Spring 2018

Visual Question Answering

Pankti Kansara
San Jose State University

Recommended Citation

Kansara, Pankti, "Visual Question Answering" (2018). *Master's Projects*. 640.
DOI: https://doi.org/10.31979/etd.8zhj-5qdc
https://scholarworks.sjsu.edu/etd_projects/640

Visual Question Answering

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Pankti Kansara

May 2018

The Designated Project Committee Approves the Project Titled

Visual Question Answering

by

Pankti Kansara

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2018

ABSTRACT

Visual Question Answering

by Pankti Kansara

There has been immense progress in the fields of computer vision, object detection and natural language processing (NLP) in recent years. Artificial Intelligence (AI) systems, such as question answering models, use NLP to provide a "comprehensive" capability to the machine. Such a machine can answer natural language queries about any portion of an unstructured text. An extension of this system is to combine NLP with computer vision to accomplish the task of Visual Question Answering (VQA), which is to build a system that can answer natural language queries about images. A number of systems have been proposed for VQA that use deep-learning architectures and learning algorithms.

This project introduces a VQA system that gains deep understanding from images by using a deep convolutional neural network (CNN) that extracts image features. More specifically, the feature embeddings from the output layer of the VGG19 [1] model are used for this purpose. Our system achieves complex reasoning capabilities and understanding of natural language, so that it can correctly interpret the query and return an appropriate answer. The InferSent [2] model is used for obtaining sentence level embeddings to extract features from the question. Different architectures are proposed to merge the image and language models. Our system achieves results comparable to the baseline systems on the VQA dataset.

ACKNOWLEDGMENTS

I would like to thank my project advisor, Dr. Katerina Potika for consistently keeping me motivated and guiding me in the right direction whenever I needed. I would also like to thank both my committee members: Dr. Chris Pollett and Dr. Natalia Khuri, for their valuable inputs and feedback about my work. I am also grateful to all my peers, classmates and friends who helped me by providing suggestions and feedback about this project. Last but not the least, I would like to express gratitude to my parents and my sister for their constant encouragement, motivation and inspiration which has always helped me overcome hurdles and accomplish my goals. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Artificial Intelligence (AI) technology has been widely used to build elements of larger systems since the late 1990s, with applications in varied fields. Progress in deep learning has led to an exponential growth in the number of AI applications. Deep learning can be thought of as a subset of AI. It is used in a variety of AI tasks such as machine comprehension, machine translation, computer vision, object recognition etc.

Tremendous research has been going on in solving problems which combine different domains of AI. These can be viewed as multi-disciplinary tasks. A simple example of this can be "image captioning". It combines simple techniques of computer vision (CV) to obtain a high-level understanding of what is being depicted in the image with basic natural language understanding (NLU) methods like *n-grams* to generate a simple caption for that image. Although image captioning is a multi-disciplinary AI task, it does not require the system to understand the intrinsic details in the image, nor does it require advanced understanding of natural language to generate the captions.

One of the challenges that requires a very advanced and in-depth understand of natural language is the task of question answering (QA). It is a concept under computer science, machine learning and deep learning which uses natural language processing (NLP), text analysis and information retrieval to build a system that can answer questions asked in natural language. Answering a query about any reading material (e.g.: passage, articles, blogs etc.) requires modeling complex interactions between the context and the query. The working of QA systems is analogous to the way humans comprehend and hence it is known as "machine comprehension." Most of the current state-of-the-art QA systems make use of advanced deep-learning architectures to gain a detailed understanding of the semantics behind the reference text and the query and then use a variety of methods to capture the relation between the two. [3] Sometimes it also becomes important that the model gain understanding about general knowledge or "common sense" so as to be able to answer the query correctly.

Another research area that has gained a lot of interest and excitement in the recent years is visual question answering (VQA). VQA can be thought of as an extension to the concept of machine comprehension. It is also a multi-disciplinary AI task which combines advanced techniques of computer vision and natural language processing to build a system that can answer a query about an image. The model attempts to understand the underlying meaning and semantics behind the image and answer questions based on its "understanding." Unlike image captioning, where basic knowledge of computer vision and NLP is sufficient to build AI models, tasks like VQA require an in-depth knowledge of advanced techniques in both these domains. Until recently, most of the AI systems built failed to match humans in high-level vision tasks due to the lack of capacity for deeper reasoning. But with the ongoing research in this field it has become possible to now attempt and build a system that is supposed to succeed at tasks like VQA. Such a system typically needs more detailed understanding of the image and complex reasoning capabilities. A more advanced version of this system would also possess factual knowledge to be able to comprehend and answer questions whose answers are not directly depicted in the image.

VQA is a relatively new and exciting concept. Most of the extensive research going on in this field, makes use different deep learning architectures and learning algorithms in the domains of computer vision, object recognition and natural language processing to accomplish the goal. Most of the existing VQA datasets and the models built on them generally focus on questions which are answerable by direct analysis of the input image. If we look at the general problem, we see that tasks like VQA pose a very diverse set of problems and challenges to overcome under the umbrella of vision, language as well as semantic representation of knowledge. With the existing research in these individual areas, it is possible to build different components of a VQA system and then integrate them in order to solve the task at hand.

This project focuses on building such a VQA system. In this report, we formulate the problem followed by a discussion about existing models, related research work and open datasets available for VQA. We then provide an overview of our proposed systems, experiments done and results obtained.

CHAPTER 2

Problem Definition

Building an AI system that can answer natural language queries about a given image is one of the most trending research areas in the recent years. Consider the image shown below:

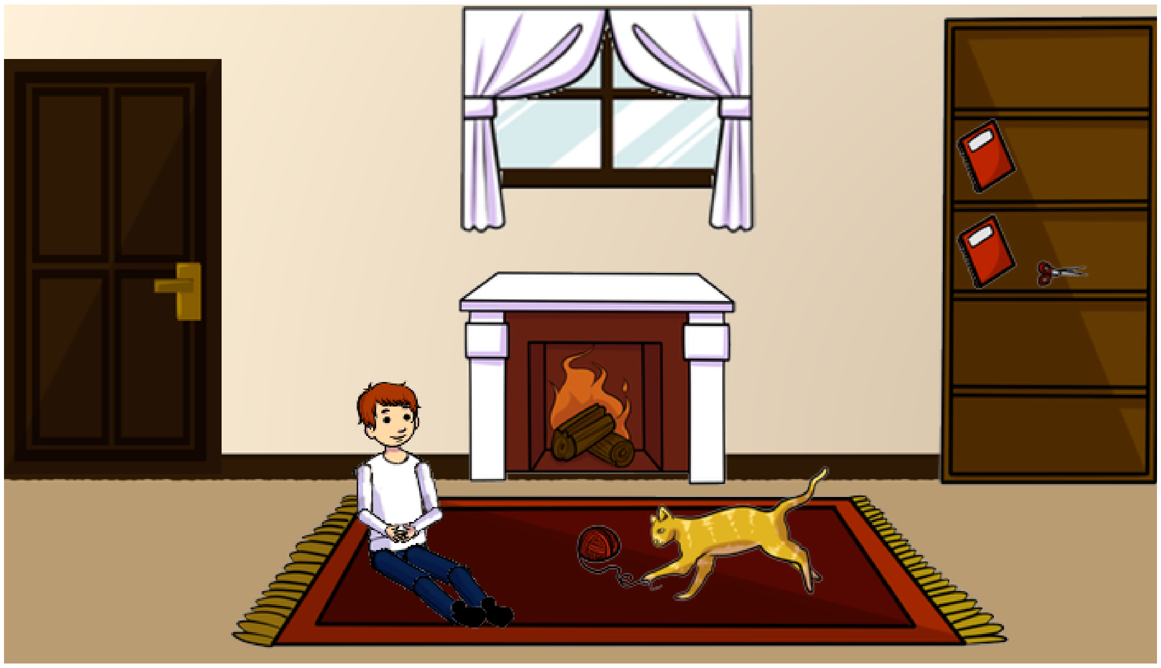


Figure 1: Sample Image

Ideally, the system should be able to answer questions about this image. Some examples of questions that can be asked are:

- How many books are on the shelf?
- What is the cat playing with?
- What color are the books?

Most of these questions can be answered by humans without any major challenges or difficulties. For a question like "How many books are on the shelf?" it is not a difficult task for humans to actually count the books and return the answer "2". But for an AI system, it is an extremely challenging task to be able to comprehend the meaning of the question, parse and understand the semantics of the image, identify the connection between the query and the image and then attempt to answer the question.

However, with emerging research in the field of deep learning, both in the domains of computer vision and natural language processing, it is now possible to build such a system which has the understanding and capability of answering these questions correctly and with great results.

Overall, the general problem of visual question answering (VQA) can be defined as building a system/algorithm that takes (ideally) any image and a question asked in natural language about that image and provides a natural language answer to that question with reference to the image as the output. A high-level view of such a system is shown in Figure 2.

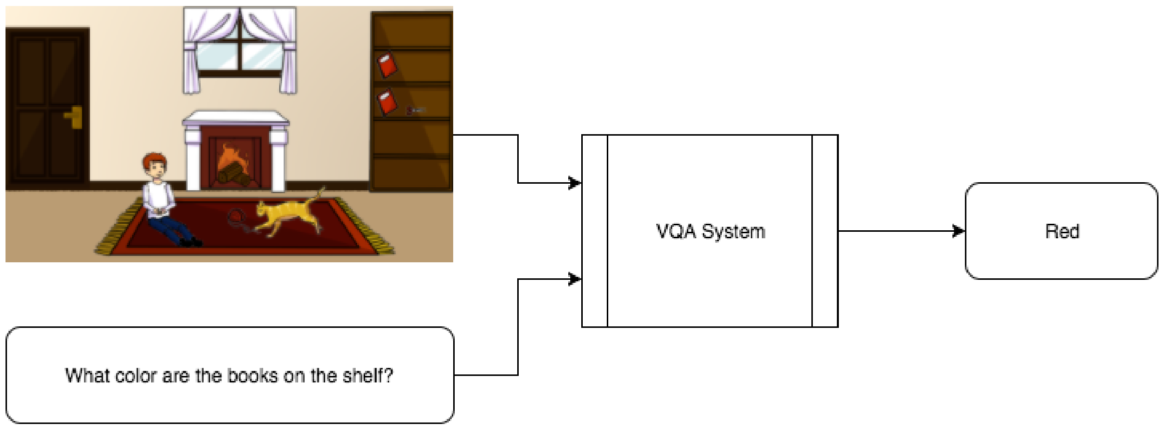


Figure 2: High-level view of a VQA system

CHAPTER 3

Terminology

3.1 Neural Network Architectures

An artificial neural network (ANN) is a network that is composed of many artificial neurons that are linked together according to a specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.

An ANN architecture consists of one input layer, one output layer and a number of hidden layers. See Figure 3.

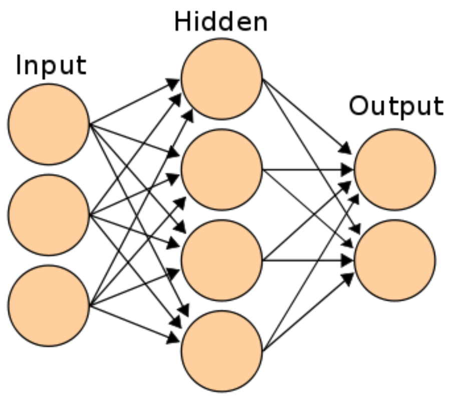


Figure 3: Example of simple neural network with one hidden layer

- **Recurrent Neural Network:**

  Reccurent Neural Networks (RNN) are those which perform the same task for every element of the input sequence. The output of an RNN depends on the current element as well as the previous computations. RNNs have a "memory" to capture all the information it has seen previously. A simple RNN unfolding in time is shown in Figure 4.

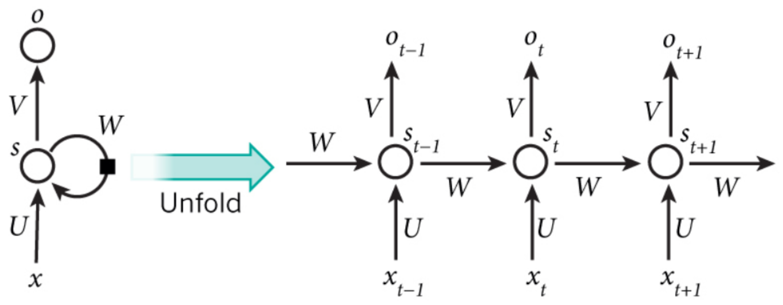


Figure 4: Example of a simple RNN unfolding in time

- **Long short-term memory:**

  Long short-term memory (LSTM) block or network is a simple recurrent neural network which can be used as a building component or block (of hidden layers) for an eventually bigger recurrent neural network. The LSTM block is itself a recurrent network because it contains recurrent connections similar to connections in a conventional recurrent neural network.

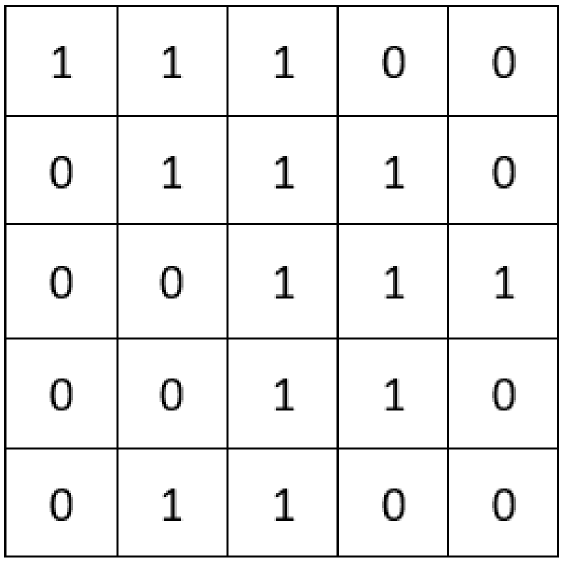
  An LSTM block is composed of four main components: a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum.

- **Convolution:**

  Convolution is a mathematical operation which is performed on two functions taken as input. It typically gives an integral of the point-wise product of the two input functions. Intuitively, it can be thought of as a modified version of one of the original functions with respect to the other function.

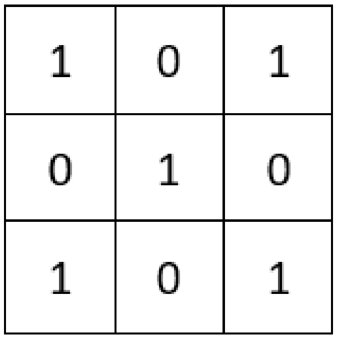
  Consider a 5x5 (Figure 5) matrix with entries as 0s or 1s (black and image

image). Another matrix with dimensions 3x3 (Figure 6), also consisting of 0s and 1s, is used as a filter. These two matrices can be thought of as the two inputs to the convolution function. After performing element-wise multiplication by sliding the filter over the original image, we get the convolved feature which is shown in Figure 7.



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Figure 5: Original image

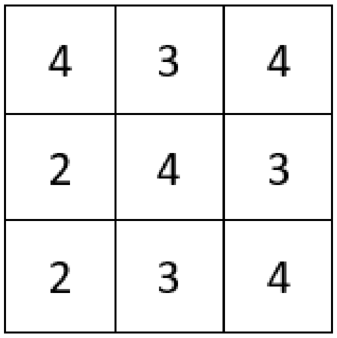


1	0	1
0	1	0
1	0	1

Figure 6: Convolution filter

- **Convolutional Neural Networks:**

Convolutional Neural Networks (CNNs) are nothing but multiple layers of con-



4	3	4
2	4	3
2	3	4

Figure 7: Convolved image

volutions stacked on one another with non-linear activation functions like *tanh* or *ReLU*. In a traditional fully-connected neural network, each node in the current layer is connected to every node in the next layer. However in CNNs convolutional filters are used to slide over the nodes in the input layer and then compute the output. In contrast to simple multi-layer perceptrons, the concept of sliding convolutions over nodes in the input layer results in regions of the input layer being connected to each node in the output layer. Each layer in the network uses different convolutional filters to compute the output. When the network sees the training data, it automatically learns which filters are to be used. See Figure 8 for a simple example of CNN with pooling.

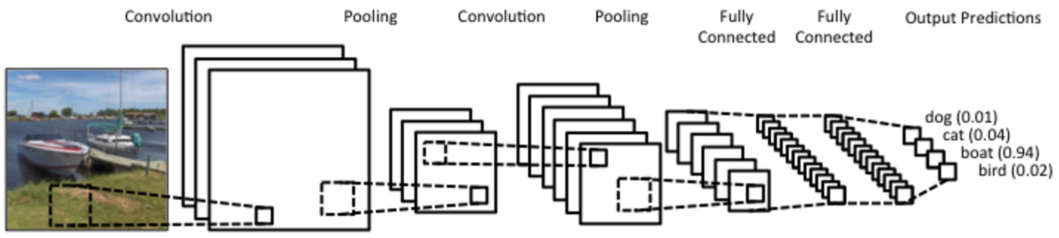


Figure 8: Example of CNN

3.2 Activation Functions

In neural networks, the role of an activation function is to determine the output of a particular node given an input set of nodes from the previous layer. There are many different commonly used activation functions, both linear and non-linear. Some of them relevant to this project are listed below.

- **Hyperbolic tangent or *tanh*:**

  The hyperbolic tangential function is a non-linear activation function given by:

  $$f(x) = tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

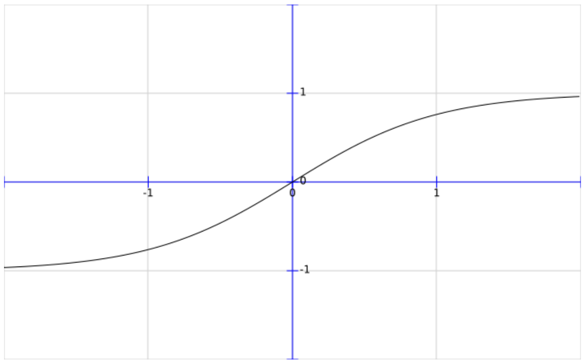
  

  Figure 9: *tanh* activation function

  tanh is a very commonly used activation function. The reason for its popularity is that it has a steep gradient, which makes it a better choice for back-propagation in the network. The first order derivative of *tanh* is given by:

  $$f'(x) = 1 - f(x)^2$$

- **Rectified Linear Unit or *ReLU*:**

  This activation function is defined as the just the positive part of the input. Mathematically, it is represented as:

  $$f(x) = 0 \quad for \quad x < 0$$

$$f(x) = x \quad for \quad x \geq 0$$

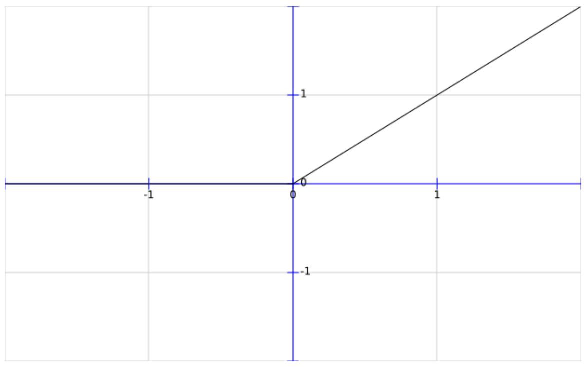


Figure 10: $ReLU$ activation function

$ReLU$ is also the most widely used activation function for deep networks, since it has been proven that its usage has resulted in improved performance of deep learning models.

- **Softmax:**

  The softmax activation function is typically used in the final output layer of the network, generally built for multi-class classification. The output of the softmax function is a probability distribution over the number of target variables and the class with the highest probability value in this distribution is predicted as the final output. Mathematically, it transforms any distribution over n classes into a probability distribution in the range of (0, 1) over the same n classes.

3.3 Feature Engineering

In systems like VQA the inputs are provided in the form of raw text and images. Typically images are represented by matrices depicting pixel values at each location. Raw inputs in these forms cannot be used as features to train AI models. Hence it becomes important to use "domain knowledge" and transform these input into a set of features that are representative of the original input. This process is called feature extraction or feature engineering. Feature engineering is a very essential step in the process of building any AI model. The number of features and their quality highly influence the training and learning process, and therefore it has a great impact on the overall performance of the system.

3.4 Evaluation Metrics

For any machine learning model built and trained, it is extremely important to evaluate its performance. Different types of models need to be evaluated in different ways and there are various metrics defined that help us to evaluate how good a model is. Generally, for classification problems the most commonly used metrics are accuracy, precision, recall and F1 score. The definitions for each of them are listed below.

- **Accuracy:**

  For any classification model, accuracy is defined as the number of times that the model predicts the correct class for inputs that it did not see during training out of the total number of inputs provided.

$$Accuracy = \frac{(Number\ of\ true\ predictions)}{(Total\ number\ of\ input\ samples)}$$

- **Precision:**

  Precision is the fraction of correctly predicted positives among all the positive predictions. Extending this for multi-class classification, precision for each class is the fraction of correct predictions for that particular class out of the total predictions that predicted that class.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:**

  Recall is the fraction of correctly predicted positives among all the actual positives. Extending this for multi-class classification, recall for each class is the fraction of correct predictions for that particular class out of the total testing instances provided for that class.

$$Recall = \frac{TP}{TP + FN}$$

- **F1 Score:**

  F1 score is used to measure the performance of the model taking into consideration both the precision and recall. Mathematically, it is computed as:

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2.\frac{Precision.Recall}{Precision + Recall}$$

CHAPTER 4

Existing Techniques and Related Work

4.1 Datasets

Machine Learning (ML) is a subset of AI in which data is extremely important to train the model and obtain good results. Deep learning algorithms, in particular, have a large number of parameters and hyper-parameters that require tuning. This tuning happens during the training process, where the model is still in the initial "learning" phase. The more data it gets trained on, the more generalized the model can be. Good quality data is therefore a very important requirement to implement any machine learning algorithm. Most VQA models also use deep learning. This implies that a lot of good quality data is required to train the models. Multiple datasets for VQA have been created since 2014. Most of these datasets are open for academic research purposes. A few of these datasets are:

- **VQA:**

  This is the most widely known and used dataset for visual question answering [4]. It was created in 2015 as a part of a VQA challenge. The dataset is split into two different parts:

  – Real-world images

  – Abstract clip-art scenes

  The use of clip-art images eliminates the need to pre-process and clean noisy images. They can be used directly to perform complex reasoning. Each image in the dataset also has a list of question-answer pairs associated with it. These were collected by crowdsourcing. In all, the VQA dataset has more than 250,000 images and around 800,000 questions.

- **Visual7W:**

  The Visual7W dataset [5] was released in 2016. This dataset contains images obtained from the MS-COCO dataset. The questions dataset was created by asking 7 "w" questions: what, where, when, why, who, how, which. The questions were multiple-choice questions and each question had 4 answer options. Every object mentioned in the question-answer pairs was marked by bounding boxes in the images. The question-answer pairs and object-level groundings were collected by crowd-sourcing using Amazon Mechanical Turks (AMT). This dataset contains approximately 47,000 images and more than 320,000 associated questions.

- **DAQUAR:**

  The DAtaset for QUestion Answering on Real-world images (DAQUAR) [6] was also released in 2015. It was the first dataset ever released (even before the original VQA dataset) for this task. The authors collected the images for this dataset from the NYU-Depth V2 dataset. Every image in the dataset is labelled based on the objects present in the image. The objects can belong to one of the possible 894 object classes. The dataset is comparatively small. It only contains a total of around 1500 images. The question-answer pairs were auto-generated using 9 pre-defined templates for the questions. The answers to these questions were obtained directly from the NYU-Depth V2 dataset.

4.2 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is extremely vital since it is the features of the data that mainly influence the training and eventually, the predictions. The quality of input features has a significant impact on the quality of the model.

- **Feature Engineering for Images:**

  To obtain feature vectors for images, there are many open-source pre-trained models available. In 2014, Google proposed one such model called GoogLeNet [7]. Since then, it has been widely used to encode images and obtain high-quality feature vectors to be fed in any deep learning model. It is a convolutional neural network (CNN) with 22 layers and has been trained on the ImageNet dataset. Another similar model is VGGNet [1], which is also a CNN. It is 16-19 layers deep and uses very small convolution filters (size of 3x3).

- **Feature Engineering for Text:**

  There are pre-trained models available to obtain feature vectors from raw text data. These feature vectors are called word vectors. There are pre-trained models available publicly to obtain corresponding word vectors from input text data. The most commonly used word vectors are GloVe (Global Vectors for word representation) [8] and Word2Vec [9].

4.3 Memory

- **Long short-term memory (LSTM):**

  LSTMs are a special type of Recurrent Neural Networks (RNNs) which can learn and identify long-term dependencies. LSTMs use a "gating" mechanism to define how much of the current input or output we actually want to let through to the next layer. For VQA, LSTMs are generally used to obtain word embeddings from the text question. The dimensionality of the word embeddings depends on the architecture of the LSTM. Both the image feature vectors and word embeddings can also be combined to obtain a single embedding using a deep LSTM.

- **Gated recurrent units (GRU):**

  GRUs are quite similar to LSTMs. The internal mechanism is slightly different. Unlike LSTMs, GRUs do not have internal memory which is different from the hidden state. The use case for GRUs is the same as that of LSTMs. They can be used to obtain question embedding as well as to provide âĂIJrememberingâĂİ capability to the model. In [9] GRUs are used to build a dynamic memory network specifically for visual question answering systems.

- **Attention models:**

  Attention models are used to solve the problem of long-term dependencies more efficiently than simple LSTMs. Simple LSTMs require that we encode the entire text as input to the network. But in models that incorporate attention mechanism, there is no need to encode the entire text as input to the model. Instead, the model pays "attention" to specific parts of the input which are more important with respect to the current context and gives higher weightage to those parts at every stage of computing the output. Many VQA systems have attempted to incorporate attention mechanism in their architecture, and the results have shown significant improvement when compared to simple LSTMs or GRUs. In most of the cases, question-based attention was used to encode image features. This means that more attention was given to those parts of the image which were more relevant to the question being asked.

4.4 Related Work

As mentioned already, the task of VQA has gained immense popularity due to the promising results of deep learning techniques on various problems involving computer vision and natural language processing. The state-of-the-art performance shown by a lot of these deep-learning models has led the researchers to take this direction while proposing a solution for VQA. Although there have been models built using traditional machine learning approaches, most of the research done in this field is inclined towards deep learning. Some of the models built also use graphical representations to depict the elements of the image and the question and then perform graph computations to identify the relation between the two to eventually answer the question.

Kafle et al. [10] propose a VQA system using a classical algorithm that uses Bayesian probability to predict the probability of the answer type given the input question and image. This answer type is then used to generate the actual answer. The model uses Bayes' probability rule to compute the probability. They use skip-thought vectors [11] to compute feature embeddings for questions.

In 2016, Teney et al. [12] proposed a VQA system that used graphs to represent the images and questions. A deep neural network was then built over this graph to identify the connection between the image and question from the graphical representation. The input image was encoded as a graph where each node in the graph represented the objects in the image and the edges between these nodes represented the relative positions of those objects in the image. The question tokens were also encoded in a graph that represented the sequential connections and syntactic dependencies between the words in the question.

Most of the other deep learning models proposed for VQA typically make use of the CNN/LSTM approach. CNNs are used to transform input images into their corresponding vector representations and extract meaningful insights from the images which can be used to train the model. One of the most popular models used for feature extraction from images is the VGGNet system proposed by Simonyan et al. [1] Similarly pre-trained word embeddings are obtained to transform the question into its vector form. The most widely used word embeddings are Word2Vec [9] and GloVe [8]. (Refer Section 4.2) These embeddings are then merged using different techniques to train the model on the combination of the image and question and then provide an answer.

In more recent times, another concept that has gained increased popularity within the field of deep learning is attention mechanism (Refer Section 4.3). Networks using attention mechanism are built for solving problems like machine translation, text question answering, generating image captions etc. For the task of VQA, attention mechanism is utilized to emphasize or focus more on relevant parts of the image based on what the question is asking and then provide an answer. In 2016, Shih et al. proposed an attention-based model for VQA called "Where to Look" [13]. VGGNet was used to obtain image embeddings and the embedding for question was computed by finding the average of word vectors for each word in the question. These two embeddings are then fused together and an attention vector is calculated to identify which region in the image are more relevant to the question so that the model can pay attention only on those regions. Some of the other attention models developed for VQA are "Ask, Attend and Answer" by Xu et al. [14], "Stacked Attention Networks" by Yang et al., [15], "Hierarchical co-attention for VQA" by Lu et al. [16]

CHAPTER 5

Proposed Method for Visual Question Answering

5.1 Data Loader

The first step of building any machine learning model is to clean and preprocess the data to convert it into the required format. For this project, the VQA dataset is used for training and evaluation. The structure of the dataset is shown in Figure 11.

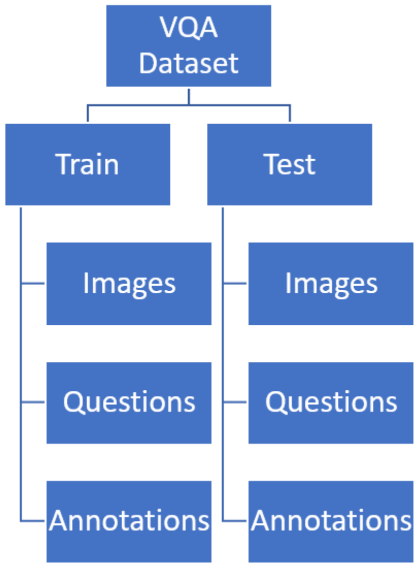


Figure 11: Structure of VQA Dataset

The VQA v1.0 training dataset which is a dataset of Abstract images contains 20k images in the "Images" folder. The "Questions" folder has a JSON file which contains 60k questions mapped to the images. The VQA v2.0 training dataset which is a dataset of Real images contains 82k images in the "Images" folder, with approximately 440k questions mapped to these images. The structure of the JSON file is as follows:

```
{
"info" : info,
"task_type" : str,
"data_type": str,
"data_subtype": str,
"questions" : [question],
"license" : license
}

question
{
"question_id" : int,
"image_id" : int,
"question" : str
}
```

The "Annotations" folder has a JSON file which provides answers to all the questions based on the image that they are mapped to. The structure of this file is as follows:

```
{
"info" : info,
"data_type": str,
"data_subtype": str,
"annotations" : [annotation],
"license" : license
}

annotation{
```

```
"question_id" : int,
"image_id" : int,
"question_type" : str,
"answer_type" : str,
"answers" : [answer],
"multiple_choice_answer" : str
}

answer{
"answer_id" : int,
"answer" : str,
"answer_confidence": str
}
```

For this project, we only use the questions with multiple-choice answers and formulate the task as a classification problem. To do this, we first identify the top k multiple choice answers that answer the most number of training questions. These answers are treated as classes and the final task is to map any new input question about an input image into one of these classes.

All other questions whose answers do not belong to one of these k answers are discarded and not used in the training process. A combined dataset is now created that maps the selected questions to its corresponding images with the correct answer. This combined dataset is also in JSON format and a snippet of this file is shown below.

```
[{"question": "Is the bench a toy?",
"image_file": "Dataset/Images/scene_img_abstract_v002_train2015/
abstract_v002_train2015_000000000000.png",
"image_id": 0,
"answer": "no",
"question_id": 2}, ... ]
```

The questions and images from this dataset are now used to perform feature engineering and obtain vector representations for each image and question.

5.2 Feature Engineering

5.2.1 Feature Engineering for Images

To obtain feature vectors for images, the model that is used is VGGNet [1], which is a CNN. It is 16-19 layers deep and uses very small convolution filters (size of 3x3). The model is trained on the ImageNet dataset [17] and it performs extremely well on classification task of images.

Architecture and Configuration: The model takes an RGB image as input, with dimensions 224x224 pixels. This image matrix is then passed through convolutional layers. In each of these layers, the output for the next layer is computed by using very small convolutional filters of size 3x3. This is the smallest filter size that is required if we need to capture the surrounding features of a pixel from all directions. The filters are moved with a stride of 1.

Some of the convolutional layers are also followed by max-pooling for which a window of size 2x2 is used. This window is moved with a stride of 2.

The model used in this project to encode images is VGG19. It contains 19 weight layers, 16 of which are convolutional layers followed by 3 fully-connected layers. The number of nodes in the first 2 fully-connected layers is 4096 and the final layer is a

Table 1: VGG19 Configuration

Number of convolutional layers	**Number of channels in each layer**
Input RGB image with dimensions 224x224 pixels	
2	64
Maxpooling	
2	128
Maxpooling	
4	256
Maxpooling	
4	512
Maxpooling	
2 (Fully-connected)	4096
1 (Fully-connected)	1000
Softmax activation	

softmax distribution over 1000 image classes and hence it has 1000 nodes. This means that the input image of dimensions 224x224 pixels is now converted to a vector of 4096 dimensions, which we obtain from the second to last layer of the model. Each of the 19 hidden layers in the network use $ReLU$ as their activation function.

Table 1 shows the configuration of the entire network.

Training: Each image in the training dataset was first rescaled to match the input dimensions of the model (224x224). The model is then trained using mini-batch gradient descent approach with back-propagation to optimize the objective of logistic regression. Regularization was done using L2 technique.

Encoding Images: Each image from the VQA training and testing dataset was encoded into a 1000 dimensional vector. These 1000 dimensions are the image features extracted by the VGG19 model. Keras applications provides this VGG19 model which is pre-trained on the ImageNet dataset.

The input images are first rescaled to match the input dimensions of VGG19. Each image is then fed in to the pre-trained model to obtain its encoding. The transformation is done for all the required images in the dataset. Since the dataset was too large to fit in the memory, it was done in batches of size 100. (Refer code below)

```
def encode_image(image):
    model = VGG19(weights='imagenet', include_top=True, pooling='avg')
    img_em = np.zeros((1, 1000))
    im = cv2.resize(cv2.imread(image), (224, 224))
    im = np.expand_dims(im, axis=0)
    img_em[0,:] = model.predict(im)[0]
    embedding = img_em
    return embedding
```

5.2.2 Feature Engineering for Text

There are pre-trained models available to obtain feature vectors from raw text data. Most of these models provide word-level embeddings. The most commonly used word vectors are GloVe (Global Vectors for word representation) [8] and Word2Vec [9]. However, it is a difficult task to understand the dependencies between the words and interpret the semantics behind each sentence. Therefore, it becomes necessary to design a system that captures these dependencies and semantics and provide sentence-level embeddings which can further be used in our question-answering system. The simplest way to obtain sentence-level embeddings, provided we already have vectors for each word, is to average all the vectors representing each word in that sentence. GloVe provides 300-dimensional vectors for each word and it is trained on the Wikipedia corpus. If these vectors are used to represent words in our text, then our sentence-level embeddings averaged from word-vectors will also have 300 dimensions. This is a very naive approach to get sentence representations. Averaging of word-vectors to obtain a single vector for each sentence generally results in loss of information. We only obtain a high-level idea of what is being said in the sentence, but we might miss important intrinsic details. For our task of visual question answering, it is very important that we capture as much information as possible from the question, to be able to answer the question correctly. For this reason, we need to use much more complex models which capture details from the text such as word dependencies, relative importance of the words in the sentence etc. Kiros et al. [11] published their research on Skip-thought vectors which used unsupervised learning to obtain sentence encoders. Alexis et al. [2] from Facebook research studied techniques to obtain universal representation of sentences. In our project, their pre-trained model InferSent is used to obtain vector representations for natural-language questions asked about images.

Model Architecture: The model uses a bidirectional LSTM (BiLSTM) with either max-pooling or mean-pooling. For any given sentence consisting of T words, each word in that sentence is first represented by its GloVe vector.

A forward LSTM first computes a set of T vectors:

$$\overrightarrow{h}_t = \overrightarrow{LSTM}_t(w_1, ..., w_T)$$

Then a backward LSTM computes a set of T vectors:

$$\overleftarrow{h}_t = \overleftarrow{LSTM}_t(w_1, ..., w_T)$$

The final set of vectors computed by the BiLSTM is thus:

$$h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$$

Each sentence now has a set of T vectors. To deal with varying number of words in sentences, these vectors are now combined either by max-pooling or mean-pooling. The final result is a 4096-dimensional vector for each sentence. (See Figure 2)

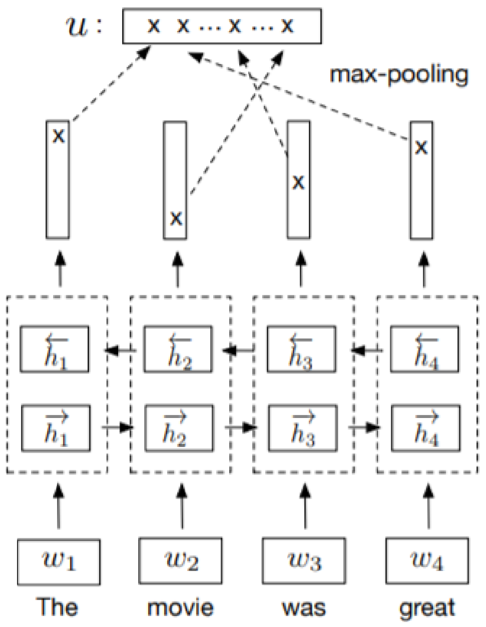


Figure 12: BiLSTM encoder with max-pooling

Training: The model is trained using the Stanford Natural Language Inference (SNLI) dataset which is a manually labeled dataset. Stochastic gradient descent algorithm is used to train the model with a learning rate of 0.1 and mini-batch size of 64.

Encoding Questions: Each question from the VQA training and test dataset was encoded into a 4096 dimensional vector. These 4096 dimensions are the sentence features extracted by the InferSent model. Each question is fed in to the pre-trained model to obtain its corresponding sentence embedding. (Refer code below)

```
def encode_questions(train_questions):
    infersent = torch.load('infersent.allnli.pickle', map_location=lambda
        storage, loc: storage)
    glove_path = 'GloVe/glove.840B.300d.txt'
    infersent.set_glove_path(glove_path)
    infersent.build_vocab(train_questions, tokenize=True)
    embeddings = infersent.encode(train_questions, tokenize=True)
    return embeddings
```

5.3 VQA Model Architectures

The encodings of images and questions are both stored in separate files to be used as inputs to the classifiers. The simplest way of combining them is to concatenate the vectors of each question with its corresponding image, resulting in a 5096 dimensional vector (4096 dimensions from question embedding + 1000 dimensions from image embedding). These 5096 dimensions are now used as features for our classification task. Experiments were performed on different model architectures, both traditional machine learning models as well as deep-learning models.

5.3.1 Traditional Support Vector Machine

SVMs are one of the most commonly used classification algorithms on supervised data. They have a lot of advantages over other classification algorithms. In this project, experimenting with an SVM was suitable because one of the most important advantages they have is that they perform effectively even if we have a very high dimensional feature set. Figure 13 shows an overview of this architecture. The use of SVM is also efficient in terms of memory because they do not use all the training points in the decision function. Only those training points which are close to the support vectors are considered, the rest are discarded.

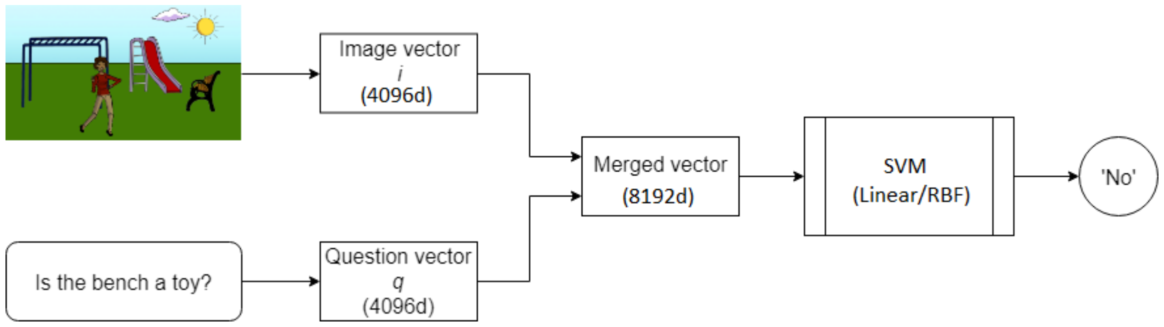


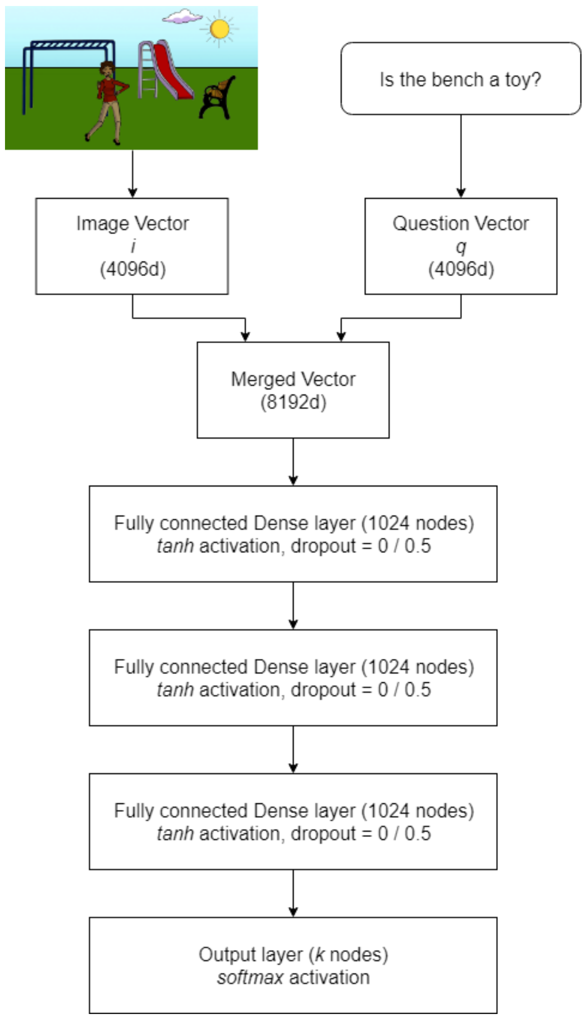
Figure 13: Linear SVM for VQA

Training: For training, first the entire dataset of 8192 dimensions was fed in to an SVM. For further experiments, we also performed dimensionality reduction using Singular Vector Decomposition (SVD) and reduced the size of feature matrix from 8192 to 1000 and 100. The hyperparameters were chosen using the technique of randomized search over the values of C and *gamma*. This set of experiments was carried out for both linear and non-linear (with RBF kernel) SVMs.

5.3.2 Simple Multilayer Perceptron

A multilayer perceptron (MLP) is a fully-connected artificial neural network which is commonly used for classification. It can be perceived as a network that uses logistic regression algorithm to transform the fed input into a non-linear transformation, which it learns while training. It is a simple feed-forward network, but it also uses back-propagation to change the network parameters based on the error encountered in the previous iteration. For this experiment, we use the encodings obtained for the images and questions as inputs to the network. The training dataset of concatenated image + question embeddings of 8192 dimensions is fed into a simple MLP, which has 3 fully-connected dense layers and 1024 nodes in each of those layers. Each of these layers use the *tanh* activation function with a dropout value of 0.5. The final output layer uses *softmax* to perform classification. Figure 14 shows an overview of this architecture.

Training: Training was done in 100 epochs and each epoch was trained using mini-batches of size 128.



Is the bench a toy?
Image Vector
i
(4096d)
Question Vector
q
(4096d)
Merged Vector
(8192d)
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Output layer (k nodes)
softmax activation

Figure 14: Simple MLP Architecture

5.3.3 Deeper network with normalized embeddings

The idea behind this architecture was to transform the embeddings to 1000 dimensions so as to reduce the size of the feature matrix. This transformation was done using a fully-connected dense layer with non-linear *tanh* activation function. The combined vector is now obtained by calculating the dot product of both the 1000 dimensional vectors each corresponding to the question and the image. The difference between this architecture and the simple MLP is that we use a concatenated question + image vector which has 8192 dimensions in the MLP, whereas here we use a 1000 dimensional vector obtained by element-wise product of the question and image embedding. Figure 15 shows an overview of this architecture.

Training: Training was done in 300 epochs and each epoch was trained using mini-batches of size 128.

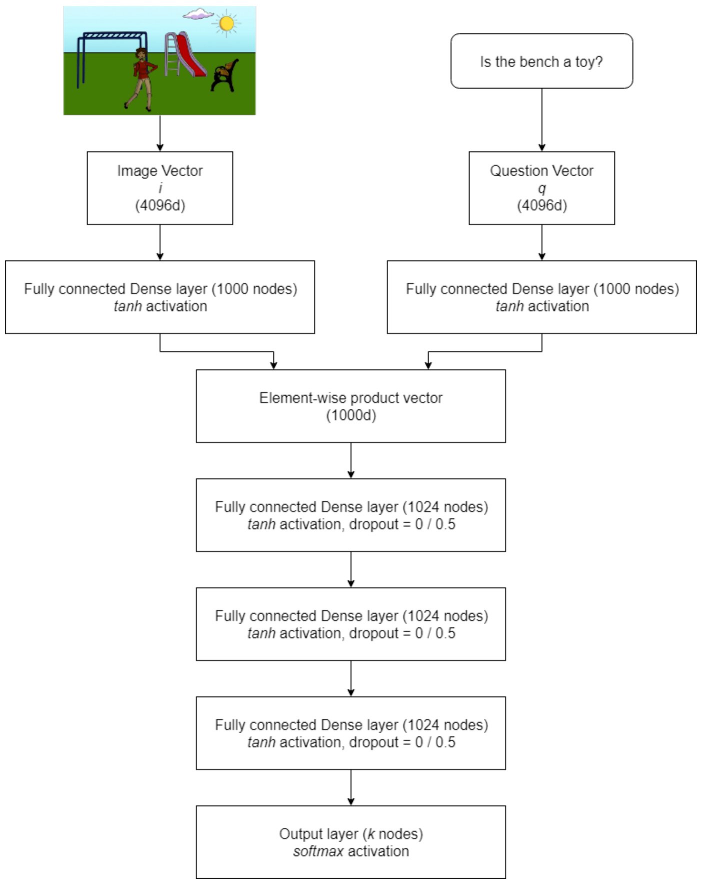

Is the bench a toy?
Image Vector
i
(4096d)
Question Vector
q
(4096d)
Fully connected Dense layer (1000 nodes)
tanh activation
Fully connected Dense layer (1000 nodes)
tanh activation
Element-wise product vector
(1000d)
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Fully connected Dense layer (1024 nodes)
tanh activation, dropout = 0 / 0.5
Output layer (k nodes)
softmax activation

Figure 15: Deep network with normalized embeddings

CHAPTER 6

Experimental Results

For all the experiments, the training and testing splits from the VQA dataset were used. A virtual machine (VM) instance was created on the Google Cloud platform. The configuration of the VM was as follows:

- 8 CPU cores
- 52 GB memory
- 100 GB SSD

Table 2 shows the size of datasets used for training each model. For the *Yes/No* subcategory, only the questions with answer-type "yes/no" were chosen to train and test the models. For experiments on the *Other* subcategory, top 10, 50 and 100 answers that occurred most frequently in the entire VQA dataset were chosen. Then the questions corresponding only to those answers were put in a subset which was then used for training and evaluation. All other questions whose answers did not belong to any of the top occurring ones were discarded.

Table 2: Data used for training and validation

	Yes/No		***Other (k=100)***	
	Images	**Questions**	**Images**	**Questions**
Abstract Training	15643	24396	12999	17824
Abstract Validation	7859	12321	6503	8961
Real Training	69797	166882	76208	219269
Real Validation	33940	80541	26007	47029

Table 3: Accuracy values for Yes/No and Other multiple-choice questions on VQA dataset for abstract images

Model	***Yes/No***	***k=10***	***k=50***	***k=100***
KNN	70.65	52.16	46.64	43.91
Linear SVM	63.68	60.94	54.32	50.27
SVM with RBF kernel	58.26	55.03	46.72	46.63
Simple MLP	68.83	59.93	53.03	52.52
MLP with normalized embeddings	**72.76**	**64.76**	**58.97**	**57.54**

All the proposed architectures were utilized to build models and these models were trained on the VM instance created in Google Cloud. Table 3 shows the results (accuracy) of all these architectures trained and tested on the VQA dataset for abstract images. Figure 16 shows precision scores for the top 10 answers from the abstract images dataset.

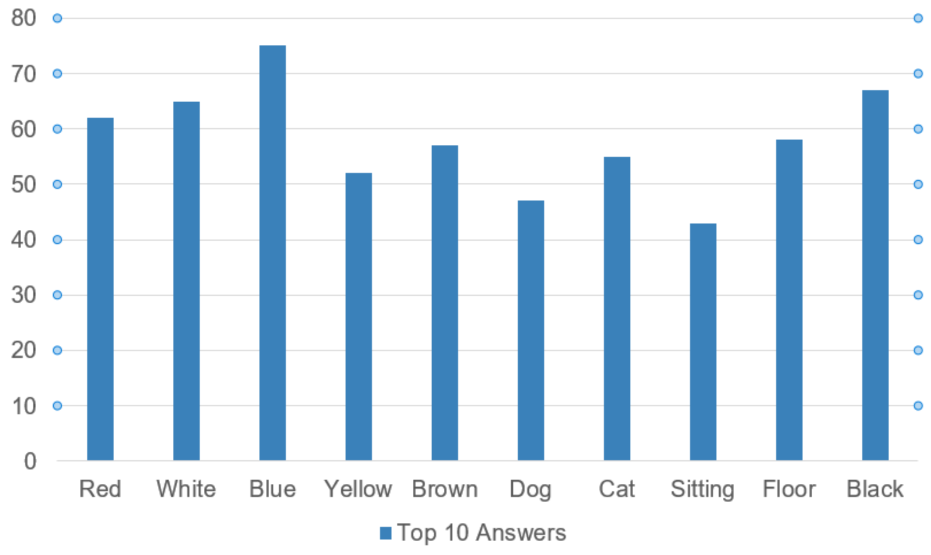


Figure 16: Precision values for the top 10 answers on the VQA validation dataset for abstract images on "Other" category (Results from normalized MLP model)

Table 4 shows the results (accuracy) of all the architectures trained and tested on the VQA dataset for real images. Figure 17 shows precision scores for the top 10

Table 4: Accuracy values for Yes/No and Other multiple-choice questions on VQA dataset for real images

Model	***Yes/No***	***k=10***	***k=100***	***k=1000***
VQA Baseline KNN	71.94	–	–	33.56
KNN (Our model)	**73.94**	52.16	43.91	–
Simple MLP (Our model)	74.83	50.61	46.73	–
VQA Baseline LSTM Q + Norm I	80.52	–	–	53.01
MLP with normalized embeddings (Our model)	**82.06**	**60.26**	**59.47**	–

answers from the real images dataset. These results are compared with the baseline accuracies provided by VQA.

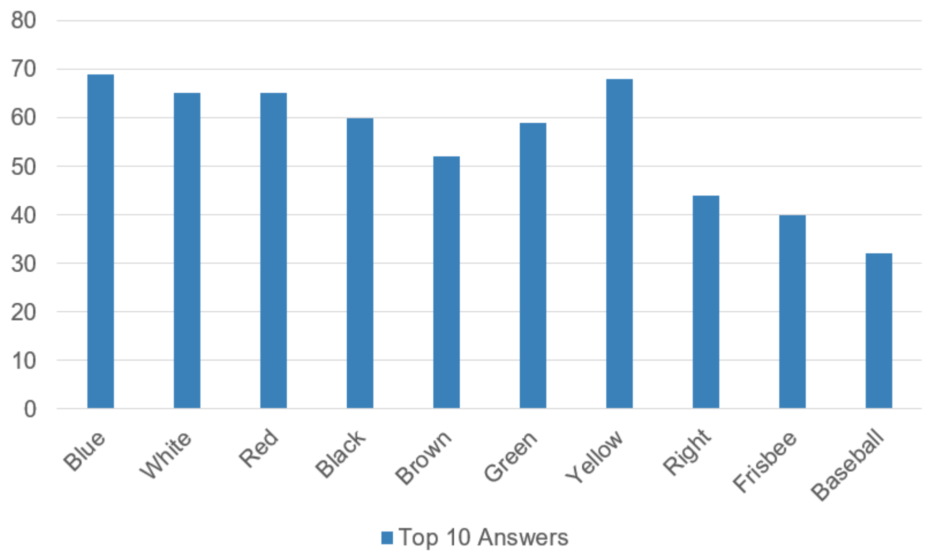
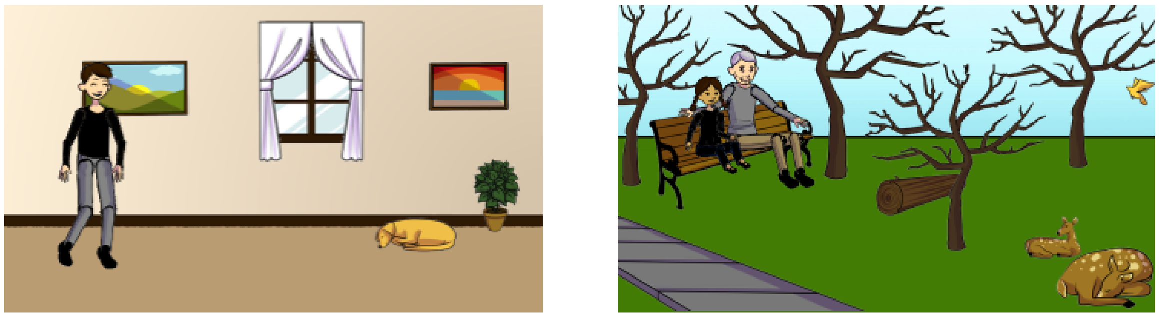


Figure 17: Precision values for the top 10 answers on the VQA validation dataset for real images on "Other" category (Results from normalized MLP model)

From the results in Table 3 and Table 4, we see that the deep learning model where the question embeddings are normalized outperforms the other models, both in *Yes/No* and *Other* subcategories of questions. Traditional machine learning models like linear and non-linear SVMs also performed considerably well, with not much difference between their accuracy scores and the accuracy scores of both the deep-learning models. But the advantage of using SVMs is that they are extremely efficient in terms of memory utilization and time taken for training. If optimal performance is required in terms of resources consumed and efficiency is more important than accuracy, then these models are definitely the right choice since the trade-off on accuracy does not result in a huge degradation of performance.

However, one of the reasons why the normalized MLP performs considerable better than the other models could be that the network learns the relation between the input question and image way better compared to the others. This could be because the embeddings of the image and question are fused together by element-wise multiplication and every element of the image embedding is, in a way, overlapped with every element in the question embedding.

A few examples of predicted answers on both subcategories of questions on abstract images are shown below.

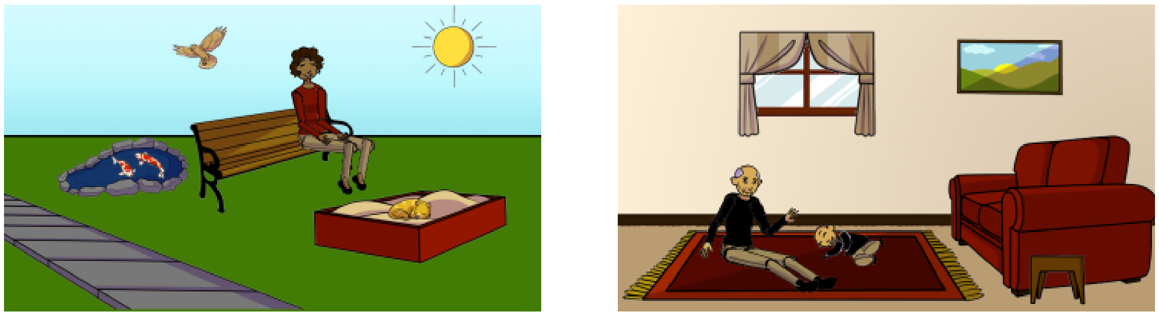


Q: Is there any furniture in the room?
A: No

Q: Is it a hot day?
A: Yes
Q: Are both people on the bench?
A: Yes

Figure 18: Examples of predicted answers from the "Yes/No" subcategory

Figure 18 shows questions about two different input images from the *Yes/No* subcategory of the validation dataset and their predicted answers. As it can be seen, the answer to the question about the first image was predicted correctly, but one of the questions about the second image was answered incorrectly (answer shown in red).



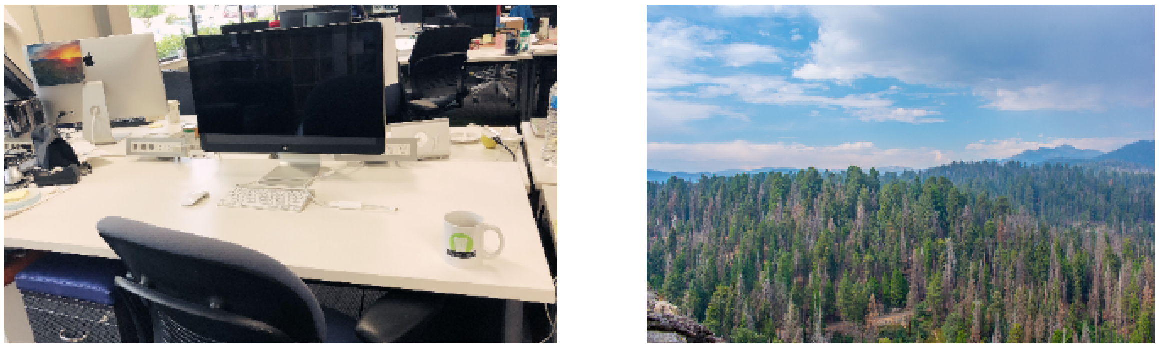
Q: What is asleep in the sandbox?
Answer choices:
Red
White
Blue
Yellow
Brown
Dog
Cat
Sitting
Floor
Black

Q: What color are the rug fringes?
Answer choices:
Red
White
Blue
Yellow
Brown
Dog
Cat
Sitting
Floor
Black

Figure 19: Examples of predicted answers from the "Other" subcategory with the top 10 answers provided as multiple choices

Figure 19 shows questions about two different input images from the *Other* subcategory of the validation dataset and their predicted answers. The top 10 most frequent answers were provided as multiple choices for each question. As it can be seen, the answer to the question about the first image was predicted correctly, but the answer to the question about the second image was predicted incorrectly (answer shown in red). The model predicted the color of the rug fringes as "brown" but the ground-truth answer for this question was "yellow".

Figure 20 shows predictions on different real images with questions asked on both *Yes/No* and *Other* subcategories. The input images were clicked by ourselves and were never seen by the model during the training or testing phase. Most questions



Q: Is the desk cluttered?
A: No

Q: What color is the screen right now?
Answer Choices:
Blue
White
Red
Black
Brown
Green
Yellow
Right
Frisbee
Baseball

Q: Are there clouds in the picture?
A: Yes

Q: How does the sky look?
Answer Choices:
Blue
White
Red
Black
Brown
Green
Yellow
Right
Frisbee
Baseball

Figure 20: Examples of predicted answers on real images

about colors in the pictures were predicted correctly, both on the abstract and real images.

CHAPTER 7

Conclusion and Future Work

VQA is a relatively new field that needs in-depth understanding of both images and text. Looking at the current research scenario in the area of deep learning, it can be expected that VQA systems will definitely improve over time, both in terms of optimality and accuracy. Deep learning has already showed an exponential increase in the performance of various computer vision and language models. Utilizing these individual components to build a system that combines them will drastically improve results seen for tasks like VQA.

In terms of metrics used to measure the performance of a VQA system, the most widely used metrics are accuracy and F1 score. This is because all the current systems built for VQA treat it as a classification problem. They assume that the answer to any natural language question asked about an image will potentially belong to one of the "answer classes" that the model is trained upon. An extension to this could be propose a system that uses a generative model to actually generate a meaningful answer based on the image. This perspective directs the technique away from pure classification, thus requiring the need to introduce new metrics for performance evaluation. But it seems extremely likely that new datasets along with new metrics will be introduced soon and this will allow researchers to look into the problem from a completely new and fresh perspective.

Taking into consideration all the existing systems (including this project) there are still ongoing discussions about several issues regarding the techniques with which the models are trained and evaluated. One of the most common and open-ended debate is whether these VQA systems that are trained and evaluated on questions with multiple choice answers in the datasets, can actually be considered systems with "good performance" in real-life scenarios. In most datasets, the questions that are used for training the model are gathered by crowdsourcing. Are these questions enough to represent all potential questions that the system will most likely encounter? The formulation of questions also has a huge impact on feature engineering and hence on the overall predictive ability of the system.

The current state-of-the-art systems on VQA are still far-fetched from human performance on the same datasets. But with the availability of published research and emerging tools and technologies, there is a huge scope for improvement in the results.

LIST OF REFERENCES

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

[2] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*, 2017.

[3] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[4] 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*, 2015, pp. 2425–2433.

[5] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, "Visual7w: Grounded question answering in images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4995–5004.

[6] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1682–1690. [Online]. Available: http://papers.nips.cc/paper/5411-a-multi-world-approach-to-question-answering-about-real-world-scenes-based-on-uncertain-input.pdf

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

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

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[10] K. Kafle and C. Kanan, "Answer-type prediction for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4976–4984.

[11] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[12] D. Teney, L. Liu, and A. v. d. Hengel, "Graph-structured representations for visual question answering," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 3233–3241.

[13] K. J. Shih, S. Singh, and D. Hoiem, "Where to look: Focus regions for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4613–4621.

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

[15] 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*, 2016, pp. 21–29.

[16] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Advances In Neural Information Processing Systems*, 2016, pp. 289–297.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.