*Joshua Wong*
*Hughes Hall*
dsjw2

PROJECT PROPOSAL

COMPUTER SCIENCE TRIPOS PART II

# A Visual Question Answering System

October 20, 2017

**Project Originator:** Joshua Wong

**Project Supervisor:** Dr Marek Rei

**Director of Studies:** Dr John Fawcett

**Project Overseers:** Dr Hatice Gunes & Dr Robert Watson

# 1 Introduction and Description of Work

Building systems which are capable of either classifying the content within images or answering natural language questions on its own has been a challenge for some time. However, the past few years have seen strong advancements in these fields, including GoogLeNet's Deep Convolution Neural Network Model for identifying image features[4], and the Long Short-Term Models for understanding and producing sequences of text[1].

A relatively new and exciting field, visual question answering takes this one step further by combining both Natural Language Processing (NLP) and Computer Vision (CV) techniques to perform reasoning about the content of an image in order to answer a question.

Figure 1: Sample of an abstract image from the VQA dataset

The task involves taking as input an image, such as the one in Figure 1, and a question such as "What did the girl kick?" and providing an answer to the question, where for this case the ground truth answer would be "ball".

Visual question answering requires a higher level of reasoning than most of today's common machine learning tasks. For a system to answer the question above, it first needs to recognize from the question what a "girl" is, what it means to "kick" and where all this fits in in the image before it can produce an answer to that question. This makes it an important step towards understanding more general forms of Artificial Intelligence. It also has many real world applications, including helping the visually impaired, providing better personal assistants (e.g. "Which dress is more fitting for the occasion?") and in automated assessments for young learners.

## 1.1 The Project

The aim of the project will be to implement and investigate different models for building a visual question answering system. From a logical perspective, considering the way humans perform such a task, I find that a system performing Visual QA should require components which perform each of these:

1. A way to understand images
2. A way to understand the question
3. Linking the understanding of the question to specific parts of the image
4. A decision process over prior knowledge
5. Answer generation

There have been a number of recent papers published using neural network architectures to tackle the task of visual question answering. However, most of these only incorporate some portion of the list above. For example, BOWIMG[7] fulfills 1, 4 and 5, but barely utilizes 2, and completely excludes 3. On the other hand, Ask Your Neuron[3] fulfills 1, 2, 4 and 5, but lacks rigour in 3.

My goal for the project will be to investigate ways of incorporating all 5 points mentioned above. The core of this project will begin with a baseline model following [7], and proceed towards a comparative study of additional approaches for building the system, including the use of LSTMs in embedding the question and combining this with a CNN for understanding the image. Further approaches which incorporate all the points above including 3, will be explored as extensions to this project, through the investigation of attention-based networks[2].

This project will not simply be replicative of those mentioned in the papers, since most of those have been built using Lua and Torch[1], whereas this project aims to take an approach which allows for easier integration into a website or mobile application, and will implement these models using Python and TensorFlow[2]. Although I will be borrowing the high level ideas from these papers, I will perform my own optimizations, evaluation and build upon these architectures. These will be trained and evaluated on the VQA dataset[3].

### 1.2 Disambiguation

In order to reduce confusion between VQA the task and VQA the dataset (since both are equivalently named), I will use the acronym VQA to refer to the dataset, while Visual QA will be used to refer to the task of visual question answering.

## 2 Starting point

In my previous internship at Amazon (Alexa), I have worked on applications using Natural Language Processing (NLP) for text generation and the use of Knowledge Bases (KBs) for textual question and answering.

The libraries and tools which I will use for the project are covered in section 5.2.

[1] http://torch.ch/
[2] https://www.tensorflow.org/
[3] http://visualqa.org/download.html

The following courses in the Computer Science Tripos will be useful for the project:

- **Artificial Intelligence I** - Introduction of the foundations of neural network architectures, knowledge representation and automated reasoning.
- **Machine Learning and Bayesian Inference** - Supervised learning techniques and further depth for neural network architectures
- **Natural Language Processing** - Natural language embedding techniques and algorithms for text processing and generation
- **Computer Vision** - Algorithms for processing images
- **Software Engineering** - Practices for managing large scale projects

A number of these courses will not occur until much later in the year. Furthermore, aspects such as LSTMs, attention-based networks and web applications are not covered in the course. In order to gain a deeper understanding of how to implement this project, I will fill in this gap by referring to online courses, academic papers and textbooks.

# 3 Substance and Structure of the Project

This project will be broken up into 3 phases - preparation, implementation and evaluation.

## 3.1 Preparation

This phase will consist mainly of self-study and research to attain the required in-depth knowledge of how to implement the specified architectures. This includes doing the following:

- Get familiar with writing python
- Udacity's Online Deep Learning Course by Google[4]
- Read papers on the following:
  - The Visual Question Answering task
  - VQA Baseline Models
  - LSTMs and CNNs
  - Attention-based architectures
  - Knowledge Bases
  - Neural Module Networks
- Tensorflow tutorial
- Fill out any further gaps in knowledge by referring to the Deep Learning textbook[5]
- Normalise the data and prepare a skeleton program for reading and using it
- Split the dataset into training, validation and test sets

[4]https://www.udacity.com/course/deep-learning–ud730
[5]http://www.deeplearningbook.org/

## 3.2 Implementation

This phase forms up the core building of the Visual QA system. The input of the model will be the question in string format and the image as downloaded in png format. The answer output will be in string format, either selected from multiple-choice answers or open-ended generated text.

Since Visual QA is a relatively new task which was proposed only after deep learning showed state-of-the-art results in computer vision and NLP, most of the work published involve the use of deep learning approaches. Similarly, the models used in this project will focus on deep learning architectures, where there have been the most promising results.

### 3.2.1 Architecture Components

This section describes the fundamental components which will form up the larger architectures in the proposed models in section 3.2.2 and why they will be used.

#### The Feedforward Neural Network

The feedforward neural network, or multilayer perceptron, is a machine learning algorithm with a goal of approximating some function which maps instances to labels. They also form the basis for a number of the next few components. Since these have been covered in Part IB's AI course, this section will not go into detail for explaining them, but will state the differences in what was taught.

The feedforward networks used in this project will follow a similar structure and mainly be trained with stochastic gradient descent using the backpropagation algorithm. However, for most their output units, instead of the standard sigmoid or regression, it will make use of the softmax function since the Visual QA task involves either multiple-choice or open-ended answers, which would be discrete variables with multiple possible values. A linear layer first computes the unnormalised probabilities in equation 1.

$$\mathbf{z} = \mathbf{W}^\top \mathbf{X} + \mathbf{b} \tag{1}$$

W refers to the weights, while b is the bias and X is the input. The softmax function in equation 2 will take its output, z, and turn it into proper normalized probabilities which sum to 1.

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{2}$$

It will make the probability large when the value is large, and all other probabilities will be small when they are comparatively smaller. This is useful for selection in the Visual QA task in choosing an answer over multiple values.

The softmax function itself will also be used as the output unit for the BOW + IMG model in section 3.2.2.1.

### The Convolutional Neural Network (CNN)

CNNs will be used as the image processing component for the Visual QA system. They are similar to regular feedforward neural networks, except that they use convolution in place of the typical matrix multiplication in at least one of their layers. They are best at processing data with a grid-like topology. The convolution function used here is similar to that used in Mathematical Methods in Part IB, as seen in equation 3.

$$f(t) = \int x(a)k(t-a)da \tag{3}$$

However, CNNs for the Visual QA task will be used for processing discrete 2D images, hence will use convolutions over more than one axis and will be a summation instead of an integration, as shown in equation 4.

$$F(i,j) = \sum_m \sum_n X(m,n)K(i-m, j-n) \tag{4}$$

X refers to the input, which can be an image taken in as a 2-D array (also called tensor) of its pixels, or an input from an earlier layer. K refers to the kernel, which in this case would also be a 2-D array of parameters adapted from the algorithm used. The kernel here is often much smaller than the image. For example, an image may have on the order of tens of thousands of pixels, while the kernel may only be a hundred parameters. For the CNNs in this project, the kernel in equation 4 may also be flipped relative to the input, which makes it easier for implementation (due to a smaller range of values) while still maintaining its use due to its commutative property.

Each CNN layer in the architectures used will consist mainly of 3 stages (which are repeated multiple times for multiple layers):

1. Convolution stage, where the mentioned convolution operation is done on the input
2. Rectified linear activation stage, where the output of the convolution will be passed through a nonlinear activation function to increase the nonlinear properties of the decision function
3. Pooling stage, which provides an overall summarizing value of a region of nearby outputs

Most of the CNN implementations in this project would follow the structure in [4], using the max pooling approach for the pooling stage, which provides the maximum value from a cluster in the previous stage. This gives the important property of invariance, which makes it insensitive to small shifts such as rotations and translations of the image, and is important for identifying the same objects in different VQA inputs.

For processing images in the Visual QA task, CNNs will be the architecture of choice over feedforward networks. The convolution operation makes the CNN strong at processing images. Since the kernel is significantly smaller than the input, it has the benefit of sparse connectivity.

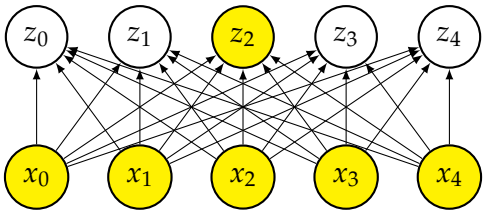


Figure 2: A normal feedforward network

In a regular feedforward network, every output node, such as $z_2$, is affected by every input node, as seen by the shaded nodes in figure 2.

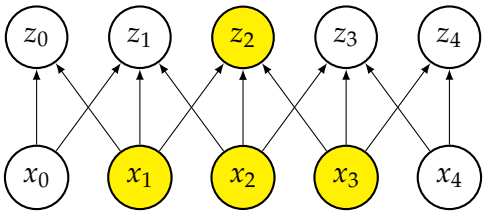


Figure 3: Sparse connectivity in a CNN

However, for a CNN layer (with a kernel width of 3 in this case), each output node is only affected by 3 input nodes, as seen in figure 3. This reduces memory requirements and increases computational efficiency, while allowing complex interactions between different pixels of an image to still be described. Furthermore, a CNN employs parameter sharing. While in regular feedforward networks every parameter is only used once, the smaller kernel means that its set of parameters only need to be learned once and is applied to all parts of the image. This makes the training phase require much less memory and is more efficient at describing transformations which apply across the entire image.

The fact that convolution has equivariance, where changes in the input will affect the output in the same way, allows CNNs to efficiently map the region of features in the 2-D input to the 2-D output. This will be essential when computing attention over specific parts of the image in section 3.3. For these reasons, CNNs will be chosen as a means of understanding the image in this task.

## The Recurrent Neural Network (RNN)

RNNs are similar to feedforward networks, except that loops occur on some nodes, as shown in figure 4. They are specialized for taking in sequences of input values, as each increment on t can process a new chunk of input.

These loops allow information taken from one time step to persist to future states, which is important when dealing with a question with different parts relating to each other. However, regular RNNs face the problem of long term dependencies, and would not do well in learning questions where the context and subject may be far apart. For example, a question such as, “There is a sad person in the picture. Is this person male or female?” has the context of a “sad person” being a distance away from the subject of whether the person is male or female. In practice, an RNN has been shown as unable to handle such

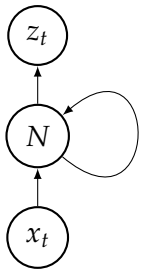


Figure 4: A node in a Recurrent Neural Network

long-term dependencies well. As such, the project will instead use a different variation of the RNN, called the Long Short-Term Model (LSTM) for this task.

## The Long Short-Term Memory Model (LSTM)

The LSTM is similar in structure to the RNN above, except that each recurrent node has the structure shown in fig 5. By default, they specialize in being able to learn long-term

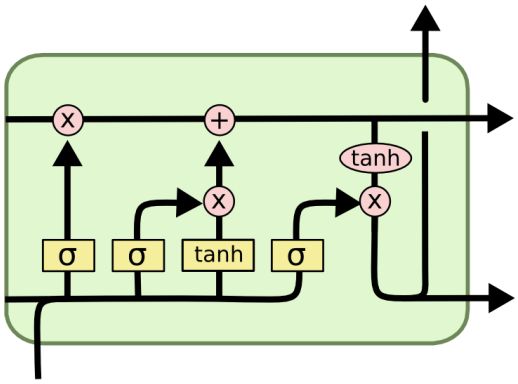


Figure 5: An LSTM node[6]

dependencies[1]. The rectangular nodes here represent neural network layers while the X represents the pointwise multiplication operation. The combination of the 2 is called a gate, which controls how much information to let through and what information to forget. This allows much longer term dependencies to be learned.

Since the questions in the VQA dataset are a sequence of text, often with multiple subjects in the question, they are best handled by an LSTM. These can be used to process the question by taking in a word at each time step.

### 3.2.2 Visual QA Models

The models described below will be implemented and investigated as part of the core project and evaluated on the VQA dataset. Although a number of different models to be investigated are listed here, the project will also look at optimization and regularization techniques within each of these architectures.

[5]Image from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

#### 3.2.2.1 BOW+IMG Model

This model takes the approach of embedding the question using a single Bag-Of-Words (BOWs) vector. For the image, GoogLeNet's pre-trained model will be used to extract its features and transform it into a feature vector. These 2 vectors will be concatenated and fed into a softmax layer to predict the answer class. This structure is shown in figure 6. It is also used as a baseline model.

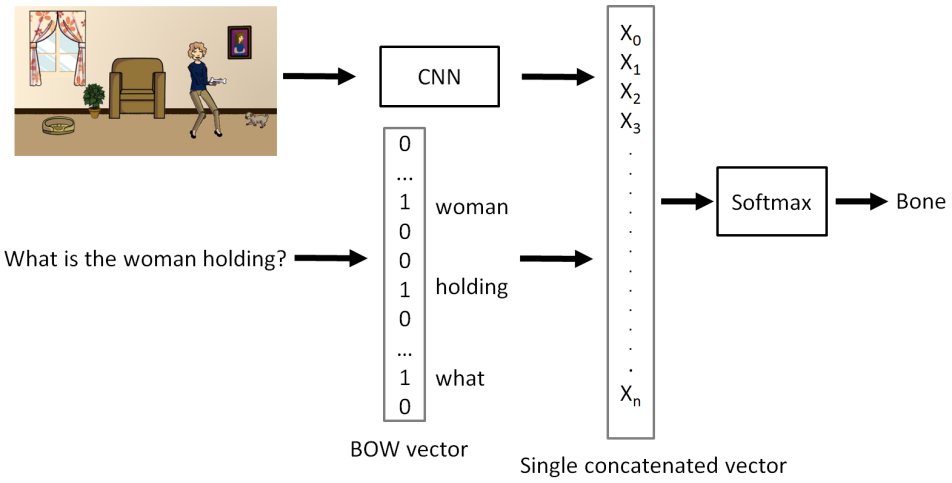


Figure 6: Architecture of the BOW+IMG model

Making use of BOWs to embed the question ignores the structure of the question, but it can still identify its main topics. GoogLeNet's model is trained on general images and not on the Visual QA task, hence it is not expected to provide an ideal performance. However, both of these provide a quick working model as a starting implementation which can be investigated and improved.

### 3.2.3 LSTM+IMG Model

This model will also use GoogLeNet for extracting the image features. However, instead of using the BOW model to input the question as a sequence, it will use an LSTM. The input at each time step of the LSTM will be one word of the question, concatenated with the encoded image vector, as shown in figure 7. The LSTM blocks in the figure show how the nodes look when they are rolled out, taking in one word each time together with the image vector. To produce an answer, the model is used to predict a sequence of output over a vocabulary, which is shown by the softmax layer after the full sequence is processed. This model will follow an architecture similar to [3]. As a start, this vocabulary will consist of the 1000 most frequent answers in the VQA dataset. Further approaches for deciding this vocabulary will also be investigated in the extended models.

As described in section 3.2.1, LSTMs are the state-of-the-art algorithms for processing natural language, and their strengths make them a reasonable choice for use in understanding the question text. This model aims to utilise the LSTM and image features, combined with a softmax layer, to understand the question, image and produce an answer from both.

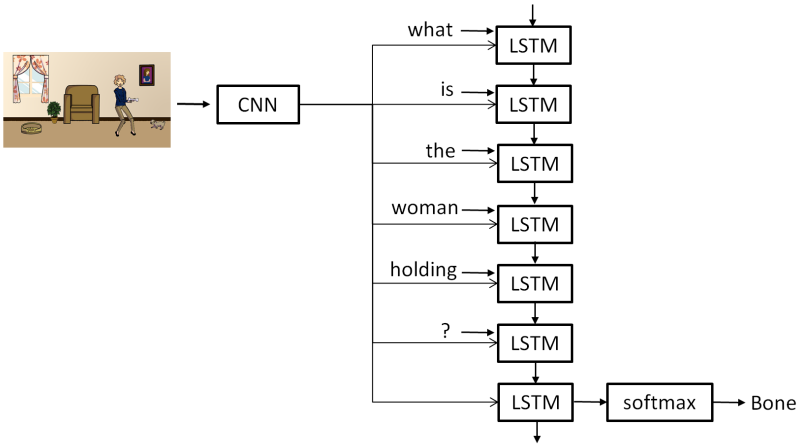


Figure 7: Architecture of the LSTM+IMG model

This architecture can be further built upon by experimenting with the depth and width of the layers, as well as using regularization techniques to reduce overfitting.

## 3.3 Extensions

The following extensions will be implemented if time permits, but are not crucial to the core of the project.

### 3.3.1 LSTM+CNN trained on VQA

All CNN models in the core project till this point have used a pre-trained model on ImageNet. While these are great at general classification, they have not been tuned towards the Visual QA task. This extension will implement its own CNN architecture and train it on the ImageNet dataset[7]. Next, it will randomly initialize and train the last layer of the CNN model together with the LSTM on the VQA task. This will make use of the same model structure as 3.2.3, but with its own CNN trained on the VQA task together with the LSTM, instead of training the LSTM separately.

### 3.3.2 Attention over the image

Standard neural network architectures often fail to produce precise answers if they require looking at a fine-grained region of an image. A higher level of reasoning is required for many of the VQA questions in order to pinpoint relevant parts of an image which are related to the question. Making use of Stacked Attention Networks (SAN) greatly improves this aspect[6].

SANs are generally made up of 3 components. The first is a method for image encoding. My implementation will make use of the CNN model trained in section 3.3.1, and the image features will be extracted from its last pooling layer so that the image spatial information is retained. The next component is the question model. My implementation will make use of the same LSTM architecture used in the previous section.

[7]http://www.image-net.org/

The output from these 2 components will be an image feature matrix, $V_i$ and a question feature vector, $V_q$. Both of these are fed into a single layer neural network and then through a softmax layer. Up to this point, the architecture can be described by figure 8.

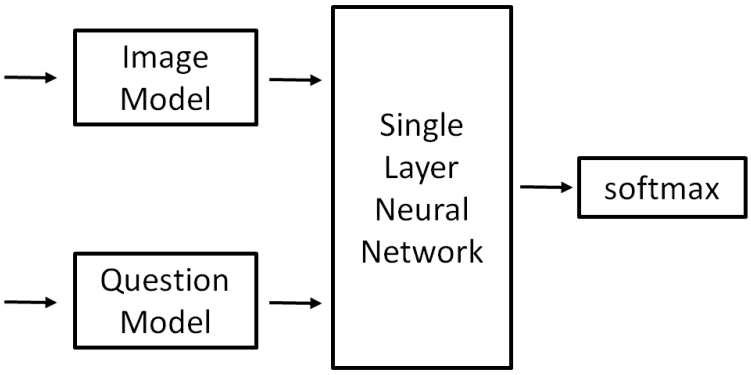


Figure 8: Computing the first attention distribution

The output of the softmax layer is an attention distribution, which is used to calculate the weighted sum of the initial image vectors, $V_i$. By combining this with $V_q$, we attain a new query vector U. This process of computing a new query vector can be iterated multiple times to enhance the attention over an image. My implementation will calculate the attention vectors twice before feeding it into a softmax layer to produce an answer to the question.

### 3.3.3 Attention over the question

Just as it is important to pinpoint parts of the image to give importance to, giving attention to parts of the question can be equally as important[2]. This architecture builds upon the previous section by making use of question attention on top of image attention. The main structure of the network is similar to the previous sections, except

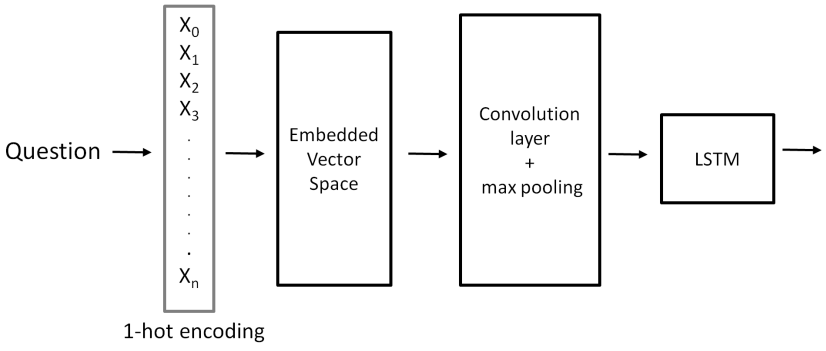


Figure 9: Question Encoding for the Question Attention Network

that the encoding of the question is done through the architecture shown in Figure 9, which begins by attaining word level features from the embedded vector space, before a 1-D convolution is applied to compute phrase level features, and subsequently question level features from the LSTM.

At each step of this hierarchy, the attention weight, $\mathbf{H}$, of a feature X, is computed through equation 5 and 6.

$$\mathbf{H} = tanh(\mathbf{WX} + (\mathbf{Wg})1^{\top}) \tag{5}$$

$$\mathbf{a}^{x} = softmax(\mathbf{w}^{\top}H) \tag{6}$$

W refers to the weights, while 1 refers to a vector with all elements as 1. g refers to an intermediate attention matrix, since this approach of computing attention over the question can be iterated together with computing attention over the image multiple times.

Once attention is computed, a feedforward network will be used to encode the attention features and the softmax output units will be used to produce an answer.

#### 3.3.4 Building a web interface.

This will be done using the Flask framework which calls one of the implemented models. The model will be pre-trained offline before it is uploaded onto a server. A Graphical User Inferface (GUI) will allow anyone using the site to ask questions on displayed images, which will be sent as input to the model, and an answer will be returned.

### 3.4 Evaluation

The VQA datasets provide fully labelled training and validation sets, containing 248,349 and 121,512 questions respectively for the real images set. The testing procedures in this project will follow that in [6] to allow a comparison with their result: the validation set will be equally split into Val1 and Val2. The training set and Val1 will be used for training and validating the models in this project. Val2 will be used for testing the models and for providing an in-depth analysis. Each answer from the dataset and the model will be normalised (e.g. all lowercase, no punctuation) and checked for equality to get a result on correct answers.

The dataset includes two types of images, abstract and real-world, and two types of question formats, multiple-choice and open-ended. The evaluation will focus on using MSCOCO images, which are the real-world images.

If time permits, the Abstract image set will also be used for an additional level of evaluation. Abstract images reduce the need to handle noise, so the models will focus more heavily on higher level reasoning. A varied performance between models on abstract and real-world datasets can help to distinguish between result improvements due to noise handling or improved reasoning. Multiple choice and open-ended questions can also help to evaluate the generative parts of the model. Furthermore, the questions can be clustered into types such as "What" or "Why" questions, and differences in the performance on different question types will aid in directing further improvements, such as whether the question type might require prior knowledge.

For the multiple-choice questions, the models will be evaluated on the percentage of answers which they are able to get correct. For the open-ended answers, they will also be evaluated on percentage of answers which are equal to the ground truth answer, but on

top of this, the similarity of the answer produced to the ground truth answer will also be taken into account using the word2vec similarity score.

### 3.5 Dissertation Writeup

In order to ensure the results and challenges faced in the project are adequately remembered and documented, the writing of the dissertation and implementation of the project will happen concurrently. Early drafts of the dissertation will be written in point form for the purpose of quick recording. As the project nears completion, this will be expanded upon and formalised.

## 4 Success Criteria

### 4.1 Main Project

The project will be evaluated upon the following success criteria:

1. Implement a Visual QA Model which embeds its question input using BOW and a pre-trained model to extract image features, and is able to produce a text answer to each question in the test set.

2. Implement a Visual QA Model which uses an LSTM for embedding the question, and a pre-trained CNN to extract image features, and is able to produce a text answer to each question in the test set.

3. An evaluation done between 1) and 2) which produces comparative statistics on their accuracy and percentage of correct answers on the test set.

### 4.2 Extensions

For the extensions, the following success criteria apply:

1. Train a CNN and implement a model using it which can produce a text answer to each question in the test set.

2. Implement a model which uses attention over an image to produce a text answer for each question in the test set.

3. Implement a model which uses attention over an image and question to produce a text answer for each question in the test set.

4. Create a web interface which can take an image and question query, call one of these models and return its answer.

## 5 Resource Required

### 5.1 Machines

I will use my personal laptop, a Dell XPS 13 (9343) with 2.4GHz Intel Core i7 CPU and 8GB RAM, dual booted with Ubuntu and Windows 8.1, for the development of this

project. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

In order to protect against hardware and/or software failures, the following measures will be taken:

- All code will be regularly pushed into my project repository on Github for revision control.
- All relevant documents and code will be regularly synchronised, as well as manually uploaded, to my Google Drive and Dropbox.
- An external hard disk will also be used to regularly back up all project files.

For training of the models, the University's High Performance Computing Service will be used. I have contacted my supervisor, Dr Marek Rei, for this and the relevant permission for using this has been granted.

## 5.2 Tools

**Tensorflow:** an open source software library for machine intelligence, to implement the neural network architectures. While building neural network architectures from scratch is possible, the aim of the project is to investigate architectures for building a Visual QA system, rather than on constructing neural networks. Thus, Tensorflow will be used for this. It is also commonly used in industry and distributes computations well to Graphics Processing Units(GPUs), and hence I would like to learn more about it.

**NLTK**[8]**:** A Natural Language Toolkit. It will be used for tasks such as tokenization in preparing the question datasets for Tensorflow.

**word2vec**[9]**:** A model used to produce word embeddings. It will be used for calculating the similarity between words for evaluating open-ended answers.

**GoogLeNet or VGGNet model:** Both of these are Deep CNNs pre-trained on ImageNet[4]. Depending on their performance and whether it is possible to access intermediate layer values, one of these will be used to extract image features for the 2 models in the core project. Caffe[10] may also be used for running these models.

**Flask**[11]**:** A micro web framework for Python. It will be used for building the web interface for the model.

## 5.3 Datasets

The VQA dataset[12] provides a large and labeled corpus of images, questions and the answers to those questions in JSON format. It is publicly available and is free to adapt and share under the Commons Attribution 4.0 International License.

[8] http://www.nltk.org/
[9] https://www.tensorflow.org/tutorials/word2vec
[10] http://caffe.berkeleyvision.org/
[11] http://flask.pocoo.org/
[12] http://visualqa.org/vqa_v1_download.html

Each dataset is downloaded in 3 separate parts:

1. Question set in JSON, contains the question, question ID and image ID
2. Image set containing the image in png format, named with the image ID
3. Annotations set in JSON, containing the question type, answer type, question ID, image ID and 10 ground truth answers.

Image and Question IDs allow questions, images and answers to be mapped to each other. The dataset does not provide any extracted features, just questions in JSON string and images in png.

For real world images, there are 248,349 training questions and 121,512 validation questions. With these, there are 82,783 training images and 40,504 validation images. For abstract scenes, there are 60,000 training questions and 30,000 validation questions. These are accompanied by 20,000 training images and 10,000 validation images. All of these are fully labelled.

Each image has an average of 3 questions about it, and each question has 10 ground truth answers. Since the 10 answers were annotated by Amazon Mechanical Turk (AMT), not all of them may be accurate. For evaluating this project, a majority vote (with tie breaker measures) will be performed on the 10 ground truth answers to attain the most certain answer for checking.

# 6 Timetable and Milestones

The project will be split into two-week sprints. While it is difficult to get an accurate measure of my sprint rate, the milestones and deliverables for each two-week block will be stated as tangibly as possible, so they can be evaluated and adjusted accordingly for the following sprint. Below is my initial estimation of the timetable for this project.

**Sprint 1: Michaelmas week 3-4 (19 October - 1 November)**

**Project Proposal Deadline: 20 October**

* Set up and get familiar with Python and Tensorflow
* Do the online Udacity Deep Learning Course
* Read papers on the visual question answering task and on each of the architectures that I plan on implementing, and refer to the deep learning textbook for how these can be implemented
* Download all datasets to be used and split them into training, validation and test sets

**Milestone:** Python and Tensorflow set up; simple programs written to test it is working; Notes made on the overall structure of the models to implement.

**Sprint 2: Michaelmas week 5-6 (2 November - 15 November)**

* Implement a skeleton structure of the program which can read VQA input
* Implement a component which can convert VQA question input into BoWs
* Implement a component which can extract image features from the VQA image input
* Continue reading up on LSTMs and CNNs for the next architecture

**Milestone:** Skeleton structure of the program written; Working components for BOW and image extraction.

**Sprint 3: Michaelmas week 7-8 (16 November - 29 November)**

* Complete the BOW+IMG model which integrates the previous 3 components and begin training it
* Build a test harness around the starting model and attain first set of results for the baseline
* Begin reading up and experimenting with LSTMs and CNNs

**Milestone:** Training for BOW+IMG model has started.

**Sprint 4: Michaelmas Vacation week 1-2 (30 November - 13 December)**

* Build the LSTM component for the LSTM+CNN model
* Test the BOW+IMG model on the VQA dataset
* Complete the implementation for the LSTM+CNN model and begin training

**Milestone:** Attain a set of results from the BOW+IMG model

**Sprint 5: Michaelmas Vacation week 3-4 (14 December - 27 December)**

* Buffer for any remaining implementation
* Start writing Introduction, Preparation and Conclusion chapters of dissertation

**Milestone:** Draft of 3 chapters prepared; LSTM+IMG model began training

**Sprint 6: Michaelmas Vacation week 5-6 (28 December - 10 January)**

* Test the LSTM+IMG model
* Evaluate the 2 models to get a set of comparative results
* Read and begin experimenting on training CNN models
* Tweak the architecture of the models and conduct tests to investigate ways of improving results

**Milestone:** Attain a set of results from the LSTM+IMG model

**Sprint 7: Michaelmas Vacation / Lent week 1 (11 January - 24 January)**

* Implement and begin training the CNN extension model
* Write a draft of the progress report

**Milestone:** Draft of progress report submitted for feedback; CNN extension model started training

**Sprint 8: Lent week 2-3 (25 January - 7 February)**

**Progress Report Deadline: 2 February**

* Test the CNN extension model and evaluate with the other models
* Read up on attention-models and begin building the attention over images model
* Complete and submit the progress report

**Milestone:** Progress report submitted; Attain set of results from CNN extension model

**Sprint 9: Lent week 4-5 (8 February - 21 February)**

* Complete and begin training for the attention over images model
* Research on attention over text
* Write a draft of implementation and evaluation chapters of dissertation

**Milestone:** Implementation and evaluation chapters draft ready

**Sprint 10: Lent week 6-7 (22 February - 7 March)**

* Begin implementing the attention over question model
* Test the attention over images model

**Milestone:** Attain a set of results for the attention over images model

**Sprint 11: Lent week 8 / Easter Vacation (8 March - 21 March)**

* Complete and begin training the attention over question model
* Begin writing a full draft of dissertation

**Milestone:** Submit introduction and preparation chapters to DOS

**Sprint 12: Easter Vacation week 2-3 (22 March - 4 April)**

* Buffer block for any remaining implementation
* Do a final test and evaluation over all models
* Complete draft of dissertation, submit to DOS and amend accordingly
* Begin on second draft

**Milestone:** Draft submitted to DOS; All code for implementations completed

**Sprint 13: Easter Vacation week 4-5 (5 April - 18 April)**

* Complete second draft, get feedback and amend accordingly
* Revision for written papers

**Milestone:** Second draft completed

**Sprint 14: Easter Vacation / Easter week 1 (19 April - 2 May)**

* Prepare final dissertation draft - get feedback, proofread, amend and iterate
* Submit final draft for feedback
* Revision for written papers

**Milestone:** Final draft completed

**Sprint 15: Easter week 2-3 (3 May - 16 May)**

* Final touching up on any further dissertation comments
* Revision for written papers

**Milestone:** Submit Dissertation

**Deadline: 18 May**

# References


[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] 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*, pages 289–297, 2016.

[3] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, pages 1–9, 2015.

[4] 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*, pages 1–9, 2015.

[5] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4622–4630, 2016.

[6] 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*, pages 21–29, 2016.

[7] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.