Visual Question Answering

A project report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Aathira Satheesh
Anagha K
Fathima Salim
Jerin Jayaraj

Federal Institute of Science And Technology (FISAT)®
Angamaly, Ernakulam

Affiliated to

APJ Abdul Kalam Technological University
CET Campus, Thiruvananthapuram

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY
(FISAT)
Mookkannoor(P.O), Angamaly-683577

CERTIFICATE

This is to certify that the report entitled "**Visual Question Answering**" is a bonafide record of the project submitted by **Aathira Satheesh (FIT16CS001) , Anagha K (FIT16CS020), Fathima Salim (FIT16CS048), Jerin Jayaraj (FIT16CS059)** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in Computer Science & Engineering during the academic year 2019-20.

Dr. Prasad J C

Staff in Charge | Project Guide | Head of the Department

ABSTRACT

Given an image and a natural language question about the image, the task is to provide an accurate natural language answer.Mirroring real-world scenarios, such as helping the visually impaired, both the questions and answers are open-ended. Visual questions selectively target different areas of an image, including background details and underlying context. As a result, a system that succeeds at VQA typically needs a more detailed understanding of the image and complex reasoning than a system producing generic image captions. Moreover, VQA is amenable to automatic evaluation, since many open-ended answers contain only a few words or a closed set of answers that can be provided in a multiple-choice format.

Contribution by Aathira Satheesh

I personally learned how back propagation works in a neural network by implementing XOR function and by checking if a mail if spam or not . I learnt the working of CNN using Keras library in python . I learned the training of a neural network in Google collab . I have got a clear idea about the working of a Keras neural network model . I was able to create our proposed model after learning about Keras and classify the answers into top thousand class labels using labelencoder class.This project has improved my knowledge about deep learning and neural networks.

Aathira Satheesh

Contribution by Anagha

I learned the practical application of neural network and learned the working of backpropogation algorithm. I implemented that by checking if a mail is spam or not, hence a clear idea is obtained. I learned the training of neural network in Google colab. I also learnt the importance of Keras neural network library for our model by understanding the layers like Dense,Dropout,Convolution2D an flatten. I extracted the question features by implementing the word embedding. For this purpose, tokenization had to be done. This project helped me improving my knowledge on neural network, associated python libraries and also deep learning.

Anagha K

Contribution by Fathima

I personally learnt and familiarized myself with CNN and implemented the same with a small project of handwritten digit classification on Keras running on Tensorflow and Python3. I did a thorough study on how LSTM networks work and also familiarized myself with data preprocessing on Keras, building and training a model with different layers and evaluating it on a test data. I created 2 sequential models in Keras for language features and image features and merged them to obtain the VQA model and tried out different activation functions in each case to evaluate and compare the same in our project. This project has helped me gain knowledge and hands on experience on neural networks, classification using CNN and keras neural network library.

Fathima Salim

Contribution by Jerin Jayaraj

I learned VGG-16 model and VGG-19 models for image classification in detail.Since VGG-19 was found to have more accuracy in the test set, it is used in the program to extract the image features. Using this network, I trained a model to perform classification on dog vs cat data set in Google colab. Also I created the User Interface using Qt and connected the UI part and python program using sockets.This project helped me to gain knowledge on deep learning field more specifically on CNNs.

Jerin Jayaraj

ACKNOWLEDGMENT

If the words were considered as symbols of Approval and token of Acknowledgement, then let the words pay the heralding role of expressing our gratitude. First and foremost, we praise the God almighty for the grace he showered on us during our studies as well as our day to day activities.

We would like to take this chance to thank the our Principal, Dr George Isaac for providing us with such an environment, where students can explore creative ideas. Equally eligible is the Head of the Department of Computer Science,Dr Prasad J.C who always guided us and rendered his help in all phases of our project.

We are extremely grateful to our project guide Mrs Roshna K I for the valuable suggestions for the project. We would also like to extend out gratitude to the project in charge Mr Mahesh C for giving us all the technical support . We are extremely grateful to the project-in-charge Mrs Hansa J Thattil for the support she has given during lab sessions.We also sincerely thank the Computer science and Engineering faculty for providing us with invaluable help.

Overall,nothing comprises the support and hard work equally put in by our teammates.Last,but not the least ,we thank all our families and friends for giving us the help, strength and courage for accomplishing the task.

Aathira Satheesh
Anagha k
Fathima Salim
Jerin Jayaraj

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

A VQA system as an algorithm that takes as input an image and a natural language question about the image and generates a natural language answer as the output. This is by nature a multi-discipline research problem. Let's take, for example, the questions about the previous image. We need NLP for at least two reasons: to understand the question and to generate the answer.The main difference in VQA is that the search and the reasoning part must be performed over the content of an image. So, to answer if there are any humans, the system must be able to detect objects. To say if it is raining, it needs to classify a scene. To answer who are the teams, the system needs world knowledge. Finally, to say which player has the ball, commonsense reasoning and, very likely, knowledge reasoning are necessary. Many of these tasks (object recognition, object detection, scene classification, etc.) have been addressed in the field of Computer Vision (CV), with impressive results in the last few years.
VQA is also amenable to automatic quantitative evaluation, making it possible to effectively track progress on this task. While the answer to many questions is simply yes or no, the process for determining a correct answer is typically far from trivial Moreover,since questions about images often tend to seek specific information, simple one-to-three word answers are sufficient for many questions. In such scenarios, we can easily evaluate a proposed algorithm by the number of questions it answers correctly.his goal-driven task is applicable to scenarios encountered when visually- impaired users or intelligence analysts actively elicit visual information.We present a large dataset that contains 204,721 images from the MS COCO dataset and a newly created abstract scene dataset that contains 50,000 scenes. The MS COCO dataset has images depicting diverse and complex scenes that are effective at eliciting compelling and diverse questions.our proposed task involves open ended, free-form questions and answers provided by humans. Our goal is to increase the diversity of knowledge and kinds of reasoning needed to provide correct answers.

1.2 Problem Statement

On seeing an image when a question is asked , answering it is a challenging task. As a human we can answer this pretty easily but for a machine to answer it , the machine needs to learn a lot of things. The search and the reasoning part must be performed over the content of an image. We need to create a model that predicts the answer of an open-ended question related to a given image. We also need to create an user interface so that user can interact easily.

1.3 Objective

Our objective is to design a model which allows user to ask questions based on an image given by the user and the model should give an accurate natural language answer.

Chapter 2

Related works

2.1 Fact Based Visual Question Answering

Visual Question Answering (VQA) has attracted much attention in both computer vision and natural language processing communities, not least because it offers insight into the relationships between two important sources of information. Current datasets, and the models built upon them, have focused on questions which are answerable by direct analysis of the question and image alone. The set of such questions that require no external information to answer is interesting, but very limited. It excludes questions which require common sense, or basic factual knowledge to answer, for example. Here we introduce FVQA (Fact-based VQA)[**?**], a VQA dataset which requires, and supports, much deeper reasoning. FVQA primarily contains questions that require external information to answer. Thus a conventional visual question answering dataset is extended, containing image-question-answer triplets, through additional image-question-answer-supporting fact tuples. Each supporting-fact is represented as a structural triplet

Several baseline models are evaluated on the FVQA dataset, and describe a novel model which is capable of reasoning about an image on the basis of supporting-facts.The proposed FVQA dataset differs from existing VQA datasets in that it provides a supporting fact which is critical for answering each visual question. A novel VQA approach is developed, which is able to automatically find the supporting-fact for a visual question from large-scale structured knowledge bases. Instead of directly learning the mapping from questions to answers,this approach learns the mapping from questions to KB queries, so it is much more scalable to the diversity of answers. all the information extracted from images and KBs are stored as a graph of interlinked RDF triples. State-of-the-art RNN approaches directly learn the mapping between questions and answers, which, however, do not scale well to the diversity of answers and cannot provide the key information that the reasoning is based on. In contrast, we propose to learn themapping between questions and a set of KB-queries, such that there is no limitation to the vocabulary size of the answers (i.e., the answer to a test question does not have to be observed ahead in the training set) and the supporting-facts used for reasoning can be provided.

A. Constructing a Unified KB- The primary step of our approach is to construct a unified KB, which links the visual concepts extracted from each image to the corresponding concepts in multiple KBs. A visual concept X of type T (T = Object, Sence or Action) extracted from the image with id I is stored in two triples (X,Grounded,I) and (X,VC-Type,T). Concepts in multiple KBs with the same meaning as X are directly linked to X. By doing so, the rich external knowledge about X is linked to the images that X is grounded in.

B. Question-Query Mapping- In this approach, three characteristics of visual questions are firstly predicted by trained LSTM models, i.e., key visual concepts (TKVC), key relationships (TREL) and answer sources (TAS). In the training data, these characteristics of a question can be obtained through the annotated supporting fact and the given answer, and we have collected 32 different combinations of (TKVC,TREL,TAS) in the proposed dataset (see Appendix). Since both question and query are sequences, the question-query mapping problem can be treated as a sequence-to-sequence problem [70], which can be solved by Recurrent Neural Network (RNN). While in this work, we consider each distinct combination of the three characteristics as a query type and learn a 32-class classifier using LSTM models , in order to identify the above three properties of an input question and perform a specific query.
C. Querying the KB- Retrieving the correct supporting-fact is the key to answering a visual question in our proposed dataset. To this end, KB queries are constructed to search for a number of candidate supporting-facts. To be specific, given a question's key relationship (TREL) and key visual concept (TKVC) as inputs, a KB-query (?X,?Y) = Query(I,TREL,TKVC) is constructed as follows:
Find ?X, ?Y, subject to
(?X,Grounded,I) and
(?X,VC-Type,TKVC) and
(?X,TREL,?Y), where I denotes the id of questioned image; Grounded is a relationship representing that a specific visual concept is grounded in a specific image; VC-Type is another relationship used to describe the type of a visual concept; ?X and ?Y are two variables to be searched and returned. There are three triple templates to be matched: (?X,Grounded,I) searches all visual concepts ?X that are grounded in image I; (?X,VC-Type,TKVC) restricts that the type of ?X should be TKVC; (?X,TREL,?Y) restricts that ?X should be linked to at least one concept ?X via relationship TREL.
D. Answering-The answer to a visual question in our collected dataset can be either the visual concept in the supporting-fact (i.e., ?X), or the other concept on the KB side (i.e., ?Y). In the test phase, whether the answer is ?X or ?Y is determined by the value of answer source TAS predicted by the LSTM classifier in Section IV-B: the answer is ?X if TAS = Image, and ?Y if TAS = KB. The last issue before arriving at the answer is how to select the most relevant supporting-fact from candidates. In this work, the supportingfact is selected by matching between the given question and concepts ?Y (or ?X) in candidate facts .

2.2 Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for VQA

The problem of Visual Question Answering (VQA), is addressed which requires joint image and language understanding to answer a question about a given photograph. Recent approaches have applied deep image captioning methods based on convolutional recurrent networks to this problem, but have failed to model spatial inference. To remedy this, a model call the Spatial Memory Network is proposed and apply it to the VQA task. Memory networks are recurrent neural networks with an explicit attention mechanism that selects certain parts of the information stored in memory. The Spatial Memory Network stores neuron activations from

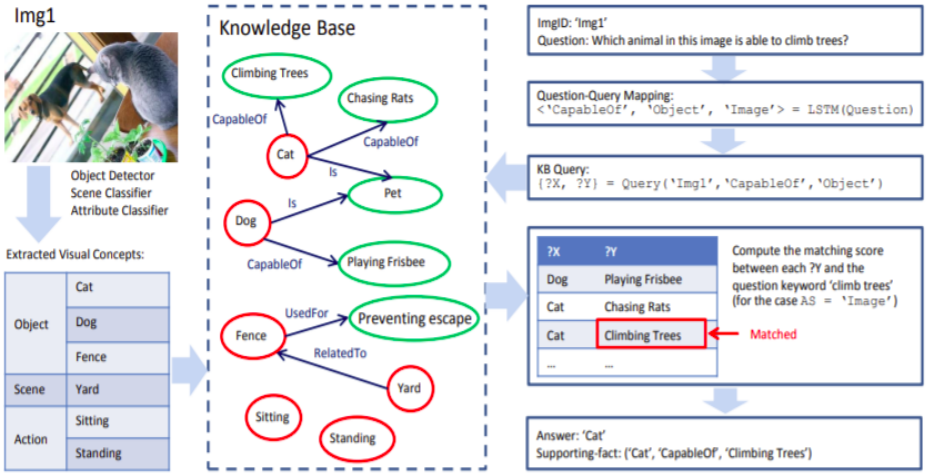


Figure 2.1: An example of the reasoning process of the proposed VQA approach.

different spatial regions of the image in its memory, and uses the question to choose relevant regions for computing the answer, a process of which constitutes a single "hop" in the network. A novel spatial attention architecture that aligns words with image patches is created in the first hop, and obtain improved results by adding a second attention hop which considers the whole question to choose visual evidence based on the results of the first hop. To better understand the inference process learned by the network, synthetic questions are designed that specifically require spatial inference and visualize the attention weights. The model is evaluated on two published visual question answering datasets, DAQUAR and VQA , and improved results is obtained compared to a strong deep baseline model (iBOWIMG) which concatenates image and question features to predict the answer.

A major drawback of existing models is that they do not have any explicit notion of object position, and do not support the computation of intermediate results based on spatial attention.Answering visual questions often involves looking at different spatial regions and comparing their contents and/or locations.The text QA memory network stores textual knowledge in its "memory" in the form of sentences, and selects relevant sentences to infer the answer.The input to network is a question comprised of a variable-length sequence of words, and an image of fixed size.Each word in the question is first represented as a one-hot vector in the size of the vocabulary, with a value of one only in the corresponding word position and zeros in the other positions.

The words in questions are used to compute attention over the visual memory, which contains extracted image features. The attention embedding projects for each visual feature vector such that its combination with the embedded question words generates the attention weight at that location. The evidence embedding detects the presence of semantic concepts or objects, and the embedding results are multiplied with attention weights and summed over all locations to generate the visual evidence vector. Finally, the visual evidence vector is combined with the question representation and used to predict the answer for the given image and question.

2.3 Fusing Attention with Visual Question Answering

In VQA, the system is provided an image and a corresponding question, then must take the two and provide a correct answer. In contrast to description of visual content, which has been well studied, VQA must take information from both a question and an image, and combine them to find an answer. This is a more complex task that image annotation or question answering using just text because it involves different types of information being processed together. One simple technique for solving this is to combine an LSTM for the language processing and a CNN for processing the image . These features are combined into an MLP, which learns to output the answer to the question. This has become the baseline technique used for evaluating in multiple dataset.

However, this technique looks at the entire image holistically, attempting to use global features to explain and collect information from multiple objects. As such, a simple architecture such as this one often has trouble with higher level reasoning such as spatial awareness of multiple different objects . On the other hand, humans have the ability to quickly process and analyze complex scenes. Rather than processing an entire scene at once, humans focus on small areas within their visual field and saccade to different area.The information gained in each saccade is then remembered by the brain and stitched into an understanding of the entire scene. In the same way, it is preferred use a divide-and-conquer process in VQA by separating images into different patches that contain the relevant information.

Saliency measures have a number of postprocessing techniques used to improve correlation with eye-tracking, such as blurring, center-weighting, and accentuating peaks. A Gaussian blur is applied to the final saliency map, however, since the dataset used here is not composed of natural images and is grayscale, the peak accentuation and center-weighting are not used.With the saliency map computed, the next step is to crop out the relevant image patches. To do this, first use an adaptive threshold to threshold the saliency map. Next, identify the thresholded object that contains the most salient point in the map.Then crop a rectangle that surrounds this object and expand the borders by 10 percent to ensure coverage of the object.After cropping each patch, the saliency map is to inhibit cropping the same area again.

By using the focus of attention, it is able to extract location and scale features as well as simplify the data being input to the CNN. By combining these aspects, it has shown to improve performance greatly in areas that require higher reasoning than just object recognition as well as reduce the computation requirements. This shows the usefulness of attention algorithms in visual problems.

2.4 Long-term Recurrent Convolutional Networks for Visual Recognition and Description

Models based on deep convolutional networks have dominated recent image interpretation tasks.A class of recurrent convolutional architectures is described which

is end-to-end trainable and suitable for large-scale visual understanding tasks, and demonstrate the value of these models for activity recognition, image captioning, and video description. In contrast to previous models which assume a fixed visual representation or perform simple temporal averaging for sequential processing, recurrent convolutional models are “doubly deep” in that they learn compositional representations in space and time. Learning long-term dependencies is possible when non linearities are incorporated into the network state updates. Differentiable recurrent models are appealing in that they can directly map variable-length inputs to variable-length outputs and can model complex temporal dynamics, yet they can be optimized with backpropagation. The recurrent sequence models are directly connected to modern visual convolutional network models and can be jointly trained to learn temporal dynamics and convolutional perceptual representations.In this paper , it is shown that the convolutional networks with recurrent units are generally applicable to visual time-series modeling, and argue that in visual tasks where static or flat temporal models have previously been employed, LSTM style RNNs can provide significant improvement when ample training data are available to learn or refine the representation.
Specifically, The paper shows that LSTM type models provide for improved recognition on conventional video activity challenges and enable a novel end-to-end optimizable mapping from image pixels to sentence-level natural language descriptions. It is also shown that these models improve generation of descriptions from intermediate visual representations derived from conventional visual models.The paper instantiates the proposed architecture in three experimental settings . First, it is shown that directly connecting a visual convolutional model to deep LSTM networks, it is possible to train video recognition models that capture temporal state dependencies.Second, the paper explore end-to-end trainable image to sentence mappings.Finally, It is shown that LSTM decoders can be driven directly from conventional computer vision methods which predict higher-level discriminative labels.

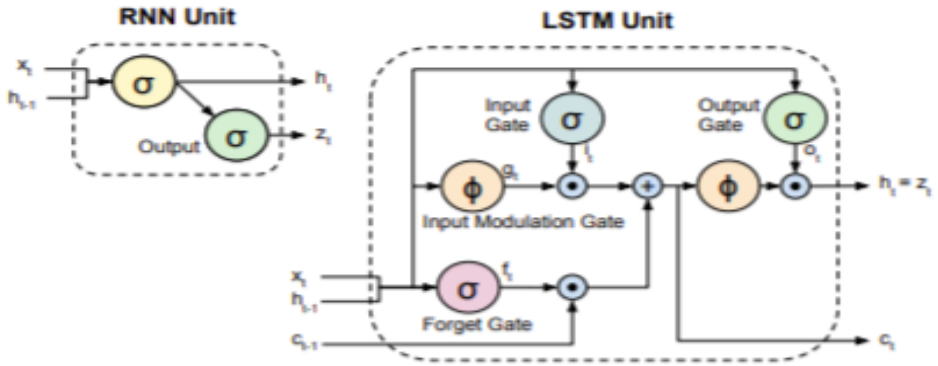


Figure 2.2: RNN vs LSTM

2.5 Where To Look: Focus Regions for Visual Question Answering

This paper present a method that learns to answer visual questions by selecting image regions relevant to the text-based query. This method maps textual queries and visual features from various regions into a shared space where they are compared for relevance with an inner product. Here the paper focuses on learning where to look but also provide useful baselines and analysis for the task as a whole. The contributions are as follows:
• Presents an image-region selection mechanism that learns to identify image regions relevant to questions.
• Presents a learning framework for solving multiple choice visual QA with a margin-based loss .
• Compares with baselines that answer questions without the image, use the whole image, and use all image regions with uniform weighting, providing a detailed analysis for when selective regions improve VQA performance.
This method learns to embed the textual question and the set of visual image regions into a latent space where the inner product yields a relevance weighting for each region. The input is a question, potential answer, and image features from a set of automatically selected candidate regions. We encode the parsed question and answer using word2vec and a two-layer network. Visual features for each region are encoded using the top two layers (including the output layer) of a CNN trained on ImageNet.The language and visual features are then embedded and compared with a dot product, which is soft maxed to produce a per-region relevance weighting. Using these weights, a weighted average of concatenated vision and language features is the input to a 2-layer network that outputs a score for whether the answer is correct.

2.6 IQA:Visual Question Answering in Interactive Environments

Challenges of IQA [**?**] are that the agent must be able to navigate through the environment,It must acquire an understanding of its environment including objects, actions, and affordances,The agent must be able to interact with objects in the environment and The agent must be able to plan and execute a series of actions in the environment conditioned on the questions asked of it To address these challenges ,HIMN (Hierarchical Interactive Memory Network) is proposed.

HIMN is factorized into a hierarchy of controllers. HIMN uses a rich semantic spatial memory that encodes a semantic representation of each location in the scene.At each time step, the esGRU(Egocentric Spatial GRU) swaps in local egocentric copies of this memory into the hidden state of the GRU, performs computations using current inputs, and then swaps out the resulting hidden state into the global memory at the predetermined location. The limitations of this method is that due to the 2D nature of the semantic spatial map, HIMN is unable to differentiate between an object being inside a container and being on top of the

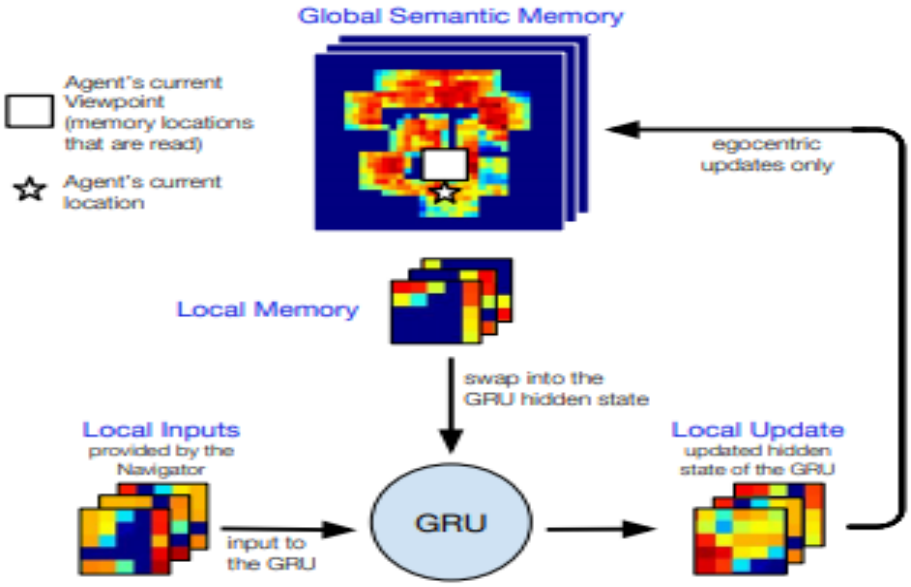


Figure 2.3: HIMN Architecture

container. HIMN is still inefficient at exploring the environment since it wastes time exploring already searches areas. A high level controller, referred to as the Planner chooses the task to be performed and generates a command for the chosen task. Tasks specified by the Planner are executed by low level controllers (Navigator, Manipulator, Detector, Scanner and Answerer) They return control to the Planner when a task termination state is reached.

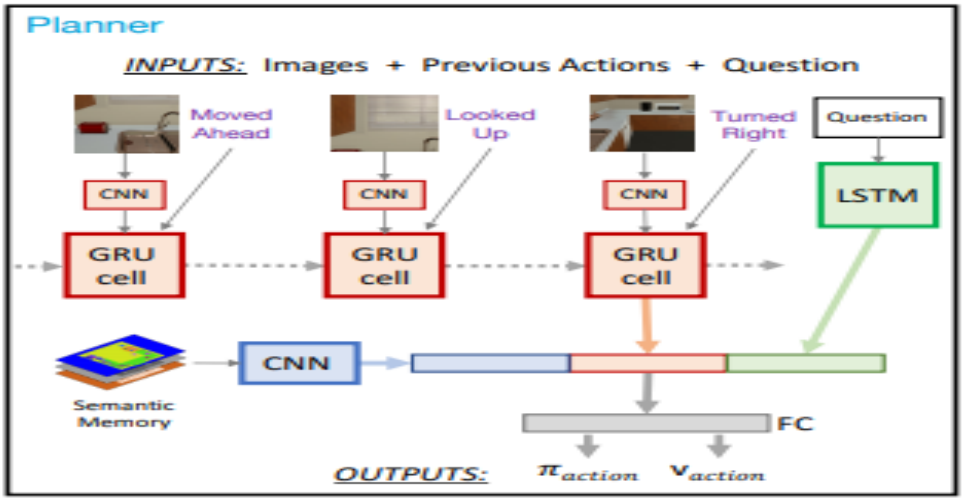


Figure 2.4: Schematic representation of planner

The high level Planner invokes low level controllers in order to explore the environment, gather knowledge needed to answer the given question, and answer the question. We frame this as a reinforcement learning problem where the agent must issue the fewest possible commands that result in a correct answer. The agent must learn to explore relevant areas of the scene based on learned knowledge (e.g.apples are often in the fridge, cabinets are openable, etc.), the current memory state (e.g. the fridge is to the left), current observations (e.g. the fridge is closed)

and the question. At every timestep, the Planner chooses to either invoke the Navigator providing a relative location in a 5x5 grid in front of the agent, invoke the Scanner with a direction such as up or left, invoke the Manipulator with open/close commands on a nearby object, or invoke the Answerer for a total of 32 discrete actions. It does this by producing a policy consisting of probabilities i for each action, and a value v for the current state. pi and v are learned using the A3C algorithm . The Planner has read only access to the semantic memory centered around the agent's current location. The output of this GRU is combined with the question embedding and an embedding of the nearby semantic spatial memory to predict pi and v.

2.7 Ask Your Neurons: A Neural-based Approach to Answering Questions about Images

By combining latest advances in image representation and natural language processing, this paper propose Neural-Image-QA, an end to-end formulation to this problem for which all parts are trained jointly. In contrast to previous efforts, the challenge is a multi-modal problem where the language output (answer) is conditioned on visual and natural language input (image and question). This approach Neural-Image-QA doubles the performance of the previous best approach on this problem.Additional insights are provided into the problem by analyzing how much information is contained only in the language part for which we provide a new human baseline. To study human consensus, which is related to the ambiguities inherent in this challenging task, two novel metrics are proposed and collect additional answers which extends the original DAQUAR dataset to DAQUAR-Consensus.The proposed method is benchmarked on a task of answering questions about images. The paper compares different variants of the proposed model to prior work In addition, the paper analyze how well questions can be answered without using the image in order to gain an understanding of biases in form of prior knowledge and common sense. A new human baseline for this task is provided. In particular, the WUPS score is extended to a consensus metric that considers multiple human answers.This model has learns biases and patterns that can be seen as forms of common sense and prior knowledge that humans use to accomplish this task. Indoor scene statistics, spatial reasoning, and small objects are not well captured by the global CNN representation, but the true limitations of this representation can only be explored on larger datasets. The extension from the existing DAQUAR dataset to DAQUARConsensus, which provides multiple reference answers allows to study inter-human agreement and consensus on the question answer task.

2.8 Exploring Models and Data for Image Question Answering

This work aims to address the problem of image-based question-answering (QA) with new models and datasets. Neural networks and visual semantic embeddings

are used , without intermediate stages such as object detection and image segmentation, to predict answers to simple questions about images. Recently, researchers studying image caption generation have developed powerful methods of jointly learning from image and text inputs to form higher level representations from models such as convolutional neural networks (CNNs) trained on object recognition, and word embeddings trained on large scale text corpora. Image QA involves an extra layer of interaction between human and computers. Here the model needs to pay attention to details of the image instead of describing it in a vague sense. The problem also combines many computer vision sub-problems such as image labeling and object detection. The answers consist of only a single word, which allows us to treat the problem as a classification problem. This also makes the evaluation of the models easier and more robust, avoiding the thorny evaluation issues that plague multi-word generation problems. The methodology presented here is two-fold. On the model side various forms of neural networks and visual-semantic embeddings are applied on this task, and on the dataset side new ways of synthesizing QA pairs from currently available image description datasets are proposed. Another dataset is created to produce a much larger number of QA pairs and a more even distribution of answers. While collecting human generated QA pairs is one possible approach, and another is to synthesize questions based on image labeling. The model builds directly on top of the LSTM sentence model and is called the "VIS+LSTM" model. It treats the image as one word of the question. In general, objects mentioned in image descriptions are easier to detect than the ones in DAQUAR's human generated questions, and than the ones in synthetic QAs based on ground truth labeling. This allows the model to rely more on rough image understanding without any logical reasoning. Lastly the conversion process preserves the language variability in the original description, and results in more human-like questions than questions generated from image labeling. As the currently available dataset is not large enough, an algorithm that helps us collect large scale image QA dataset from image descriptions ia developed. The question generation algorithm is extensible to many image description datasets and can be automated without requiring extensive human effort. The focus is on a limited domain of questions. However, this limited range of questions allows to study the results more in depth. Lastly, it is also hard to interpret why the models output a certain answer. By comparing models with some baselines one can roughly infer whether they understood the image. Visual attention is another future direction, which could both improve the results as well as help explain the model prediction by examining the attention output at every timestep.

Chapter 3

Design

3.1 Design

Design is the first step in the development phase for any engineered product or system. It may be described as "The process of applying various techniques and principles for the purpose of defining a device,a process or a system in sufficient detail to permit its physical realisation".

In early days software development consisted of a programmer writing code to solve a problem or automate a procedure.Nowadays,systems are so big and complex that teams of architects , analysts, programmers,testers and users must work together to create the millions of lines of custom-written code that drive our enterprises.
There are a sequence of stages in which the output of each stage becomes the input for the next

1. Project Planning, feasibility study: establishes a high level view of the intended project and determines its goal.

2. System analysis, requirements definition : Refines project goals into defines functions and operations of the intended applications.

3. System Design: Describes desired features and operations in detail,including screen layouts,process diagrams etc.

4. Implementation : The real code is written.

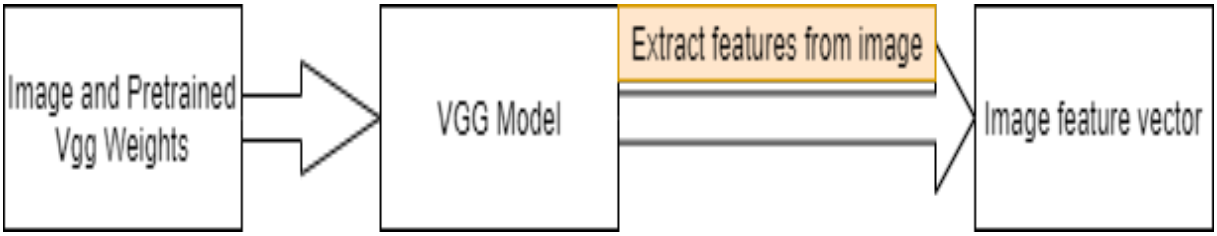
5. Integration and Testing : Brings all the pieces together into a special testing environment

6. Acceptances,installation,deployment : the final stages of initial development where the software is put into production and runs actual business.

3.2 Architecture

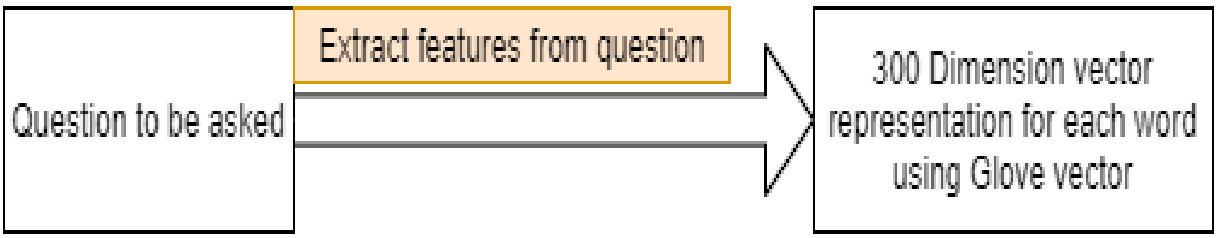
3.2.1 Extracting Image Features

We have used VGG-19 pre-trained model weights to obtain the image features . VGG-19 is trained on images of size 224x224 so every image we give must be reshaped to this size.We have used channel first approach as the image data format.This means that the image data is represented in a three-dimensional array where the first channel represents the color channels. Since we are using RGB channel , the value of channel will be 3.
Since we use channel first approach the dimension of the image will be 3x224x224. We need to expand the dimension include the number of batches.This axis dimension is required because VGG was trained on a dimension of (1,3,224,224) where first axis is for the batch size and even though we are using only one image, we have to keep the dimensions consistent. The last layer of the actual VGG-19 model is removed. The output of this is a (1,4096) dimension feature vector .



3.2.2 Extracting Question features

To extract question features word embedding of the question has to be done. First,the en_vectors_web_lg model is installed using python command. This package has exactly every vector provided by the Glove model. Experiments showed that having full vector gives more accuracy since it would not miss out any extra vectors. Then this package is loaded to the program using spacy. Then, tokenization is done. Tokenization allows the document to break down into standardized word representation eliminating punctuations.
A three dimensional array is created and initialized to zero using numpy. Now for each word in question,a vector is created. The embedding is done on 300 dimension space. Because the english language has verbs,noun,parts of speech etc. So a dimension needs to be selected which is computationally inexpensive and uses less computational resources. Dimensions around 50 to 500 can be used. Here, 300 is used. The input question is mapped onto a space (max_length,300), where max_length is the maximum number of words allowed in the question. Glove is the unsupervised learning algorithm internally used by Spacy. Glove reduces a given token to a 300-dimensional representation. The final vector is then returned.

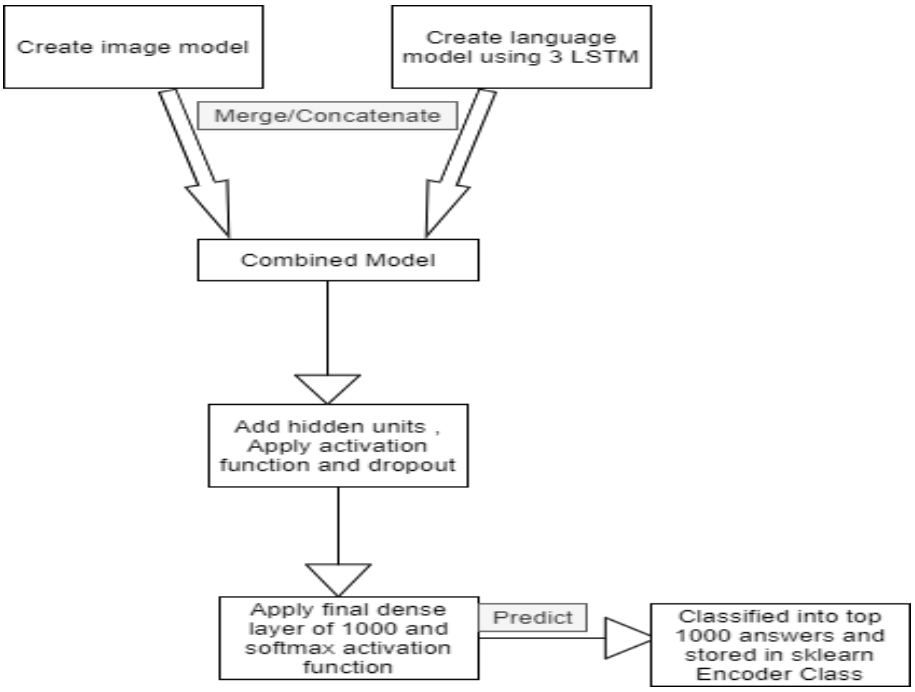


3.2.3 Combined Model

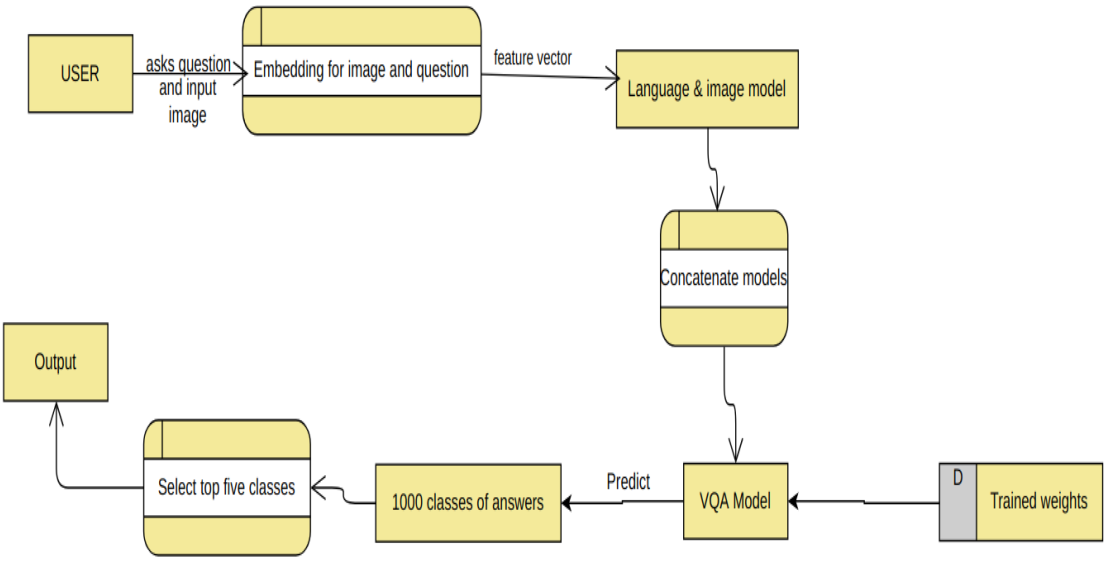
After extracting image features and language features from the input, a sequential model is created in keras respectively for both.

For the image model, image feature size is given as the input for the first layer in the sequential model and is reshaped to the required size. Similarly for the language model, a sequential model with three LSTM layers with 512 hidden units in each is created.

Both the sequential layers are combined together to form a new sequential model representing our VQA model. To this combined model a fully connected dense layer with 1024 hidden units, a layer with tanh activation and another layer with 0.5 dropout rate to fix overfitting is added sequentially in a set of three. Finally a dense layer with 1000 neurons and a layer with softmax activation is added to it in order to predict and select the top 1000 answers. This is the final VQA model returned. The labels then are converted and normalized into an understandable numerical data using sklearn label encoder class.



3.3 Data Flow Diagram



3.4 Algorithm

- First the image is given as input and it is resized into 224x224 to match the size of images that VGG was trained with
- Image features are extracted by using the pre-trained weight of the VGG 19 model .
- Next input the question where each word of the question is transformed into a 300 dimensional representation produced by spaCy .
- These features are then passed to a VQA Model which is a combination of LSTM and MLP
- This VQA model was trained on COCO train set available on VQA website for questions.
- The images for VQA model were trained on VGG-19 model.
- Our VQA model consists of an image model that defines structure of input image , language model that contains LSTM to memorise sequential data and then these two models are concatenated.
- Then we add dense layer , a tanh non linear activation function and also dropout for each of the three hidden layer.
- We then classify our answers to top 1000 answers.
- Finally we apply softmax layer that normalise the values to fit between 0 and 1 . It gives the probability value .

3.5 Implementation

The implementation is the stage in the project where the theoretical design is turned into a working system.It involves careful planning investigation on the current system and its constraints on its implementation design of methods to achieve the change over.

The most complex system being implemented the more involved will be the system analysis and design effort required just for implementation. According to the plan, the activities are carried out , discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system

Implementation is the final important phase. The most critical stage in achieving a successful new system and in giving the users confidence that the new system will work and be effective.

3.5.1 Implementation details

Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. tf.keras is TensorFlow's high-level API for building and training deep learning models. It's used for fast prototyping, state-of-the-art research, and production. Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors. Keras models are made by connecting configurable building blocks together, with few restrictions.Keras writes custom building blocks to express new ideas for research and creates new layers, metrics, loss functions, and develops state-of-the-art models.

Convolution 2d

2D convolutional layers take a three-dimensional input, typically an image with three color channels. They pass a filter, also called a convolution kernel, over the image, inspecting a small window of pixels at a time, for example 33 or 55 pixels in size, and moving the window until they have scanned the entire image. The convolution operation calculates the dot product of the pixel values in the current filter window with the weights defined in the filter.
Parameters:
filters: Integer, the dimensionality of the output space
strides: An integer or tuple or list of 2 integers, specifying the strides of the

convolution along the height and width. It can be a single integer to specify the same value for all spatial dimensions.

Flatten

Flattening a tensor means to remove all of the dimensions except for one. This is exactly what the Flatten layer do. Flatten make explicit how you serialize a multidimensional tensor (tipically the input one). This allows the mapping between the (flattened) input tensor and the first hidden layer. If the first hidden layer is "dense" each element of the (serialized) input tensor will be connected with each element of the hidden array. If you do not use Flatten, the way the input tensor is mapped onto the first hidden layer would be ambiguous.

Dense

A dense layer thus is used to change the dimensions of your vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector. A dense layer is a classic fully connected neural network layer : each input node is connected to each output node. "model.add(Dense(32,input_dim=16))"
16 is column numbers ,it must take that as input data (well python counts from 0 by the way).
Then right after this "Dense" comes "32" , this 32 is classes you want to categorize your data.

Dropout

Dropout is a a technique used to tackle Overfitting . The Dropout method in keras.layers module takes in a float between 0 and 1, which is the fraction of the neurons to drop.A dense layer is a classic fully connected neural network layer : each input node is connected to each output node.
keras.layers.Dropout(rate, noise_shape=None, seed=None)
Rate: the parameter p which determines the odds of dropping out neurons.
Noise shape: if you wish to share noise across one of (batch, timesteps, features), you can set the noise shape for this purpose. Seed: if you wish to fixate the pseudo-random generator that determines whether the Bernoulli variables are 1 or 0.

Maxpooling

A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer. The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map.

The results are down sampled or pooled feature maps that highlight the most present feature in the patch. The window is shifted by strides in each dimension.

Zeropadding

This layer can add rows and columns of zeros at the top, bottom, left and right side of an image tensor.
padding: Int, or tuple of 2 ints, or tuple of 2 tuples of 2 ints.
If int: the same symmetric padding is applied to height and width.
If tuple of 2 ints: interpreted as two different symmetric padding values for height and width: (symmetric_height_pad, symmetric_width_pad).

Activation Function

Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a mode and have a major effect on the neural network's ability to converge and the convergence speed. Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated or not, based on whether each neuron's input is relevant for the model's prediction. It also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer. The activation function is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

Some of the activation functions used are:

- SIGMOID
  - Smooth gradient, preventing "jumps" in output values.
  - Output values bound between 0 and 1, normalizing the output of each neuron.
  - Clear predictions - For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.
  - Vanishing gradient - for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
  - Outputs not zero centered.
  - Computationally expensive

- TANH
  - Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
  - Otherwise like the Sigmoid function.
- RELU
  - Computationally efficient—allows the network to converge very quickly
  - Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation
  - The Dying ReLU problem—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.
- SOFTMAX
  - Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.
  - Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

LSTM

A LSTM network is a kind of recurrent neural network. A recurrent neural network is a neural network that attempts to model time or sequence dependent behaviour. This is performed by feeding back the output of a neural network layer at time t to the input of the same network layer at time t + 1.

The network has LSTM cell blocks in place of standard neural network layers. The cell is responsible for keeping track of the dependencies between the elements in the input sequence. These cells have various components called the input gate, the forget gate and the output gate. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. There are connections into and out of the LSTM gates, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate.

To specify an LSTM layer, first you have to provide the number of nodes in the hidden layers within the LSTM cell like the number of cells in the forget gate layer, the tanh squashing input layer and so on. The argument return sequences ensures that the LSTM cell returns all of the outputs from the unrolled LSTM cell through time. If this argument is left out, the LSTM cell will simply provide the output of the LSTM cell from the last time step.

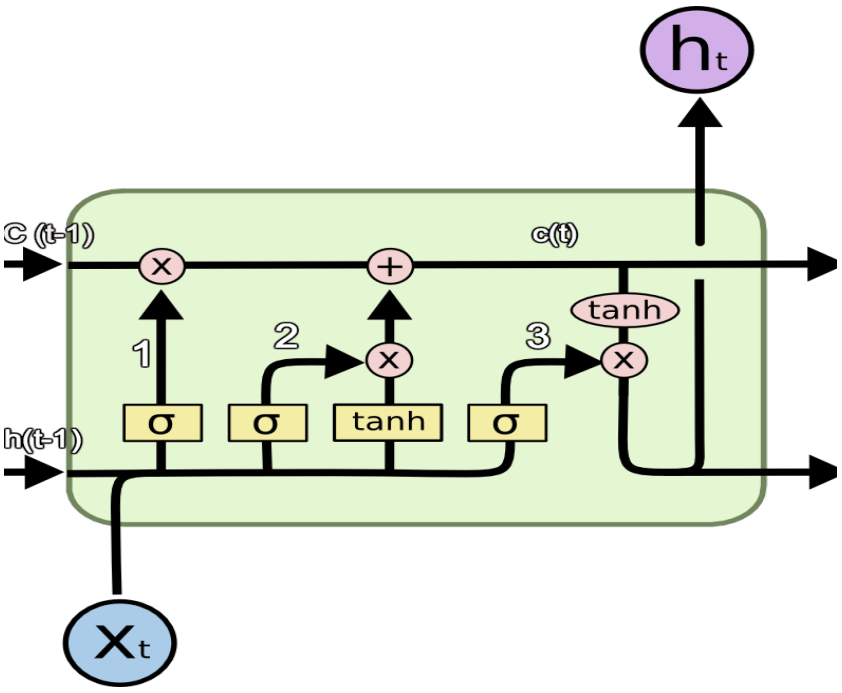


Figure 3.1: LSTM Architecture

The symbols used here have following meaning:

- X : Scaling of information
- + : Adding information
- sigma : Sigmoid layer
- tanh: tanh layer
- h(t-1) : Output of last LSTM unit
- c(t-1) : Memory from last LSTM unit
- X(t) : Current input
- c(t) : New updated memory
- h(t) : Current outpuT

The working of LSTM is as follows:

- LSTM has a special architecture which enables it to forget the unnecessary information .The sigmoid layer takes the input X(t) and h(t-1) and decides which parts from old output should be removed (by outputting a 0). This gate is called forget gate f(t). The output of this gate is f(t)*c(t-1).
- The next step is to decide and store information from the new input X(t) in the cell state. A Sigmoid layer decides which of the new information should be updated or ignored. A tanh layer creates a vector of all the possible values from the new input. These two are multiplied to update the new cell sate. This new memory is then added to old memory c(t-1) to give c(t).

- Finally for deciding what we're going to output, a sigmoid layer decides which parts of the cell state are going to output. Then, the cell state is put through a tanh generating all the possible values and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to. Our model does not learn this answer from the immediate dependency, rather it learnt it from long term dependency.

LSTM outperforms the other models when we want our model to learn from long term dependencies. LSTM's ability to forget, remember and update the information pushes it one step ahead of RNNs.

Chapter 4

Results

4.1 Sample 1

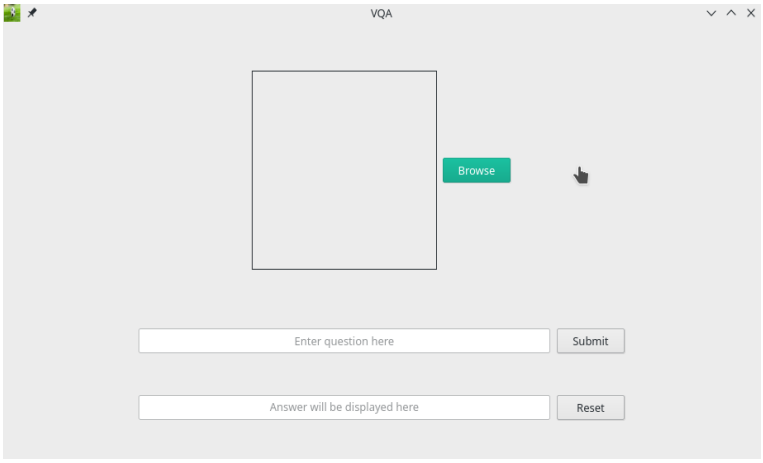


Figure 4.1: UI at initial stage

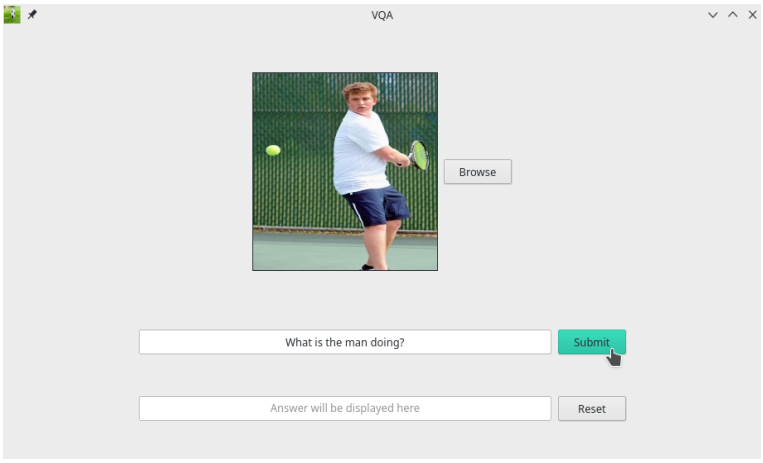


Figure 4.2: After selecting the image and question

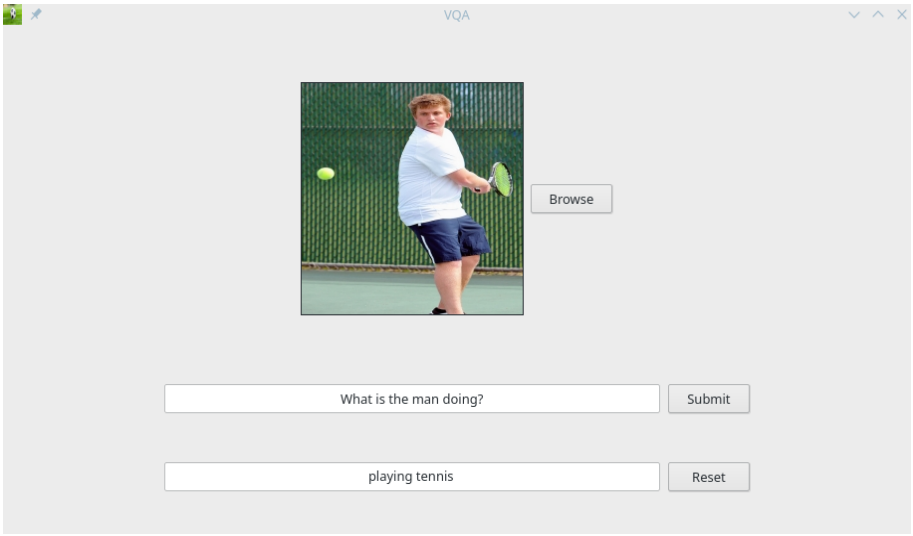
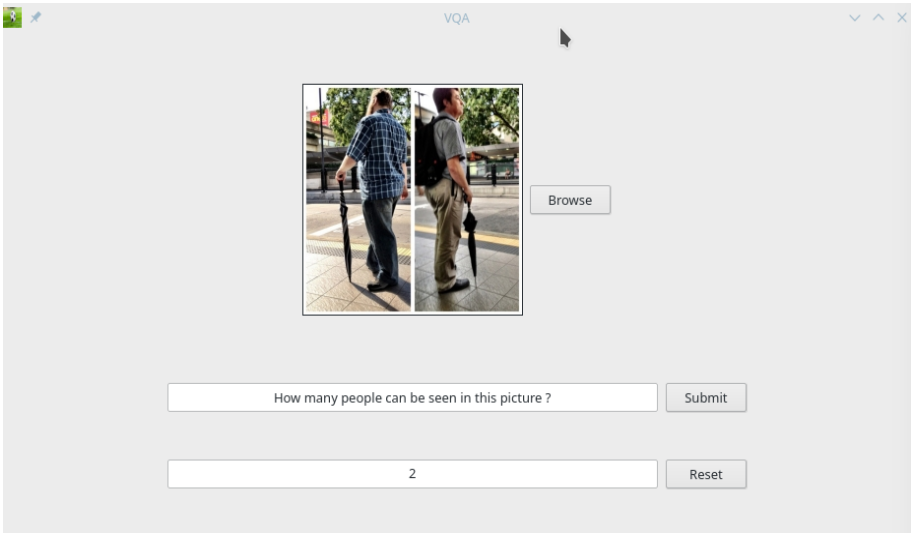


Figure 4.3: Displaying Result

4.2 Sample 2



Chapter 5

Conclusion

In conclusion, we have created a neural network model which can answer questions based on the image given . In general, we can outline the approaches in VQA as follows:
1)Extract features from the question.
2)Extract features from the image.
3)Combine the features to generate an answer.
For text features, Long Short Term Memory (LSTM) encoders are used. In the case of image features, CNNs pre-trained on ImageNet(VGG19) is the most frequent choice.Regarding the generation of the answer, the approaches usually model the problem as a classification task.

Appendices

Appendix A

MAIN CODE

```
import socket
import os, argparse
import cv2, spacy, numpy as np
from keras.models import model_from_json
from keras.optimizers import SGD
from sklearn.externals import joblib
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras import backend as K
K.set_image_data_format('channels_first')
K.common.image_dim_ordering() == 'th'
import time
import warnings


VQA_weights_file_name   = 'models/VQA/VQA_MODEL_WEIGHTS.hdf5'
label_encoder_file_name = 'models/VQA/FULL_labelencoder_trainval.pkl'
CNN_weights_file_name   = 'models/CNN/vgg16_weights.h5'

verbose = 1

def get_image_model(CNN_weights_file_name):

    from models.CNN.VGG import VGG_16
    image_model = VGG_16(CNN_weights_file_name)
    # this is standard VGG 16 without the last two layers

    return image_model

def get_image_features(image_file_name, CNN_weights_file_name):


    image_features = np.zeros((1, 4096))
   im = cv2.resize(cv2.imread(image_file_name), (224, 224))
    mean_pixel = [103.939, 116.779, 123.68]
    im = im.astype(np.float32, copy=False)
    for c in range(3):
        im[:, :, c] = im[:, :, c] - mean_pixel[c]
```

```
    im = im.transpose((2,0,1))
    im = np.expand_dims(im, axis=0)
    image_features[0,:] = get_image_model(CNN_weights_file_name).
    predict(im)
   return image_features

def get_question_features(question):

    word_embeddings = spacy.load('en_vectors_web_lg')
    tokens = word_embeddings(question)
    question_tensor = np.zeros((1, 30, 300))
    for j in range(len(tokens)):
            question_tensor[0,j,:] = tokens[j].vector
    return question_tensor


def get_VQA_model(VQA_weights_file_name):


    from models.VQA.VQA import VQA_MODEL
    vqa_model = VQA_MODEL()
    print(vqa_model.summary())
    vqa_model.load_weights(VQA_weights_file_name)

    vqa_model.compile(loss='categorical_crossentropy',
    optimizer='rmsprop')
    return vqa_model


def main():
    print("Server Listening")
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("127.0.0.1", 6000))
    print("\n\n\nPredicting result ... Hello")
    print("Server Listening")
    s.listen(1)
    while True:
        conn, address = s.accept()  # accept new connection
        start_time = time.time()
        print("Time started")
        print("Connection from: " + str(address))
        data = conn.recv(1024).decode()
        print("from connected user: " + str(data))
        arr=list(map(str,data.split("\n")))
        print(arr)
        image_file_name=arr[0]
        question=arr[1]

        if verbose : print("\n\n\nLoading image features ...")
        image_features = get_image_features(image_file_name,
```

```
        CNN_weights_file_name)

        if verbose : print("Loading question features ...")
        question_features = get_question_features(str(question))
        if verbose : print("Loading VQA Model ...")
        vqa_model = get_VQA_model(VQA_weights_file_name)

       if verbose : print("\n\n\nPredicting result")
        y_output = vqa_model.predict([question_features, image_features]
        print(y_output.shape)
        y_sort_index = np.argsort(y_output)

        labelencoder = joblib.load(label_encoder_file_name)
        for label in reversed(y_sort_index[0,-1:]):
            print(label)
            print(str(round(y_output[0,label]*100,2)).zfill(5), "% ",

            labelencoder.inverse_transform(label))
        res=''
        res=str(labelencoder.inverse_transform(label))

        print(res)
        conn.send(res.encode())

        print("--- %s seconds ---" % (time.time() - start_time))
        conn.close()  # close the connection

if __name__ == "__main__":

    main()
```

Appendix B

VQA Model

```
from keras.models import Sequential
from keras.layers.core import Reshape, Activation, Dropout
from keras.layers import LSTM, Merge, Dense

def VQA_MODEL():
    image_feature_size          = 4096
    word_feature_size           = 300
    number_of_LSTM              = 3
    number_of_hidden_units_LSTM = 512
    max_length_questions        = 30
    number_of_dense_layers      = 3
    number_of_hidden_units      = 1024
    activation_function         = 'tanh'
    dropout_pct                 = 0.5

    # Image model
    model_image = Sequential()
    model_image.add(Reshape((image_feature_size,), input_shape=

    (image_feature_size,)))

    # Language Model
    model_language = Sequential()
    model_language.add(LSTM(number_of_hidden_units_LSTM,

    return_sequences=True, input_shape=(max_length_questions,

    word_feature_size)))
    model_language.add(LSTM(number_of_hidden_units_LSTM,

    return_sequences=True))
    model_language.add(LSTM(number_of_hidden_units_LSTM,

    return_sequences=False))

    # combined model
    model = Sequential()
    model.add(Merge([model_language, model_image], mode='concat',
```

```
    concat_axis=1))

    for _ in range(number_of_dense_layers):
        model.add(Dense(number_of_hidden_units, kernel_initializer=

        'uniform'))
        model.add(Activation(activation_function))
        model.add(Dropout(dropout_pct))
    model.add(Dense(1000))
    model.add(Activation('softmax'))

    return model
```

Bibliography

[1] R. Burt, M. Cudic and J. C. Principe, "Fusing attention with visual question answering," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 949-953, doi: 10.1109/IJCNN.2017.7965954.

[2] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell ,Long-term Recurrent Convolutional Networks for Visual Recognition and Description

[3] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox and A. Farhadi, "IQA: Visual Question Answering in Interactive Environments," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4089-4098, doi: 10.1109/CVPR.2018.00430.

[4] Malinowski, Mateusz, Marcus Rohrbach, and Mario Fritz. "Ask Your Neurons: A Neural-Based Approach to Answering Questions About Images." 2015 IEEE International Conference on Computer Vision (ICCV) (2015): n. pag. Crossref. Web.

[5] Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, Anton van den Hengel,FVQA: Fact-based Visual Question Answering , Volume: 40 , Issue: 10 , Oct. 1 2018

[6] Xu H., Saenko K. (2016) Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9911. Springer, Cham

[7] VQA: Visual Question Answering - Stanislaw Antol and Aishwarya Agrawal and Jiasen Lu and Margaret Mitchell and Dhruv Batra and C. Lawrence Zitnick and Devi Parikh , International Conference on Computer Vision (ICCV), 2015

[8] Kevin J. Shih, Saurabh Singh, Derek Hoiem ,Where To Look: Focus Regions for Visual Question Answering In CVPR, 2016