cell consists of a control unit, read unit, and write unit, which interact with control and memory hidden states (see Figure 3.6).

   (a) The **control unit** successively attends to different parts of the questions, and then updates the control state to represent the reasoning operation the cell intends to perform.

   (b) The **read unit** extracts relevant information from image guided by the control state.

   (c) The **write unit** integrates the retrieved information into the memory state, yielding the new intermediate result, by applying the current reasoning operation.

3. **Output unit.** The final memory state and question vector are concatenated, and then fed into a softmax classifier to produce an answer.

As a result, the MAC networks were able to achieve 98.94% accuracy on CLEVR dataset, halving the error rate from the best prior model. Analysis reported in the paper also shows

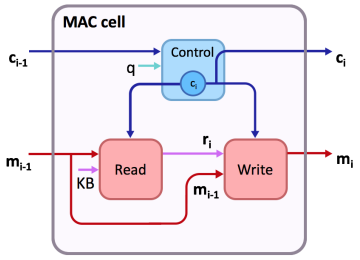


FIGURE 3.6: MAC cell [29]

that MAC networks are able to learn significantly faster and more data efficient than other approaches (using only 10% of the data to achieve 85% accuracy). This work shows that inductive bias can improve both the accuracy as well as the efficiency of learning. In addition, MAC networks allow greater transparency and interpretability, as the attention maps of the model can reveal how the model come up with an answer. This is in contrast to the traditional deep learning models, which are considered a black-box system. However, the biggest weakness of the MAC networks is that the performance of this model on the datasets containing real-world images, such as GQA [6] is not very competitive.

3.2.2 The Neuro-Symbolic Concept Learner

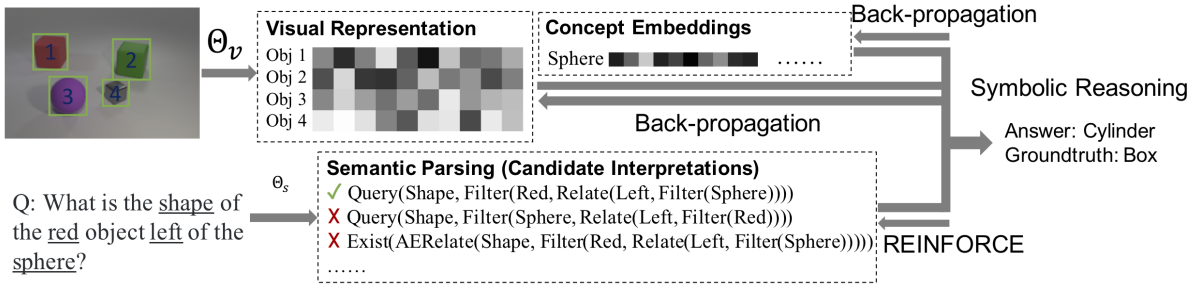


FIGURE 3.7: Overview of Neuro-Symbolic Concept Learner [30]

Unlike all previous models, which are based on end-to-end fully differentiable neural network, Neuro-Symbolic Concept Learner (NS-CL) [30] is a model rooted in the concept of **neural-symbolic AI**. The aim of neural-symbolic approach is to combine symbolic AI's ability to reason with neural networks' ability to learn from experience [31]. Similar to human concept learning, NS-CL is able to learn visual concepts, words, and semantic parsing of sentences without explicit supervision. In NS-CL framework, deep neural network is applied for visual recognition and language understanding, while the reasoning part is solved by symbolic program execution. In addition, NS-CL employed curriculum learning to help with optimisation, where images and questions are presented to the model lesson by lesson with increasing difficulty.

As shown in Figure 3.7, the model is composed of four components:

1. **Visual Module**. Given the input image, object proposals are generated by a pre-trained object detector (Mask R-CNN), which is then sent to ResNet-34 to extract the region-based feature by performing an ROI Align.

2. **Concept Embeddings.** For each object in a scene, the extracted features are linearly mapped (neural operator) to an attribute embedding space. Then the object attribute (e.g. sphere) can be determined by measuring the cosine distance between the concept vector with the mapped object vector.

3. **Semantic Parser**. The semantic parsing module translates a natural language question into a hierarchy of pre-defined domain specific language (DSL) of executable program. The translation is performed by using a recurrent neural network (GRU) stacked in an encoder-decoder manner.

4. **Symbolic Program Executor**. The program executor executes the parsed program and derives the answer based on the object-based visual representation.

The experiments reported in [30] shows that this method requires significantly less amount of training data to accurately answer questions and generalise well to scenes with more objects and more complex questions. Moreover, its results are fully interpretable, as the execution trace is visible. Figure 3.8 shows the result of this model on the VQS dataset, along with the generated symbolic programs. However, one significant limitation is that the symbolic functions (DSL) need to be pre-defined. As shown in Figure 3.8, functions such as filter, count and query are hard coded into the system, which make it difficult to translate into real-world system.

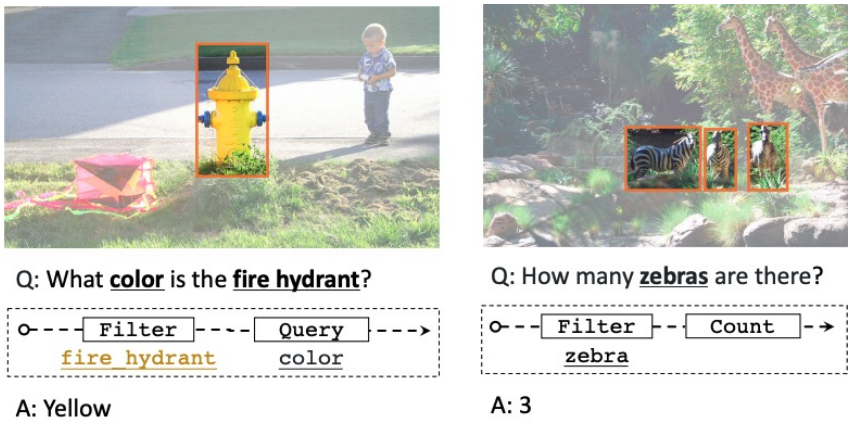


FIGURE 3.8: Result of NS-CL on the VQS dataset [30]

3.3 State of the art VQA Models

In this section, we will look at the current state-of-the-art deep learning model that has won both the 2019 and 2020 VQA challenge[1].

3.3.1 Deep Modular Co-Attention Networks

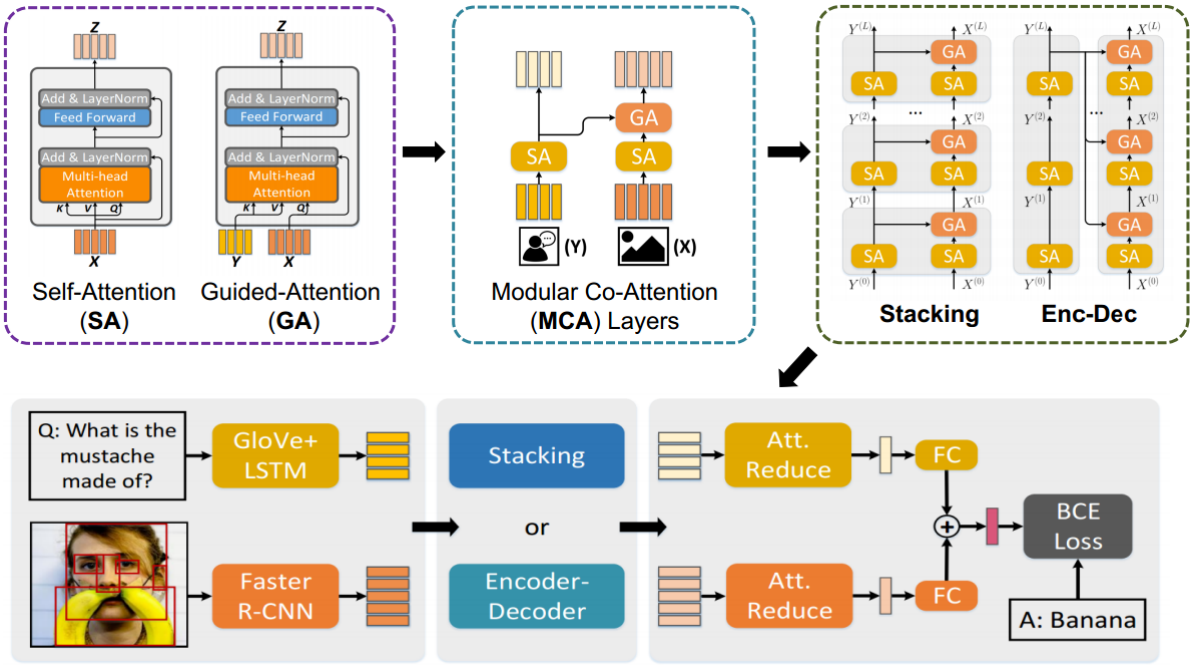


FIGURE 3.9: Overall architecture of Deep Modular Co-Attention Networks [32]

Deep Modular Co-Attention Networks (MCAN) is a VQA model inspired by the Transformer architecture [32]. It employs self-attention and guided-attention unit proposed in the Transformer model (discussed in section 2.5). As shown in Figure 3.9 (top), these two basic units comprise the Modular Co-Attention Layer (MCA), which is the core component of this model. The MCA layer enables both dense intra-modal and inter-modal interactions. The self-attention unit enable the model to learn word-to-word relationship in questions and region-to-region relationship in images, while the guide-attention unit facilitates the understanding of image-question relationship. In addition, the MCA layer can be stacked in depth, which allows for more accurate visual reasoning.

Figure 3.9 (bottom) shows the overall framework of MCAN. To obtain the question features, words in the question are first transformed into vector by using GloVe word embedding [33]. Then word vectors are sent to LSTM to obtain the contextual representation of each word. Image features are represented in a bottom-up manner (see section 3.1.2) obtained from Faster R-CNN object detector model. Both question and image features are then sent through

[1] https://visualqa.org/challenge.html

multiple layers of MCA, to obtain fine-grained representation in each modality. Finally, outputs from the final MCA layer go through the attention reduction stage, which is then fused by element-wise addition for classification.

Analysis from the paper shows that there are four key aspects that enable MCAN to achieve state-of-the-art result in a VQA task: inter-modal attention modeling, intra-modal attention modeling, dense interactions modeling, and a deep model. Fundamentally, these characteristics are due to the design of the model that is based on the Transformer architecture. It was also reported that modeling self-attention for image features greatly improved the performance of object counting, which has been challenging for VQA.

3.4 Pre-trained Vision and Language Models

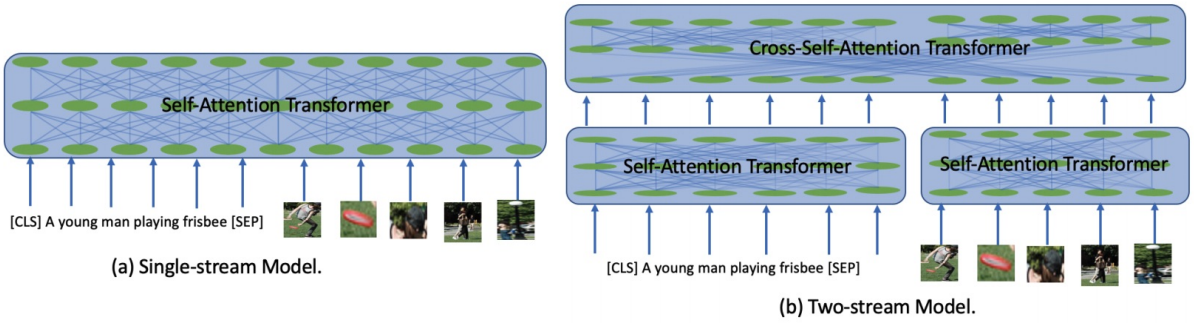


FIGURE 3.10: Two types of pre-trained vision and language models [34]

Inspired by the success of BERT (see section 2.6), many large-scale pre-trained models based on the Transformer architecture have emerged in vision and language (V+L) research [35, 36, 37, 38, 39]. These "BERT with vision" models use Transformer to jointly model words and image regions, resulting in a visually grounded language model. The core idea is the use of self-attention mechanism to implicitly align words of the input text with regions in the input image. As shown in Figure 3.10, there are mainly two categories of model design: single-stream model and two-stream model. In a single stream model, both text and image features are input to a single Transformer. On the other hand, in a two-stream model, two Transformers are independently applied to text and image features, which is then fused by another layer of Transformer. The aim of these models is to learn task-agnostic joint representations of image content and natural language, which can then be applied to a variety of V+L downstream tasks, such as VQA, image captioning and image-text retrieval.

As shown in Figure 3.10, these models typically use regions of image detected from Faster R-CNN (bottom up attention) as a visual input, while the textual input are kept the same as the original BERT. In addition, segment embedding is introduced to distinguish between the text and visual token. In contrast to BERT, which can be pre-trained on an unlabeled text corpus, these BERT with vision models requires training data that contains a pair of image

and text caption. In particular, image captioning dataset such as COCO [40] and conceptual caption [41] have been extensively used as a pre-training data.

To learn generic visual-linguistic representation, several pre-training objectives have been introduced:

1. **Masked language modeling with image.** Similar to the original BERT, some element in the text input are masked, and the objective is to predict the word based on other words and image regions.
2. **Sentence-image alignment.** Given an image and a sentence as input, the objective is to predict whether the text describes the image.
3. **Masked visual-feature classification.** The objects in the visual input are masked randomly, and the objective is to classify the masked region based on visible object tokens as well as the text input.

When finetuned, these pre-trained models were able to achieve new state-of-the-art results on a variety of downstream tasks, including VQA, visual reasoning, image captioning and visual entailment. This indicates that the pre-trained models have learned substantial amount of visual and linguistic knowledge. Analysis on these models [34, 42] have shown that some attention heads in Transformer perform entity grounding (e.g. image regions are correctly mapped to the corresponding word) and syntactic grounding (e.g. non-entity words such as wearing and sitting are correctly mapped to its subject in the image).

3.5 Other Approaches

In addition to the methods discussed, there are a variety of models proposed for VQA and visual reasoning based on alternative approaches. These models build on other recent advances in deep learning, such as the use of external memory (Memory Augmented Neural Networks) [43, 44] and graph structure (Graph Neural Networks) [45]. For example, the model proposed in [46] employs external memory to accurately predict answer which rarely occurs in the training set. Another recent model called Neural State Machine [47] decomposed image into probabilistic graph that captures the semantic relation of the visual scene, then sequential reasoning process is performed over the graph structure to derive the answer. Another common approach for solving visual reasoning is based on modular neural network [48, 49, 50]. This type of network composes of a collection of pre-defined neural modules, where each module is responsible for an elementary reasoning operation, such as identifying object's colour and counting. In modular neural network, the questions are first translated into an action plan, then the network is constructed dynamically based on the plan using neural modules to obtain the answer.

Chapter 4

Proposed VQA Model

In this chapter, we will explain the architecture of our proposed VQA model in detail. We will start by discussing the rationale behind the design of each component. We will also mention the ideas that are inspired by recent developments in deep learning and vision and language research (V+L) literature. Finally, we will briefly talk about the initial design of our VQA model that fails to achieve the desired accuracy in a visual reasoning task.

4.1 Model Design Motivation

As stated previously, the goal of this project is to develop a VQA model based on Transformer architecture for solving a visual reasoning task. The hypothesis is that Transformer's self-attention mechanism would increase the model's capability in performing multi-step reasoning, as well as high level skills, such as counting and making comparison, which are the required skills in this task. Similar to human, the model with multiple layers of attention mechanism would be able to perform multiple glimpses of the image and question before coming up with an answer. In addition, self-attention and co-attention mechanism would enable dense intra-modal and inter-modal interaction between image and question, which would result in higher quality feature representations.

Similar to current state-of-the-art VQA model, MCAN [32] (see section 3.3.1), our model employs GloVe word embedding [33], followed by biLSTM to extract the input question, while region-based representation from bottom-up attention [26] is used to extract visual features. These input models are chosen, because they are already proved to be effective for a variety of VQA and visual reasoning datasets. To adapt the Transformer architecture for a multimodal task, we take inspiration from two-stream pre-trained vision and language models (see section 3.4), such as ViLBERT [37] and LXMERT [38]. In particular, our model adopted the Co-Attention Transformer layer and pooling strategy from VilBERT, due to the simplicity and similarity to the original BERT model.

4.2 Model Architecture

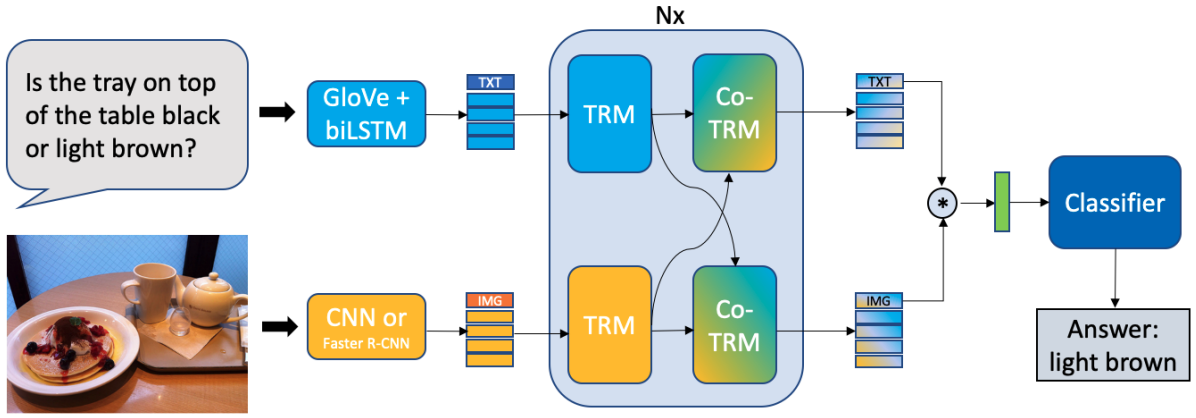


FIGURE 4.1: Our proposed model architecture

As shown in Figure 4.1, the architecture of our model consists of input feature extractors, a stack of multimodal Transformer layer (Nx), multimodal fusion layer, and a classifier. The main configurable parameters of our model are the hidden dimension of the model (d_{model}), the number of Transformer layer (N), the number of attention heads in the Transformer layer (A), and the hidden dimension of feed-forward networks in the Transformer layer (d_{ff}). These hyperparameters can be adjusted based on the difficulty of the dataset.

The input feature extractors (CNN, LSTM) and the classifier components are similar to most existing VQA models. However, the main differences between our proposed model and existing VQA models are the use of Co-Attention Transformer layer and how they are stacked together, as well as the pooling strategy, which we utilise the [IMG] and [TXT] token as a pooled representation of the image and question. The details of each component are explained as follows.

4.2.1 Input Representations

4.2.1.1 Image Representation

The input image can be represented either using the region-based visual features extracted from Faster R-CNN (see section 3.1.2) or the grid-based feature map from pre-trained ResNet. In the case of region-based visual features, up to 100 detected objects (m regions) are used as the representation of the input image, and each region x_i has the dimension of 2048 (d_x):

$$F_I = \text{FasterRCNN}(Image) \tag{4.1}$$

where $F_I \in \mathbb{R}^{m \times d_x}$ is the feature matrix of the input image.

In the case of grid-based visual features, the output feature maps from conv4 layer from ResNet-101 are used as the representation of the input image. The obtained visual features have the spatial dimension of 14 x 14, which are then flattened to be in 1D before going through the next stage, resulting in 196 grid regions (m). Each grid location x_i contains the feature with 1024 dimension (d_x). This can be represented as follows:

$$F_I = \text{ResNet}(Image) \tag{4.2}$$

where $F_I \in \mathbb{R}^{m \times d_x}$ is the feature matrix obtained from conv4 layer of the pre-trained ResNet-101.

The obtained visual features are then linearly projected to be the same hidden dimension of the model (d_{model}) in order to be sent to the Transformer layer:

$$F_v = W_v F_I + b_v \tag{4.3}$$

where $F_v \in \mathbb{R}^{m \times d_{model}}$ is the projected image features, $W_v \in \mathbb{R}^{d_x \times d_{model}}$ is the weight matrix, and $b_v \in \mathbb{R}^{d_{model}}$ is the bias of the linear function.

4.2.1.2 Question Representation

The question text is first tokenised into a sequence of words $q = [q_1, ..., q_n]$, with each word is represented as a one hot vector from the index of the dataset's vocabulary. Each word vector is then transformed into a distributed vector representation using 300D GloVe word embedding:

$$E = qW_e \tag{4.4}$$

where $E \in \mathbb{R}^{n \times 300}$ is the embedded question, n is the number of words in the longest question, and W_e is the GloVe embedding weight. The question with number of words less than n is padded with the [PAD] token. Then the embedded question is fed through a bidirectional LSTM with d_{model} hidden dimension. In contrast to SAN [24], the output from each timestep of LSTM is used to represent the contextual word vectors with both left and right contexts:

$$\overrightarrow{h}_t, \overleftarrow{h}_t = \text{biLSTM}(E_t), t \in 1...n \tag{4.5}$$

$$h = [\overrightarrow{h}_0; \overleftarrow{h}_0, ..., \overrightarrow{h}_n; \overleftarrow{h}_n] \tag{4.6}$$

where $h \in \mathbb{R}^{n \times (2d_{model})}$ is the question features matrix. Finally, the obtained feature matrix is projected to be the same dimension as the hidden dimension of the model:

$$F_q = W_Q h + b_Q \tag{4.7}$$

where $F_q \in \mathbb{R}^{n \times d_{model}}$ is the final textual representation.

4.2.1.3 Special Tokens

Similar to BERT, we use two special tokens: [IMG] and [TXT], as the pooled representation of the image and question respectively. These tokens are added to extracted information related to answer from both visual and textual input in the Transformer layer. This pooling strategy is relatively simple and has shown to be effective in many Transformer based models [8, 51, 37].

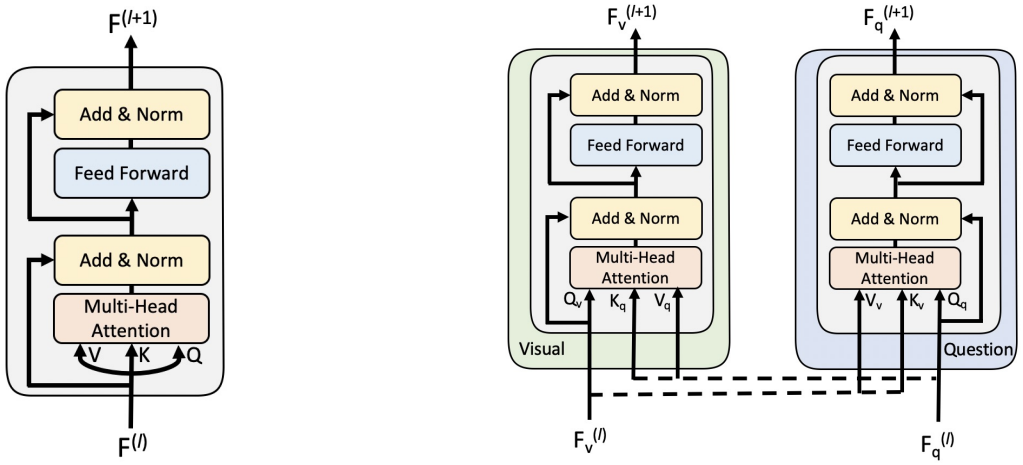
First, both [IMG] and [TXT] token need to be added to the list of vocabulary in the dataset. Then these tokens can be generated by multiplying each of the token id with the word embedding weight and transformed to the dimension of the model d_{model}. Finally, the [IMG] token (f_{img}) is inserted to the beginning of the image features F_v, while the [TXT] token (f_{txt}) is inserted to the beginning of the question features F_q:

$$F_v = [f_{img}; F_v] \tag{4.8}$$

$$F_q = [f_{txt}; F_q] \tag{4.9}$$

where F_v and F_q are the final input representation, which are sent to the Transformer layer, and $[.;.]$ represent the concatenation function (see Figure 4.1).

4.2.2 Multimodal Transformer



(A) Standard Transformer layer (TRM)

(B) Co-attention Transformer layer (Co-TRM)

FIGURE 4.2: Multimodal Transformer block

The multimodal Transformer encoder block consists of two components: standard Transformer layer (TRM), which is the same as the original BERT model, and Co-attention Transformer layer (Co-TRM). These two layers are based on multi-head attention concept discussed in section 2.5.1. However, in Co-TRM layers, the keys and values from each modality are swapped (see Figure 4.2 for the difference). The attention mechanism in Co-TRM layers can be interpreted as question-conditioned image attention in the textual

stream and image-conditioned question attention in the visual stream. The image attention conditioned on the question is similar to visual attention found in many existing VQA models (discussed in section 2.4.1 and 3.1.1).

In our multimodal Transformer layer, each input modality goes through the standard Transformer layer, follows by the co-attention Transformer layer (see Figure 4.1). Similar to the original Transformer model, our multimodal Transformer layer can be stacked in order to enable the model to perform multiple glimpse of the image and question, as well as to capture more complex relationship between the two modalities.

As shown in Figure 4.2, both Transformer blocks contain a multi-head attention module and a position-wise feed forward neural network with a residue connection and layer normalisation after each module.

4.2.2.1 Multi-Head Attention

Both TRM and Co-TRM layers rely on the scaled dot product attention as the underlying mechanism to calculate the attention score between its own input sequence (self-attention) and two sequences of input (co-attention). First, both the extracted question features (F_q) and image features (F_v) are projected to a set of query (Q), key (K) and value (V) matrices. Then the attention weight assigned to each value is computed by dot product between the queries (Q) and the keys (K). Finally, the output is represented as the weighted sum of the values (V):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.10}$$

where d_k is the dimension of the key. The only difference between TRM and Co-TRM units is that, in Co-TRM layers, the key and value matrices from each modality are passed as input to the other modality to calculate the scaled dot product attention.

As in the original Transformer, multi-head attention is introduced to improve the representation capacity of the attended features. The multi-head attention module consists of h parallel heads, with each head performs independent scaled dot product attention. The output of the multi-head attention is given by:

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}(head_1, ..., head_h)W^O \\ \text{where } head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^W) \end{aligned} \tag{4.11}$$

where the projection matrix of each head has the following dimensions:
$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model} \times d_k}$ and $W^O \in \mathbb{R}^{hd_k \times d_{model}}$ is the projection matrix for the concatenate output of the multi-head attention. All the projection weight matrices are learned during training.

4.2.2.2 Position-Wise Feed Forward Networks

In addition to the attention layer, each Transformer block contains a fully connected feed-forward network, which consist of two layers MLP with a RELU activation function in between. The same network is applied to each element in the input sequence (called position-wise), but not to different blocks.

$$\text{FFN} = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{4.12}$$

4.2.2.3 Residue Connection and Layer Normalisation

Residue connection and layer normalisation are applied after each sublayer to help with optimisation:

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \tag{4.13}$$

where sublayer includes the multi-head attention and position-wise feed forward networks.

4.2.3 Multimodal Fusion and Classifier

The output of [IMG] and [TXT] tokens from the final layer of multimodal Transformer are used as the pooled representation of image and question. Similar to BERT model, both outputs are transformed before passing through the classification layer:

$$f^*_{img} = \tanh(f^n_{img}W_i + b_i) \tag{4.14}$$

$$f^*_{txt} = \tanh(f^n_{txt}W_t + b_t) \tag{4.15}$$

where f^n_{txt} is the hidden state of [TXT] token from the final layer of Transformer, and f^n_{img} is the hidden state of [IMG] token from the final layer of Transformer.

After obtaining the projected features, f^*_{img} and f^*_{txt}, they are fused by element wise multiplication.

$$f_{fuse} = f^*_{img} * f^*_{txt} \tag{4.16}$$

Finally, the fused features (f_{fuse}) are passed through the 2 layers fully connected softmax classifier to obtain the distribution over candidate answers.

$$\text{answer} = \text{softmax}(\text{ReLU}(f_{fuse}W_1 + b_1)W_2 + b_2) \tag{4.17}$$

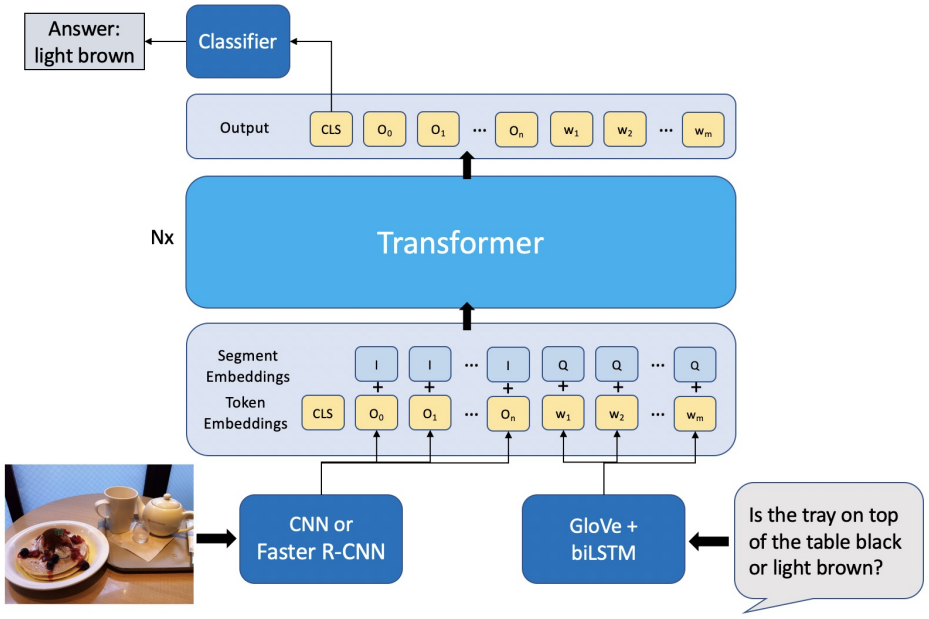


FIGURE 4.3: Initial model design

4.3 Initial Design

Initially, our VQA model was designed to be based purely on self-attention mechanism from standard Transformer layer, which would be similar to single-stream pre-trained vision and language models (see section 3.4), such as Pixel-BERT [39] and VisualBERT [35]. In this architecture, the features extracted from image and question are simply concatenated then fed through multiple layers of standard Transformer encoder. Similar to the original BERT model, segment embedding is introduced to distinguish between the two modalities. In this design, [CLS] token is used to extract information related to the answer, and the hidden state of [CLS] token from the final layer of Transformer is used as an input to the classifier layer. Figure 4.3 shows the overall architecture of our initial design.

The simplicity of this architecture is compelling; however, the model performs poorly in a visual reasoning task. We have tried different learning rates, optimisers, regularisation methods. We have also incorporated positional embeddings for both image and text input similar to [37], however the model was unable to achieve the desired accuracy, only achieving result similar to baseline models in a visual reasoning task.

Chapter 5

Experiments and Results

In this chapter, we will discuss in detail the experiment we performed on two visual reasoning datasets. We will first talk about the implementation of our VQA model, and the environment that we have set up to run the experiment. We will also give an overview of the two datasets, namely CLEVR [5] and GQA [6] and specify the detail of our hyperparameter choices and model settings for each of the experiment. Then we will show the results of our proposed model from both datasets and compare them with current state-of-the-art models. Finally, analysis of the model and discussion on the results will be given.

5.1 Model Implementation

Our VQA model is implemented using Python and PyTorch deep learning library [52]. We chose PyTorch because of its flexibility and simplicity. Moreover, there are several PyTorch based frameworks, developed to facilitate vision and language research. In this project, we make use of two frameworks, MMF [53] and OpenVQA [54], to help with the development of our VQA model. These frameworks provide built-in support for various VQA datasets, training loop, configuration managements, etc, with minimal modification required to train the new model, enabling us to focus on the development of the model. In addition, we based our implementation of Transformer related components, such as multi-head self-attention unit on Hugging Face's Transformer library [55]. This is to ensure the correctness, as well as to speed up the development process of the Transformer component.

In addition to the proposed model, we also implemented several baseline VQA models, such as CNN+LSTM and SAN (see section 3.1.1), to gain further understandings of the library and the underlying implementation of several deep learning components, such as attention mechanism. From implementing the baseline models, we obtained the benchmark to improve upon. The code for our initial model and baseline VQA models is available at `https://github.com/markvasin/vqa_project`. This repository contains the models that are

made to run on MMF framework. However, due to the ease of usage, we later changed the framework to OpenVQA. Thus, our final model can be found at https://github.com/markvasin/openvqa.

5.2 Experimental Environment

We have set up GPU server on Google cloud platform specifically for training the model. The server has 8-core Intel(R) Xeon(R) CPU @ 2.00GHz, Tesla T4 GPU, 30GB RAM and 500GB SSD persistent disk. In addition, we also utilised experiment tracking tools, such as *Weights and Biases* and TensorBoard to keep track of the hyperparameters, training loss, learning curve and accuracy of each run. All of the experiments can be viewed online at https://app.wandb.ai/mvsmark/.

5.3 Experiment on CLEVR Dataset

We first evaluated our VQA model on the CLEVR dataset [5]. Introduced in 2017, CLEVR is a diagnostic dataset designed to test a range of visual reasoning abilities. The dataset consists of synthetic images and questions generated from functional program. It was considered to be quite challenging, as many standard deep learning based VQA models performed very poorly.

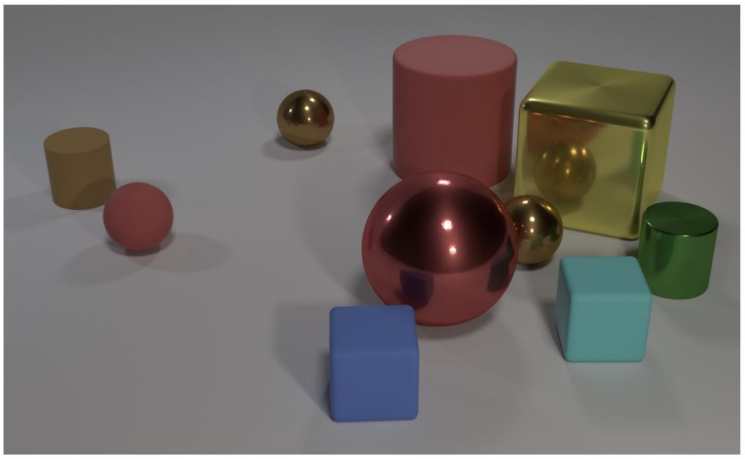


FIGURE 5.1: Example of CLEVR dataset [5]

5.3.1 CLEVR Dataset

The CLEVR dataset [5] was developed to test a wide range of reasoning abilities of the VQA models, including counting, comparing, logical reasoning, spatial reasoning, and attribute identification. Its aim is to study the ability of VQA systems to perform visual reasoning, as well as to discourage the VQA models from developing the Clever Hans effect. Clever Hans was a horse in the 1900s era who appeared to be able to perform arithmetic tasks, but later revealed to be relying on the clues of the trainer. Similarly, statistical learning systems, like many VQA models, may develop a cheating process to answer questions by exploiting biases of the dataset without any reasoning.

As illustrated in Figure 5.1, the questions in the dataset are designed to be highly compositional and requires long chains of complex reasoning. In order to correctly answer these questions, the model would need to be able to perform multi-step reasoning. For example, to answer the question of *"Are there an equal number of large things and metal spheres?"*, the model would first need to identify all the large things, and then identify the metal spheres, then count the number of identified items in each category, and finally compare the number of objects between the two categories.

The dataset is made up of over 850K automatically generated questions and 100k synthetic images. Each image comes with a scene graph that give ground-truth locations, attributes, and relationships for objects, and each question is provided with the tree-structured functional program that was used to generate the question. Images consist of 3D-rendered objects of three shapes (cube, cylinder, and sphere), two materials (shiny "metal" and matte "rubber"), two sizes (small and large), and eight colours. There are four types of spatial relationship between objects in the image: "left", "right", "behind", and "in front". Every image contains between three and ten objects with random positions, shapes, materials, sizes, and colours. The distribution of answer is also carefully balanced by performing rejection sampling, to minimise question-conditional bias in the dataset.

In the CLEVR dataset, there are mainly five types of question: counting, integer comparison, existence, querying attributes, and comparing attributes. Counting questions ask for the number of objects satisfying certain conditions (e.g. *"How many red cubes are there?"*). Integer comparison questions ask which of two object sets is larger (e.g. *"Are there fewer cubes than red things?"*). Existence questions ask whether a certain type of object exists in the image (e.g. *"Are there any cubes to the right of the red thing?"*). Query questions ask about an attribute of a particular object (e.g. *"What color is the thing right of the red sphere?"*). Finally, attribute comparison questions ask whether two objects have the same attribute (e.g. *"Is the cube the same size as the sphere?"*).

5.3.2 Experimental Settings

The hyperparameters of our Transformer component are selected based on a smaller setting of the original BERT_{BASE} model. This is because the structure of the data in the CLEVR dataset is less complex compare to natural language datasets used in BERT. In CLEVR, the images contain minimal noises, as they are synthetically generated, and the questions are created by pre-defined program with small vocabulary size. Table 5.1 summarises the hyperparameters of our VQA model for the CLEVR experiment. To train the model, we use ADAM optimiser [56] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, and use cross entropy as a loss function. The model is trained for 16 epochs with batch size of 64. The learning rate is set to $4e^{-5}$, and after 13 epochs, the learning rate is decayed by $1/5$ every 2 epochs.

Model Hyperparameters	**Value**
Model's hidden dimension (d_{model})	512
Number of multimodal Transformer layer (N)	6
Number of attention heads (A)	8
Hidden dimension of feed-forward networks (d_{ff})	2048
Dropout	0.1

TABLE 5.1: Model hyperparameters for CLEVR experiment

For this experiment, we use grid-based feature map extracted from pre-trained ResNet-101 to represent the visual features. Although some previous works make use of scene graph and/or functional program associated with each image and question as additional supervisory signal, we intentionally did not use them to train our model, as we want the model to infer the underlying reasoning processes directly from the question and answer pairs in an end-to-end approach. With the GPU server we have set up, the training took roughly around 60 hours (3.5 hours per epoch).

5.3.3 Results

Table 5.2 summarises the performance of our model along with other methods on CLEVR's validation set. Our model achieved the overall accuracy of **98.3%**, outperforming humans and many existing methods. This accuracy is comparable to current state-of-the-art models, such as MAC and NS-CL (discussed in section 3.2), which achieved 98.8%. Comparing to SAN (the baseline model mentioned in section 3.1.1), our model achieved significantly higher accuracy (98.3% vs 73.2%). In addition, our model outperforms module based neural networks (N2NMN and IEP), even though they rely on functional program as extra supervision.

As shown in Figure 5.2, our model consistently achieved high accuracy in all question categories. This result shows that utilising Transformer's self-attention and co-attention unit highly helps improve the performance for a visual reasoning task, since the main difference between SAN (baseline model) and our model is the use of multimodal Transformer layers

Model	Count	Compare Numbers	Exist	Query Attribute	Compare Attribute	Overall
Human [49]	86.7	96.6	86.5	95.0	96.0	92.6
Q-type Baseline [49]	34.6	50.2	51.0	36.0	51.3	41.8
LSTM [49]	41.7	61.1	69.8	36.8	51.8	46.8
CNN+LSTM [49]	43.7	65.2	67.1	49.3	53.0	52.3
SAN (baseline model) [24]	59.7	77.9	75.1	80.9	70.8	73.2
N2NMN* [50]	68.5	85.7	84.9	90.0	88.7	83.7
IEP* [49]	92.7	98.7	97.1	98.1	98.9	96.9
Relation Networks [57]	90.1	97.8	93.6	97.9	97.1	95.5
FiLM [58]	94.3	99.3	93.4	99.3	99.3	97.6
NS-CL† [30]	98.2	99.0	98.8	99.3	99.1	98.9
MAC [29]	97.1	99.1	99.5	99.5	99.5	98.9
Our Model	95.5	98.3	99.1	99.5	99.0	98.3

TABLE 5.2: Results of CLEVR dataset (validation set). (∗) denotes use of extra supervisory information through functional program. (†) denotes use of pre-defined symbolic functions and attribute templates.

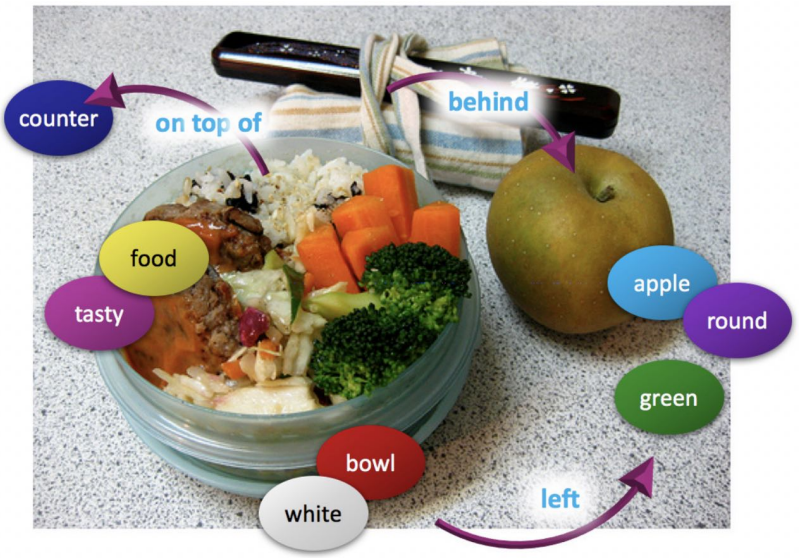
instead of SAN's stacked attention model. However, our model seems to struggle the most on counting questions, achieving only 95.5% accuracy. This appears to be in consistent with [59], which reported that end-to-end neural networks trained to minimise the cross entropy classification loss do not perform well on counting questions.

5.4 Experiment on GQA Dataset

In this experiment, we focus on evaluating our VQA model using real-world images. In particular, we performed the experiment on GQA [6], a dataset for real-world visual reasoning and compositional question answering. The GQA dataset is inspired from CLEVR, and is designed to solved various issues from the CLEVR dataset. Due to the artificial nature and low diversity of object types and attributes of CLEVR, it was vulnerable to memorisation for all combinations. As shown in the previous experiment, many models have already outperformed human and achieved accuracy close to 100%. Thus, the CLEVR dataset could be already be considered as *solved*. On the other hand, GQA contains real images and a much larger semantic space, making it much more difficult.

5.4.1 GQA Dataset

The GQA dataset [6] is developed to test a wide range of visual reasoning abilities, such as object and attribute recognition, transitive relation tracking, spatial reasoning, logical inference and comparisons. The dataset consists of 20M compositional questions and 113k real-world images. Similar to CLEVR, many questions require multi-step inference in order to



*Is the **bowl** to the right of the **green** **apple**?*
*What type of **fruit** in the image is **round**?*
*What color is the **fruit** on the right side, red or **green**?*
*Is there any **milk** in the **bowl** to the left of the **apple**?*

FIGURE 5.2: Example of GQA dataset [6]

come up with an answer. Each image is provided with a scene graph that describe the object's attribute, location and relations, and each question is associated with a functional program that represents its semantics. Questions are generated using a rule-based engine which focus on linguistic diversity and a large vocabulary. The distributions of answer are also controlled to minimise language priors and prevent educated guesses. In total, there are 3097 words in the questions (vocabulary size), and 1878 possible answers.

In addition to the standard accuracy metric, five new evaluation metrics are introduced as part of the dataset: *consistency*, *validity*, *plausibility*, *grounding* and *distribution*. Consistency metric measures responses consistency across different questions using entailment relations between questions (e.g. the model should not respond red to a new question about an apple it has just identified as green). Validity metric evaluates whether the model gives valid answer (e.g. respond with colour to a colour question or yes/no to a binary question). Plausibility metric checks whether the answer is reasonable in the real world (e.g. red and green is considered as plausible apple colours, while purple is not). Grounding metric measures whether the model attends to regions within the image that are relevant to the question (for attention based model only). Finally, distribution metric measures the overall match between the model predicted distribution and the true answer distribution (lower is better), which reveals if the model can correctly predict answers which occur less frequently.

5.4.2 Experimental Settings

We kept the settings of our model to be mostly the same as the previous experiment, even though the GQA dataset is considered more challenging than the CLEVR dataset. We had hoped to increase the size of the multimodal Transformer component, however due to time constraint, we did not have a chance to try a larger model size. The hyperparameters of our model is summarised in Table 5.3. Identical to the previous experiment, we use ADAM optimiser [56] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ to train the model, and cross entropy as the loss function. The model is trained for 11 epochs with batch size of 64. The initial learning rate is set to $5e^{-5}$, with decay factor of 0.2 starting in the 8^{th} epoch.

Model Hyperparameters	**Value**
Model's hidden dimension (d_{model})	512
Number of multimodal Transformer layer (N)	6
Number of attention heads (A)	8
Hidden dimension of feed-forward networks (d_{ff})	2048
Dropout	0.1

TABLE 5.3: Model hyperparameters for GQA experiment

For this experiment, we use object-based features extracted from pre-trained Faster R-CNN (bottom up attention) to represent the image. We neither use scene graph information nor functional program to train our model, since we want the model to learn the underlying reasoning processes directly from question and answer pair in an end-to-end manner. Using the GPU server we have set up, it took roughly around 40 hours to train the model.

5.4.3 Results

Table 5.4 summarises the result of our model along with other methods on the GQA's test set. For fair comparison, the table only shows the models which do not leverage additional supervisory information, such as scene graph and functional program. We obtained the result of the test set from submitting to EvalAI's GQA challenge [1]. The team name is shown as "MVSMARK (MMT-VQA)" in the leaderboard. Our model achieved the overall accuracy of **56.28%**, which is higher than baseline methods, such as BottomUp model (discussed in section 3.1.2), and the model specifically designed for multi-step reasoning, MAC Network (discussed in section 3.2.1). However, the model fell short in comparison to current state-of-the-art models, such as NSM [47], which operates on inferred graph structure of the image, and LXMERT [38], which is the large scaled pre-trained vision and language model. Nevertheless, we suspect that the accuracy of our model could still be further improved, with larger model size and more extensive hyperparameter search. As shown in the table, there is

[1] https://evalai.cloudcv.org/web/challenges/challenge-page/225/leaderboard/733

still a large margin between the best performing model and human performance, which demonstrates the difficulty of this dataset.

Model	Binary	Open	Consistency	Validity	Plausibility	Distribution	Accuracy
Human [6]	91.20	87.40	98.40	98.90	97.20	-	89.30
LSTM [6]	61.90	22.69	68.68	96.39	87.30	17.93	41.07
CNN+LSTM [6]	63.26	31.80	74.57	96.02	84.25	7.46	46.55
BottonUp [26]	66.64	34.83	78.71	96.18	84.57	5.98	49.75
MAC [29]	71.23	38.91	81.59	96.16	84.48	5.34	54.06
MCAN [32]	76.49	42.45	87.36	96.98	84.47	1.29	58.38
LXMERT [38]	77.76	44.97	92.84	96.30	85.19	8.31	60.34
NSM [47]	78.98	49.25	93.25	96.41	84.28	3.71	63.17
Our Model	73.73	40.87	86.86	96.01	84.20	5.78	56.28

TABLE 5.4: Results of GQA dataset (test set)

5.5 Analysis

In this section, we conduct an extensive analysis of our VQA model to understand its inner workings, as well as to understand its strengths and weaknesses. First, the **interpretibility** of our model is explored and discussed by visualising the attention distributions produced by the model during the inference process. For simplicity, we chose the CLEVR dataset to study how the model learns to infer the answer. Given that the questions are highly compositional, we are also particularly interested in understanding how the model would perform multi-step reasoning. Finally, in the error analysis section, we will look into potential reasons why the model came up with incorrect answers.

5.5.1 Qualitative Analysis

To obtain insights into the underlying reasoning process of our model, we can visualise the attention components learned by the model. For this analysis, we decided to visualise the attention map across all the layers and attention heads of the [IMG] and [TXT] tokens, since they are used as the pooled representation of an image and a question and are the input to answer classifier layer. As discussed in section 4.2.1.3, these tokens are used to extracted information related to answer from both visual and textual input, so they should hold highly critical information. The attention distribution of [IMG] token can be viewed as the visual attention (section 2.4.1), which can reveal which specific parts of the image the model is looking at. Conversely, the attention distribution of the [TXT] token can reveal which words in the question the model is focusing on.

To generate the visualisation of [IMG] token's attention distribution, we use method similar to [22]. In the CLEVR experiment, the input image is resized to 224x224 before sending to ResNet-101 to extracting the visual features. We use the output from conv4 layer, which have

the dimension of 14x14. Thus, the visualisation can be generated by upsampling the attention weights by a factor of 16 to make them the same dimension as the input image, and applying the Gaussian filter. Figure 5.3b and Figure 5.4 shows the example of attention distribution of [IMG] and [TXT] token across different layers and attention heads of Multimodal Transformer's self-attention unit. For the attention visualisation of [IMG] token, the brighter region of the image indicates where the model attends to (higher attention weight). Similarly, for the [TXT] token, darker orange indicates high attention on specific words in the question.

For this analysis, we will look into two examples with different image, question, and difficulty (more visualisation is shown in Appendix A). In the first example (see Figure 5.3a), the question is *"How many other things are there of the same shape as the green thing?"*. To correctly answer this question, human would first need to locate the *green thing*, then find other things with the same shape, and finally count the objects. Surprisingly, our VQA model also exhibits this behaviour, as shown in the visualisation of the visual attention (see Figure 5.3b). In the first layer (layer 0), we can see that the model looks at the overall image, without specific focus on any of the object in all of the attention heads. Then in layer 1, the model is able to locate the *green thing* with high attention in most of the heads. In layer 2 and 3, the model starts to look at other objects, with high attention weight on different object in different head. Then in layer 4, the model starts to identify the objects with the same shape (cube), as indicated in head 3 and 5. Also notice that the model keeps looking at the cyan cyclinder behind the blue sphere, as shown in many heads of layer 2 to 4. Our hypothesis is that model is not certain on the shape of the object, as it is highly occluded. Finally, the model becomes confident of the three objects it identifies as the same shape as the *green thing*, as shown in most of the attention heads of the last layer, before coming up with the correct answer which is three.

Similarly, analysing the visualisation of [TXT] token's attention distribution also reveals certain qualitative patterns. As shown in Figure 5.4, there are many attention heads across layers which focus on keywords of the question. For this example, the words "How many" are highly important, since they indicate the task of counting. In layer 1, 3 and 4, the model sharply focuses on these two words, as indicated by dark orange colour. In layer 0, the model mostly focuses on these sequence of words "same shape as the green thing", which indicate the first object the model needs to look at. The model also looks at the words "other thing" in layer 3, and "shape" in layer 2, which are another highly relevant keywords.

In the second example (see Figure 5.5a), the question is "There is a large yellow metallic object; is it the same shape as the green thing that is behind the large gray shiny object?". Similar to the previous example, the model learns to look around the image, and iteratively focus on objects which are relevant to finding an answer. We can see that the model looks at all the objects in the image in the first layer (layer 0). Then the model starts to look around different part of the image in layer 1. Interestingly, the model looks at the all the objects behind the large gray cylinder, indicating that it understands the concept of "behind", as shown in head 1 and 3 in layer 1. Then the model is able to locate the green object in layer 2. Finally, the

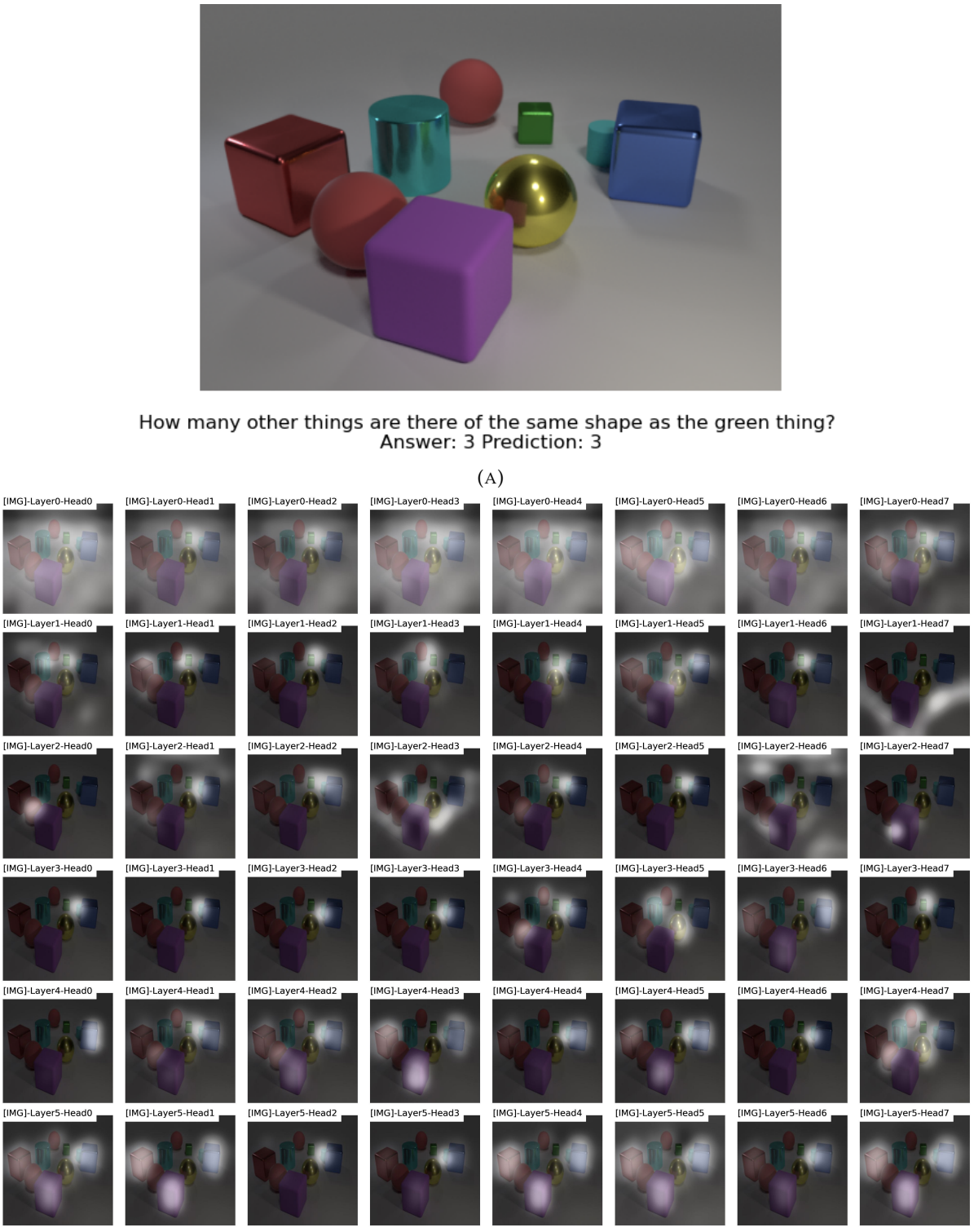


FIGURE 5.3: **First example.** (A) The original image and question, along with predicted answer. (B) Visualisation of the attention distributions across all the layers and attention heads of the [IMG] token (visual attention). Each row indicates different layer of the self-attention unit. Each column indicates different head of the multi-head attention in each layer. Bright region indicates the part of the image where the model focus on. In (B) we can clearly see that the model learns to look around the image and iteratively attends to parts of the image that are relevant to finding an answer, indicating that it is capable of performing multi-step reasoning.

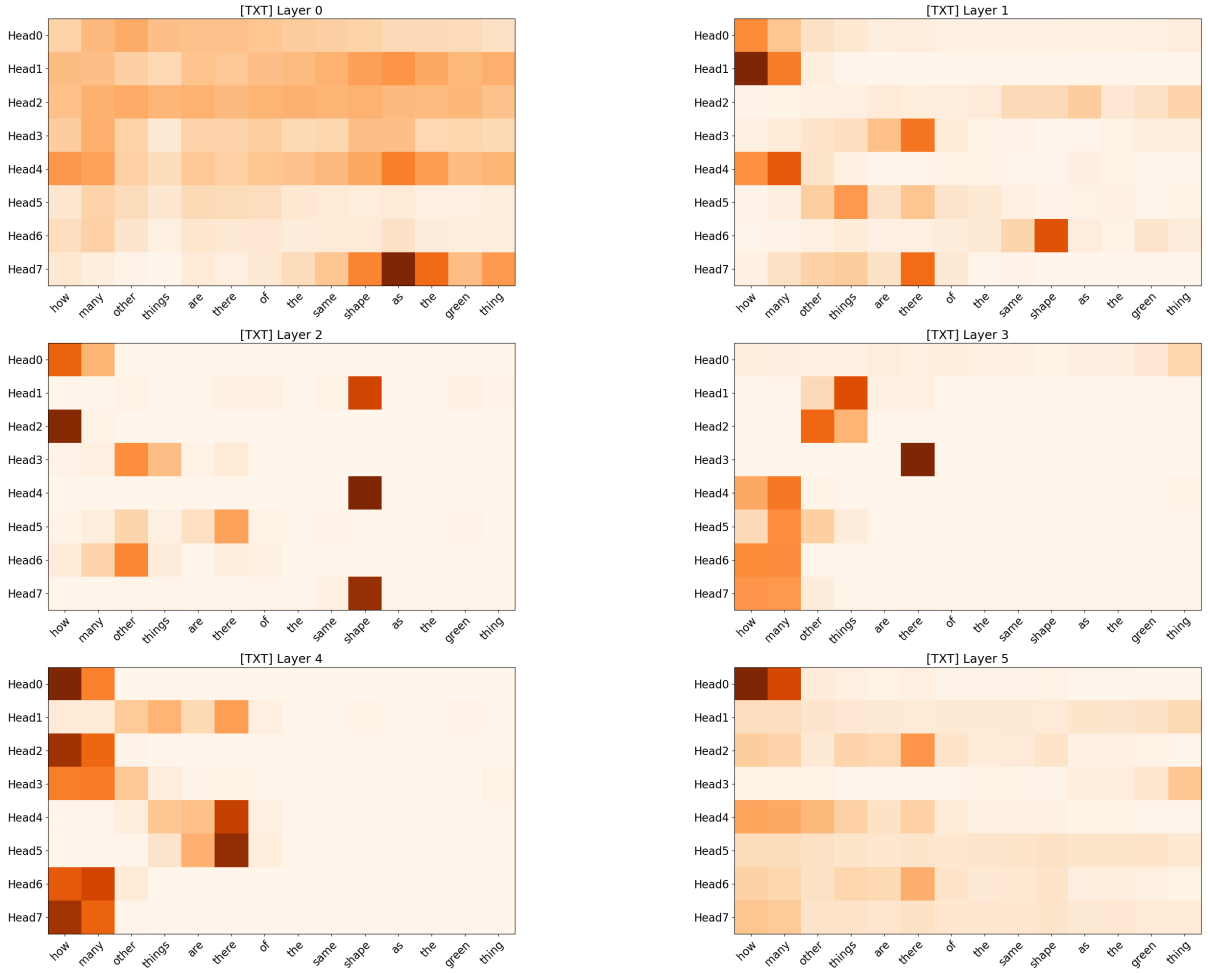
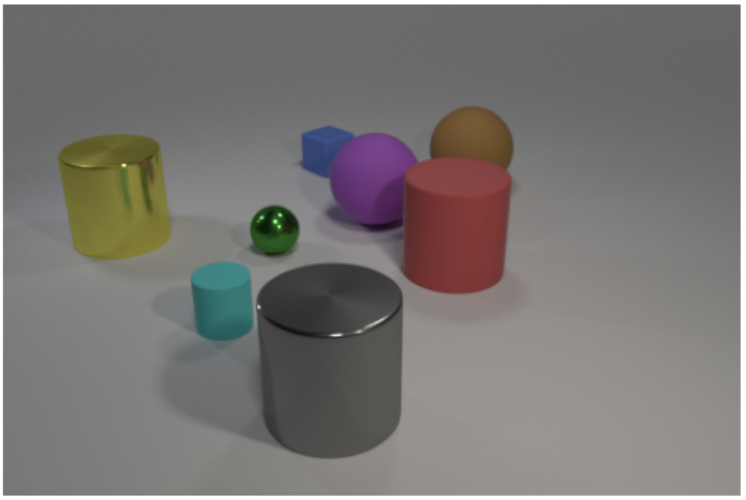


FIGURE 5.4: **First example.** Visualisation of the attention distributions across all the layers and attention heads of the [TXT] token (question attention). There is a total of six layers (shown in separate figure) Column in each figure represents the word in the question, while row indicate different attention head. Dark orange indicates high attention weight. Refer to Figure 5.3a for the corresponding image and question.

model sharply attends to the small green sphere and large yellow cylinder in layer 3 and 4, in which it can infer that the shape of these two objects are not the same. For this example, layer 4 and 5 seem to be non-essential, as the model can already infer the answer in layer 3.

Like the previous example, the visualisation of question attention (see Figure 5.6) shows that the model learns to focus on specific keywords. The model first focus on "gray shiny object" and the word "behind" in layer 0 and layer 1 respectively. Then the model sharply focuses on the word "shape" in layer 2 and 3. This is expected since shape is the main topic of this question. Finally, the model focus on "large yellow metallic object" in the last layer, which is the main subject of this example. All these words are highly relevant, however the sequence of attention on keywords between layers is not very clear compare to visual attention.

In addition to the analysis mentioned, we also found other interesting insights obtained from other samples. These are summarised as follows. In the first layer of visual attention, the model tends to look at the overall picture before focus on specific objects in the successive



There is a large yellow metallic object; is it the same shape as the green thing that is behind the large gray shiny object?
Answer: no Prediction: no

(A)

(B)

FIGURE 5.5: **Second example.** (A) The original image and question, along with predicted answer. (B) Visualisation of the attention distributions across all the layers and attention heads of the [IMG] token (visual attention). Each row indicates different layer of the self-attention unit. Each column indicates different head of the multi-head attention in each layer. Bright region indicates the part of the image where the model focus on. Refer to the text for explanation.

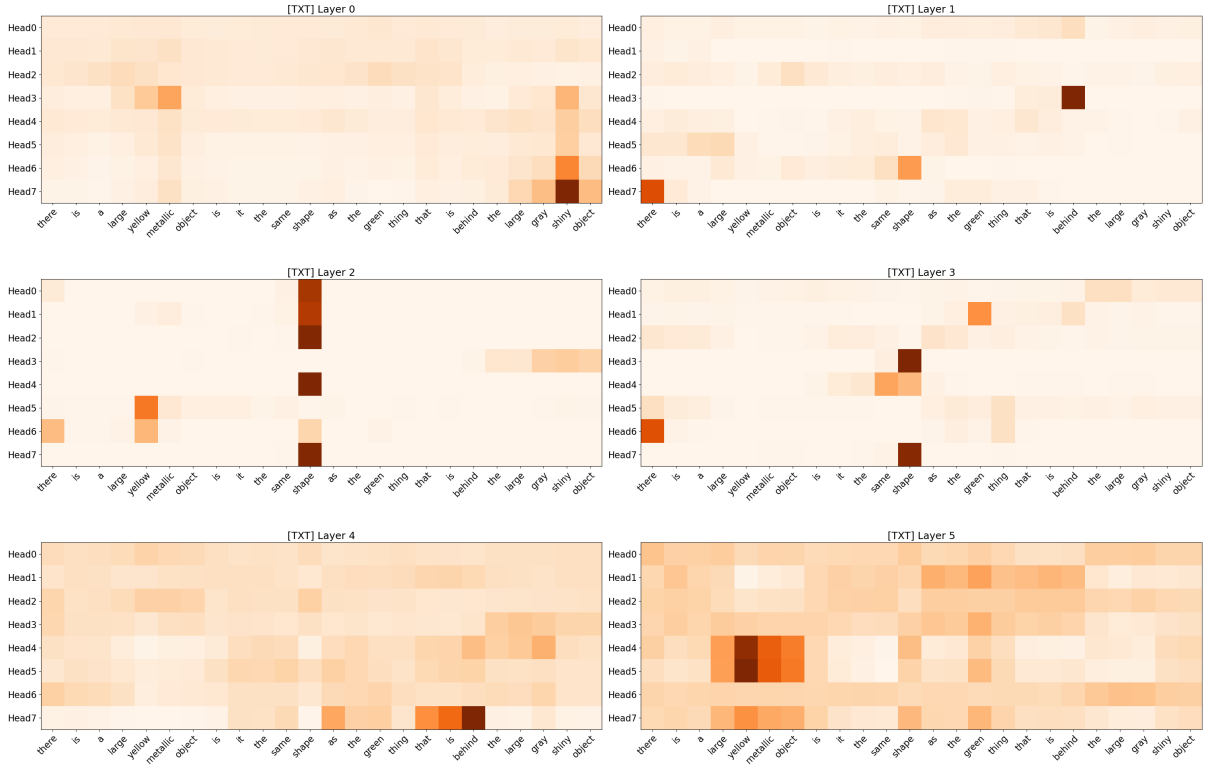


FIGURE 5.6: **Second example.** Visualisation of the attention distributions across all the layers and attention heads of the [TXT] token (question attention). There is a total of six layers (shown in separate figure) Column in each figure represents the word in the question, while row indicate different attention head. Dark orange indicates high attention weight. Refer to Figure 5.5a for the corresponding image and question.

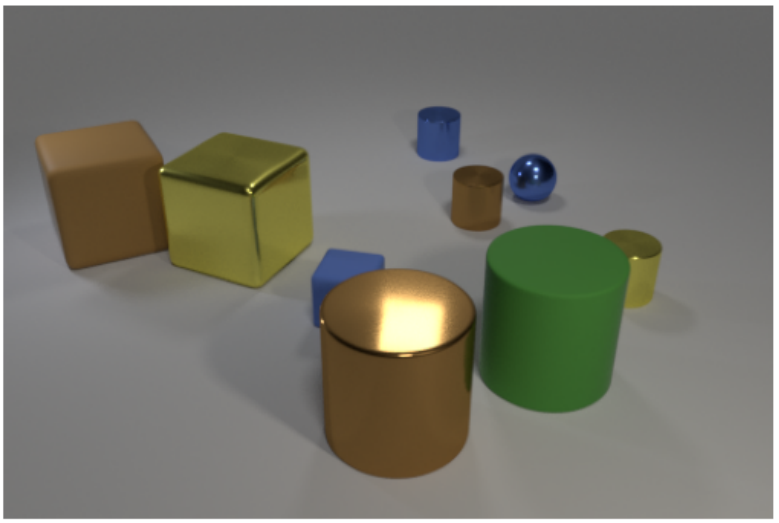
layers. The model is able to learn the concept of spatial relation (left, right, in front, behind), as the visualisation clearly show that it can focus on the correct region corresponding to the preposition. The model can also focus on multiple objects with the same attribute at the same time, which helps with the model's performance on counting questions. In addition, the model is adaptable to different level of question's difficulty. For example, the model doesn't look for new information in the last layers when dealing with easier question. We suspected that this is aided by self-attention mechanism, as the [IMG] or [TXT] token can just focus on itself. To conclude, it is clear that model is capable of multi-step inferences, which confirms the hypothesis that stacking multiple layers of self-attention enables the model to iteratively look around the input sequence.

5.5.2 Error Analysis

To understand why the model makes mistakes, we run the model on the validation set, and visualise the attention map to see the process of how the model comes up with answer. Two examples of the mistakes made by the model are shown in Figure 5.7 and Figure 5.8 (more example can be found in Appendix A). In the first example, we can see that the inference

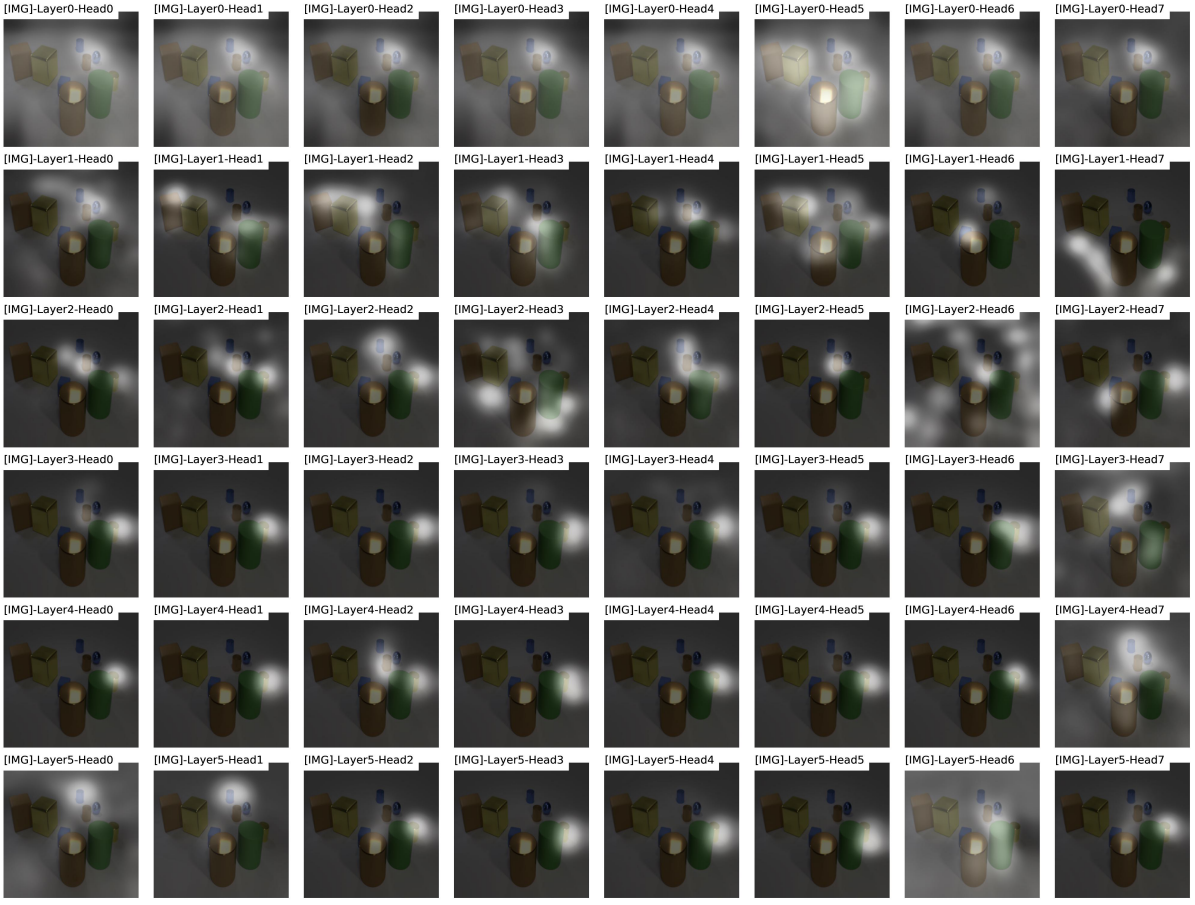
process of the model is valid. The model starts to look at the overall image, then focus on different objects, and finally locate the correct object in final layers. Even though the model focuses on the correct region, it produces an incorrect answer, responding with *brown* instead of *yellow*. We suspect that the reason might be because the object is highly occluded, and the colour between brown and yellow is quite similar.

In the second example (see Figure 5.8), we can see that the underlying reasoning process seems to be valid. Similar to the other examples, the model first looks around different parts of the image, and then finally locates all the relevant objects. Surprisingly, the model is able to identify the *tiny metal cylinder*, which is almost entirely occluded. Since the model can only see a small portion of the relevant objects, we suspect the model is less certain about their colour, thus providing incorrect result. In general, most of the mistakes occur when the objects are highly overlapping. The model also tends to give incorrect answer when the occluded object's colour is *yellow* or *brown*, since they are quite similar. Even when the model makes mistake, most of the time it attends to the correct regions of the image.



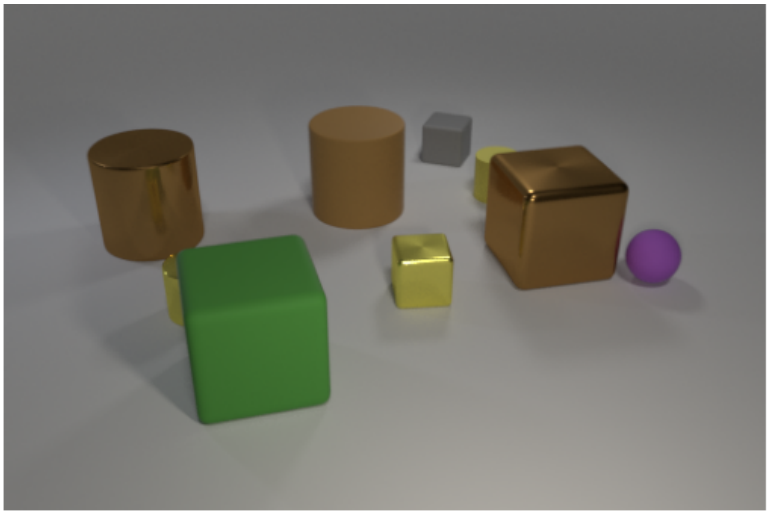
There is a small metal cylinder that is to the right of the green rubber object; what color is it?
Answer: yellow Prediction: brown

(A)



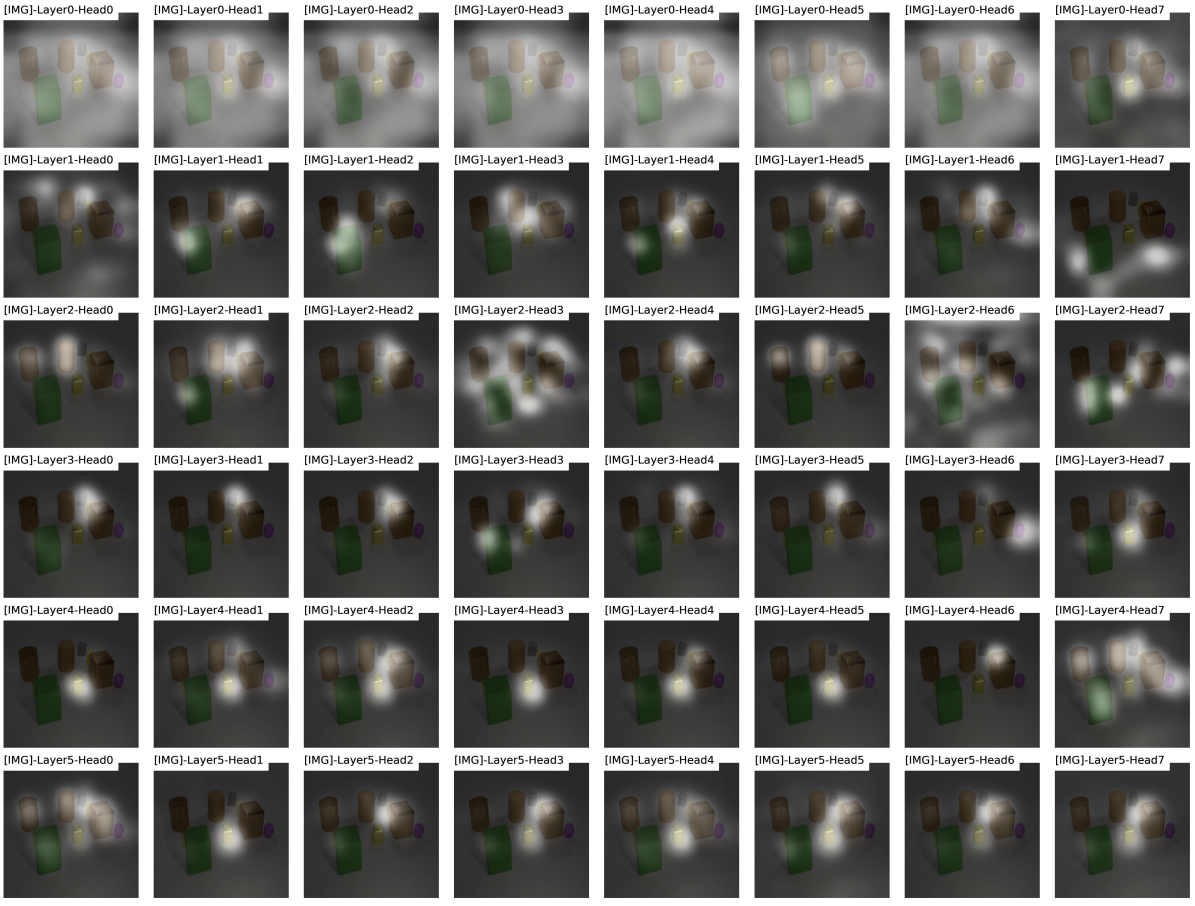
(B)

FIGURE 5.7: Example of the mistake made by the model. Even though the model produces incorrect answer, it is able to focus on the correct region of the image in the final layers.



What number of objects are the same color as the tiny metal cylinder?
Answer: 2 Prediction: 1

(A)



(B)

FIGURE 5.8: Another example of the mistake made by the model. In this image, the main objects relevant for answering the question are highly occluded.

Chapter 6

Conclusions

In this work, we presented a model capable of performing multi-step inferences in a visual reasoning task. The model is based on Transformer architecture and utilises both self-attention and co-attention unit for dealing with multimodal input. We performed the experiments on two visual reasoning datasets, CLEVR [5] and GQA [6]. The model achieved result comparable to current state-of-the-art models on the CLEVR dataset with the accuracy of 98.3%, and a relatively good performance on the GQA dataset with the accuracy of 56.28%. We also performed extensive analysis of the model by visualising the model's attention distribution on both images and questions. Visualisations show that the model learns to look around the image, and iteratively focus on relevant objects. Surprisingly, the multi-step reasoning process of the model is clearly visible. This confirms our hypothesis that stacking multiple layers of Transformer does enable the model to perform multi-step inferences in a task with a compositional structure.

We also suggested one effective way to adapt the Transformer architecture for a multimodal task. We have found that the model could not achieve high accuracy by simply using only the self-attention unit. As a pooling strategy, we added the [IMG] and [TXT] tokens as part of the input to aggregate information from each modality using self-attention and co-attention mechanism. These special token can help us visualise the attention distributions in every layers, which subsequently helps improve the interpretability of the model. The results from experiments demonstrated that the Transformer architecture is very powerful, flexible, and can be effective in multimodal task. As shown in the analysis, the model was able to perform multi-step reasoning, without the use of explicit reasoning architecture like MAC network [29] and Neural Module Networks [48]. To conclude, this work shows that deep learning model with the right inductive bias can indeed learns to reason from raw data.

6.1 Future Work

We believe that the model's performance on both datasets could be further improved. Due to time constraint and resources requirement to train the model, we could not perform extensive hyperparameter search. It would have been interesting to know the effects of different model sizes and depth of the Transformer layers. In addition, there are further analysis of model which could be performed. For instance, we have only visualised the attention distribution from the [IMG] and [TXT] token in the self-attention unit, so visualisation on other input tokens and co-attention unit could provide additional insights to the inner working of the model. Furthermore, it would be interesting to see the generalisation ability of the model by testing it on other datasets, such as VQAv2 [12].

As future work, better visual features could be incorporate to help with the performance of the model. In VQA, ResNet architecture, which is pre-trained on classification are typically used to represent the image. However, it is reported that changing the pre-training task to object detection can boost the accuracy of the VQA model [60]. We believed that this would definitely increase the model's performance, especially on the GQA dataset, as it contains real-world images. Moreover, it would be interesting to incorporate curriculum learning as the training strategy to see whether the model can learn faster. Another interesting research direction is to apply the model on other input modalities, and other tasks which require multi-step reasoning, such as textual question answering and reading comprehension.

References

[1] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[2] F. Chollet, "The limitations of deep learning." https://blog.keras.io/the-limitations-of-deep-learning.html, Jul 2017.

[3] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[4] M. Garnelo, K. Arulkumaran, and M. Shanahan, "Towards deep symbolic reinforcement learning," *arXiv preprint arXiv:1609.05518*, 2016.

[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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910, 2017.

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

[7] 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*, pp. 5998–6008, 2017.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[10] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," *arXiv preprint arXiv:1906.04341*, 2019.

[11] R. Yampolskiy, "Ai-complete, ai-hard, or ai-easy: Classification of problems in artificial intelligence," *The 23rd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH, USA*, 01 2012.

[12] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

[13] S. Manmadhan and B. C. Kovoor, "Visual question answering: a state-of-the-art review," *Artificial Intelligence Review*, pp. 1–41, 2020.

[14] 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*, pp. 770–778, 2016.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] S. Saha, "A comprehensive guide to convolutional neural networks — the eli5 way." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[18] C. Olah, "Understanding lstm networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs, Aug 2015.

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

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] S. Chaudhari, G. Polatkan, R. Ramanath, and V. Mithal, "An attentive survey of attention models.," *arXiv preprint arXiv:1904.02874*, 2019.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, 2015.

[23] J. Uszkoreit, "Transformer: A novel neural network architecture for language understanding." https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html, Aug 2017.

[24] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 21–29, 2016.

[25] V. Kazemi and A. Elqursh, "Show, ask, attend, and answer: A strong baseline for visual question answering," *arXiv preprint arXiv:1704.03162*, 2017.

[26] 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*, pp. 6077–6086, 2018.

[27] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *European Conference on Computer Vision*, pp. 451–466, Springer, 2016.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[29] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning," in *International Conference on Learning Representations (ICLR)*, 2018.

[30] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision," in *International Conference on Learning Representations*, 2019.

[31] A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," *arXiv preprint arXiv:1905.06088*, 2019.

[32] 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*, pp. 6281–6290, 2019.

[33] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[34] J. Cao, Z. Gan, Y. Cheng, L. Yu, Y.-C. Chen, and J. Liu, "Behind the scene: Revealing the secrets of pre-trained vision-and-language models," *arXiv preprint arXiv:2005.07310*, 2020.

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

[36] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[37] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Advances in Neural Information Processing Systems*, pp. 13–23, 2019.

[38] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," *arXiv preprint arXiv:1908.07490*, 2019.

[39] Z. Huang, Z. Zeng, B. Liu, D. Fu, and J. Fu, "Pixel-bert: Aligning image pixels with text by deep multi-modal transformers," *arXiv preprint arXiv:2004.00849*, 2020.

[40] 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*, pp. 740–755, Springer, 2014.

[41] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2556–2565, 2018.

[42] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "What does bert with vision look at?," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5265–5275, 2020.

[43] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[44] S. Sukhbaatar, J. Weston, R. Fergus, *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, pp. 2440–2448, 2015.

[45] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[46] C. Ma, C. Shen, A. Dick, Q. Wu, P. Wang, A. van den Hengel, and I. Reid, "Visual question answering with memory-augmented networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6975–6984, 2018.

[47] D. Hudson and C. D. Manning, "Learning by abstraction: The neural state machine," in *Advances in Neural Information Processing Systems*, pp. 5903–5916, 2019.

[48] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

[49] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Inferring and executing programs for visual reasoning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2989–2998, 2017.

[50] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 804–813, 2017.

[51] D. Kiela, S. Bhooshan, H. Firooz, and D. Testuggine, "Supervised multimodal bitransformers for classifying images and text," *arXiv preprint arXiv:1909.02950*, 2019.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, pp. 8026–8037, 2019.

[53] A. Singh, V. Goswami, V. Natarajan, Y. Jiang, X. Chen, M. Shah, M. Rohrbach, D. Batra, and D. Parikh, "Mmf: A multimodal framework for vision and language research." `https://github.com/facebookresearch/mmf`, 2020.

[54] Z. Yu and Y. Cui, "Openvqa: A lightweight, scalable, and general framework for visual question answering research." `https://github.com/MILVLG/openvqa`, 2019.

[55] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

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

[57] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in neural information processing systems*, pp. 4967–4976, 2017.

[58] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.

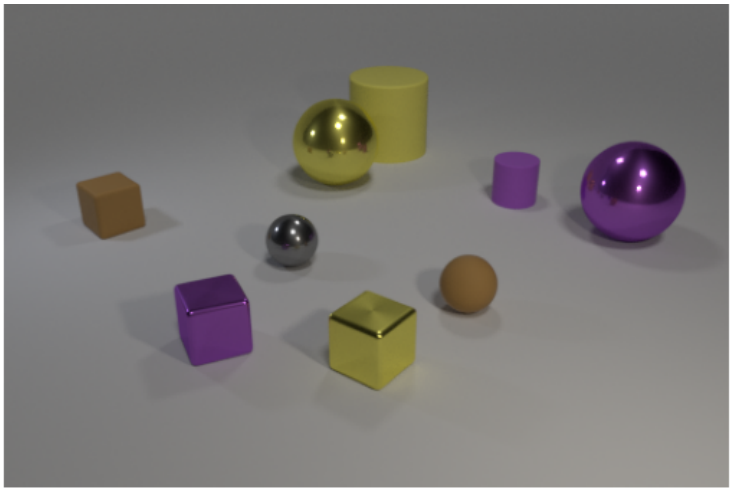
[59] P. Chattopadhyay, R. Vedantam, R. R. Selvaraju, D. Batra, and D. Parikh, "Counting everyday objects in everyday scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1135–1144, 2017.

[60] H. Jiang, I. Misra, M. Rohrbach, E. Learned-Miller, and X. Chen, "In defense of grid features for visual question answering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10267–10276, 2020.

Appendix A

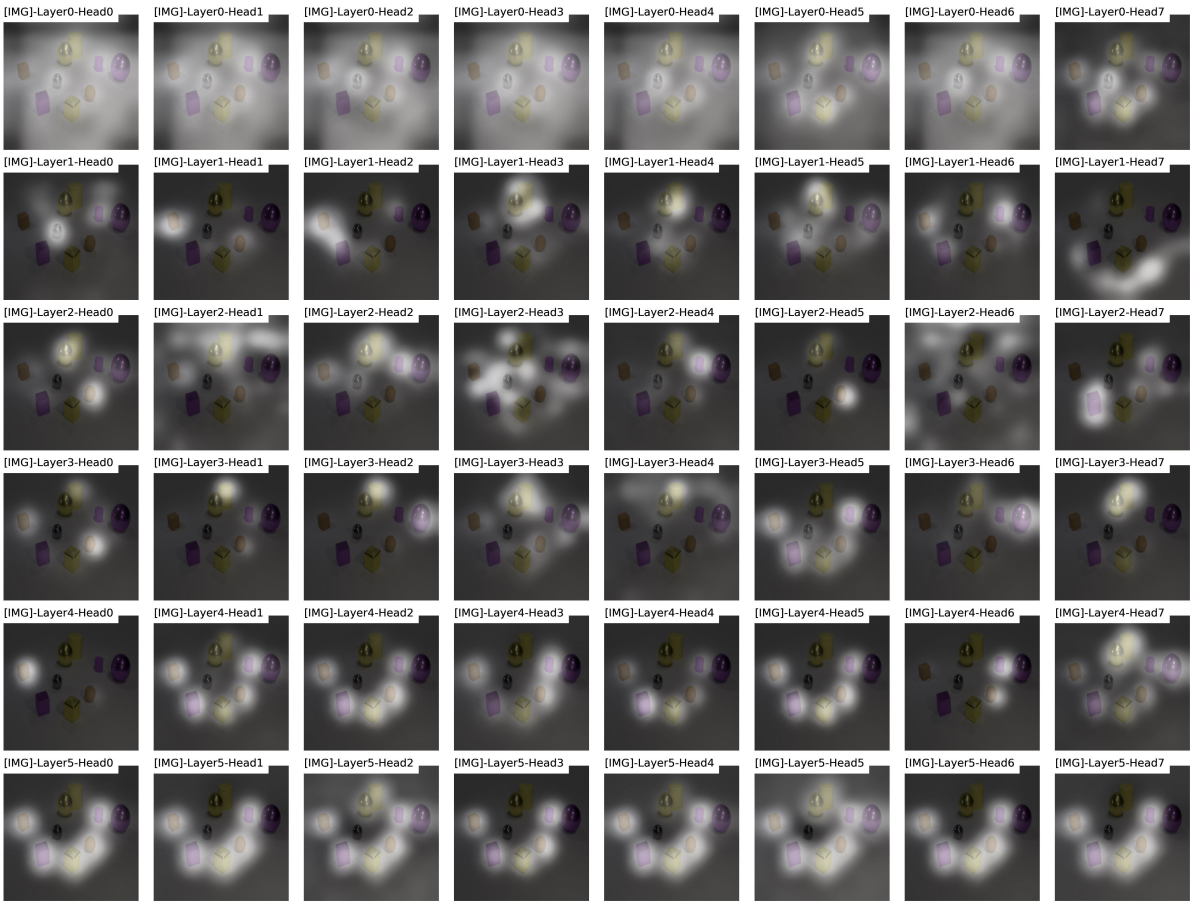
Attention Visualisation

Similar to the visualisation from section 5.5, we show four additional examples with different types of question and varying degree of difficulty. For each example, we visualise the attention distribution of the [IMG] token from all the attention heads (8) and layers (6). We also show another case where the model fails to produce the correct answer (see Figure A.4). From the visualisation, we observed similar behaviors as explained in section 5.5. As shown in the figures, we can clearly see the reasoning process of our VQA model to predict the answers.



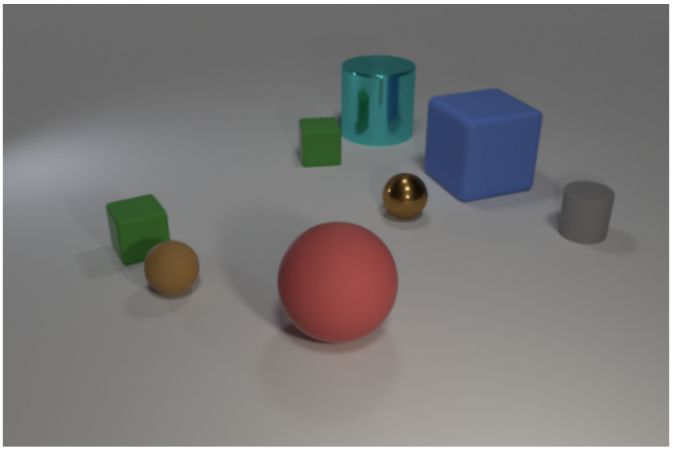
What number of other objects are there of the same size as the gray sphere?
Answer: 5 Prediction: 5

(A)



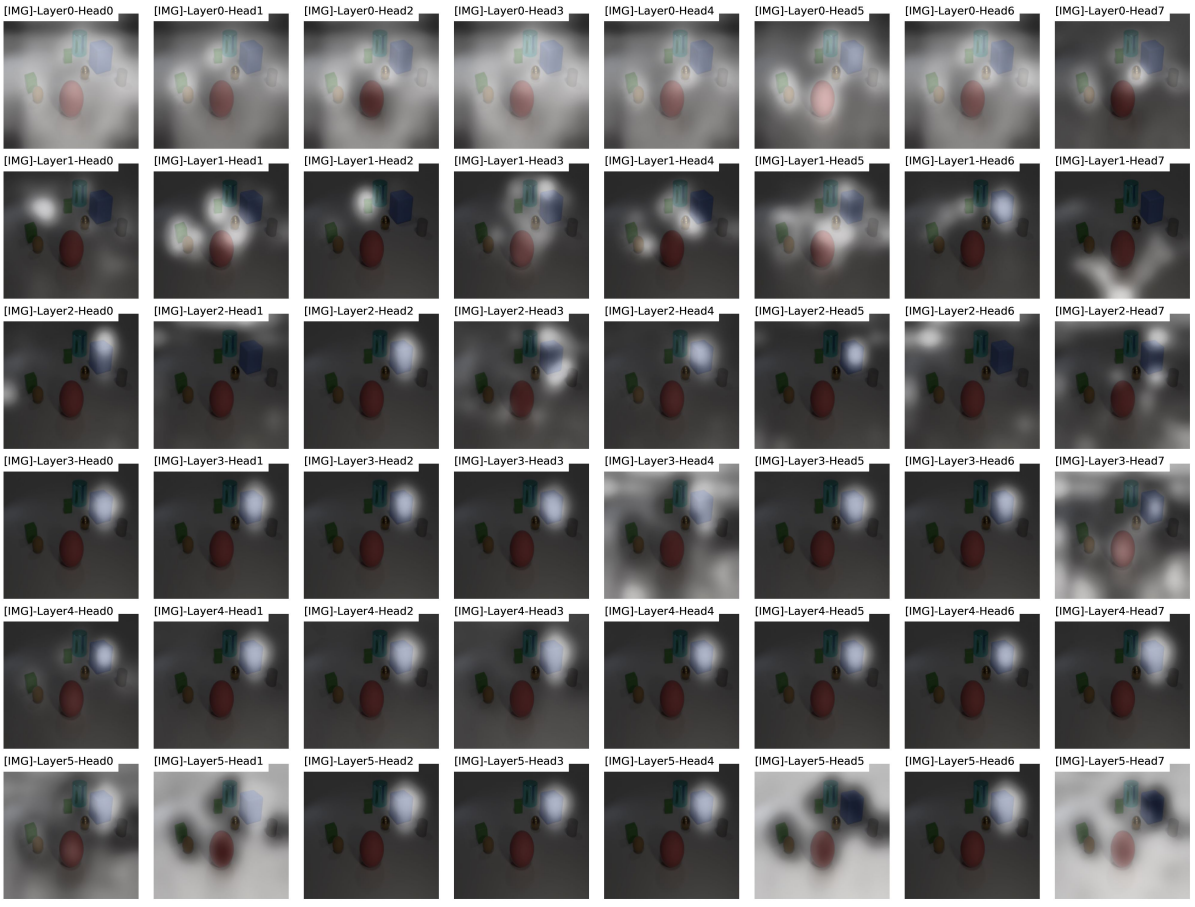
(B)

FIGURE A.1: Example of challenging counting question. In the last layer, the visualisation clearly shows that the model is able to detect all five objects which have the same size as the grey sphere.



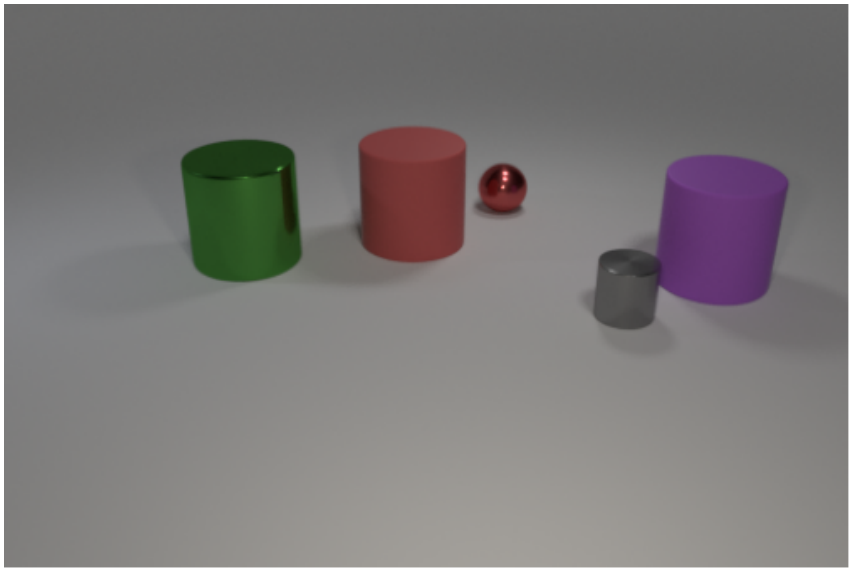
Is the number of big matte blocks greater than the number of tiny red metal cubes?
Answer: yes Prediction: yes

(A)



(B)

FIGURE A.2: Example of how the model learns to infer the answer in comparison question.

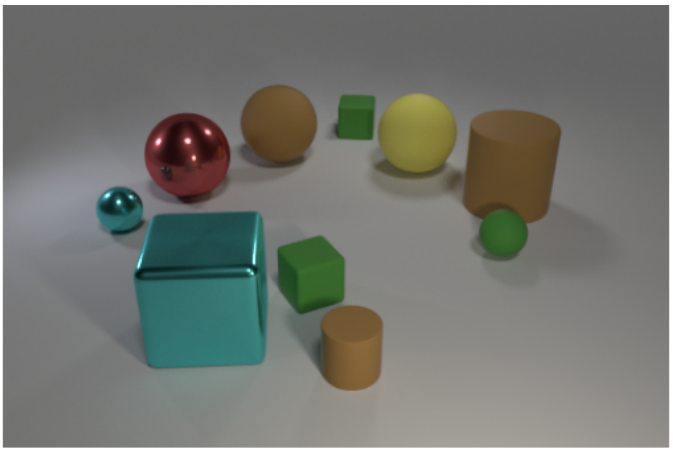


What shape is the tiny red thing?
Answer: sphere Prediction: sphere

(A)

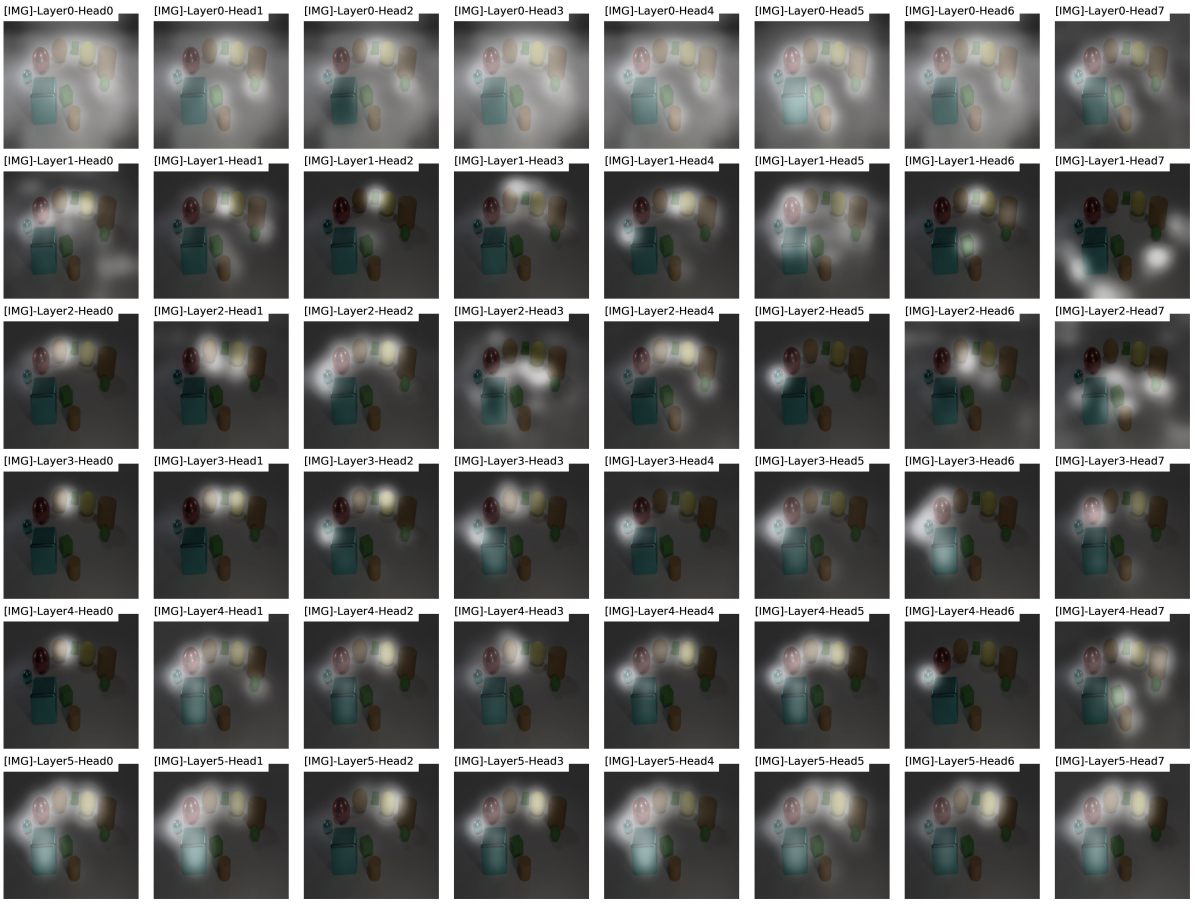
(B)

FIGURE A.3: Example of how the model learns to answer in an easy question. From the visualisation, the model already found the correct object in layer 2, so it did not look for new information in layer 3 to 5.



How many objects are metallic objects or rubber balls on the left side of the tiny brown object?
Answer: 4 Prediction: 5

(A)



(B)

FIGURE A.4: Example of mistake from the model. In this case, the model seems to not be able to detect that the yellow ball is not on the left side of the tiny brown cylinder.