# 

**Ain Shams University**  
**Faculty of Computer & Information Sciences**

**Computer Science Department**

Visual question answering system

*This is the cover page, so put your own design, colors*

# 

**July 2018**

**Ain Shams University**  
**Faculty of Computer & Information Sciences**

**Computer Science Department**



Visual question answering system

**By:**

Rabah Mohamed Gamal [cs]

Abdallah Shaaban Elsayed[cs]

Abdullah mahmoud Abdullah [cs]

Abdullah Abdelkader Roshdy [cs]

**Under Supervision of:**

[Dr. Sally Saad]

Computer science Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

[Dr. Ahmed Salah]

Computer science Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

# Acknowledgement

If you want to include thank you notes to any one you should put it here. (The acknowledgement is optional)

# Abstract

The abstract is a one page summary of the whole project including: why the project is needed, what are the main features of the project and what are the final results obtained by the developed system.

It’s the most important page in the whole documentation, it should be the last thing you write.

**Table of Contents**

[i](#_Toc516412287)

[Acknowledgement iii](#_Toc516412289)

[Abstract iii](#_Toc516412290)

[List of Figures vi](#_Toc516412291)

[List of Abbreviations vii](#_Toc516412292)

[1- Introduction 1](#_Toc516412293)

[1.1 Motivation 1](#_Toc516412294)

[1.2 Problem Definition 1](#_Toc516412295)

[1.3 Objective 1](#_Toc516412296)

[1.4 Time Plan 1](#_Toc516412297)

[1.5Document Organization 1](#_Toc516412298)

[2- Background 2](#_Toc516412299)

[3- Analysis and Design 3](#_Toc516412300)

[3.1 System Overview 3](#_Toc516412301)

[3.1.1 System Architecture 3](#_Toc516412302)

[3.1.2System Users 3](#_Toc516412303)

[3.2 System Analysis & Design 3](#_Toc516412304)

[3.2.1 Use Case Diagram 3](#_Toc516412305)

[3.2.2 Class Diagram 3](#_Toc516412306)

[3.2.3 Sequence Diagram 3](#_Toc516412307)

[3.2.4 Database Diagram 4](#_Toc516412308)

[4- Implementation and Testing 5](#_Toc516412309)

[**4.1 Implementation** 5](#_Toc516412310)

[5-User Manual 25](#_Toc516412311)

[6-Conclusion and Future Work 26](#_Toc516412312)

[6.1 Conclusion 26](#_Toc516412313)

[6.2 Future Work 26](#_Toc516412314)

[References 27](#_Toc516412315)

# List of Figures

[**Figure 1- Neural Network general architecture** 4](#_Toc417423219)

Note: Always number your figures and include a caption under each one like this. Then if you update the list above it’ll be updated automatically.

When adding a figure, right click on the image ->insert caption.

After you finish the document, write click on the table and choose update field, then update entire table.



**Figure 1- Neural Network general architecture**

Add list of Tables if you have tables in your text in the same manner

# List of Abbreviations

* **Cbow continuous bag of word**
* **Cnn convolution neural network**
* **Relu rectified linear unit**
* **Rnn recurrent neural network**
* **Lstm Long short term memory**
* **Nlp natural language processing**

# Introduction

## 1.1 Motivation

This section should include why the project is needed. This includes what is happening in the field of the project that motivated the idea.

## 1.2 Problem Definition

What is the specific problem the project is solving?

## 1.3 Objective

What are the objectives of the project and what was done to achieve these objectives?

## 1.4 Time Plan

## 1.5Document Organization

Include a paragraph for each chapter describing what was discussed in this chapter starting with chapter 2.

# Background

This chapter should introduce the following:

* A detailed description of the field of the project.
* All the scientific background related to the project.
* A survey of the work done in the field.
* Description of existing similar systems.
* Description of any technology used: Bluetooth, GPS… etc.

# Analysis and Design

## 3.1 System Overview

### 3.1.1 System Architecture

Include a figure of the system architecture and a description of all modules.

You may add Functional and non-functional requirements section –If needed–

### 3.1.2System Users

1. *Intended Users:*

To whom the system is built, and how each group of users will use the system.

1. *User Characteristics*

What kind of experience or skills are required from the users to be able to operate the project effectively.

## 3.2 System Analysis & Design

### 3.2.1 Use Case Diagram

The use case diagram + fully dressed use cases describing each function of the project if applicable.

### 3.2.2 Class Diagram

The diagram + description of all the main classesif applicable.

### 3.2.3 Sequence Diagram

if applicable

### 3.2.4 Database Diagram

If you are implementing a database include the database schema plus a description of the tables.

# 4- Implementation and Testing

## **4.1 Implementation**

In this chapter, we talk about the implementation phases for each module, you will see a detailed explanation about how the entire deep learning model work for each phase and the intuition behind each phase with some mathematical explanation.

We also mentioned a lot of references that we have used during the implementation phases, so, you can check them if you want to find more details about something not clear in the illustration.

Our model trying merging between natural language processing represented in question hierarchy and convolution neural network using co-attention, we will begin with an explanation of question hierarchy then convolution neural network for an image then co-attention between them.

**4.1.1 Preprocessing**

At the preprocessing, we start by removing the unnecessary characters that have no value during the training process like backslash or comma, also we return some of the abbreviations to its original for example " I 'm " to " I am", the reason for doing this is to make the training faster and to consider just to the words that make difference in the training.

In the second part, we start counting each word in the data-set and assign a unique number for the top thousand words and the other be assigned to one unique number which relevant to "UNK".

For every word in a given question we replace it with its unique id that has been assigned to and finally we pad the sequence to zeros to make every question has the same number of a maximum number of words so the training be more flexible.

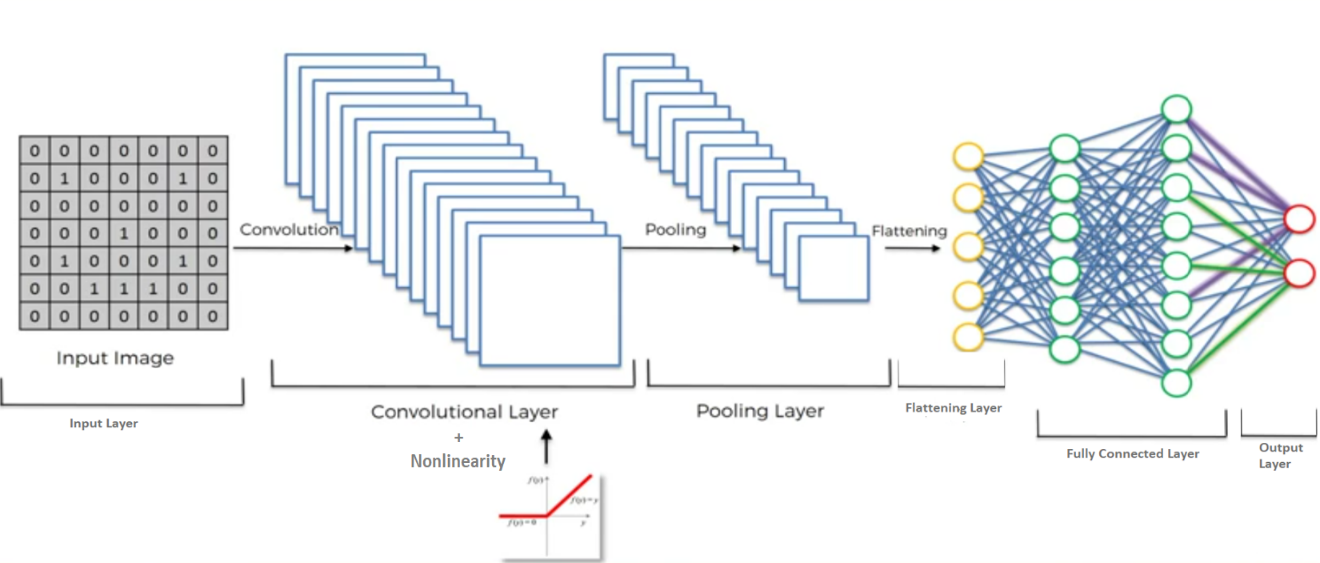
For every answer to a question in the data-set, we replace it with one hot encoding through all category types.

**4.1.2 Convolution neural networks**

In this subsection, we will talk about the convolution neural network, we will talk about its components, architectures, and features extraction.

The convolution neural networks used in our model for extracting the features of an input image.

1. **CNN Components**

****

**Figure 4.1.1.1 - Convolution Neural Networks general architecture**

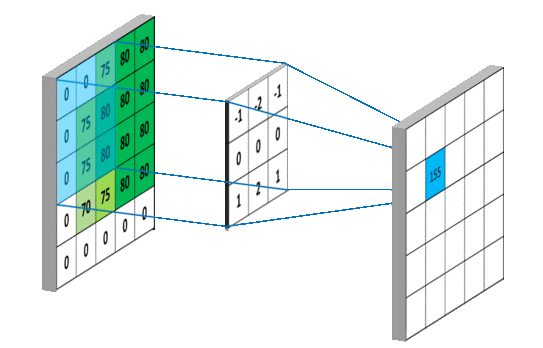
* **Input layer**

The input layer of a neural network is composed of artificial input neurons, and brings the initial data into the system for further processing by subsequent layers of artificial neurons. The input layer is the very beginning of the workflow for the artificial neural network.

* **Convolution Layer**

The primary purpose of convolution is to extract features from the input image.

Convolution preserves the spatial relationship between pixels by learning image features using small squares (filter) of input data.

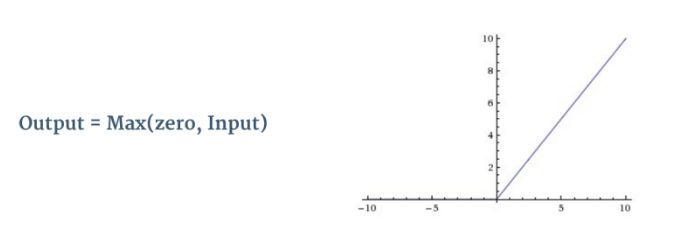


**Figure 4.1.1.2– Convolution Operation**

As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing element-wise multiplications). These multiplications are all summed up So, now you have a single number. Next step would be moving the filter to the right by 1 unit, then right again by 1, and so on. Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that you’re left with a smaller array of numbers, which we call an activation map or feature map.

* **activation function (RELU)**

An additional operation is to add an activation function, one of the most common used is “Relu” has been used after every Convolution operation in **Figure 4.1.1.2** above. Its output is given by:

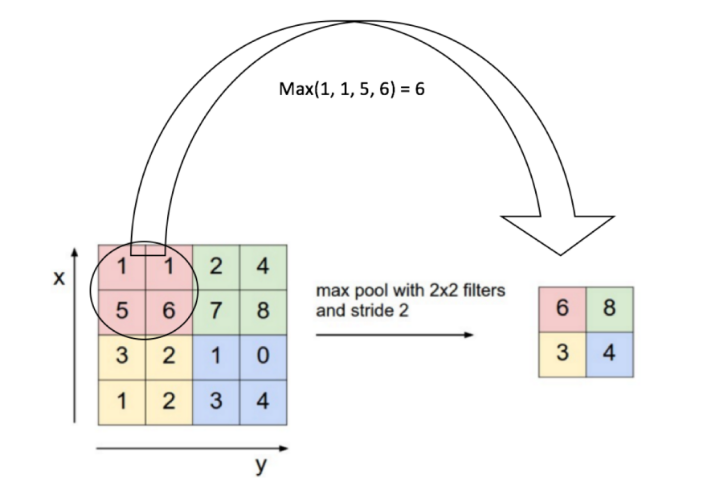


**Figure 4.1.1.3 - the RELU operation**

Relu is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of Relu is to introduce non-linearity, since most of the real-world data would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like Relu ).

* **Pooling layer**

Spatial Pooling (also called sub sampling or down sampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.



**Figure 4.1.1.4- Max Pooling**

combines the outputs of neuron clusters at one layer into a single neuron in the next layer.

* **Flattening Layer**

Convert the output of the convolution part of the CNN into a 1D feature vector, to be used by the ANN part of it. This operation is called flattening. It gets the output of the convolution layers, flattens all its structure to create a single long feature vector to be used by the dense layer for the final classification.

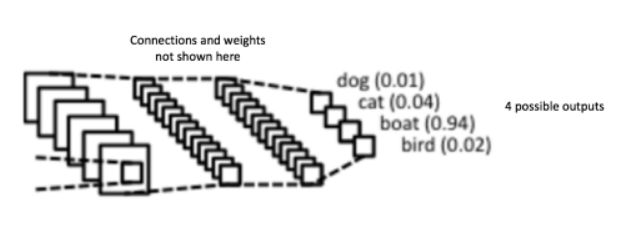
* **Fully connected layer**

The Fully Connected layer is a traditional Multi Layer perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used). The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer

The purpose of the fully connected layer is to use output features of the flattening layer for classifying the input image into various classes based on the training dataset.

* **Output layer**

The last layer of the CNN that has m artificial neurons for m classes, at the end of classification each neuron will have a probability that the input image belongs to its class.

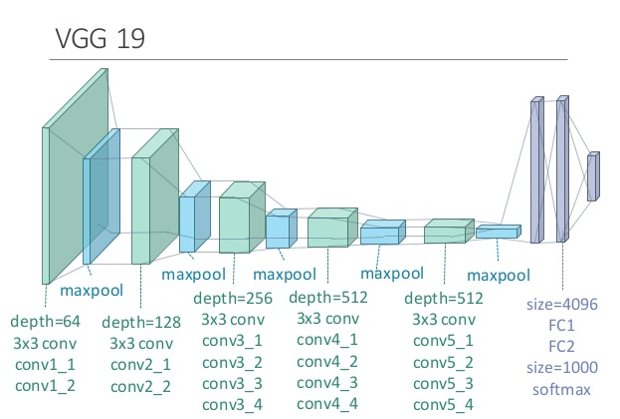


**Figure 4.1.1.5 - fully connected layer and output layer**

The sum of output probabilities from the output Layer is 1. This is ensured by using the [Softmax](http://cs231n.github.io/linear-classify/#softmax) as the activation function in the output layer

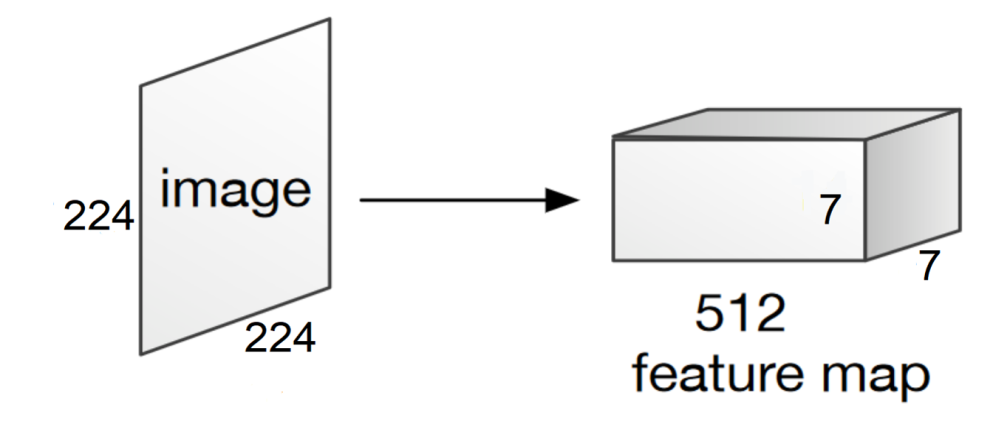
**2) CNN Architectures**

* There is several CNN architectures: Alex Net**,** VGG**,** GoogLeNet**,**ResNet etc.
* In our model we will use **VGG19[1]**

****

**Figure 4.1.1.6 - VGG16**

**3) Features Extraction**

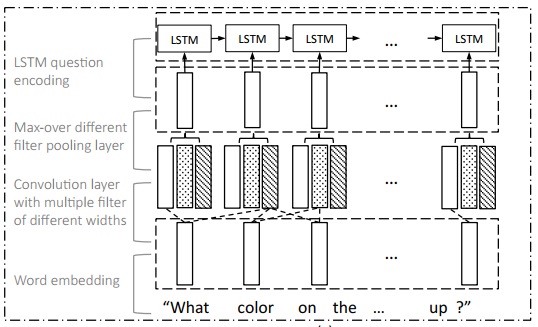
* We resize the input image into **224x224**.
* We extract the features of the input image from the **last pooling layer**.
* The last pooling layer has dimensions of **7x7x512**.

**Figure 4.1.1.7 – output of the last pooling layer**

* We consider that the image has **49 regions** as 7x7 dimension according to the first and the second dimension of the output 7x7x512.
* Each region has **feature vector of length 512** as the third dimension of the output 7x7x512.
* So, we have **49 regions** each one has feature vector of **length 512**.

**4.1.3 Question hierarchy**

Question hierarchy is an model architecture that co-attends to the image and question at three levels: word level, phrase level and question level.



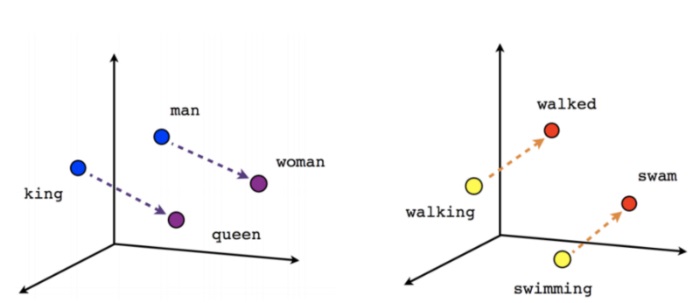
**Figure 4.1.2.1– Question hierarchy**

At the word level, we embed the words to a vector space through an embedding matrix, at the phrase level, one dimensional convolution neural networks are used to capture the information contained in unigrams, bigrams, and trigrams, specifically, we convolve word representations with temporal filters of varying support and then combine the various n-gram responses by pooling them into a single phrase level representation, at the question level, we use recurrent neural networks to encode the entire question. For each level of the question representation in this hierarchy, we construct joint question and image co-attention maps, which are then combined recursively to ultimately predict a distribution over the answers.

1. **Word level**

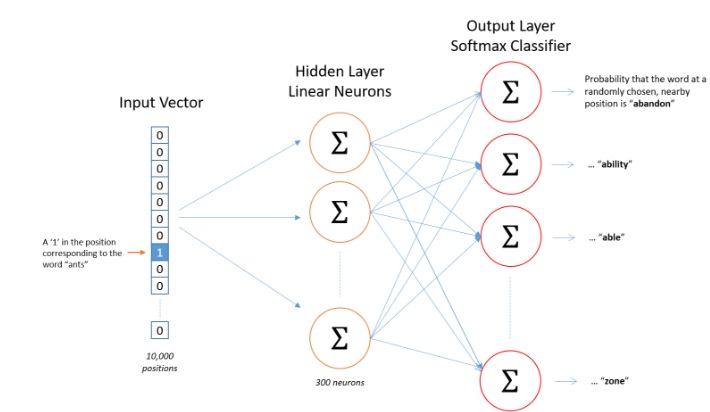
At the word level, we apply one hot encoding for the most frequent words in a given data-set which is a traditional way of representing words, a vector with only one target element being 1 and the others being 0, then generate word embedding vector for each word.

Word embedding is a learned representation of text where words that have the same meaning have a similar representation, trained over corpus "large amount of text" to find the similarity between words using conditional probability p(w1|w2).



**Figure 4.1.2.2– word embedding**

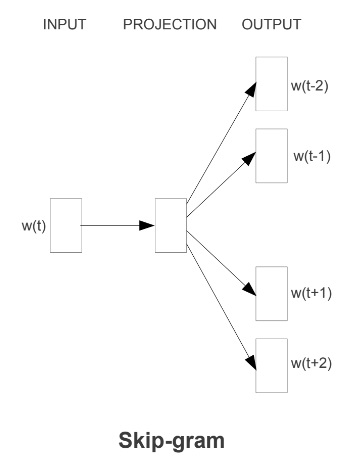
There are many word embeddings training algorithms like “Skip gram model and Cbow” each of them use a shallow neural network with one hidden layer for training.



**Figure 4.1.2.3– Question hierarchy**

* **skip gram**

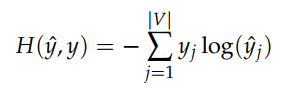
Skip gram is a word embedding model which pick a word from a sentence to be an input word then pick a random word from neighboring fixed size window to be target word, then neural network starts to train to begin to find the similarity of each two words.



**Figure 4.1.2.4– skip gram model**

The number of neurons in the hidden layer is the number of features that we want to generate for a given data-set, so we train the neural network for generating the optimized weight for the hidden layer.

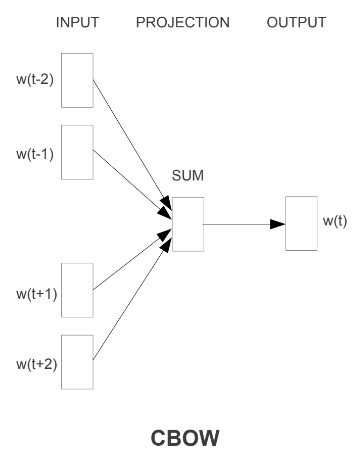
At the output layer we apply softmax function which assigns a probability for each word in the corpus given an input word using cross entropy loss function



**Figure 4.1.2.5– cross entropy**

* **continuous bag of words (Cbow)**

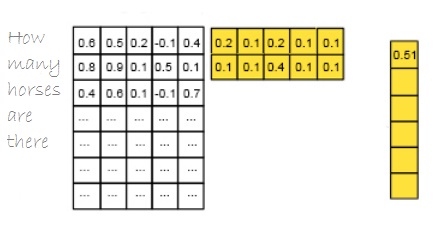
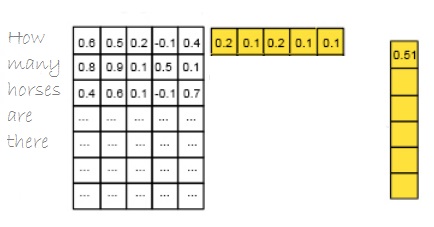
On the other side, the CBOW model aim is to fill in the missing word given its neighboring context, for example, "The cat \_\_ on the mat", the neural network will try to find the highest probability for words to fill the missing word.



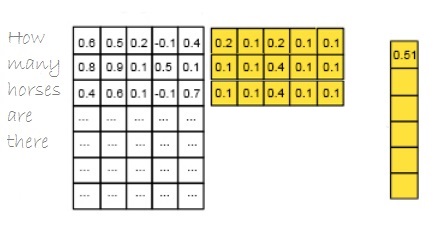
**Figure 4.1.2.6– cbow model**

1. **Phrase level**

At the phrase features, we apply 1-D convolution on the word embedding vectors. Concretely, at each word location, we compute the inner product of the word vectors with filters of three window sizes: unigram, bigram and trigram.



1. (b)



(c)

**Figure 4.1.2.7– (a) unigram , (b) bi gram , (c) tri gram**

For the *t*-th word, the convolution output with window size *s* is given by

ph.jpg

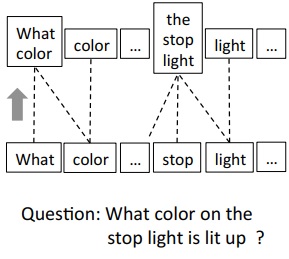
**Figure 4.1.2.8– nlp convolution**

The word-level features are appropriately 0-padded before feeding into bigram and trigram convolutions to maintain the length of the sequence after convolution, given the convolution result, we then apply max-pooling across different n-grams at each word location to obtain phrase-level features.

**ph.jpg**

**Figure 4.1.2.9– max pooling over text**

The reason for applying phrase level is to pick the most important phrase features from the given question, for each word we obtain the most important feature according to its neighboring word, for example the best representation for the word “what” is “what color" and the word “stop" is "the stop light" .

****

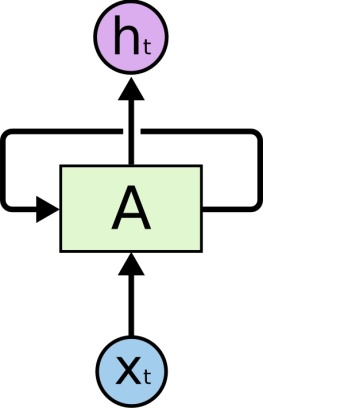
**Figure 4.1.2.10– phrase level**

The output of this level will be the input for the last level in question hierarchy " lstm recurrent neural network".

1. **Question level**

At the last step of question hierarchy, we encode the features generated from phrase level using lstm recurrent neural network.

Recurrent neural network are popular model that have shown great promise in many nlp tasks, The idea behind RNN is to make use of sequential information. In a traditional neural network, we assume that all inputs are independent of each other, for the given question word's many words depend on the previous word. If you want to predict the next word in a sentence you better know which words came before it. RNN are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.



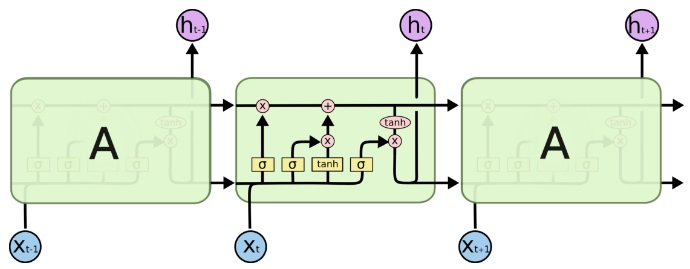
**Figure 4.1.2.11– recurrent neural network**

Vanishing gradient problem is one of the disadvantages of the recurrent neural network, for a given long sentence during the training process if the weight parameter bigger than zero the gradient will explode on the other side if it smaller than zero it will vanish,so the solution for this problem is using lstm recurrent neural network.

* **Long Short Term Memory** “ **Lstm ”**

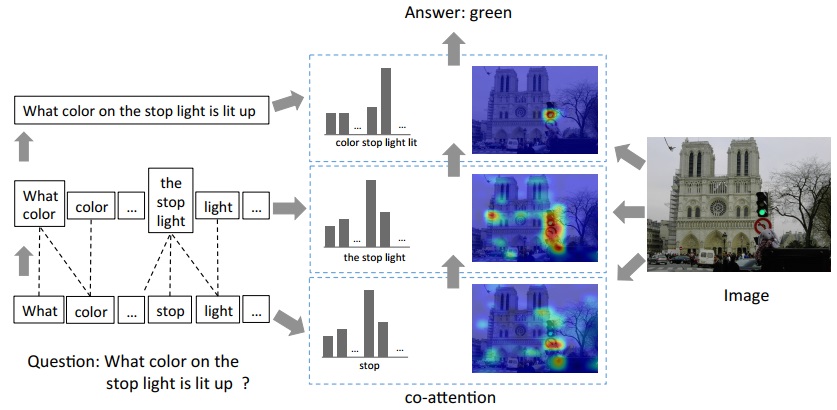
Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies, Remembering information for long periods of time is practically their default behavior.

The reason of why lstm network does not vanish is the activation function used is an identity function which has a derivative of 1, so it will never vanish or explode, also it has gates functions controlling memory cell uses for saving the output of the hidden state at each time step.



**Figure 4.1.2.12– lstm**

At each level of question hierarchy, co-attention is applied to both question and image in order to generate the best features related to each other between an image and question for generating the best answer.



**Figure 4.1.2.13–hierarchy co-attention highlight**

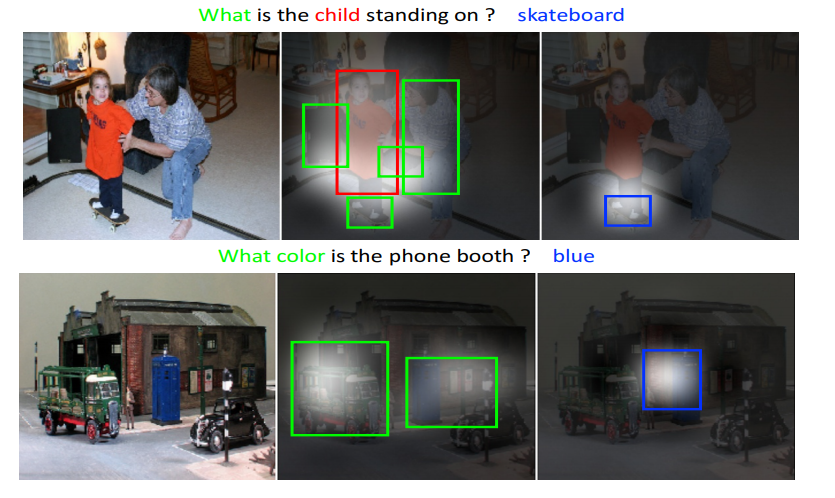
**4.1.4 Attention**

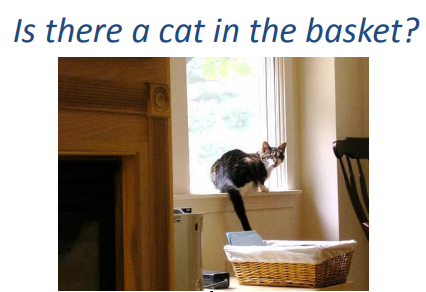
So now we have question features extracted from question hierarchy phase, and image features extracted from VGG19 Convolution Neural Network. But how to combine information from different domains (text and image)? Here comes the role of the Attention phase. Attention has been widely used in image or video captioning, and now it comes to VQA. With it, we can now focus on specific regions of an image base on the question which correspond to how a human understand an image and vice versa.

In this section you will see a detailed explanation for attention on the image which called visual attention, attention on the question and the co-attention which consider as a combination of both (visual and question attentions) and this is what we use in our project.

1. **Visual Attention**

This attention mechanism typically produces a spatial map highlighting image regions relevant to answering the question, where answering visual questions often involves looking at different spatial regions and comparing their contents and/or locations. For example, to answer the questions in Figure 1, we need to look at a portion of the image, such as the child or the phone booth. Similarly, to answer the question “Is there a cat in the basket?” in Figure 2, we can first find the basket and the cat objects, and then compare their locations.

**Figure 1:** Describes the visual attention.



**Figure 2**

1. **Question Attention**

So far, most of the previous attention models for VQA have focused on the problem of identifying “where to look” or visual attention. In this model, the problem of identifying “which words to listen to” or question attention is equally important. Consider the questions about the image in figure 3 “how many horses are in this image?” and “how many horses can you see in this image?". They have the same meaning, essentially captured by the first three words. A machine that attends to the first three words would arguably be more robust to linguistic variations irrelevant to the meaning and answer of the question.



**Figure 3**

1. **Co-Attention**

The co-attention mechanism is a combination of visual attention and question attention. The idea of the co-attention is that the model has a natural symmetry between the image and question, in the sense that the image representation is used to guide the question attention and the question representations are used to guide image attention.

There are two co-attention mechanisms that differ in the order in which image and question attention maps are generated. The first mechanism, which called parallel co-attention, generates image and question attention simultaneously. The second mechanism, which called alternating co-attention, sequentially alternates between generating image and question attentions. These co-attention mechanisms are executed at all three levels of the question hierarchy.

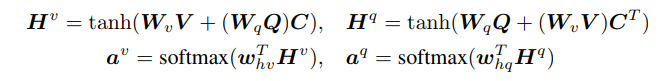
1. **Parallel Co-Attention**

Parallel co-attention attends to the image and question simultaneously. We connect the image and question by calculating the similarity between image and question features at all pairs of image-locations and question-locations. Specifically, given an image feature map ∈, and the question representation , we calculate the affinity matrix by.



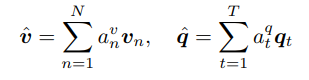
**Figure 4**

wherecontains the weights, is the image representation dimension, is the number of the spatial locations on the image and is the number of words in the question. After computing this affinity matrix, one possible way of computing the image (or question) attention is to simply maximize out the affinity over the locations of other modality, i.e. and . Instead of choosing the max activation, the best way to improve the performance is to consider this affinity matrix as a feature and learn to predict image and question attention maps via the following.



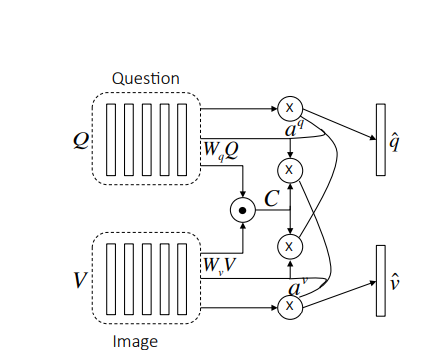
**Figure 5**

where , are the weight parameters and k is a constant number. and are the attention probabilities of each image region and word respectively. The affinity matrix C transforms question attention space to image attention space (vice versa for ). Based on the above attention weights, the image and question attention vectors are calculated as the weighted sum of the image features and question features.



**Figure 6**

The parallel co-attention is done at each level in the hierarchy, leading to and where r ∈ {word level, phrase level, question level}.



**Figure 7:** Visualizes the parallel co-attention

1. **Alternating Co-Attention**

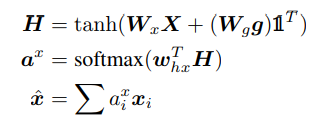
In this attention mechanism, the generating of image and question attention is sequentially alternated. Briefly, this consists of three steps (marked in Figure 9):

1) Summarize the question into a single vector q.

2) Attend to the image based on the question summary q.

3) Attend to the question based on the attended image feature.

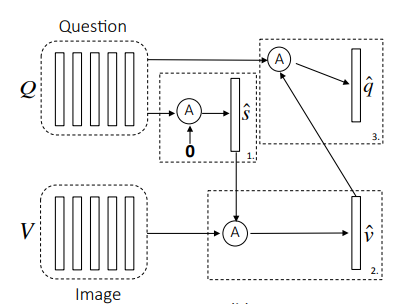
Concretely, the attention operation is defined as , which takes the image (or question) features and attention guidance derived from question (or image) as inputs, and outputs the attended image (or question) vector. The operation can be expressed in the following steps:



**Figure 8**

where**1** is a vector with all elements to be 1. andare parameters. is the attention weight of feature .

The alternating co-attention process is illustrated in Figure 9. At the first step of alternating co-attention, , and is 0. At the second step, where is the image features, and the guidance is intermediate attended question feature from the first step. Finally, the attended image feature is used as the guidance to attend the question again, i.e., and . Similar to the parallel co-attention, the alternating co-attention is also done at each level of the hierarchy.

****

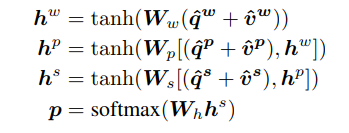
**Figure 9:** Visualizes the alternating co-attention

**4.1.5 Encoding for Predicting Answers**

So far, we have finished the features extraction phase and the co-attention phase. In this section, we will talk about the last phase of the VQA system to get the answer to the question out from the system.

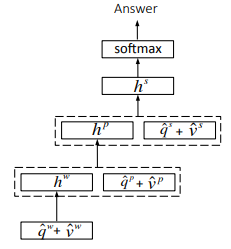
The VQA problem is treated as a classification problem where each answer should be considered as a class. The COCO-QA dataset contains 450 different answers for all questions, so we have 450 unique classes we need to classify each question to one of them.

The answer is predicted based on the co-attended image and question features from all three levels. Then a multi-layer perceptron (MLP) is used to recursively encode the attention features as shown in figure 10.



**Figure 10**

Whereandare the weight parameters. is the concatenation operation on two vectors. p is the probability of the final answer.



**Figure 11:** Visualizes the process of predicting answers

# 5-User Manual

This chapter should describe in details how to operate the project along with screen shots of the project representing all steps.

This chapter should also include an "Installation Guide" that would describe how to install the program, and all required third party tools that needs to be available for the project to run. The installation guide will also be included as a readme file in the CDs delivered at the end of the year.

# 6-Conclusion and Future Work

## 6.1 Conclusion

In our system, we tried to make an effective application to answer an image related to a given question by merging between natural language processing and convolution neural network. We got an overall accuracy “ “

Making such a system helps us to learn many new concepts and understanding too many new basics in deep learning and natural language processing.

## 6.2 Future Work

Our future work include :

* Deal with more different types of questions.
* Deal with low quality of images.
* Generate complete answers using encoder decoder model.
* Reduce the overall time complexity of the whole application.
* Increase the classifier training data.
* Try different methods of attentions.
* Try using Dynamic memory network for visual and textual answering.
* Visualizing the attention between the Image and the question.

# References

[1]Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition.CoRR, abs/1409.1556, 2014.

[2] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering.InNIPS, 2015

[3] Jiasen Lu, Jianwei Yang, DhruvBatra, and Devi Parikh. Hierarchical Question-Image Co-Attention for Visual Question Answering, 2017

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In ICCV, 2015.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár,and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.InCVPR, 2016.