

Towards Devising an Efficient VQA in the Bengali Language

by

S M Shahriar Islam

18101456

Riyad Ahsan Aunor

18101358

Minhajul Islam

18101304

Tahmin Haider Chowdhury

18101056

Mohammad Yousuf Hossain Anik

18101586

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering

Brac University

December 2021

© 2021. Brac University

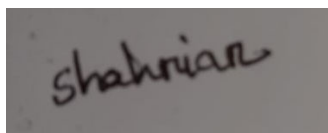
All rights reserved.

Declaration

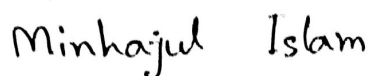
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

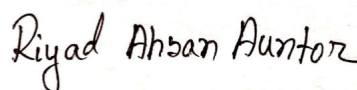
Student's Full Name & Signature:



S M Shahriar Islam
18101456



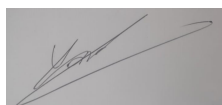
Minhajul Islam
18101304



Riyad Ahsan Auntor
18101358



Tahmin Haider Chowdhury
18101056



Mohammad Yousuf Hossain Anik
18101586

Approval

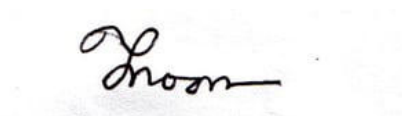
The thesis/project titled “Towards Devising an Efficient VQA in the Bengali Language” submitted by

1. S M Shahriar Islam (18101456)
2. Riyad Ahsan Auntor (18101358)
3. Minhajul Islam (18101304)
4. Tahmin Haider Chowdhury (18101056)
5. Mohammad Yousuf Hossain Anik (18101586)

Of Spring, 2018 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December 23, 2021.

Examining Committee:

Supervisor:
(Member)



Jannatun Noor
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Name of Program Coordinator
Designation
Department
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We, hereby declare that this thesis is based on the results we obtained from our work. Due acknowledgement has been made in the text to all other material used. This thesis, neither in whole nor in part, has been previously submitted by anyone to any other university or institute for the award of any degree.

Our Contributions

Through our research our aim was to establish a system which will convey the purpose of Visual question answering for Bangla language. As there is no specific or established dataset for Bangla VQA we formulated our own Bangla VQA datasets. These datasets were create using existing famous datasets. Our newly created datasets are Bangla VQA which consists of 5000 questions, answers and images and our second dataset is Bangla CLEVR dataset which consists of 12000 data points including questions, answers and images. Our next work was related to creating a perfect model which was used to train these datasets in approach to that we have successfully created and implemented a model which was capable to reach accuracy of 75% on Bangla CLEVR dataset and 63% on Bangla VQA dataset. Now, our established model is able to answer questions asked particularly in Bengali for an image and generate a correct answer in Bengali language.

What Motivated Us

Majority of the people prefers using smart systems in Bangla. But the latest additions to this field only support English. Not much research has been done in Bangla and There is a lacking of Bangla datasets for working on this. We wanted to remove the language barrier from establishing VQA. We wanted to create more scopes of research in this field.

In a nutshell, we found a lacking of Bangla resource in this particular field which we wanted to fill and this opportunity of leaving such a mark on this field motivated us.

Abstract

This paper aims to provide insight into how Visual question answering might work on Bangla datasets versus English datasets. Several studies have been conducted on deep learning methods applied to Bangla datasets up to this point. However, a Bangla dataset with images and questions embedded in each of them has yet to be created. We attempted to create a Bangla dataset suitable for such implementation through our re search. The step-by-step procedures in our work demonstrate how various barriers can be overcome while developing datasets. We attempted to use existing visual question answering datasets because there are no actual Bangla datasets created for this specific task. In the end we successfully created our own Bangla visual question answering datasets and proposed a model to train and compare among existing datasets. Following that, the comparison was provided to show how the Bangla dataset differs from the English datasets in terms of the VQA model. Our work should make more than enough room for future research and implementation of visual question answering tasks in Bangla.

Keywords: Natural Language Processing; CLEVR; VQA V1; Visual Question Answering;

Acknowledgement

Firstly, all praises to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Mrs. Jannatun Noor maam for her kind support and advice in our work. She helped and guided us whenever we needed.

And finally to our parents. For their support and blessings we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Contributions	v
What Motivated Us	vi
Abstract	vii
Acknowledgment	viii
Table of Contents	ix
List of Figures	xi
List of Tables	xiii
Nomenclature	xiv
1 Introduction	1
1.1 Objective	1
1.2 Thesis Outline	2
2 Related Work	3
3 Background Analysis	7
3.1 Recurrent Neural Networks	7
3.1.1 Types of recurrent neural networks	8
3.1.2 Common activation functions	9
3.2 Long Short Term Memory	12
3.3 Natural Language Processing	13

3.4	Convolved Neural Networks	14
3.5	MobileNetV2	15
3.5.1	Architecture	16
3.5.2	Experiments and Performance	18
4	Research and Methodology	20
4.1	Introduction to Existing Dataset	20
4.2	Our Proposed Dataset	25
4.3	Data preprocessing	30
4.3.1	Data splitting	30
4.3.2	Image preprocessing	30
4.3.3	Text preprocessing	31
4.3.4	Vocabulary set	33
4.3.5	Padding	33
4.4	Model Implementation	33
4.4.1	Input layer	35
4.4.2	Mobilenetv2 layer	35
4.4.3	Embedding layer	36
4.4.4	Bidirectional LSTM layer	36
4.4.5	Dropout layer	36
4.4.6	Output layer	36
4.4.7	Loss function	36
4.4.8	Optimizer	36
4.4.9	Batch size	37
5	Experimental Evaluation	38
5.1	Experimental Setup	38
5.2	Result and analysis for VQA v1 dataset	38
5.3	Result and analysis for Bangla VQA dataset	40
5.4	Result and analysis for CLEVR dataset	42
5.5	Result and analysis for Bangla CLEVR dataset	44
6	Limitations And Future Work	48
7	Conclusion	49
	Bibliography	53

List of Figures

3.1	RNN (to the left) vs Feedforward neural network	8
3.2	Sigmoid Activation Function	10
3.3	Tanh Activation Function	11
3.4	Relu Activation Function	12
3.5	Modules in RNN.	13
3.6	Modules in LSTM.	13
3.7	An overview on Convolutd Neural Networks	15
3.8	MobileNet[10]	16
3.9	MobileNetV2[10]	16
3.10	MobileNet[25]	17
3.11	MobileNetV2 Architecture[25]	17
3.12	MoblileNetV2 vs NasNet[16]	18
4.1	Boxplot of number of questions per image VQA v1	21
4.2	Length of the questions vs Distribution graph for VQA v1 training dataset.	21
4.3	Sample data from the VQA dataset	22
4.4	Sample data from the VQA dataset	23
4.5	Total Number of questions along with question per image distribution of CLEVR dataset	24
4.6	Length of the questions vs Distribution graph for CLEVR training dataset.	24
4.7	Sample data from our proposed Bangla dataset	26
4.9	Length of the answer vs Distribution dataset of Bangla VQA dataset	27
4.10	Sample data from our Bangla clever datase	28
4.11	Never of questions on a image on Bangla CLEVR dataset	29
4.12	Length of the questions vs distribution for our Bangla CLEVR dataset	29
4.13	Tokenizer split the sentence into small segments	31
4.14	Encoder encodes a value and assigns a unique integer.	31
4.15	Example question from Bangla VQA dataset	32

4.16	Tensorflow tokenizer tokenizing Bangla text	32
4.17	Output from our tokenizer	32
4.18	Model structure	34
4.19	A detailed architecture of our proposed model	35
5.1	accuracy vs epoch for VQA v1 dataset	39
5.2	loss vs epoch	39
5.3	Example of output of from our Model trained on VQA dataset h . . .	40
5.4	Accuracy on Bangla VQA dataset	41
5.5	Loss Function on Bangla VQA dataset	41
5.6	Example output of our model trained on Bangla VQA dataset	42
5.7	Accuracy of our model trained on CLEVR dataset	43
5.8	lost function of our model trained on CLEVR dataset	43
5.9	Example output of our model trained on CLEVR dataset	44
5.10	Accuracy vs epoch graph of our model trained on Bangla CLEVR dataset	45
5.11	loss vs epoch graph of our model trained on Bangla CLEVR dataset .	45
5.12	Example output of our model trained on Bangla CLEVR dataset . . .	46

List of Tables

5.1	Performace Table	47
-----	----------------------------	----

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

KVQA Knowledge-aware Visual Question Answering

SGD Stochastic gradient descent

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

FCNN Fully Connected Neural Network

GRU Gated Recurrent Unit

KRISP Knowledge Reasoning with Implicit and Symbolic representation

LSTM Long Short Term Memory

MAC Number of Multiply Accumulates

MLP Multi Layer Perceptron

NLP Natural Language Processing

RNN Recurrent neural network

SSD Single Shot Detector

VGG Visual Geometry Group

VQA Visual Question Answering

Chapter 1

Introduction

Computer vision is a wonder that has allowed us to identify an object almost flawlessly. But its ultimate goal is to be able to describe its attributes and its interactions with the environment, according to [7]. Combining natural language processing helps us to achieve that goal. The combination of computer vision and natural language processing can accomplish the tasks of visual question answering, visual grounded dialogue, image captioning etc. Considering the fact that ours is not an English speaking country, it would not be absurd to assume that a vast majority of people are either unfamiliar or uncomfortable with interactions in English. In such a case a smart interactable system should support the Bengali language to facilitate our need. A lot of work has been done on visual question answering but most of it involves English datasets. There is a lack of Bangla dataset and research in this field. A few works do however establish image captioning [33] and text-input based question answering [31] in Bangla, but extracting information from a visual input and processing the answer to a corresponding question in Bangla has not yet been done. Being the 7th most spoken language in the world, Bangla does not yet have a proper dataset that can facilitate our need to establish a visual question answering system. The closest work done would be Chittron [15] an automatic image captioning system that is capable of explaining a scene in Bengali. BanglaLekha-image is the most popular Bengali dataset used for this purpose. This dataset contains around 16 thousand images with captions embedded to each. Our study aims to cope with this necessity and reduce the lack of Bangla resources in this field.

1.1 Objective

The objective of our study are as follows:

- Our objective is to establish a proper bangla dataset that can facilitate the VQA

based algorithms and provide a satisfactory output.

- A proper comparison is shown in our study to shed some light on how the bangla implementation works compared to the English datasets.
- We aim to provide rooms for future implementation of our study on establishing various bangla language based smart systems.

1.2 Thesis Outline

Through our study we tried to establish a bangla language based dataset suitable enough to work on visual question answering based models. This chapter gives a thorough idea on what this research is all about. In the second chapter relevant works have been discussed. The third chapter will shed some light into the tools and properties we are going to use for our study. Chapter 4 explains our research methodology, Chapter 5 shows the result accuracy, Chapter 6 shows the future work and limitations. Chapter 7 marks the end of this paper containing the conclusion.

Chapter 2

Related Work

Throughout the years many research works have been carried on Visual Question Answering as such we have come across numerous papers on structure of Visual Question Answering and their implementations.

Antol et. al. (2015) in his paper proposed the Vanilla VQA which is the simplest and easiest visual question answering model, and the performance of this model functions as a barometer for the VQA work [2]. In this model, VGG [5] which is a convolutional neural network that is used to recover image extracted features and a Recurrent neural network namely an LSTM or a GRU is used to recover the linguistic embeddings of the queries. Both the feature representations are then passed to an MLP.

Design paradigm, question encoding modules, image feature extraction, fusion and classification are followed for Visual Question Answering (VQA) models[9]. Visual Genome dataset uses Faster R CNN for object detection purposes. The computation of the top-down attention is used for the question text of each object in an image. By using fine tuning features, overall performance will increase. The overall performance on test-dev will reach upto 68.49% with 0.1 as the rate of fine-tuning. This performance improves upto 69.81% when the bottom-up features are combined with the image features on grid-level[20].

Neural-symbolic approach for visual question answering is used to understand language from reasoning[11].Object segmentation, object oriented scene representation can be done using neural networks.To convert questions in natural language and symbolic representation the program executor plays a vital role by executing the program according to natural language instructions using scene representation. Neural-symbolic approach increases accuracy level and reduces memory cost and computational cost.The NS-VQA model consists of 3 things: 1) scene parser 2) question parser and 3) program executor.

The scene parser extracts from the image a structural representation of the scene and Mask R-CNN is used for extracting images. Mask R-CNN is used for segmentation of the object and prediction of the categorical leveling of the objects. The question parser converts input questions into natural language to a latent program. The model structure is like a flow from one sequence to another with encoder and decoder. LSTM encoder is used for conversion of the questions. CLEVR, CLEVR-CoGen, CLEVR-Humans dataset and Minecraft world use a neural symbolic approach for VQA.

Kafle et al. (2017) claims that embedding absurd questions and evaluation metrics allows the VQA algorithms to gain a deeper understanding[6]. They used the TDIUC dataset consisting of 12 different question types. They also proposed an evaluation metric to compensate for the biases that exist in the dataset.

Most of the models that have been developed before maintained the one path: The image features from the scene have been obtained using a CNN and to decode the associated question words, LSTM is used. But the performance of this approach can vary from the one that uses attention based approach[3]. According to Agrawal et al. (2016) the VQA models constructed with attention can reach up to an accuracy of 57.02% accuracy on the VQA validation dataset[2]. They have compared the two types of approach, with and without attention and gave a review on the performances.

A mixed bottom-up with top-down attention mechanism has been presented by Peter et al.(2017) allowing attention to be measured just at levels of objects as well as other prominent visual regions[8]. It is the natural premise for considering attention. The bottom-up process of their approach suggests picture regions, having feature vectors for each of them, whereas the top-down process makes use of the feature weights. The scores for CIDER reached 117.9, SPICE reached 21.5 and BLEU-4 reached 36.9 using this approach on the MSCOCO test server. They won top position in the 2017 VQA Challenge, demonstrating its method's wider application by applying the same methodology to VQA.

Kenneth et al. (2019) provides a benchmark called OK-VQA which addresses the challenges of knowledge based visual question answering[14]. The main target of this research is merging visual recognition with data extraction from places other than the picture. Situations in which the image content is insufficient to respond to questions, this model is used. That's why this paper promotes approaches that solely depend on external knowledge - based resources. More than 14,000 queries in their new dataset require outside information to answer. Here random images from the COCO dataset have

been used, making 80K as test data and 40k as validation data. Here splits should be tested. When these photos were compared in terms of visual complexity, they are useful for categorizing knowledge-based inquiries due to their similarity to other datasets. They show a dramatic fall in the overall performance within this environment for the traditional VQA models. Compared to other knowledge-based VQA datasets, their study suggests that their knowledge-based VQA work is diversified, tough, and huge.

Sanket et al. (2019) states in traditional VQA, questions regarding an image can be asked that can be answered only depending on its material[17]. A common VQA question might, for example, query about the number of individuals in an image having people in it. More lately, there has been an increasing interest in answering queries relating to common nouns like cows, horses, buildings etc. that are visible in the image. Despite this development, previous research has not covered the key problem of answering queries requiring knowledge about the things that have names like the Eiffel tower, Nelson Mandela etc shown by the image. Inside the image. In this study, they fill that vacuum by introducing KVQA. KVQA has 183K question and answer pairings with over 18K named entities and 24K photos in total. As far as we have known, this is the largest dataset for investigating visual question answering over knowledge graphs. On KVQA, the baseline performances using the cutting edge methodologies have been provided.

When responding to a question that requires outside knowledge that isn't available, it's one of the most difficult forms of VQA questions[30]. They investigate the situations that took place when the required information to generate an answer for a question is not present during the testing or training period. They make use of two different kinds of knowledge representations as well as logic. First, with transformer-based models, implicit information may be learned efficiently from supervised training and unsupervised language pre training data. Second, knowledge bases store explicit, symbolic data. The method combines the use of transformer models' strong implicit logic for response prediction with the integration of expressions from a graph structure, all while maintaining their explicit semantics. They acquired and mixed information from many sources to cope up the huge number of information required to answer knowledge-based inquiries. On OK-VQA, they show that the Knowledge Reasoning with Implicit and Symbolic representation a.k.a KRISP approach surpasses the state-of-the-art[26]. KRISP combines symbolic knowledge and implicit knowledge successfully.

To handle the query and photo, KRISP employs a multi-modal BERT-pretrained converter that utilizes implicit knowledge. To use symbolized sources of knowledge, BERT

and a graph network is used[18]. To accommodate the large range of knowledge needed in OK-VQA, they built their knowledge graph using four distinct knowledge sources: DBpedia, ConceptNet, VisualGenome, and PartKB. Visual data, encyclopedic data, data related to daily things, science knowledge including knowledge regarding specific persons and places and events all are covered[12]. Their approach maintains the symbolic meaning of data which uses a late fusion strategy by making predictions depending on the hidden layer of different nodes. They demonstrate that, besides having the capabilities of harnessing implicit knowledge logic, their model also contains the symbolic response, which links the answer language with the graph database. This is the key to their method's effectiveness and it generalizes to unusual solutions.

Yu et al. (2018) covers Pythia v0.1, the winning submission in the VQA Challenge 20181 from Facebook AI Research (FAIR-STAR)'s team[9]. They began by reimplementing the bottom-up concept in a modular fashion. This research shows that the model's performance can be improved significantly from 65.67% to 70.24% on the VQA v2 dataset, simply through minor but noticeable changes in its architecture and having to learn standard cost techniques, good image features and attaching data augmentation. Moreover, it was capable of improving by 1.31% using a heterogeneous ensemble of training images with different characteristics and on various data. They score 72.27% on the test-standard split.

To improve training precision and agility, a few tweaks were made to the up-down model. Without using gated hyperbolic tangent activation, weight leveling followed by ReLU was used[19]. The main goal of this modification was to reduce computing time. When calculating top-down attention, researchers also substituted characteristic concatenation with component multiplying to mix characteristics from text and visual modalities. 300D GloVe vectors were used to embed the words. This word embedding was then served to the GRU network as input with a query awareness module to recover attentive textual features to calculate the query representation[23]. They discovered that 5000 was the ideal concealed size for combining image and text data. With these changes, they were able to enhance the precision by 1.59% on VQA v2.0 test-dev.

Chapter 3

Background Analysis

Visual questions target specific areas of an image, such as background details and underlying context. As a result, a system that succeeds at VQA typically necessitates a more detailed understanding of the image as well as more complex reasoning than a system that produces generic image captions.

3.1 Recurrent Neural Networks

Widely known as RNNs, the recurrent neural networks work with and specialize in sequential or temporal inputs of data[13]. Algorithms like these are generally used for problems that are ordered by time or sequence, such as speech recognition, language to language translation, processing natural language(NLP), and the captioning of images. Popular applications like Apple's Siri, Voice Search applications and Google Translate use such neural networks.

Training data is used in CNN and RNN to provide functionality. Their "memory" distinguishes them since current inputs and outputs are influenced by the previous ones. While generic deep learning neural networks are based on the assumption that inputs and outputs do not affect each other. Shared parameters across all layers of the network is another distinguishing feature of RNNs. In RNN each node has the same weight as the parameter but in the feedforward neural network we will get a different weight for each node. These weights are still adjusted using gradient descent and backpropagation to facilitate reinforcement learning. RNN uses backpropagation through time (BPTT) to determine gradients. It is different from traditional backpropagation which is specific to sequence data. BPTT works just like traditional backpropagation.

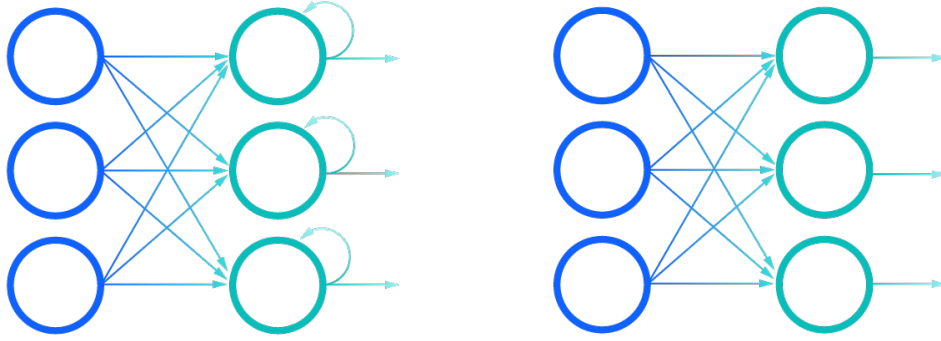


Figure 3.1: RNN (to the left) vs Feedforward neural network

Model training is done by calculating the error value which is produced in the output layer and these calculations is used to adjust the model's parameters. BPTT is different from the traditional approach in the fact that it sums errors at each time step. Where in feed forward neural networks it is not required to sum errors because sharing of parameters does not occur across different layers. During this however, RNNs frequently come across two setbacks: the exploding gradient problem and vanishing gradient gradient problem. The error curve which is the slope of the loss function, is created by the issues made by the gradient's size. When the gradient becomes too small, it continues to get smaller. It keeps updating the weight parameters until they become insignificant—that is, 0—and then stops functioning. In such cases, the algorithm stops learning. A gradient will explode if it becomes too large, which will result in an unstable model. We need to reduce the number of hidden layers to reduce the complexity of the RNN model otherwise the weights will be large enough to make the model pretty much unstable.

3.1.1 Types of recurrent neural networks

We observe the input and output has a one to one mapping in Feedforward neural network, and while recurrent neural networks are depicted in the figure 3.1 in this manner, RNN does not have this constraint. In recurrent neural networks the input length and the output length are not the same. We can see a wide range of applications of RNNs, including music generation, sentiment classification, and machine translation. The following diagrams are commonly used to represent various types of RNNs: One-to-one, One-to-many, Many-to-one, and Many-to-many are all examples of One-to-many relationships.

3.1.2 Common activation functions

An activation function controls whether or not a neuron is activated. A neural network is nothing more than a linear regression model without consideration of an activation function. The activation function is responsible for the non-linearity required to extract patterns from complex data. There are two types of activation functions: non-symmetric activation functions and symmetric activation functions. A symmetric function is the hyperbolic tanh function, and a non-symmetric activation function is the ReLU activation function. To convert the output into a value between 0 and 1 or -1 and 1 we need to use a nonlinear function. Let us discuss some commonly used activation functions which play a vital role in neural networks.

Sigmoid: This function is commonly used as a nonlinear function which takes a real value as an input and turns it into a value between 0 and 1. If the input's value is large enough it will be converted into a value closer to 1 conversely, if the value is very small in measure it will be mapped to a value closer to 0[22]. We can observe a lot of machine learning models utilizing this function lets find out a few advantages of this activation function:

- It is commonly used when the probability needs to be considered as the output. The sigmoid function is positively considered due to its range, since the probability of any event occurring lies between 0 and 1. implementation works compared to the English datasets.

- Since the function gives a smooth gradient and can be differentiated, the output value does not jump. The sigmoid activation function has an S-shape to represent this.

The limitations of sigmoid function are discussed below:

- Sigmoid function has the derivative, $f'(X) = \text{sigmoid}(X) \times (1 - \text{sigmoid}(X))$. Which may result in vanishing gradient problems.

- Close to 0, the logistic function's output is not symmetric. This results in all the outputs to be of the same sign. This makes neural network training more difficult and unstable.

It is shown by this formula $g(x) = 1/(1 + e^{-x})$.

Tanh: Tanh function and sigmoid function have the same S-shape but its output range is from -1 to 1. Tanh assumes that as the input gets more and more positive, the

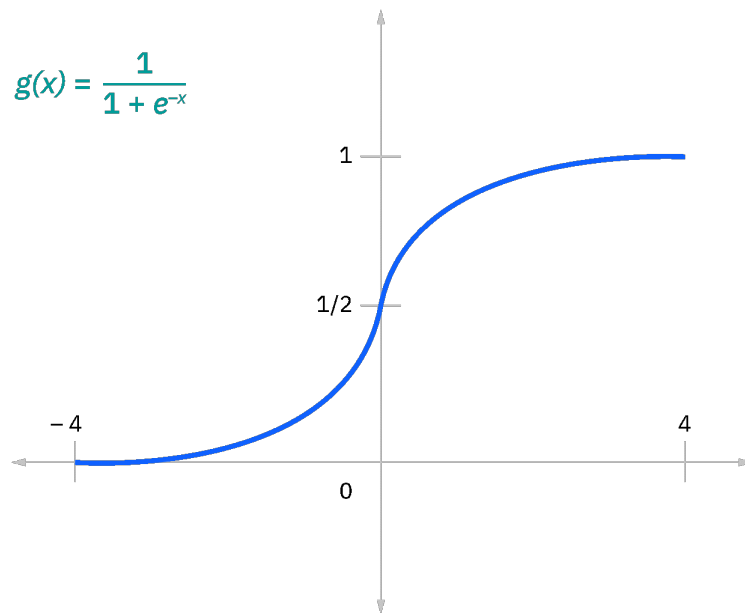


Figure 3.2: Sigmoid Activation Function

output gets closer to 1.0, and the lesser it gets, the closer the output is to -1.0[22]. Advantages of using this activation function are:

- Tanh activation function's output is 0 centered. Which is why the outputs can easily mapped to strongly negative, neutral, or strongly positive.
- Since its value ranges within -1 to 1, it helps in centering the data. This helps the next layer learn more easily.

The disadvantages include, but are not limited to, the fact that it also has the vanishing gradient problem. Furthermore, the tanh function has a much steeper gradient. Despite the fact that both the sigmoid and the tanh suffer from the vanishing gradient problem, the tanh is zero centered, and the gradients do not only move in a certain fixed direction.

The non-linearity is established with the formula $g(x) = (e^{-x} - e^x) / (e^{-x} + e^x)$.

Relu: ReLu is the abbreviation for Rectified Linear Unit. Even though ReLu seems like a linear function, it contains a derivative function that allows backpropagation efficiently[27]. ReLu cannot activate all the neurons at the same time. If the linear

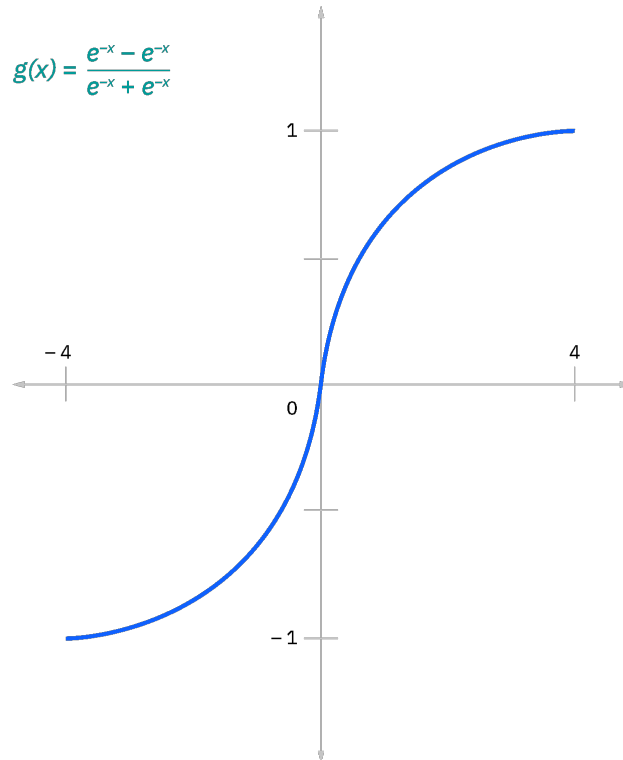


Figure 3.3: Tanh Activation Function

transformation output ever becomes less than 0, only then the neurons become deactivated. The advantages of using ReLU as an activation function are as follows:

- ReLU function computationally efficient than both tanh and sigmoid.
- Since ReLU is linear and non-saturating, it tends to push the gradient descent to converge at the global minimum of loss function.

Limitation:

- Dying ReLU is the major problem of ReLU activation function, however this can be solved by a few modifications to the activation function.

This is represented with the formula $g(x) = \max(0, x)$

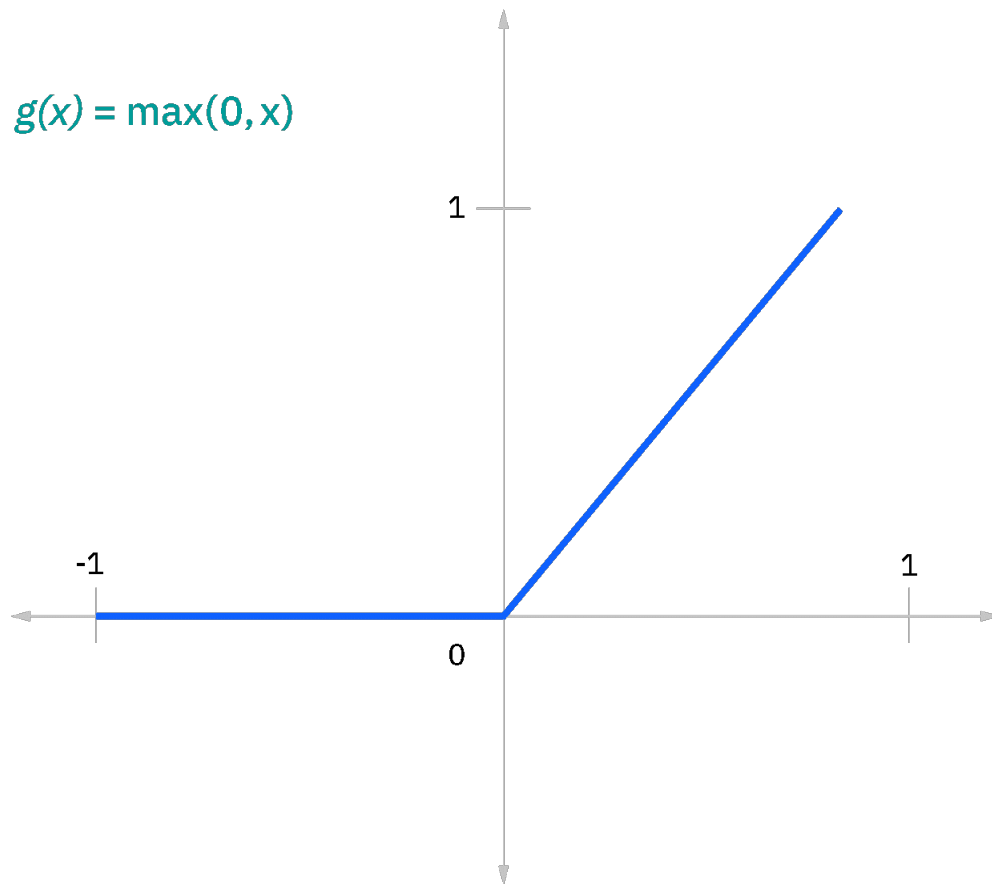


Figure 3.4: Relu Activation Function

3.2 Long Short Term Memory

A special type of RNN is Long Short Term Memory which can learn long-term dependent sequence data. Its use was initiated by Hochreiter and Schmidhuber (1997). They are widely used because they perform admirably in a variety of situations. Long-term dependency is explicitly avoided by LSTMs. Long-term memory is almost their default behavior; it is not difficult for them to learn it! All RNNs are composed of network modules that repeat in a series[24]. This represents a simple structure, one like a single tanh layer.

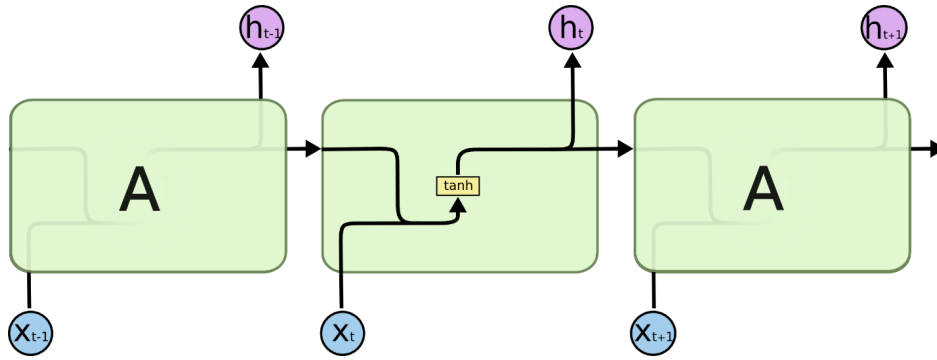


Figure 3.5: Modules in RNN.

These modules do not have a chain-like structure like LSTMs. There are four network layers, each with its own set of interactions.

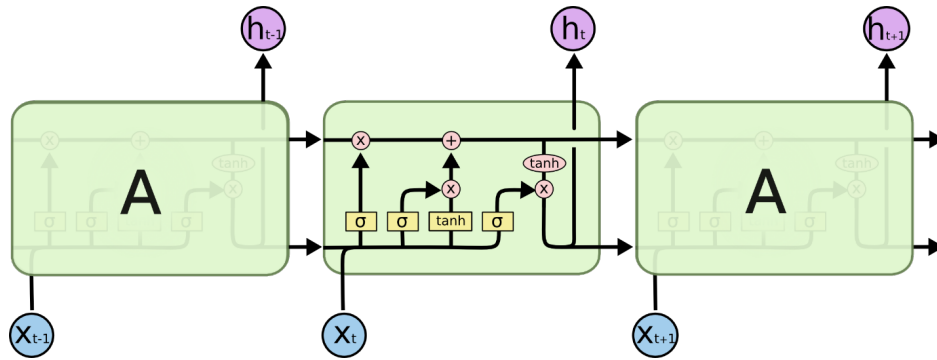
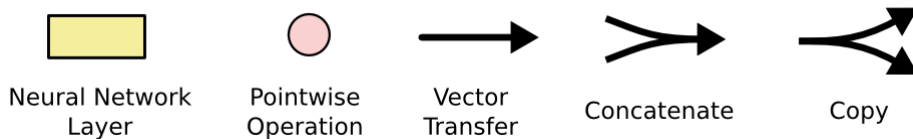


Figure 3.6: Modules in LSTM.



An entire vector is transferred from one node to another node through each of the lines in the above diagram. Pink circles represent vector addition operations, while yellow boxes represent the neural networks that have already learned. Concatenation is represented by lines that merge, whereas forking is represented by lines that copy and send the copies to different locations.

3.3 Natural Language Processing

Natural language processing enables machines to understand texts and speech just like us humans. It is a subset of artificial intelligence. NLP is a revolutionary invention which took machine and human communication to the next level[21]. With the help of this combination, NLP allows computers to process the textual and voice data and

understand the underlying meaning behind it. Through NLP, the computers can translate languages, respond to commands and also summarize paragraphs in real time[32].

The utilization of NLP can be seen in multiple applications like voice-controlled GPS systems, digital assistants, chatbots etc. Natural language processing is also seen to be used in helping streamline business operations and other business processes. Human language usually has irregularities in it. We often tend to express our thoughts indirectly through sarcasm, idioms etc. which might make sense to another human being but making a machine decipher is not that simple. These tasks are very much complicated and require a thorough understanding of the approach. The programmers need to have a clear concept themselves and thus construct the language driven applications through proper visualization of the entire scenario. NLP is the ultimate solution of breaking and reconstructing the language to understand it and make some sense of what is going on.

3.4 Convoluted Neural Networks

For image processing purposes convolutional neural networks perform really well. A digital image contains pixels whose value indicates what color it should be and what its brightness should be. Our brain processes a huge amount of data at the time of seeing an image. To cover the entire visual field each of these neurons is linked with each other. In the vision system each neuron responds to stimuli occurring in a limited region called the receptive field; similarly, CNN works in a similar fashion[29]. The arrangement of layers allows simpler patterns (lines, curves, and so on) to be detected first, and they precede more complex patterns (faces, objects, etc.). Vision can be provided to the computers using CNN. A CNN is constructed of 3 layers: 1) convolution layer 2)max pool layer and finally 3)fully connected layer. The convolution layer is what typically defines a CNN. Most of the computational load is carried by this layer. The layer functions by calculating the dot product of two matrices, one is preferably known as the filter and the other is the receptive field. The filter carries more information even though it does not take as much space as the image. If the image is of RGB type, it has 3 types of channels. The filter depth then spans all the 3 channels but width and the height are small. The filter goes through the image along its height and width and produces the image representation of that particular receptive field.

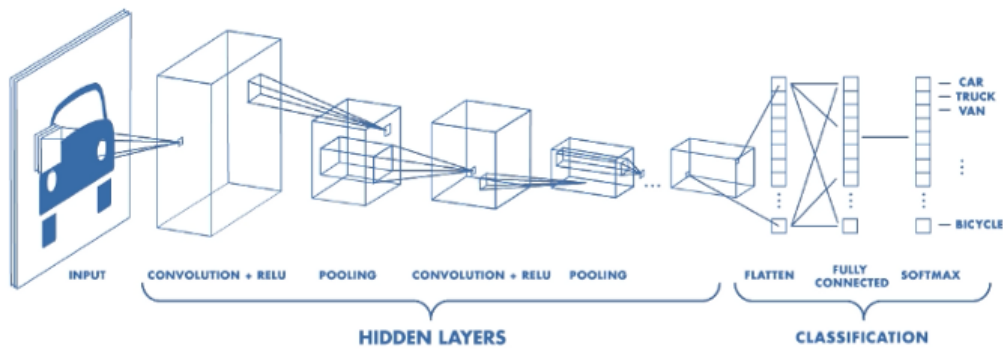


Figure 3.7: An overview on Convolutional Neural Networks

A feature map is a representation of a 2 dimensional representation which contains specific features of the image data. A stride is an amount by which the filter slides. The pooling layer is responsible for replacing network output at specific points which are determined by calculating the summary statistic of nearby outputs. The pooling operation processes each slice of the representation individually. Pooling functions include the rectangular neighborhood average, the L2 norm of the rectangular neighborhood, and a weighted average based on distance from the central pixel. Max pooling is the most popular process in CNN that takes the maximum of each layer in the feature map. Pooling provides some translation invariance in all cases, which means that an object is recognizable regardless of where it appears on the frame. Neurons in the fully connected layer, like those in regular FCNN, are completely connected to all neurons in the preceding and following layers. As a result, it can be computed using matrix multiplication followed by a bias effect as usual. The FC layer assists in mapping input and output representations.

3.5 MobileNetV2

MobileNet introduced by Google is one kind of convolutional neural network created for the vision applications for mobile. In the early stage it was created for face attribute detection. Sandler et al.[10] proposed the inverted residual structure which further advanced the previous MobileNet as we knew it. They proposed the module: inverted residual with linear bottleneck which could take in compressed low dimensional inputs and provide output by initially expanding and afterwards putting through lightweight convolution. They introduced the MobileNetV2 with the linear bottleneck layers applied within, which stopped the non-linearity from destroying too much information within the convolution layers. Unlike other CNN two stage object detection models,

MobileNetV2 detects objects in a single stage[28]. Meaning, it can predict the object location and classes directly instead of generating sparse objects in the first stage and predicting objects in the second stage. It uses predefined anchor boxes and treats object detection as a regression problem.

3.5.1 Architecture

Initially the MobileNet had a 3x3 depthwise layer and a pointwise 1x1 convolution layer, both with Relu6 activation function for s number of strides.

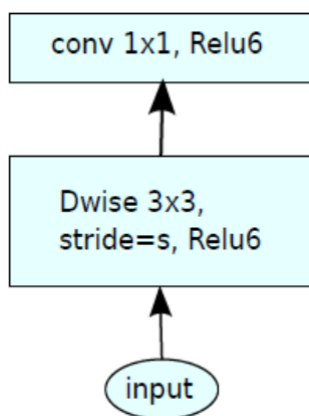


Figure 3.8: MobileNet[10]

But in the MobileNetV2, an additional convolution layer of 1x1 dimension has been provided before the depthwise 3x3 layer for stride = 1 block as explained[10]. On the second block, the feature map obtained from the convolution 1x1 layer is passed into the depthwise 3x3 layer with stride set to 2 and it is then passed to another convolution layer with dimension 1x1. All the layers contain the Relu6 activation function.

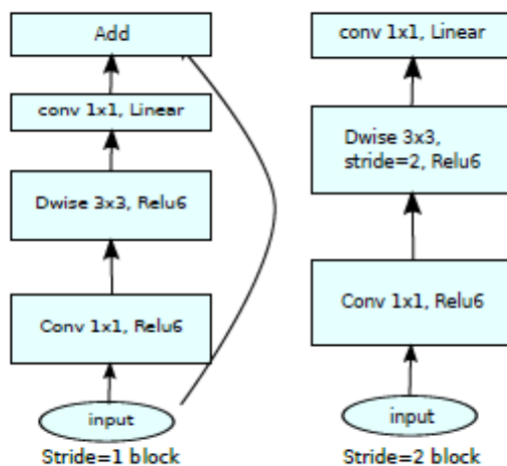


Figure 3.9: MobileNetV2[10]

Nguyen [25] further explained the architecture of MobileNetv2 presented by Sandler et al. [10] According to them the key changes the mobileNet architecture brought was the depthwise convolution layer and the pointwise convolution layer shown in Figure 3.10 as an extended version of Figure 3.8.

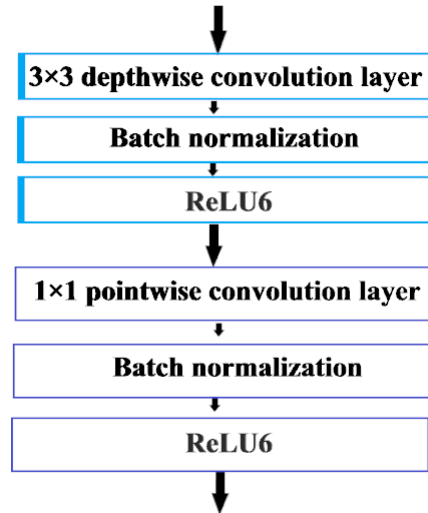


Figure 3.10: MobileNet[25]

The depthwise and pointwise convolution layers together offer much faster workflow than the regular convolution networks. MobileNetV2 is an even more efficient and powerful alternative to that. An extended version of mobilenetv2 architecture is shown in Figure 3.11.

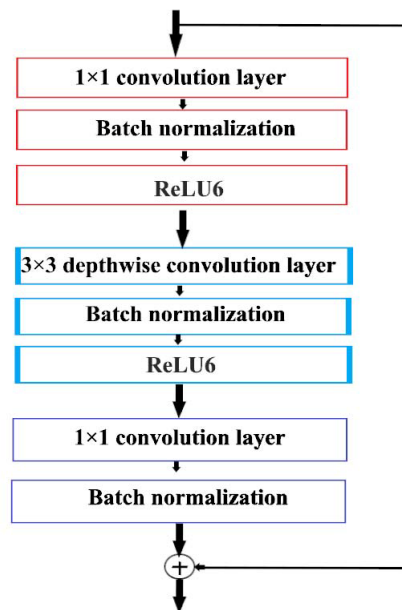


Figure 3.11: MobileNetV2 Architecture[25]

The first 1x1 convolution layer expands the number of channels obtained from the input

feature map before entering the depthwise convolution layer. The middle layer works in a similar fashion to MobileNetv1. The final bottleneck layer as shown in figure D and explained by Sandler et al.[10] converts the number of channels in the input feature map into an even smaller size. The residual connection in the MobileNetv2 helps the flow of gradients through the network. The MobileNetV2 has 17 depthwise separable convolution blocks that precedes a 1x1 convolution layer. But initially a 3x3 convolution layer has been applied containing 32 channels.

3.5.2 Experiments and Performance

The MobileNetV2 shows significant results in the ImageNet dataset as well as the COCO dataset in respect to the accuracy and model complexity[10]. The mobileNetV2 in terms of object detection outperformed the MobileNetV1 in the experiment done on the COCO dataset. Both versions of MobileNet were used as feature extractors. As a baseline they took YOLOv2 and the original SSD into consideration. This same task has also been done using the DeepLabv3 which is a semantic segmentation architecture. This part has been done on the PASCAL VOC dataset. MobileNetV2 allows to separate the network expressiveness from its capacity. Saxen et al.[16] also did a similar experiment on facial attributes extraction. They showed a comparison between the two architectures, MobileNetV2 and the NasNet as they wanted to consider two lightweight CNN architectures. In their paper a new version of MobileNet V2 was introduced which consists of approximately 3.47 million parameters with 300 million MACs. The experiment was done on the CelebA dataset. In their conclusion they stated that the NasNet compared to the MobileNetV2 needed 1.5 million more parameters and was about 40% slower while taking MACs(number of multiply accumulates) into account.

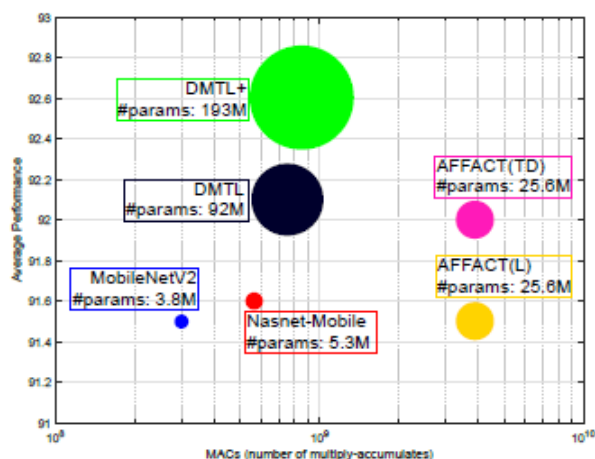


Figure 3.12: MobileNetV2 vs NasNet[16]

Applying the mobileNetV2 built on depthwise separable convolution layers and point wise separable convolution layers allows fewer multiply-add operations during each forward propagation. Such is claimed by Li et al.[28] They intended to design an object detection model deployable on mobile devices. But the noteworthy aspect here is how mobileNetV2 eased up their calculation.

Chapter 4

Research and Methodology

We started off our work by looking for a suitable dataset optimized for visual question answering. Even though we found quite a few such datasets, none of them supported Bangla.

The following sub sections describes our research methods in detail.

4.1 Introduction to Existing Dataset

Our primary task was to collect or select a suitable dataset for our research purpose, so we looked into datasets related to visual question answering, as our thesis topic is Visual Question Answering in Bangla. We were unable to locate any datasets related to the Bangla language. Although a lot of work has been done in the field of Visual Question Answering for the English language, there has been very little work done for the Bangla language. The majority of the datasets are in English. So, in order to meet our requirement, we created two datasets in Bangla for visual question answering. These datasets were created with the assistance of existing datasets related to the English language. So, first we will look at the existing datasets in English, and then move on to our newly created Bangla datasets. VQA v1 [4] is a well-known and widely used dataset for Visual Question Answering. On the 200k images from COCO [1], this balanced dataset contains approximately 1.1 million (image, question) pairs with approximately 13 million associated answers.

Total Number of Question: 248349
[3 3 3 ... 3 3 3]
Max number of questions on a image 3
Min number of questions on a image 3
Mean of questions on a image 3.0

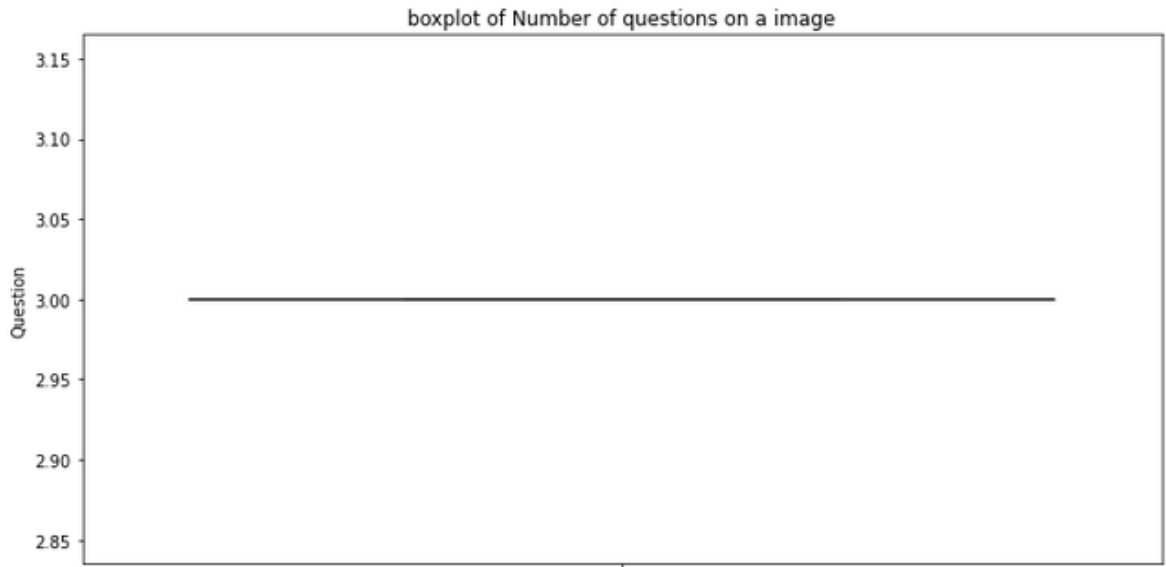


Figure 4.1: Boxplot of number of questions per image VQA v1

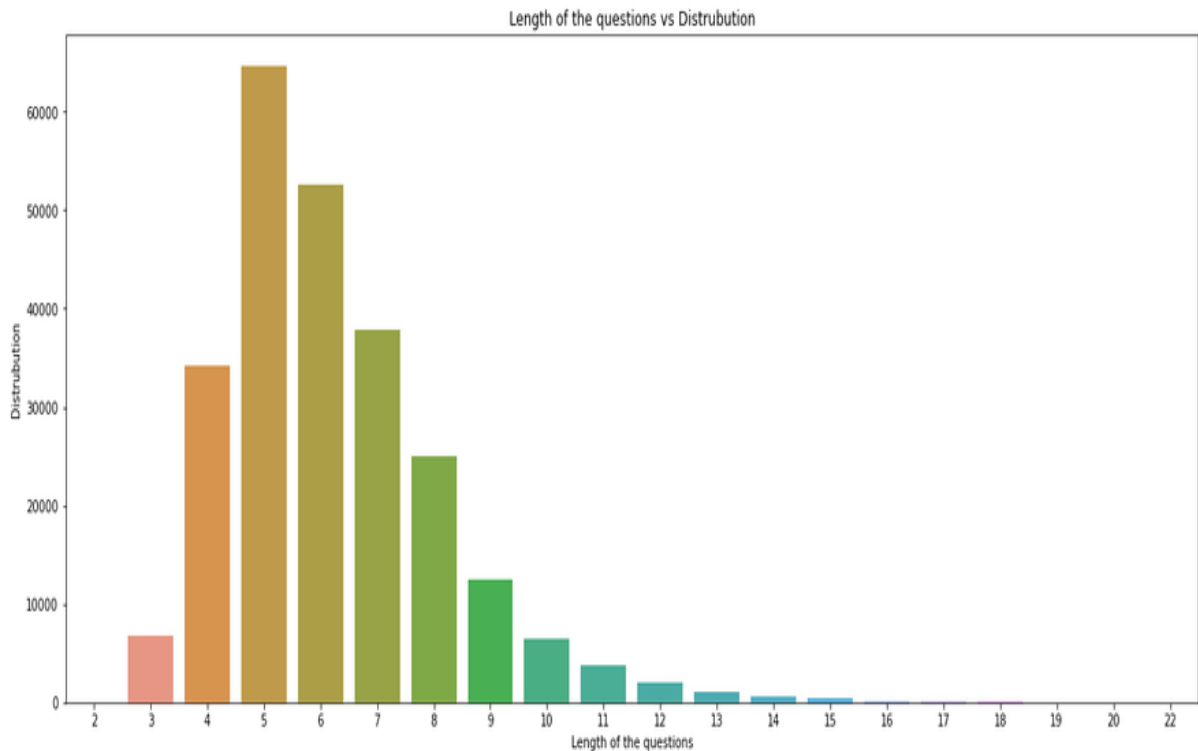


Figure 4.2: Length of the questions vs Distribution graph for VQA v1 training dataset.

As described in the paper and shown in the figure 4.1, the dataset is balanced with each image associated with three questions. From the graph above we can see that the majority of the questions have length of 5. More than 50k questions have length

of 5 to 6 words and very few questions have 15. Next we are going to see the images along with their corresponding questions and answers.

Ques. How many people are wearing glasses here?

Ans. 2



Ques. What room is the man in?

Ans. living room



Ques. Why is the plane in the water?

Ans. its landing



Figure 4.3: Sample data from the VQA dataset

The images are from the COCO [1] dataset. Each question is accompanied by an image id and a response. If we look at the length distribution of the answers in figure 4.4, we can see that the majority of the answers are only one-word long.

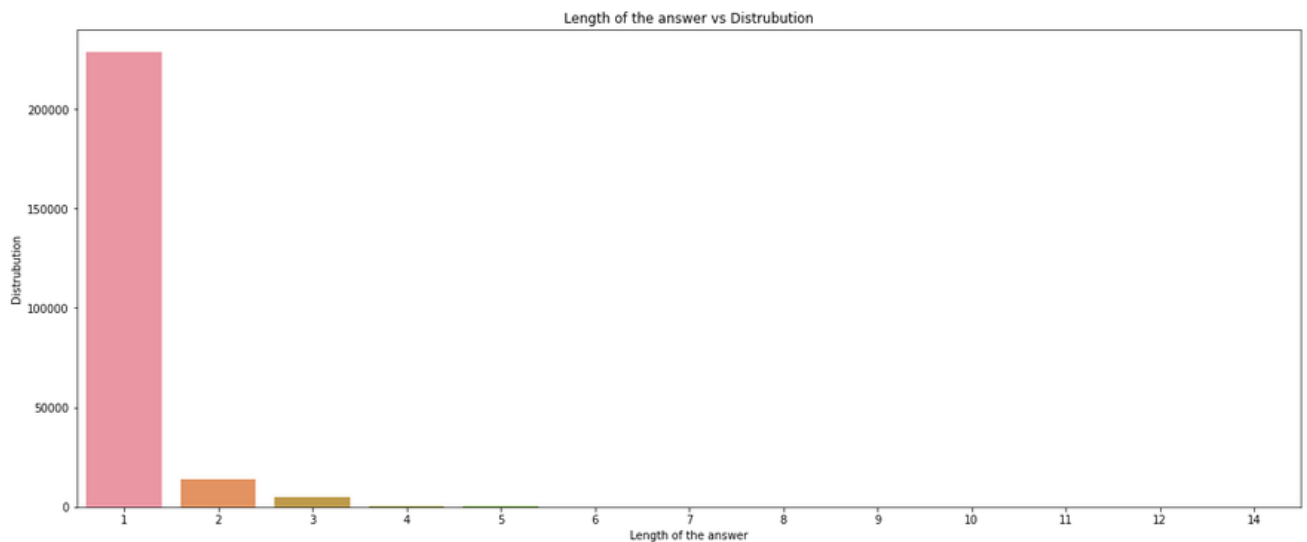


Figure 4.4: Sample data from the VQA dataset

The second dataset is CLEVR [5]. This dataset is intended for basic visual reasoning. The images do not come from the COCO dataset; they only contain objects and elements. The objects are all 3D shapes. CLEVR includes 100k rendered images and approximately one million automatically generated questions, 853k of which are unique. The dataset’s primary goal was to allow detailed analysis of visual reasoning.

From the boxplot shown in figure 4.5 we can see an image might have a maximum of 10 questions and the total question is 600k.

A key note here is that multiple questions are associated with a single image. This reduces the unavailability of probable answers to questions raised from different perspectives. All of these questions are provided with corresponding answers.

Total Number of Question: 699989
 [10 10 10 ... 10 10 10]
 Max number of questions on a image 10
 Min number of questions on a image 2
 Mean of questions on a image 9.999842857142857

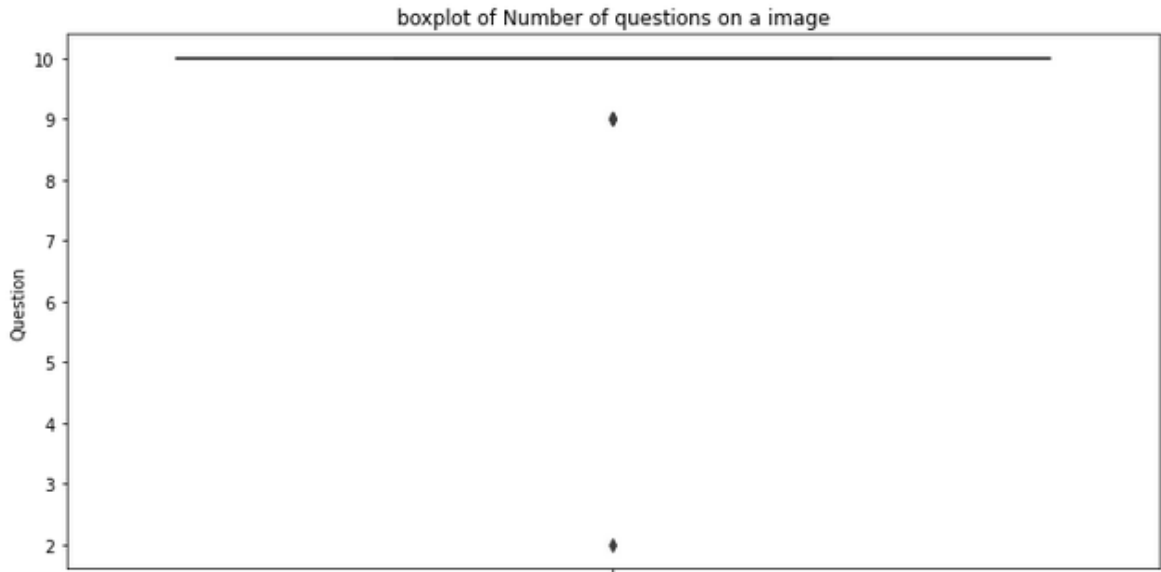


Figure 4.5: Total Number of questions along with question per image distribution of CLEVR dataset

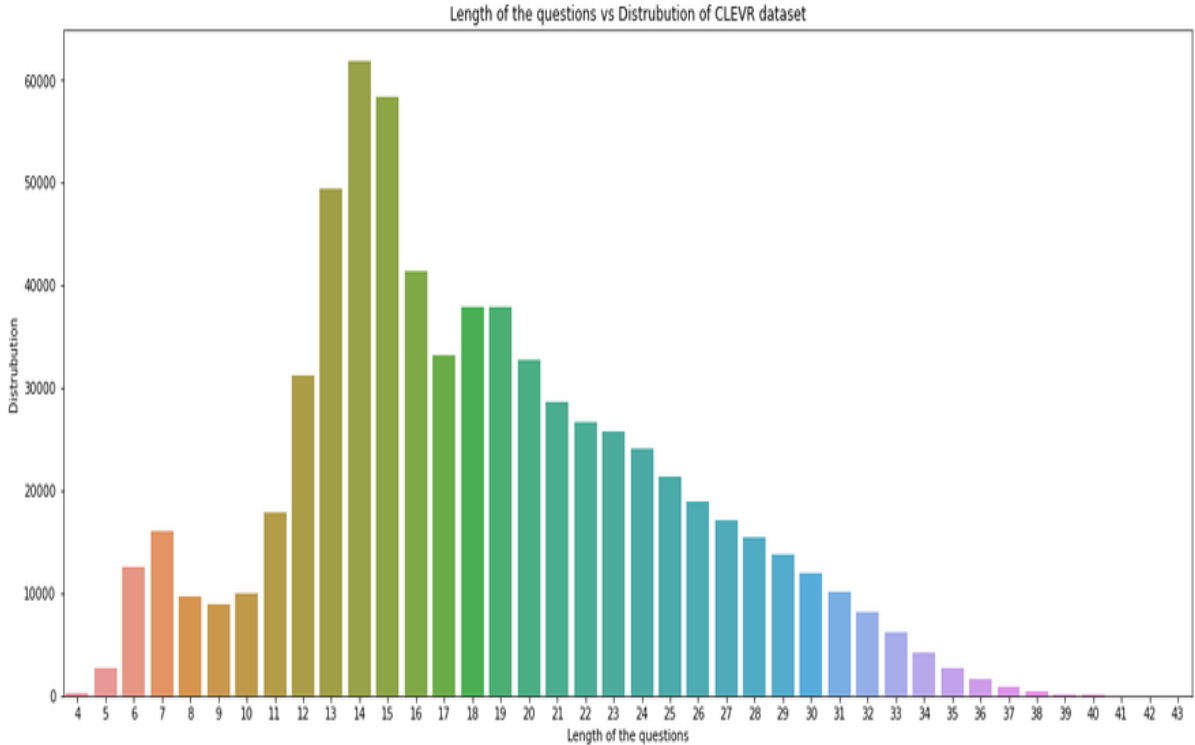


Figure 4.6: Length of the questions vs Distribution graph for CLEVR training dataset.

Figure 4.6 shows that the length of the questions vary. More than 60k questions can have a maximum length of 14. The majority of the questions in this dataset have an

answer. 57k questions have a length of 15 characters and 50k questions have a length of 13 characters.

4.2 Our Proposed Dataset

We couldn't find any dataset related to visual question answering, so we formulated our own Bangla dataset through the combination of the existing VQA datasets. Essentially, we translated the existing dataset's questions and answers and trained them using the same images. We created an automated pipeline for this translation. This pipeline consists of several Python scripts that will convert existing questions and answers to Bangla and create the new dataset. We used Google translator for this purpose.

```
translate_to_bangla( text ):
    converted_text_to_bangla = googletrans(text)
    return converted_text_to_bangla
for i in dataset:
    new_question=translate_to_bangla(i.Question)
    new_answer = translate_to_bangla(i.Answer)
    new_dataset.append(new_question)
    new_dataset.append(new_answer)
```

At first we used VQA v1 to create our Bangla dataset. Which consists of 3903 training data and 1901 validation data

All of these questions and answers are derived from the VQA v1 dataset via a pipeline, and we used images from the COCO dataset. Because questions are converted from English to Bangla, it is not guaranteed that the length of the questions will remain the same. The actual dataset contains over 1.1 million questions. However, for our thesis, we converted approximately 5000 questions and answers from this dataset. The image below shows that the majority of the questions are no more than four words long.

The length of the answers is very similar to the length of the actual dataset. When we translate a sentence from English to Bangla, we frequently find that the translation no longer conveys the meaning. In our case, we also had to deal with similar issues. Also, we discovered that not all of the questions are correctly converted. A number of converted questions did not bear the same meaning as the English questions. We had to manually inspect and remove all problematic data from our dataset. We used the same mechanism to analyze the CLEVR dataset. We used the dataset's existing 3D object images and our pipeline to convert the questions and answers to Bangla.



Question = জিরাফের পিছনে একটা গাছ আছে কি?
Actual Answer =হ্যাঁ



Question = এই ছবিটি কি খেলা?
Actual Answer =স্কিইং



Question = টেবিল কি রঙ?
Actual Answer =বাদামী



Question = এই সব অ্যাপল কম্পিউটার?
Actual Answer =না



Question = মহিলাটি কার সাথে ভ্রমণ করছে?
Actual Answer =শিশু



Question = সে কি বল আঘাত করবে?
Actual Answer =হ্যাঁ

Figure 4.7: Sample data from our proposed Bangla dataset

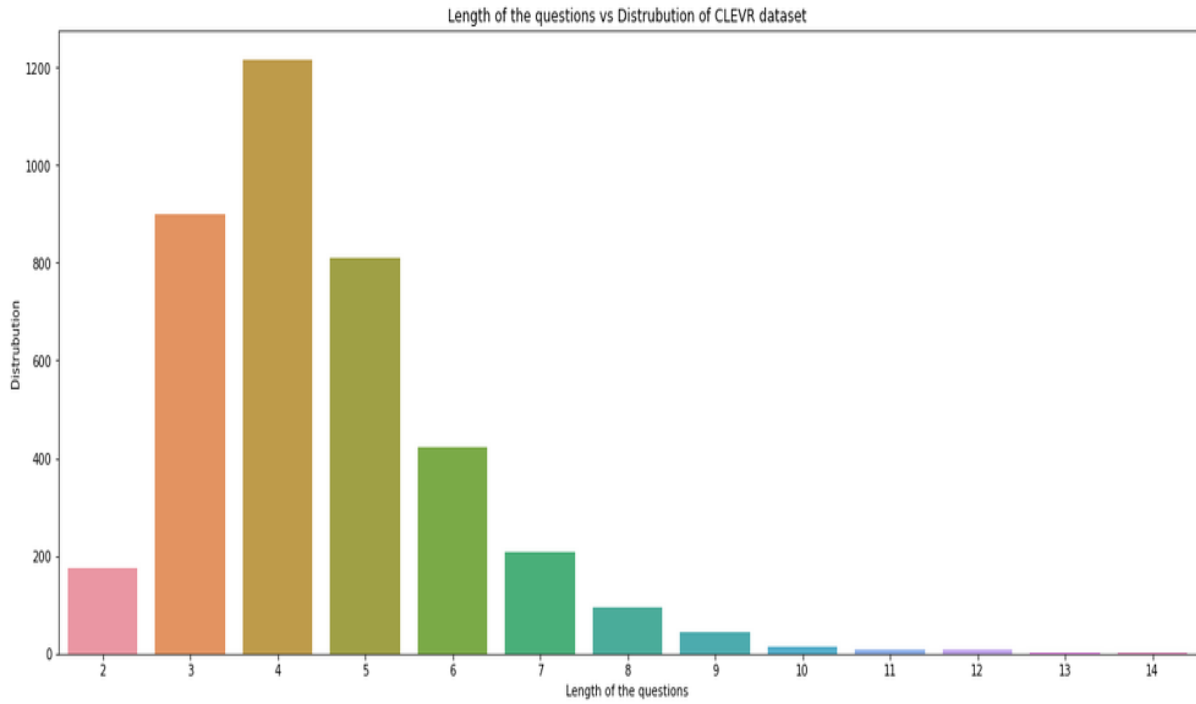


Figure 4.8: Length of the questions vs Distribution dataset of Bangla VQA dataset

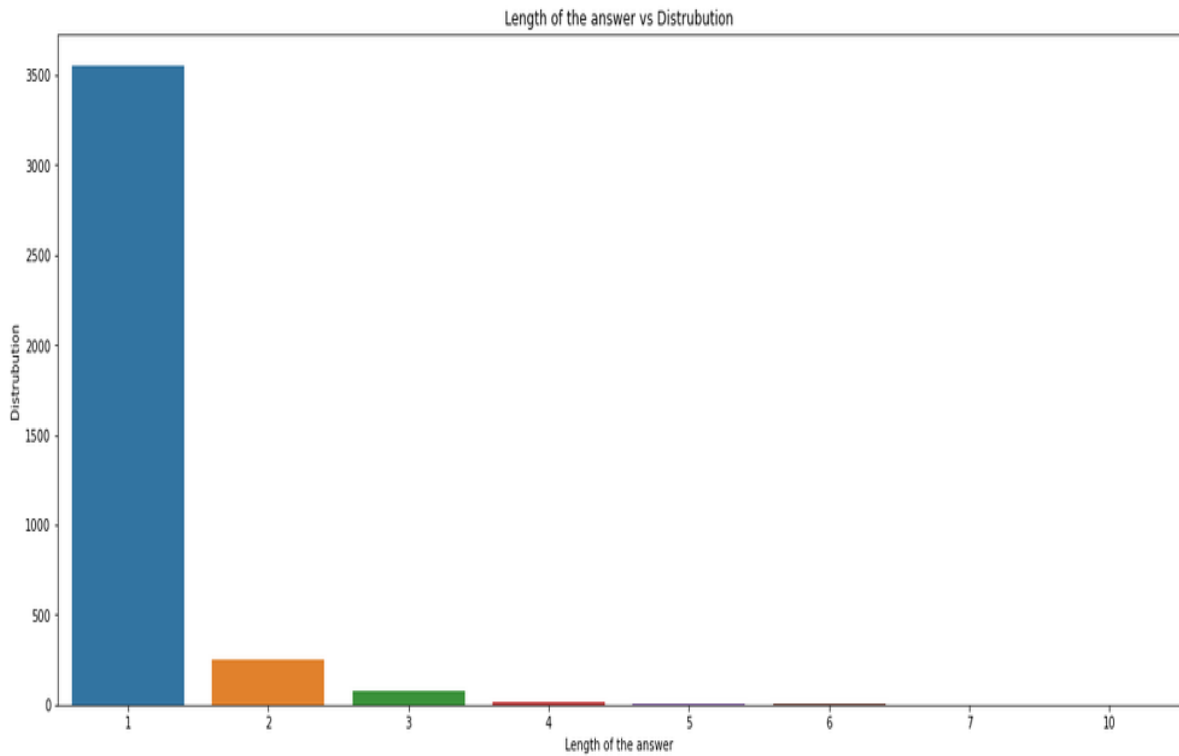


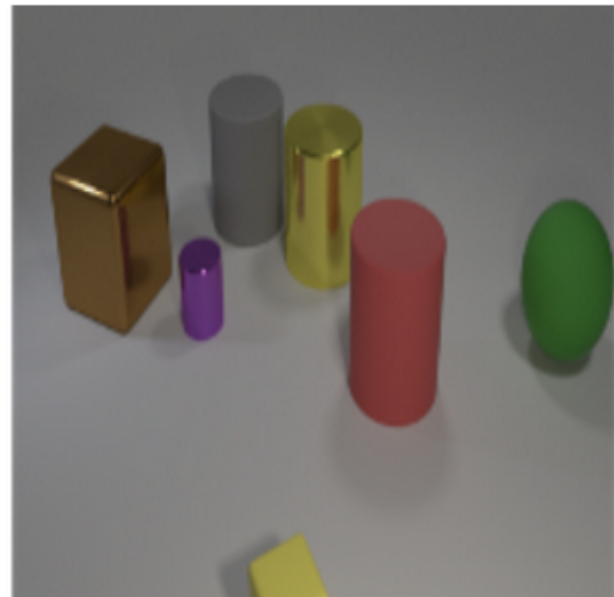
Figure 4.9: Length of the answer vs Distribution dataset of Bangla VQA dataset

This dataset contains over 12k questions accompanied by images and objects. Figure 4.11 shows that all of the questions have at least one answer. A single image can only have a maximum of ten questions. This has only images related to objects, and the

questions are not your typical ones. So when we changed it to Bangla, we noticed a significant difference from the original dataset.



Question = ক্ষুদ্র বাদামী জিনিস উপাদান কি?
Actual Answer =মেটাল



Question = রাবার ঘনক্ষেত্রের মতো একই আকারের রক্তবর্ণ বস্তুর আকৃতি কী?
Actual Answer =সিলিভার



Question = নীল জিনিসের ডানদিকে কোন বড় সিলিভার আছে?
Actual Answer =হ্যাঁ



Question = বড় সায়ান বস্তুর পিছনে বড় চকচকে বল কি রঙ?
Actual Answer =হলুদ

Figure 4.10: Sample data from our Bangla clever dataset

Total Number of Question: 12291
 [10 9 9 ... 9 10 1]
 Max number of questions on a image 10
 Min number of questions on a image 1
 Mean of questions on a image 9.670338316286388

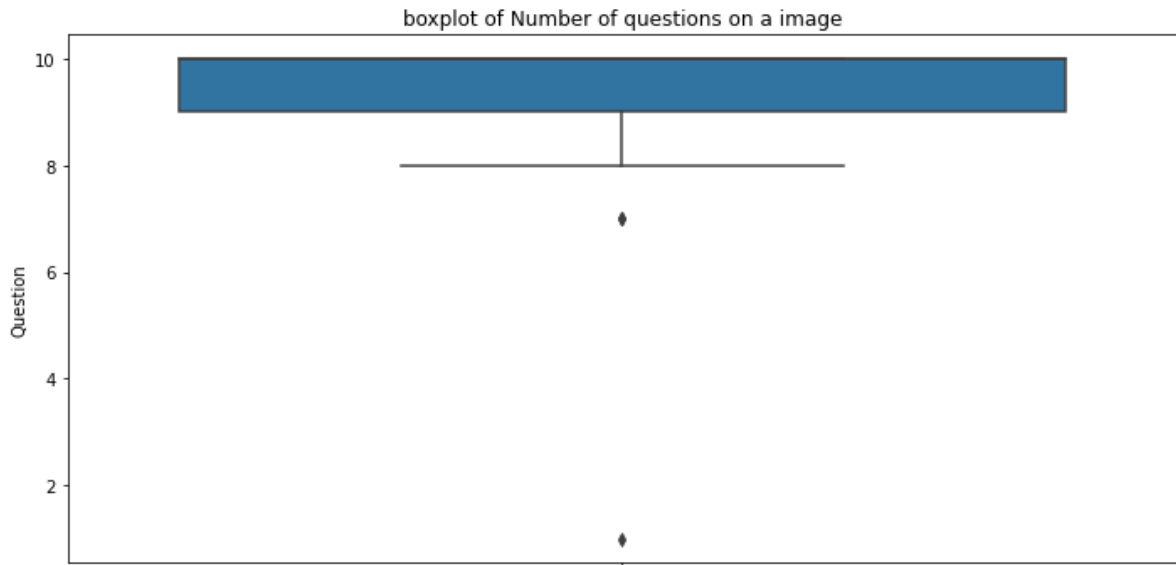


Figure 4.11: Never of questions on a image on Bangla CLEVR dataset

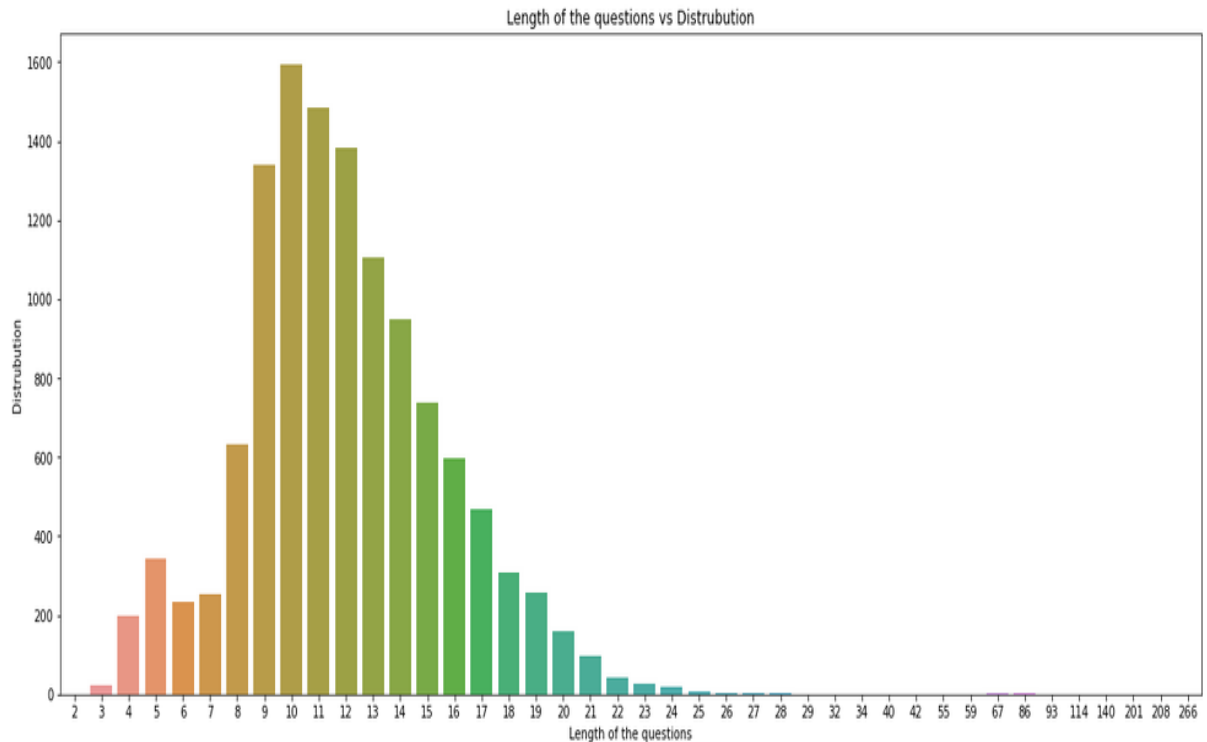


Figure 4.12: Length of the questions vs distribution for our Bangla CLEVR dataset

We can get nearly 1600 answers with a length of 10 words here. The length of our language model's input shape is extremely important to note. So, in our proposed model, which we have to use on these datasets, everything is similar except for some

minor differences, such as input length for our language model because the length of the questions, words per question, or answers are not the same in all datasets, so we had to tweak a little bit with the input length of our language model, which we will see later in section 4.4.

4.3 Data preprocessing

We had to face a few complications while preprocessing our data. Apparently after converting to Bangla, the structure of the datasets change and our model had to cope up with that. The following subsections explain how we pre processed our data.

4.3.1 Data splitting

Our dataset is made up of two types of data: image data and textual data. First our dataset was split into train and test datasets. We split our dataset into a ratio of 80:20.

4.3.2 Image preprocessing

Following the loading of an image, we need to decode the image data into pixel data in order to work with it. Tensorflow was used to decode the image data. The decoding function is determined by the image format. Generally we can use `tensorflow.image.decode_image` for generic decoding, but for JPEG decoding, we used `tensorflow.image.decode_jpeg`. Keyword argument is used to change the pixel format of the decoded image. Channel argument represents the number of integer value per pixel. Channels are set to 0 by default, which leads the decoding function to use the interpretation specified in the raw data. If we want grayscale image we can set the value to 1, if we set channel value to 3 it will allow us to use a RGB image. We know that a pixel value ranges from 0 to 255, but we can't use the exact pixel value in our dataset because the model will become biased toward higher pixel values, so we must normalize the pixel value by decoding. Data normalization is the process of converting real-valued numeric attributes into a value between 0 and 1. This assists our model in becoming less biased toward higher numeric values. To normalize our pixel values, we divided the value by 255. It will scale our data points from 0 to 1. We chose a fixed input size for the image data in our model, which is 200*200, so we had to resize the shape of our images to 200*200. To do so, we used the `tensorflow.image.resize` function.

4.3.3 Text preprocessing

This step was not as simple as the image data step. Our different datasets have different question lengths, which must be acceptable to the LSTM input layer, so the size of the input layer is not the same for the Bangla VQA dataset and the Bangla CLEVR dataset. We had to start by developing an encoding mechanism for our text data. There are several methods for encoding categorical data. 1. Label Encoding 2. One-hot Encoding

Ordinal Encoding : In this form of encoding we map a unique label to an unique integer value .

One-hot Encoding :In this kind of encoding each label is mapped to a binary vector .For our datasets we have used an Ordinal Encoding mechanism which will assign a unique integer value to our words. Before using the encoder, we need to tokenize the full sentence. Tokenizer will split the sentence into small segments and return an array then our encoder will encode the words and assign a unique integer to each word.

["ls" , "there" , " a" , "shadow"]

Figure 4.13: Tokenizer split the sentence into small segments

["ls" , "there" , " a" , "shadow"]
↑ ↑ ↑ ↑
[108 11 105 99]

Figure 4.14: Encoder encodes a value and assigns a unique integer.

In most cases we get one single integer value for a word but in some cases, we get an extra integer value in the array. So, we have to keep in mind that the length of the sentence and the length of the encoded array might not be the same as always. For encoding the data, we used the same tensorflow encoder for both Bangla and English

text but we did not use the same tensorflow tokenizer for both languages. Tensorflow tokenizer was not splitting our Bangla sentences as expected.

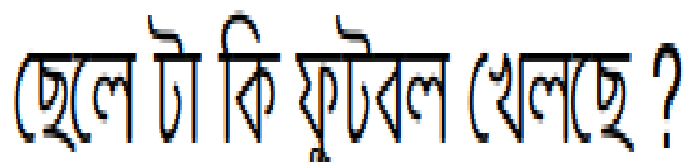


Figure 4.15: Example question from Bangla VQA dataset

For example, if the sentence is like figure 4.15, we expect our tokenizer to split the sentence into words but the tensorflow tokenizer was tokenizing the sentence like the figure 4.16.

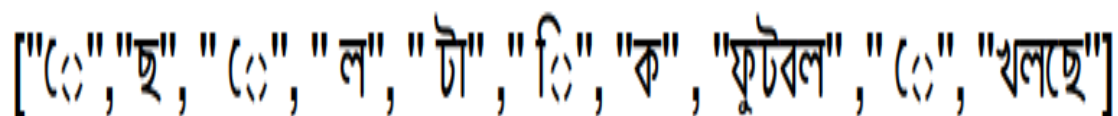


Figure 4.16: Tensorflow tokenizer tokenizing Bangla text

So, our encoder was assigning values to each of these absurd tokens and we could not obtain our expected result. To avoid this issue, we created our own tokenizer. Basically, our tokenizer splits the sentence by whitespace and return the value in an array

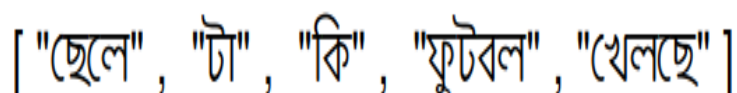


Figure 4.17: Output from our tokenizer

We used the same encoder to encode these values but we get extra integer values than the English text. So, we have different size in our Embedding layer for the English text and for the Bangla text. The output of our tokenizer is shown in figure 4.17.

4.3.4 Vocabulary set

We created a vocabulary set which contains all the unique words from our dataset. All the unique words we have in the questions and answers, is placed in this set and passed as constructor input to our tensorflow tokenizer. Tensorflow tokenizer expects a set containing all the unique words since it will assign a unique integer value to each of these unique words. So, when we call the encoder to encode a word the encoder returns a unique integer number according to the vocabulary set

4.3.5 Padding

We have used padding only for the text data. All the questions do not consist of the same length. A question might have 3 words another question might have 14 words so we need to use padding to fill up the gap. Our input text will be an encoded array and we need to keep a fixed size for the input shape after tweaking some values with respect to our question length. We have fixed a shape of 50 for the English text for both VQA v1 and CLEVR dataset and we used padding, accordingly. So now the length of the padding would be:

$$\text{padding} = [[0 , 50 - \text{shape_of_the_question}]]$$

All padding values will be set to a fixed value of zero. To add padding to our question, we used `tf.pad`. However, in the case of our Bangla text, we considered the question shape to be 150 in the Bangla CLEVR dataset and 50 in the Bangla VQA dataset. While working with the CLEVR dataset, we discovered that some questions have lengths of 111 or 147 after encoding. So, we chose 150 as a fixed shape otherwise we would have an error while passing the question in our language model. If the language model has a fixed shape less than the Max length of the encoded question, it will not fit into the model. So, now that we've established a fixed question input shape of 150 for the Bangla CLEVR dataset, our padding will look like this:

$$\text{padding} = [[0 , 150 - \text{shape_of_the_question}]]$$

4.4 Model Implementation

Till now we have discussed our dataset and how we have prepared our dataset .In this chapter we will explain about our proposed model and its implementation. Basically our model is a combination of CNN and LSTM layers. Our input to the model will be the image along with a question and the output will be an answer . So we concat

CNN with LSTM . We have trained 4 datasets using this model. We had to make some slight changes to run our model using these datasets. For example when we used the Bangla CLEVR dataset we provided 150 as the input shape for the LSTM input layer but for the Bangla VQA dataset we used 50 as our input shape for the LSTM layers . The length of the vocabulary set discussed in the previous chapter plays an important role in our model as this length is considered as the shape of the Embedding layer and it is also considered as the shape of the final output layer.

```
....._.
```

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	[(None, 150)]	0	
embedding_4 (Embedding)	(None, 150, 256)	95488	text_input[0][0]
bidirectional_12 (Bidirectional)	(None, 150, 512)	1050624	embedding_4[0][0]
image_input (InputLayer)	[(None, 200, 200, 3)]	0	
bidirectional_13 (Bidirectional)	(None, 150, 512)	1574912	bidirectional_12[0][0]
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2257984	image_input[0][0]
dropout_9 (Dropout)	(None, 150, 512)	0	bidirectional_13[0][0]
global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 1280)	0	mobilenetv2_1.00_224[0][0]
bidirectional_14 (Bidirectional)	(None, 1024)	4198400	dropout_9[0][0]
concatenate_4 (Concatenate)	(None, 2304)	0	global_average_pooling2d_4[0][0] bidirectional_14[0][0]
dropout_10 (Dropout)	(None, 2304)	0	concatenate_4[0][0]
output (Dense)	(None, 373)	859765	dropout_10[0][0]

```
=====
```

Total params: 10,037,173
Trainable params: 10,003,061
Non-trainable params: 34,112

Figure 4.18: Model structure

A more detailed architecture of our model would be like the figure 4.19

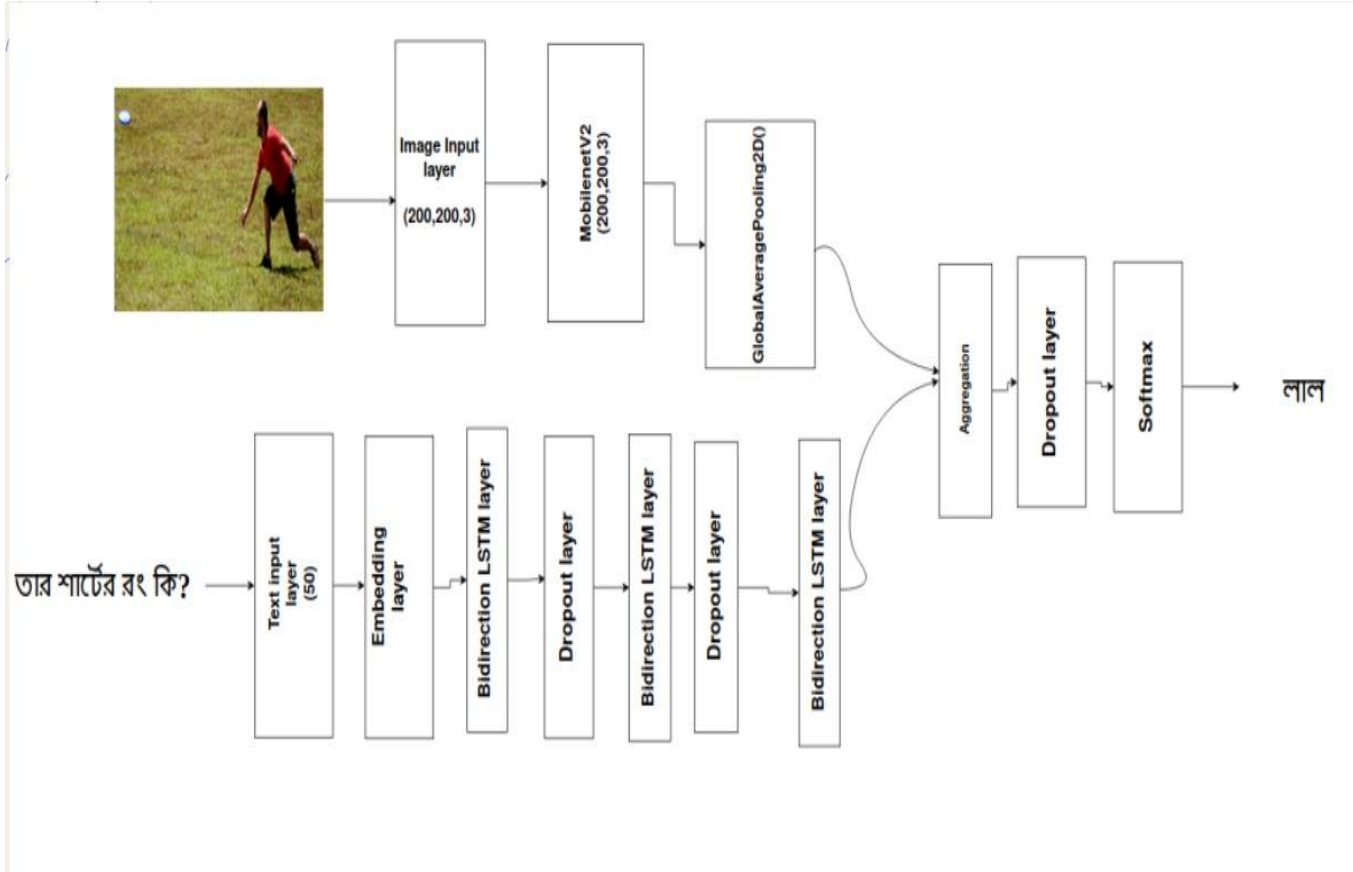


Figure 4.19: A detailed architecture of our proposed model

4.4.1 Input layer

For our image dataset we have chosen a fixed shape of (200,200,3) . And for the question part we have a fixed input shape of 50 for CLEVR English dataset , VQA v1 dataset , and for Bangla VQA dataset . Only for Bangla CLEVR dataset we chose 150 as an input shape as described in chapter 4 . Sometimes we get a length of 111+ from our encoded questions in Bangla CLEVR dataset. So this is the only change we had to make in our model architecture to work with all four datasets.

4.4.2 Mobilenetv2 layer

After the input layer we used a pre-trained mobilenetv2 model which has an input shape of (200,200,3) and an alpha value of 1.0 . This alpha value controls the width of the network . This is known as the width multiplier in the mobilenet paper[10]. If $\alpha = 1$ default number of filters from the paper are used at each layer. If $\alpha > 1.0$ proportionally increases the number of filters in each layer. If $\alpha < 1.0$ the number of filters is proportionally decreased

4.4.3 Embedding layer

We added an embedding layer which has the shape of the vocabulary set which contains all the unique words from the questions and the answers. This layer helps us to convert each word into a fixed length vector of defined size.

4.4.4 Bidirectional LSTM layer

We have 3 bidirectional LSTM layers . First two layers return the sequences but the last layer does not. First two bidirectional layers have the dimensionality of 256 of the output space . The last bidirectional layer has the output dimensionality of 526.

4.4.5 Dropout layer

This layer is useful for avoiding the overfitting problem. An issue with LSTMs is that they can easily overfit training data, reducing their predictive skill. Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. At first while training our dataset we faced issues related to overfitting then after adding the Dropout layer after the first two LSTM layers the overfitting problem got resolved. For dropout we can choose a value which is suitable for us. It is a hyper parameter which we need to determine. Most popular value for dropout is 0.5 - 0.9 . We chose 0.5 for our model.

4.4.6 Output layer

After concatenating the CNN model and the LSTM layer we put another layer of dropout and then we added a softmax activation which takes the length of the vocabulary set as dimensional size.

4.4.7 Loss function

As we have used Integer encoding, it means that we have assigned a unique value to each word in our vocabulary set as we have described in chapter 4. So we choose sparse_ categorical_ crossentropy . This is a loss function for multi-class classification models where the output label is assigned an integer value (0, 1, 2, 3...).

4.4.8 Optimizer

We have explored two optimizers SGD (stochastic gradient descent) and ADAM optimizer.

SGD: A class that implements the stochastic gradient descent optimizer we can pass a learning rate and momentum value in the constructor.

Adam: Another stochastic gradient descent based optimizer is ADAM optimizer.

For our model we use Adam optimizer as it was generalizing the data better than SGD. Here the default learning rate was 0.001 and momentum was 0.9.

4.4.9 Batch size

Batch size plays a vital role in Deep Learning algorithms. Batch size is a term used in machine learning that refers to the number of training examples used in one iteration. We can have 3 kinds of batch sizes. If the batch size is 1 then the algorithm will follow stochastic gradient descent if the batch size is greater than 1 but less than the length of the training dataset then the algorithm will follow mini batch gradient descent.

Chapter 5

Experimental Evaluation

In the following subsections we will provide our findings from the experiments, the accuracy of our datasets, and a comparison between our datasets and the English CLEVR and VQA v1 datasets.

5.1 Experimental Setup

For our experimental setup, we used tensorflow with CUDA core GPU support. Our machine has a 4 core i5 CPU and 1050 ti GPU of 4GB memory. In another PC setup we had 4 core CPU and 1660 GPU which consist of 8GB memory. We used Ubuntu 20.04 as our operating system. The support of CUDA core made our training much faster.

5.2 Result and analysis for VQA v1 dataset

We ran our model on the English VQA v1 dataset first. Figure 5.1 shows the accuracy vs epoch graph obtained from VQA v1.

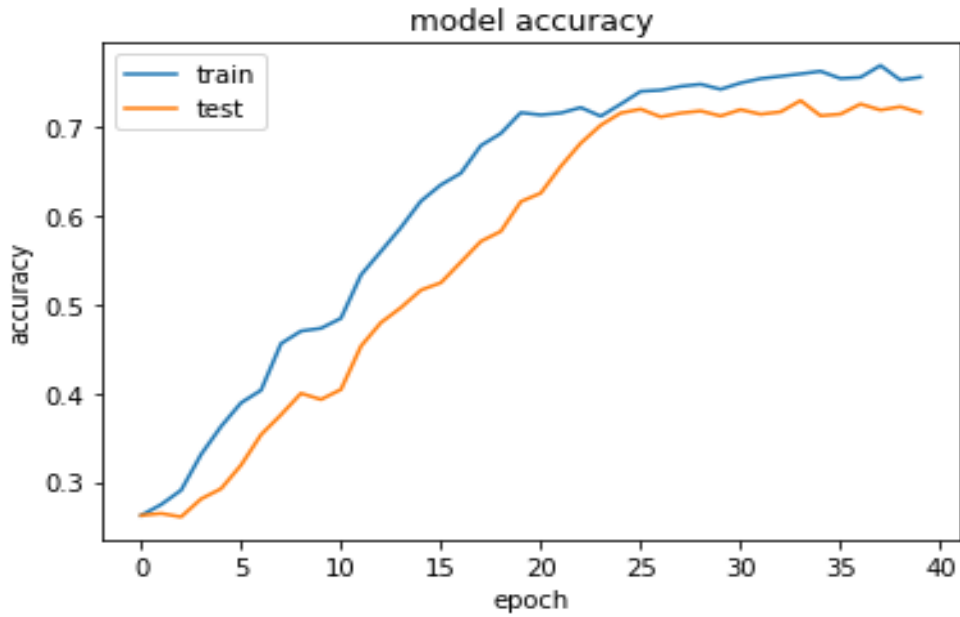


Figure 5.1: accuracy vs epoch for VQA v1 dataset

The figure 5.2 provides the loss vs epoch graph. As we ran for 40 epochs, our test and train loss both decreased gradually and ended up at 0.5% for test loss and we got below 0.5% of train loss.

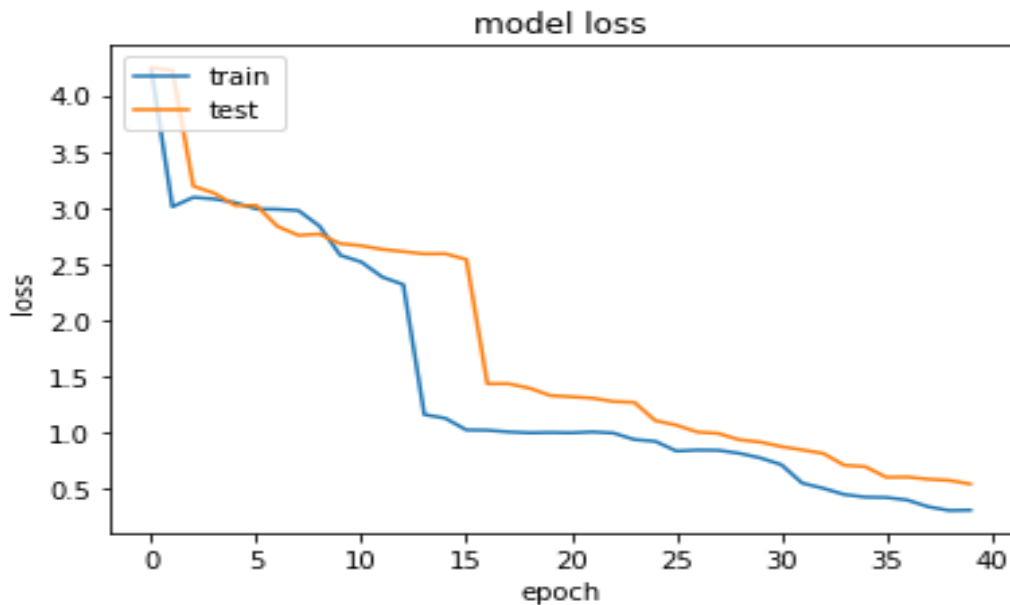


Figure 5.2: loss vs epoch

We took 5000 traindata and ran the model for 40 epochs. Our Train accuracy is 75% Test accuracy is 71%.

Figure 5.3 shows a sample output from our experiment.

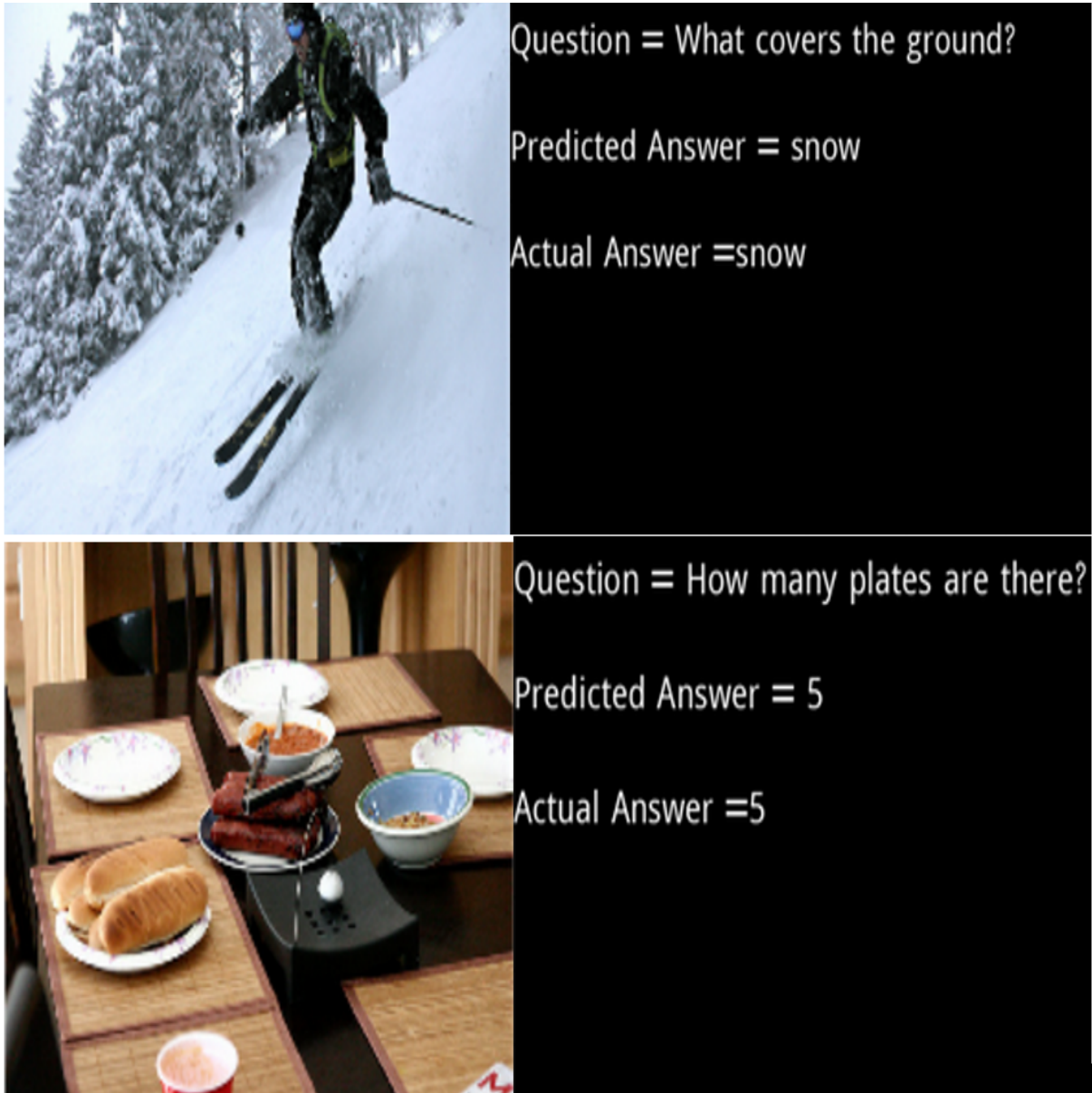


Figure 5.3: Example of output of from our Model trained on VQA dataset h

5.3 Result and analysis for Bangla VQA dataset

Figure 5.4 shows the model accuracy on Bangla VQA dataset.

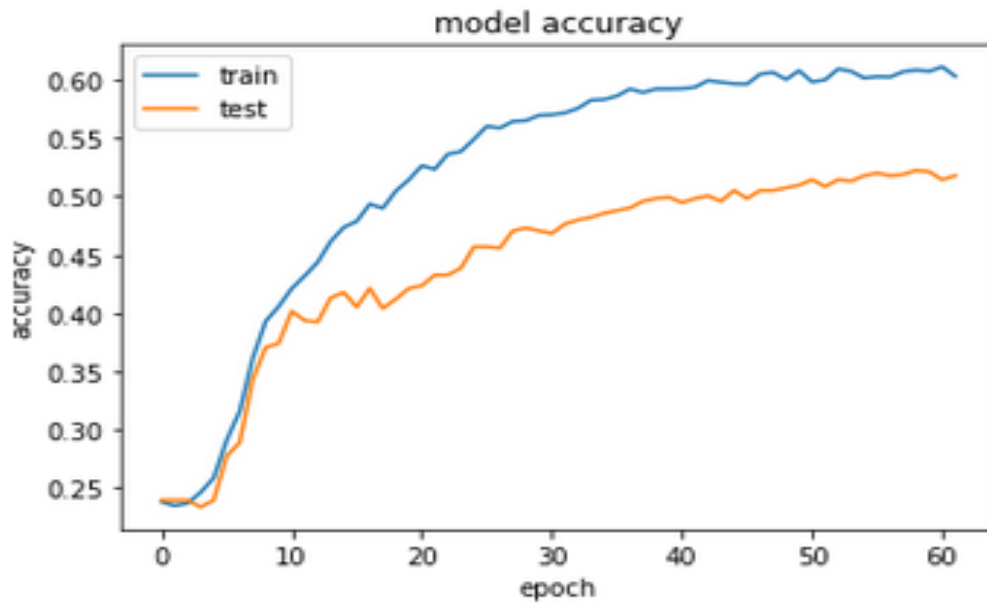


Figure 5.4: Accuracy on Bangla VQA dataset

The fig 5.5 shows the loss function on Bangla VQA dataset which we ran for 60 epochs. where we can see the test loss suddenly increased after 5 epochs and gradually drops after 5 epochs. Then the test loss function maintain almost a constant value of 2.6% and train loss function decreases from 5% to 2.2%.

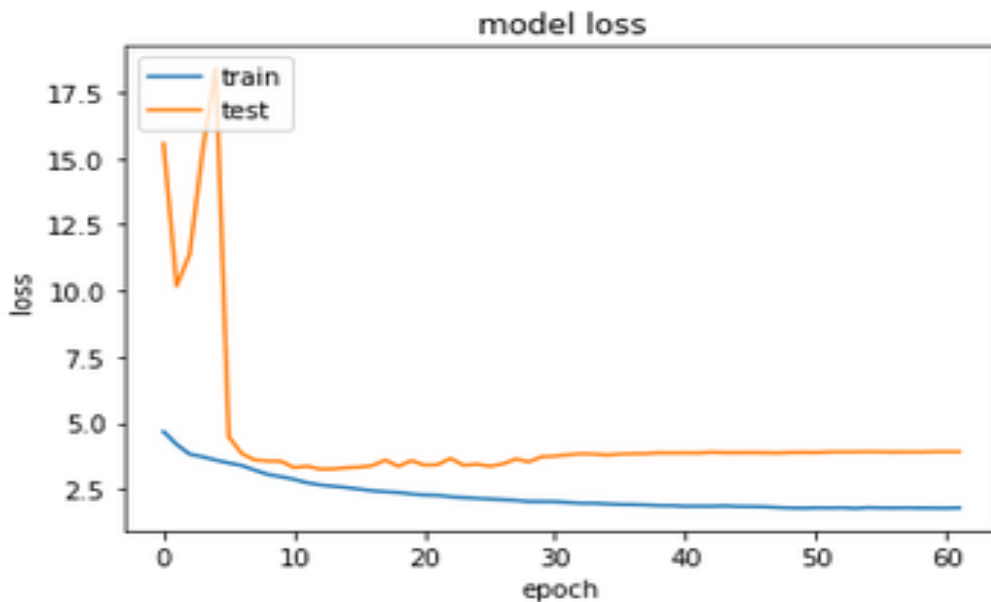


Figure 5.5: Loss Function on Bangla VQA dataset

For our Bangla VQA dataset we took about 4000 dataset from our Bangla VQA dataset and we have run for 60 epoch and our train accuracy was 63% And our test accuracy 55%.

Fig 5.6 is an example output of our Bangla VQA dataset.

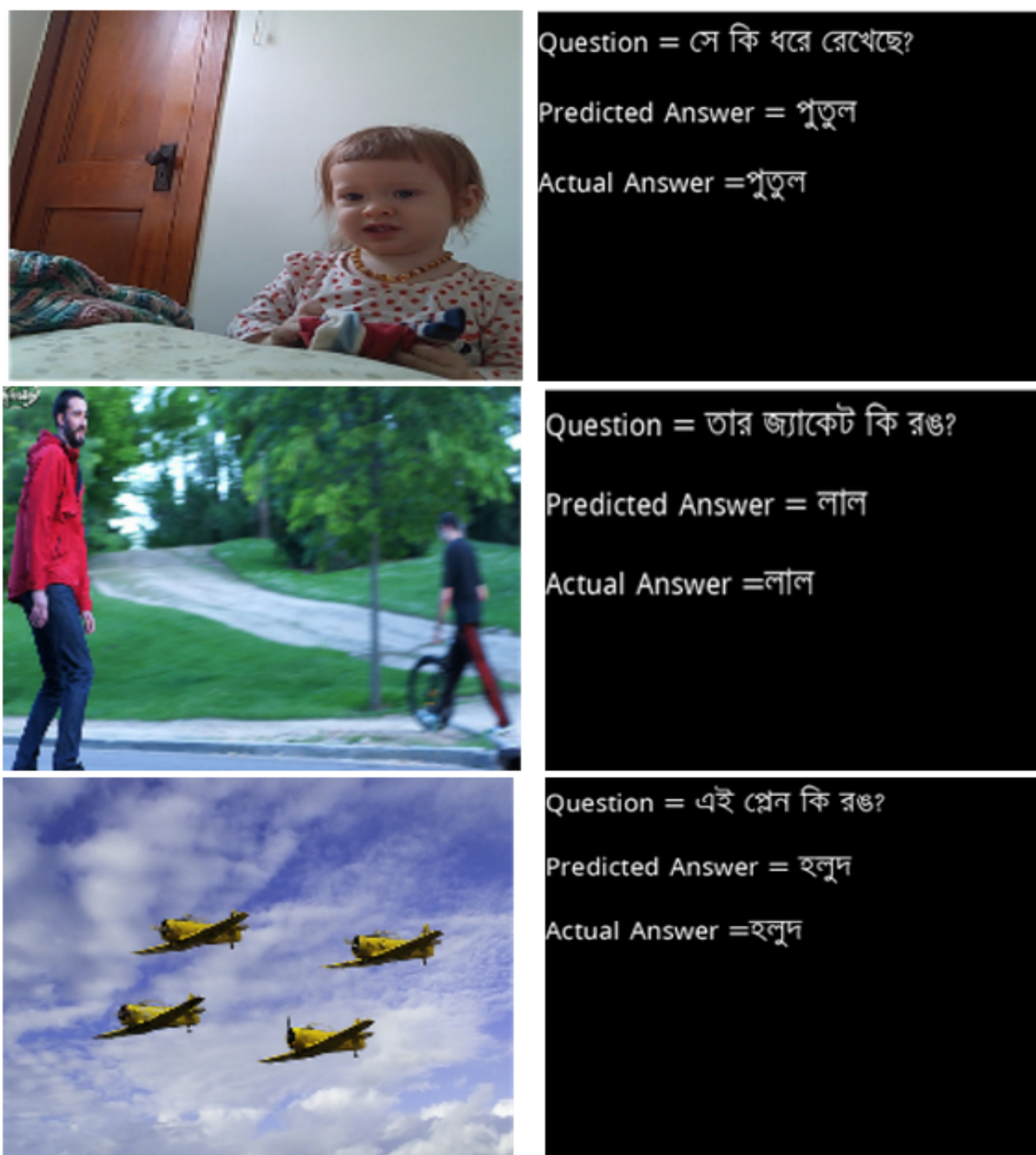


Figure 5.6: Example output of our model trained on Bangla VQA dataset

5.4 Result and analysis for CLEVR dataset

Fig 5.7 contains the accuracy graph of our model trained on CLEVR dataset. From the graph we can see our train accuracy reached upto 77% whereas our test accuracy

reached upto 66%. We ran our model for 20 epochs.

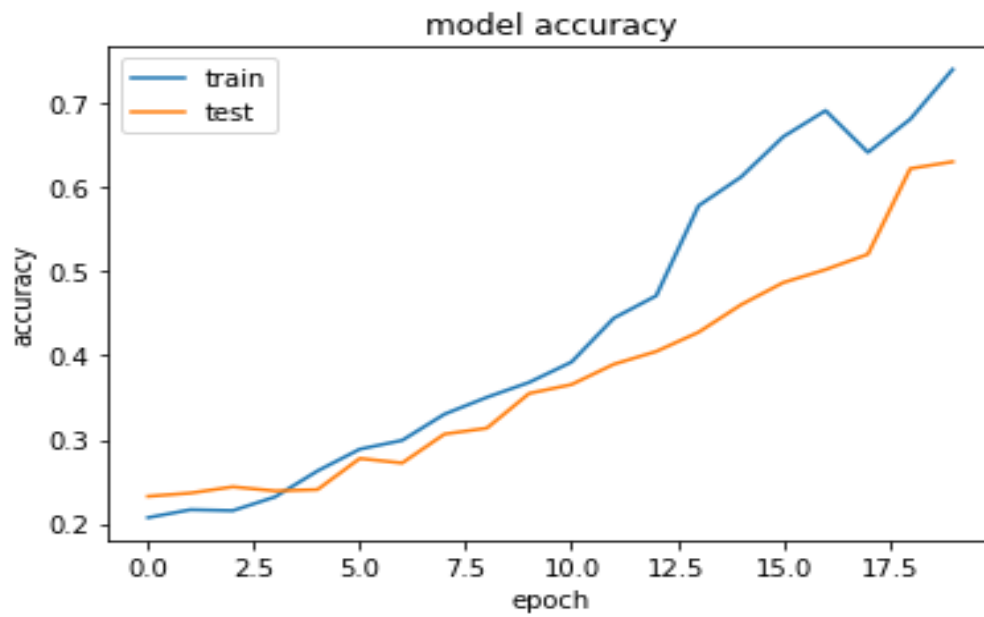


Figure 5.7: Accuracy of our model trained on CLEVR dataset

As we ran for 20 epochs our test loss decreased to 0.5% and the train loss decreased to below 0.5%. And both of them decreased gradually. The figure 4.8 shows the loss function graph of CLEVR English dataset.

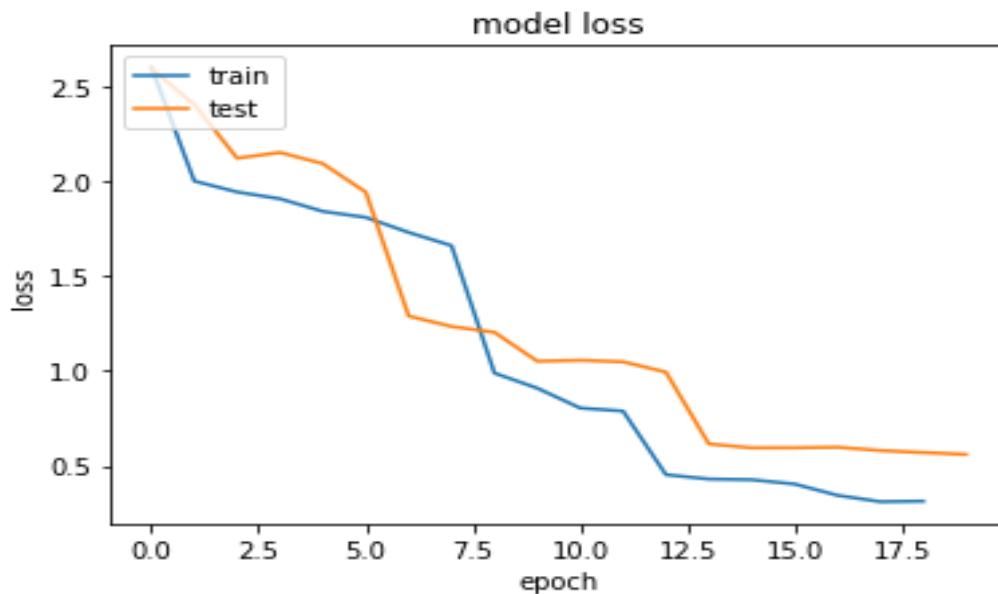


Figure 5.8: lost function of our model trained on CLEVR dataset

Figure 5.9 shows an example output from the CLEVR dataset.

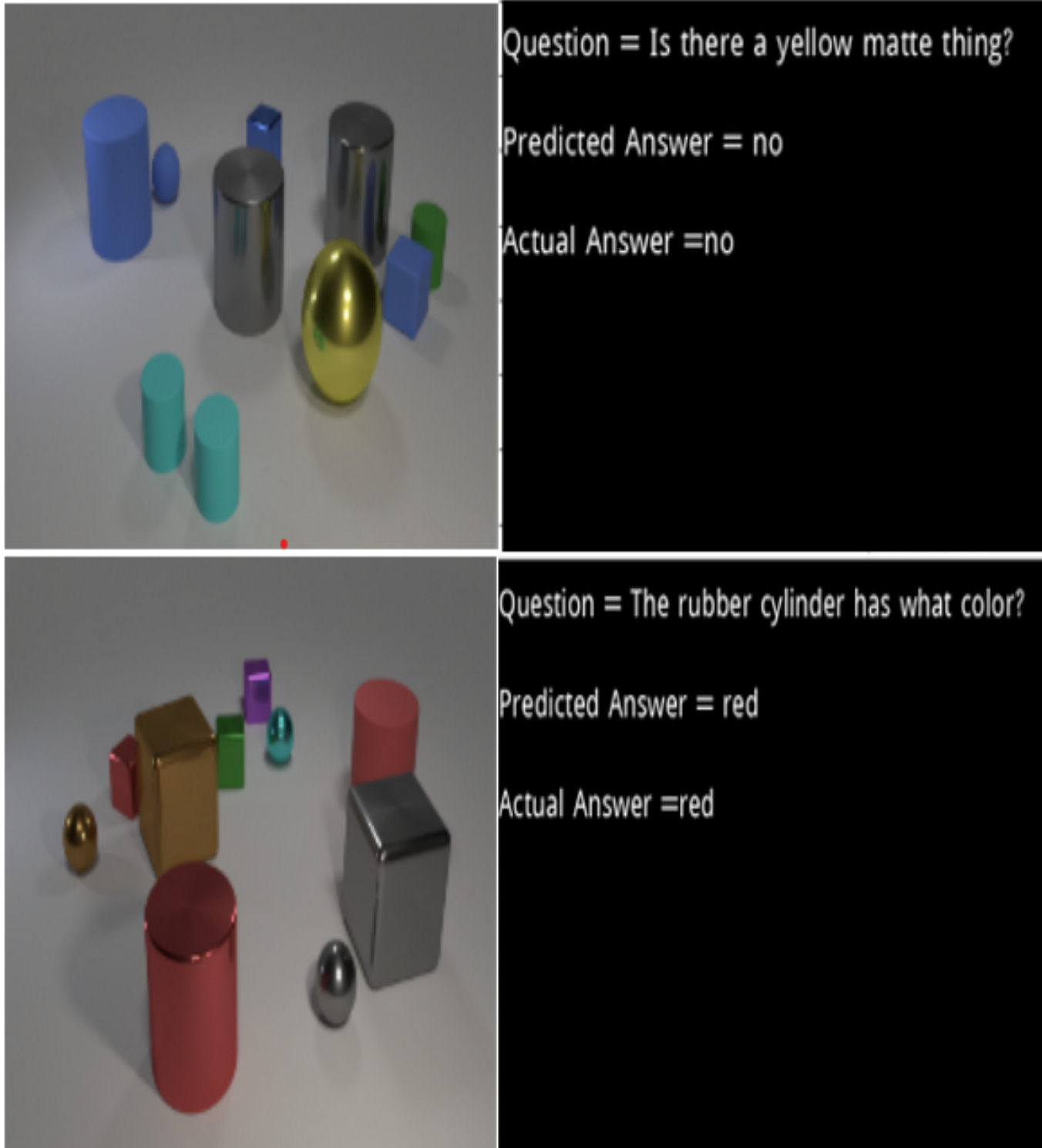


Figure 5.9: Example output of our model trained on CLEVR dataset

5.5 Result and analysis for Bangla CLEVR dataset

Figure 5.10 contains the accuracy vs epoch graph of our model trained on the Bangla CLEVR dataset.

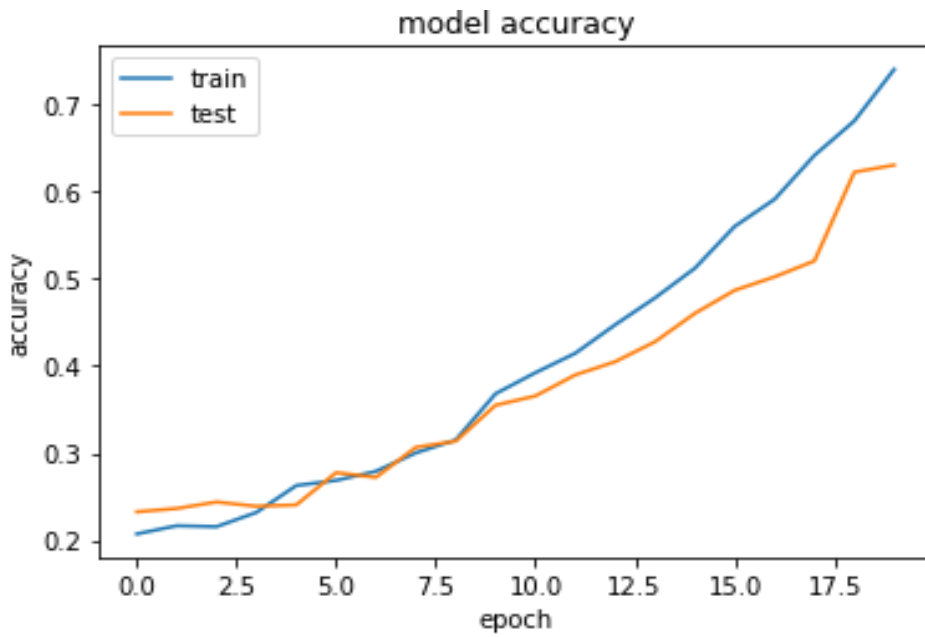


Figure 5.10: Accuracy vs epoch graph of our model trained on Bangla CLEVR dataset

For our Bangla CLEVR dataset we ran for 20 epochs and both of the train and test losses reached 0.5% and we can see a smooth decrease of the loss function. The figure 5.11 shows the loss vs epoch graph for our Bangla CLEVR.

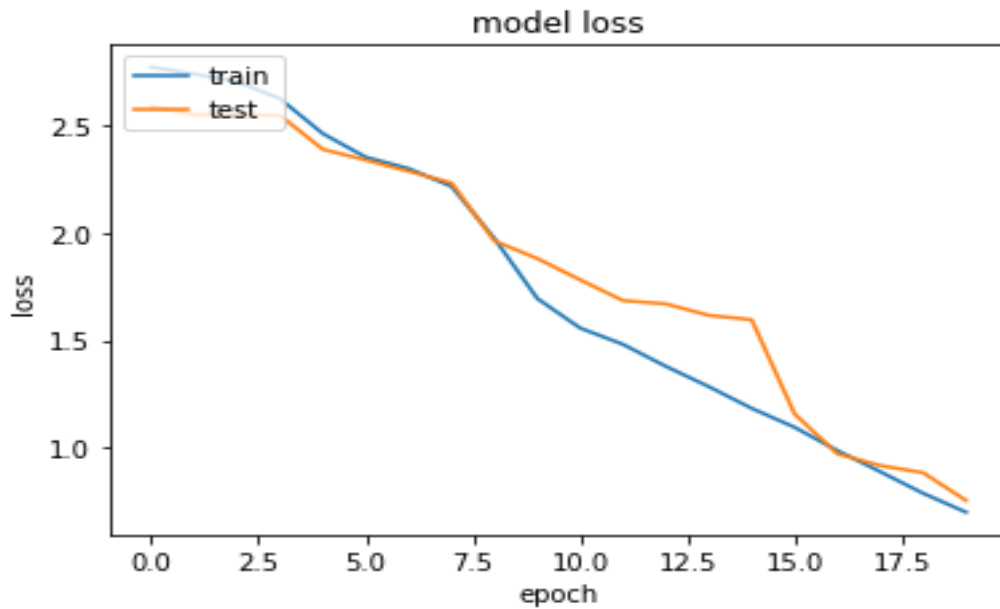


Figure 5.11: loss vs epoch graph of our model trained on Bangla CLEVR dataset

An example output from our Bangla CLEVR dataset is shown in figure 5.12.

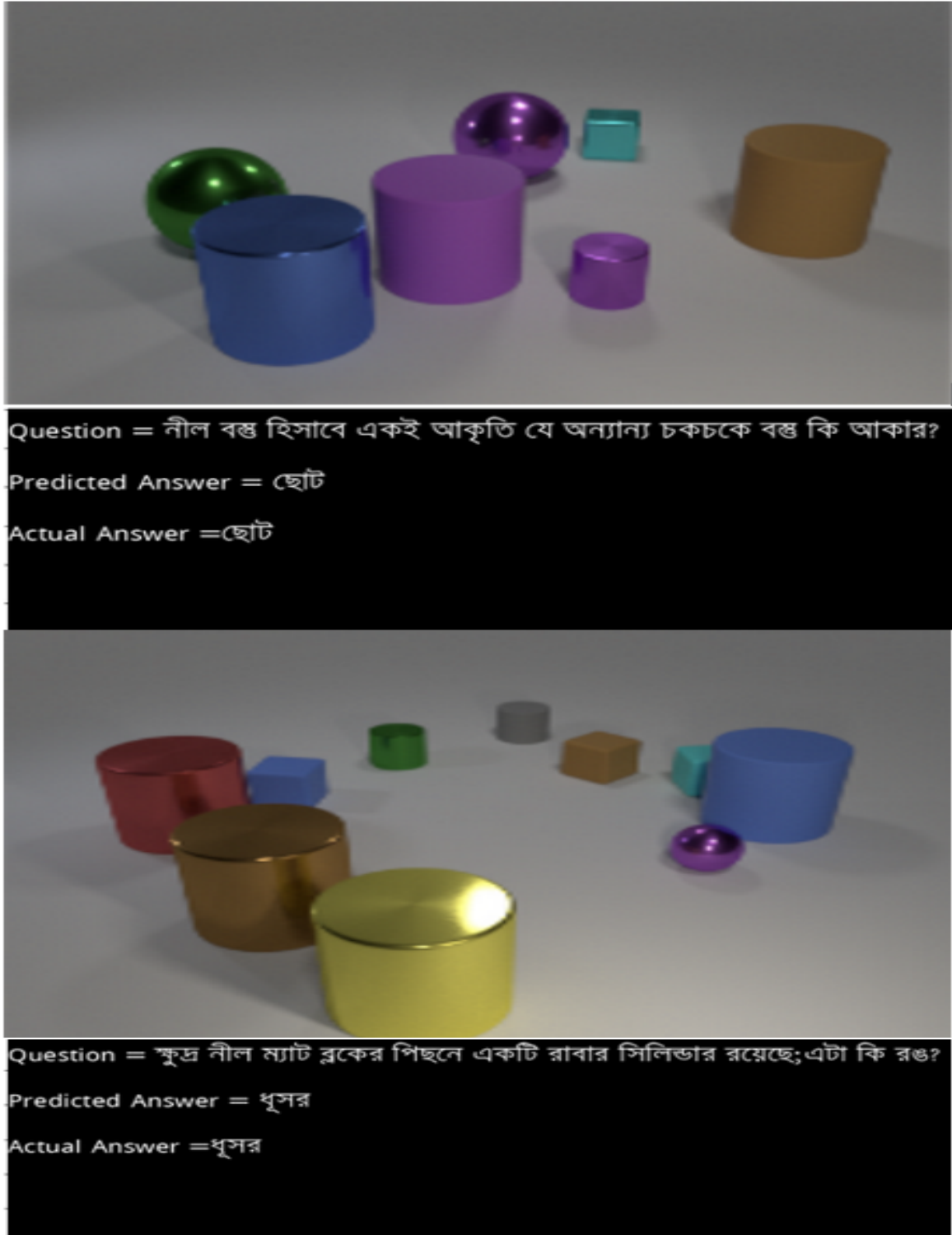


Figure 5.12: Example output of our model trained on Bangla CLEVR dataset

For this dataset we have used 4000 data and run for 20 epochs . Our train accuracy is 75 % and our test accuracy is 63% .We can improve this result with more training dataset and if run the training process for more epochs . For now we have chosen 4000 data and run the model for 20 epochs.The dropout layers have helped the model to get rid of overfitting.

The table below shows the comparison between our datasets and the English datasets.

Dataset	Accuracy(Train)	Accuracy(Test)
VQA v1	75%	71%
Bangla VQA dataset	63%	55%
CLEVR	77%	66%
Bangla CLEVR dataset	75%	63%

Table 5.1: Performace Table

Chapter 6

Limitations And Future Work

Our model had more or less similar outputs in case of running on both the datasets. But in a few cases the accuracy was not up to the mark. Even though it was able to process and analyze the questions with longer length, the outputs could not be made descriptive. Providing answers with multiple words during training gave us errors. It could only process and generate answers containing one word. Due to this we were not able to generate a detailed and more descriptive answer for each question. For example, in case of a question where it was asked what color dress the person is wearing, it was only able to answer with the color's name ex: white. A full sentence is not generated. One other thing is that, some of the questions got translated but did not convey the same meaning as the English question. This led to the translated sentence becoming unclear. We therefore had to exclude those questions from the dataset which led to a loss of data while preparing the dataset. Our Bangla VQA dataset has less accuracy than our Bangla CLEVR dataset because it has less training data. If we increase the amount of training data in Bangla VQA dataset, we can improve the accuracy. We wish to alleviate all of these complications in the future. For our future we wish to work on implementing a fully functional visual question answering system that could communicate in Bangla just as well as English. So, various VQA based smart systems such as smart glasses for the visually impaired, navigation system in cars, fully automated virtual assistants etc. could work on our language as per our need.

Chapter 7

Conclusion

The entirety of our research focuses on enabling Visual Question Answering in Bangla. We started off our work with finding a proper dataset. But apparently no such dataset was found that properly facilitated our intentions. So, we improvised. Using the existing VQA datasets, we tried to formulate our own Bangla version of VQA dataset. For this purpose, we specifically used the Clevr dataset and the VQA v1 dataset. Accumulating and preprocessing the newly formulated datasets were a bit complicated so we had to modify the model to our necessity. Finally, we were able to propose two such Bengali datasets that could be used for our work. The Bangla clever dataset contains 12k image data and corresponding questions and answers while the Bangla VQA dataset contains 5k data. For our Bangla VQA dataset we got 63% train accuracy and 55% test accuracy and in Bangla CLEVR dataset we got a train accuracy of 77% and test accuracy of 66%. Through our work we tried to lay a foundation of visual question answering in Bangla.. Our desire was to work for the people of Bangladesh and support the 7th most popular language in the world. The reality suggests that most of the people in this country prefers to communicate in Bangla. Why should language hold us back from utilizing the best gifts modern technology has to offer? So, we tried to remove that unnecessary barrier. We pray and hope that our work can inspire thousands of minds to take it even further and achieve something wonderful for the greater good of the people of Bangladesh and for the greater good of mankind in general.

Bibliography

- [1] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, “Microsoft coco: Common objects in context,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.
- [2] S. Antol, A. Agrawal, J. Lu, *et al.*, “Vqa: Visual question answering,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2425–2433. DOI: 10.1109/ICCV.2015.279.
- [3] A. Agrawal, D. Batra, and D. Parikh, “Analyzing the behavior of visual question answering models,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Jun. 2016. DOI: 10.18653/v1/D16-1203.
- [4] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, “Making the v in vqa matter: Elevating the role of image understanding in visual question answering,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6325–6334. DOI: 10.1109/CVPR.2017.670.
- [5] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1988–1997. DOI: 10.1109/CVPR.2017.215.
- [6] K. Kafle and C. Kanan, “An analysis of visual question answering algorithms,” Oct. 2017, pp. 1983–1991. DOI: 10.1109/ICCV.2017.217.
- [7] R. K. and Yuke Zhu, O. Groth, J. Johnson, *et al.*, “Visual genome: Connecting language and vision using crowdsourced dense image annotations,” *International Journal of Computer Vision*, vol. 123, pp. 32–73, 2017. DOI: 10.1007/s11263-016-0981-7.
- [8] P. Anderson, X. He, C. Buehler, *et al.*, “Bottom-up and top-down attention for image captioning and visual question answering,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 6077–6086. DOI: 10.1109/CVPR.2018.00636.
- [9] Y. Jiang, V. Natarajan, X. Chen, M. Rohrbach, D. Batra, and D. Parikh, *Pythia v0.1: The winning entry to the vqa challenge 2018*, 2018. arXiv: 1807.09956 [cs.CV].

- [10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” Jun. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.
- [11] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum, “Neural-symbolic vqa: Disentangling reasoning from vision and language understanding,” Oct. 2018. [Online]. Available: <http://nsvqa.csail.mit.edu>.
- [12] R. Cadene, C. Dancette, H. Ben younes, M. Cord, and D. Parikh, “Rubi: Reducing unimodal biases for visual question answering,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/51d92be1c60d1db1d2e5e7a07da55b26-Paper.pdf>.
- [13] M. Kaur and A. Mohta, “A review of deep learning with recurrent neural network,” in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2019, pp. 460–465. DOI: 10.1109/ICSSIT46314.2019.8987837.
- [14] K. Marino, M. Rastegari, A. Farhadi, and R. Mottaghi, “Ok-vqa: A visual question answering benchmark requiring external knowledge,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 3190–3199. DOI: 10.1109/CVPR.2019.00331.
- [15] M. Rahman, N. Mohammed, N. Mansoor, and S. Momen, “Chittron: An automatic bangla image captioning system,” *Procedia Computer Science*, vol. 154, pp. 636–642, 2019, Proceedings of the 9th International Conference of Information and Communication Technology [ICICT-2019] Nanning, Guangxi, China January 11-13, 2019, ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2019.06.100>.
- [16] F. Saxen, P. Werner, S. Handrich, E. Othman, L. Dinges, and A. Al-Hamadi, “Face attribute detection with mobilenetv2 and nasnet-mobile,” Sep. 2019, pp. 176–180. DOI: 10.1109/ISPA.2019.8868585.
- [17] S. Shah, A. Mishra, N. Yadati, and P. P. Talukdar, “Kvqa: Knowledge-aware visual question answering,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8876–8884, Jul. 2019. DOI: 10.1609/aaai.v33i01.33018876.
- [18] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, “Deep modular co-attention networks for visual question answering,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6274–6283. DOI: 10.1109/CVPR.2019.00644.
- [19] M. Basirat and P. M. Roth, “L*relu: Piece-wise linear activation functions for deep fine-grained visual categorization,” in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1207–1216. DOI: 10.1109/WACV45572.2020.9093485.

- [20] H. Jiang, I. Misra, M. Rohrbach, E. Learned-Miller, and X. Chen, “In defense of grid features for visual question answering,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 264–10 273. DOI: 10.1109/CVPR42600.2020.01028.
- [21] K. Jiang and X. Lu, “Natural language processing and its applications in machine translation: A diachronic review,” in *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, 2020, pp. 210–214. DOI: 10.1109/IICSPI51290.2020.9332458.
- [22] V. K. and S. K., “Towards activation function search for long short-term model network: A differential evolution based approach,” *Journal of King Saud University - Computer and Information Sciences*, 2020, ISSN: 1319-1578. DOI: <https://doi.org/10.1016/j.jksuci.2020.04.015>.
- [23] C. Kolling, J. Wehrmann, and R. C. Barros, “Component analysis for visual question answering architectures,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9206679.
- [24] N. S. Malinović, B. B. Predić, and M. Roganović, “Multilayer long short-term memory (lstm) neural networks in time series analysis,” in *2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, 2020, pp. 11–14. DOI: 10.1109/ICEST49890.2020.9232710.
- [25] H. Nguyen, “Fast object detection framework based on mobilenetv2 architecture and enhanced feature pyramid,” *Theoretical and Applied Information Technology*, vol. 98, no. 05, 2020, ISSN: : 1992-8645.
- [26] Z. Huang, H. Zhu, Y. Sun, D. Choi, C. Tan, and J.-H. Lim, “A diagnostic study of visual question answering with analogical reasoning,” in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 2463–2467. DOI: 10.1109/ICIP42928.2021.9506539.
- [27] R. Jie, J. Gao, A. Vasnev, and M.-n. Tran, “Regularized flexible activation function combination for deep neural networks,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 2001–2008. DOI: 10.1109/ICPR48806.2021.9412370.
- [28] W. Li and K. Liu, “Confidence-aware object detection based on mobilenetv2 for autonomous driving,” *Sensors*, vol. 21, no. 7, 2021, ISSN: 1424-8220. DOI: 10.3390/s21072380.
- [29] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021. DOI: 10.1109/TNNLS.2021.3084827.

- [30] K. Marino, X. Chen, D. Parikh, A. Gupta, and M. Rohrbach, “Krisp: Integrating implicit and symbolic knowledge for open-domain knowledge-based vqa,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 14 106–14 116. DOI: 10.1109/CVPR46437.2021.01389.
- [31] T. T. Mayeesha, A. M. Sarwar, and R. M. Rahman, “Deep learning based question answering system in bengali,” *Journal of Information and Telecommunication*, vol. 5, no. 2, pp. 145–178, 2021. DOI: 10.1080/24751839.2020.1833136.
- [32] T. P. Nagarhalli, V. Vaze, and N. K. Rana, “Impact of machine learning in natural language processing: A review,” in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021, pp. 1529–1534. DOI: 10.1109/ICICV50876.2021.9388380.
- [33] M. A. H. Palash, M. D. A. A. Nasim, S. Saha, F. Afrin, R. Mallik, and S. Samiappan, “Bangla image caption generation through cnn-transformer based encoder-decoder network,” *CoRR*, vol. abs/2110.12442, 2021.