Stage 2 Report on

"Visual Question answering using LSTM and CNN Techniques"

Submitted in partial fulfilment of the requirements of the degree of Bachelor of Technology

By

RIDDHI N. NISAR (161071018)

DEVANGI P. BHUVA (161071040)

B. Tech. (Computer Engineering)

2019-20

Under the guidance of

Mrs. P. M. CHAWAN

Department of Computer Science and Information Technology

Veermata Jijabai Technological Institute

(An Autonomous institute affiliated to University of Mumbai)

Mumbai-400019

2019-20

STATEMENT OF CANDIDATES

We state that work embodied in this Project entitled "**Visual question answering using LSTM and CNN techniques**" forms our own contribution of work under the guidance of Mrs. P. M. Chawan at the Department of Computer Engg. And Information Technology, Veermata Jijabai Technological Institute. The report reflects the work done during the period of candidature but may include related preliminary material provided that it has not contributed to an award of previous degree. No part of this work has been used by us for the requirement of another degree except where explicitly stated in the body of the text and the attached statement.

(Signature)	(Signature)
Riddhi N. Nisar	Devangi P. Bhuva
Roll: 161071018	Roll: 161071040

Date: _____

APPROVAL SHEET

The stage 2 report, "**Visual Question answering using LSTM and CNN Techniques",**

by Riddhi N. Nisar (161071018) and Devangi P. Bhuva (161071040) is found to be satisfactory and is approved as successfully completed first phase of Bachelor of Technology project.

_____ _____ _____

Examiner 1 Examiner 2 Examiner 3

_____

Project Guide

(Prof. P. M. Chawan)

Date: _____

Place: VJTI, MUMBAI

CERTIFICATE

This is to certify that Ms. Riddhi N. Nisar (161071018) and Ms. Devangi P. Bhuva (161071040), a student of Bachelor of Technology in Computer Engineering at Veermata Jijabai Technological Institute (VJTI), Mumbai has successfully completed the First phase of the Master of Technology project, entitled "**Visual Question answering using LSTM and CNN Techniques**" under the guidance of Mrs. P. M. Chawan.

_____

Prof. P. M. Chawan

Project Guide

_____

Dr. M. M. Chandane

Head, CS & IT Department

Date: _____

Place: VJTI, MUMBAI

TABLE OF CONTENTS

ABSTRACT ……..…..7

1. INTRODUCTION ……..……..……..8
   1.1. Neural Networks…….……...…..8
   1.2. Supervised Learning …….……...……...11
   1.3. Recurrent Neural Networks …….……..…….12
2. LITERATURE REVIEW …….……..…….....15
   2.1. Convolutional Neural Networks ……..……...15
   2.2. Long short term memory ……..…….…...……..15
   2.3. Multilayer perceptron ……..…….……..15
   2.4. Research gap …….....16
   2.5. Dataset……..……18
3. PROPOSED SYSTEM …….…..……....21
   3.1. Problem statement …….....21
   3.2. Problem elaboration ……..…..…....21
   3.3. Objectives ……..…….21
   3.4. Significance …….……..……..21
4. METHODOLOGY ……..……..22
   4.1. Proposed method ……..…..….22
   4.2. Proposed System Architecture …….……...……24
5. IMPLEMENTATION
   5.1.Building Dataset
   5.2.Algorithm
   5.3.Experimental implementation/execution

6. CONCLUSIONS …….……...….25

REFERENCES ……26

List of Figures Figure 1: Layers of interconnected nodes……9

13

15

16

17

Figure 6: VQA Dataset…….....18

39

Figure 8: FigureQA Dataset……20

Figure 9: Work flow……24

Figure 10: Ask Question with selected image…….……42

Figure 11: Answer using trained model:…… ……...…42

ABSTRACT

This project aims to perform the task of Visual Question Answering using Keras deep learning frameworks that combine the image features and the question vector to produce answers for the questions posed by making use of a pre-processed image dataset along with spacy word embeddings. In this article, we have explained how we can merge the CNN and RNN models with a dense multi-layer perceptron to produce categorical classes that correspond to our answer. The underlying problem here is to merge the two different models as these are two different domains of the machine learning regime. We have solved it by merging the image vector and question vector in a multi-layer perceptron with the number of layers and class of activation mentioned further in the article. In the implementation, we use the Keras python package with the backend of Tensorflow and the pre-processed VGG-16 weights for extracting image features using CNN and the question vector using the Spacy word embeddings, and finally we use the multi-layer perceptron to combine the results from the image and question.

.

Chapter 1

INTRODUCTION

The issue of solving visual question answering goes past the ordinary issues of image captioning and natural language processing, as it is a blend of both the strategies which makes it a perplexing system. Since language has a complex compositional structure, the issue of taking care of vision and language becomes a tough task.

It is very simple to get a decent shallow exhibition of accuracy when the model disregards the visual content on account of the predisposition that is available in the dataset. For this situation, the model doesn't genuinely comprehend the information embedded in the picture and just focuses on the language semantics, which is not what we need. For example, in the VQA dataset, the most widely recognized game answer "cricket" is the right response for 41% of the inquiries beginning with "What game is", and "white" is the right response for half of the inquiries beginning with "What shading is". These dialects can make a bogus impression of accomplishing precision. It is very conceivable to get cutting edge results with a moderately low comprehension of the picture. This should be possible by misusing the factual inclinations as well, which are present in the datasets. They are commonly obvious in standard language models as well. Presently, we need language to posture difficulties including the visual comprehension of rich semantics. The frameworks should not have the option to get rid of ignoring the visual data.

In this work, we propose to counter these language biases and make the role of image understanding in VQA more impactful by utilizing the underlying image features and the corresponding semantics of language.

1.1 Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

How does deep learning attain such impressive results?

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of **labeled data**. For example, driverless car development requires millions of images and thousands of hours of video.

2. Deep learning requires substantial **computing power**. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Examples of Deep Learning at Work

Deep learning applications are used in industries from automated driving to medical devices.

Automated Driving:

Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

Aerospace and Defense:

Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

Medical Research:

Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

Industrial Automation:

Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

Electronics:

Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

How Deep Learning Works

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

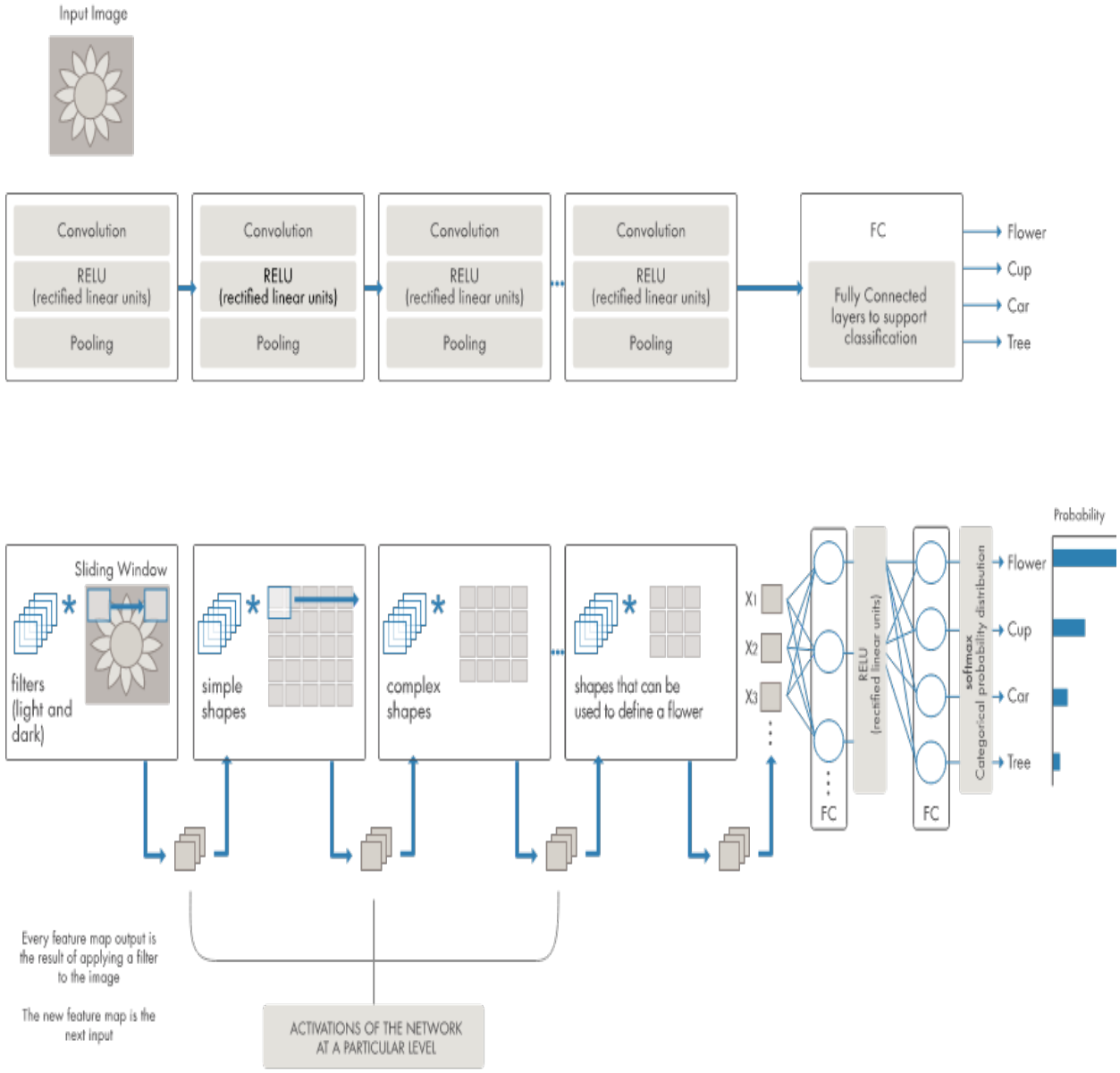


Figure No. 1 Example of a network with many convolutional layers

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs "end-to-end learning" – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

1.2 Neural Network

What is a Neural Network?

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

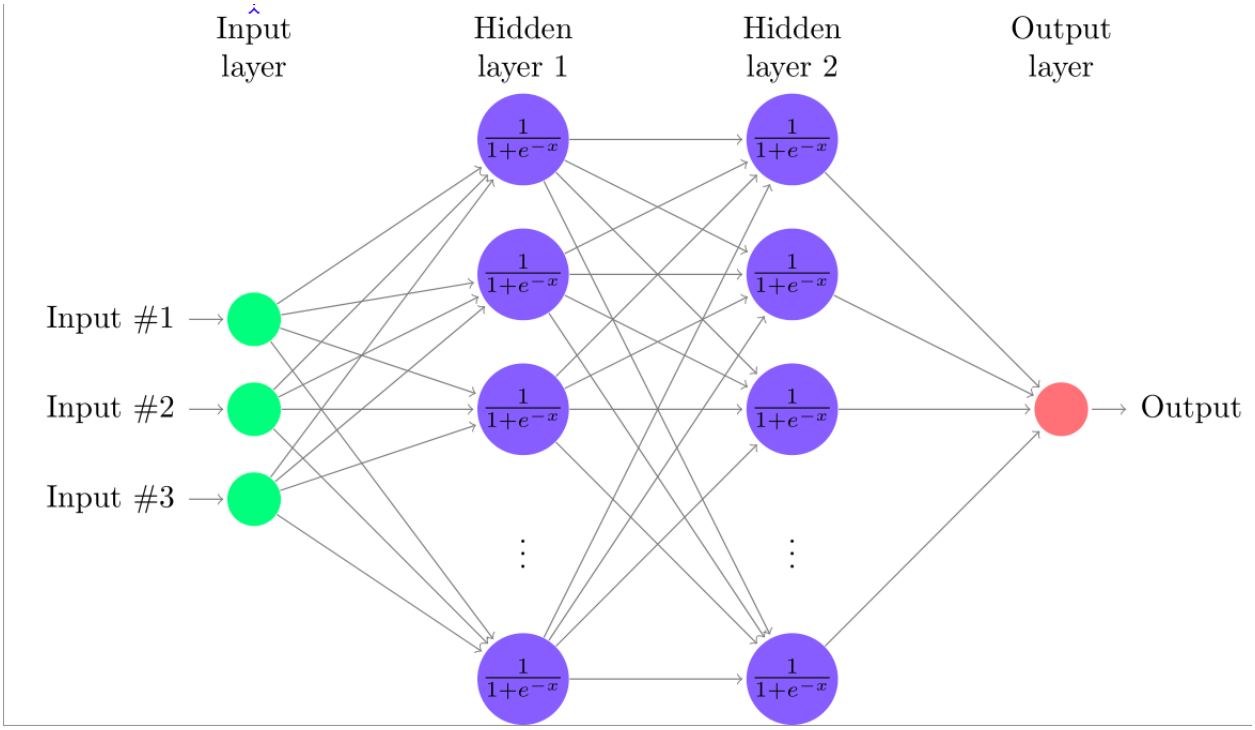


Figure 2: Layers of interconnected nodes

What is a Node?

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

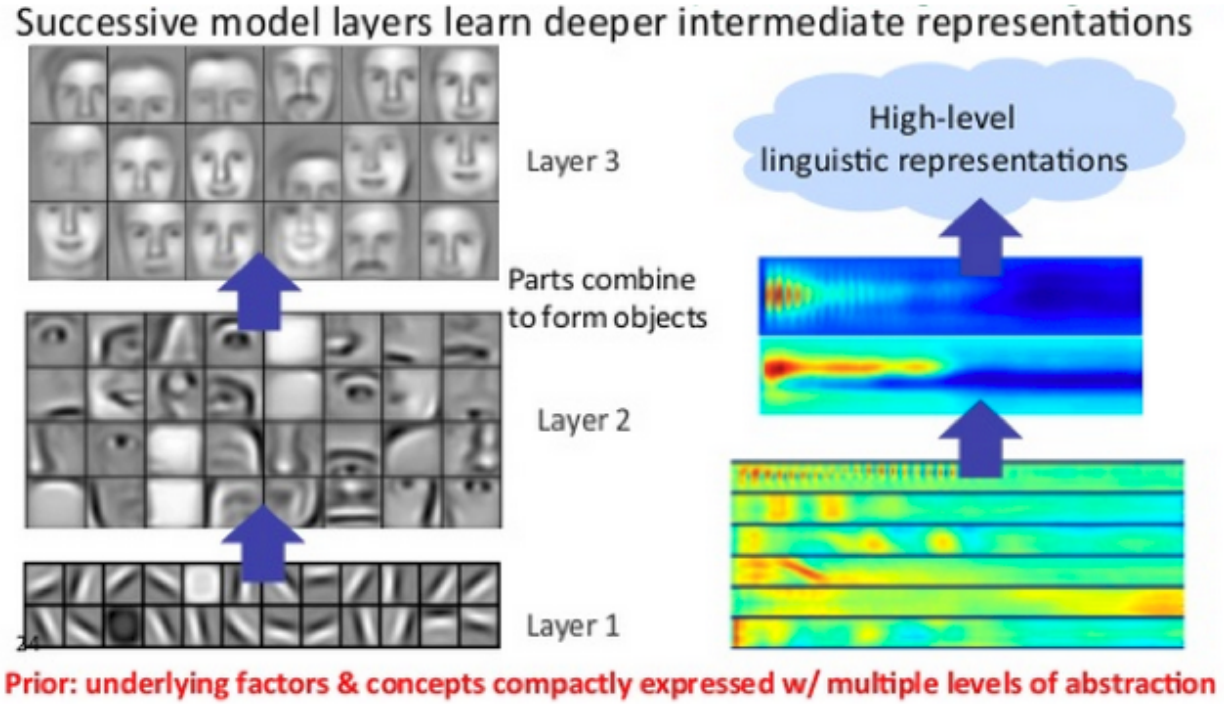


Figure 2: Understanding complexity of layers

This is known as **feature hierarchy**, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions.

For example, deep learning can take a million images, and cluster them according to their similarities: cats in one corner, ice breakers in another, and in a third all the photos of your grandmother. This is the basis of so-called smart photo albums.

Key Concepts of Deep Neural Networks

Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms. Given that feature extraction is a task that can take teams of data scientists years to accomplish, deep learning is a way to circumvent the chokepoint of limited experts. It augments the powers of small data science teams, which by their nature do not scale.

When training on unlabeled data, each node layer in a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network's guesses and the probability distribution of the input data itself. Restricted Boltzmann machines, for examples, create so-called reconstructions in this manner.

In some circles, neural networks are thought of as "brute force" AI, because they start with a blank slate and hammer their way through to an accurate model. They are effective, but to some eyes inefficient in their approach to modeling, which can't make assumptions about functional dependencies between output and input.

1.3 Supervised Learning

Supervised learning is when the model is getting trained on a labelled dataset. **Labelled** dataset is one which have both input and output parameters. In this type of learning both training and validation datasets are labelled as shown in the figures below.

Training the system:

While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and rest as testing data. In training data, we feed input as well as output for 80% data. The model learns from training data only. We use different machine learning algorithms(which we will discuss in detail in next articles) to build our model. By learning, it means that the model will build some logic of its own. Once the model is ready then it is good to be tested. At the time of testing, input is fed from remaining 20% data which the model has never seen before, the model will predict some value and we will compare it with actual output and calculate the accuracy.

Types of Supervised Learning:

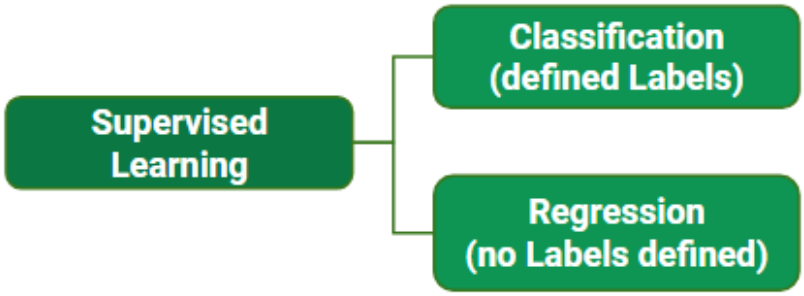


Figure 3: **Types of Supervised Learning**

Classification

It is a Supervised Learning task where output is having defined labels(discrete value). The goal here is to predict discrete values belonging to a particular class and evaluate on the basis of accuracy. It can be either binary or multi class classification. In binary classification, model predicts either 0 or 1 ; yes or no but in case of multi class classification, model predicts more than one-class. **Example:** Gmail classifies mails in more than one classes like social, promotions, updates, forum.

Regression

It is a Supervised Learning task where output is having continuous value. The goal here is to predict a value as much closer to actual output value as our model can and then evaluation is done by calculating error value. The smaller the error the greater the accuracy of our regression model.

1.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) add an interesting twist to basic neural networks. A simple neural network takes in a fixed size vector as input which limits its usage in situations that involve a 'series' type input with no predetermined size.

What is RNN?

Recurrent Neural Network remembers the past and it's decisions are influenced by what it has learnt from the past. Note: Basic feed forward networks "remember" things too, but they remember things they learnt during training. For example, an image classifier learns what a "1" looks like during training and then uses that knowledge to classify things in production.

While RNNs learn similarly while training, in addition, they remember things learnt from prior input(s) while generating output(s). It's part of the network. RNNs can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series.

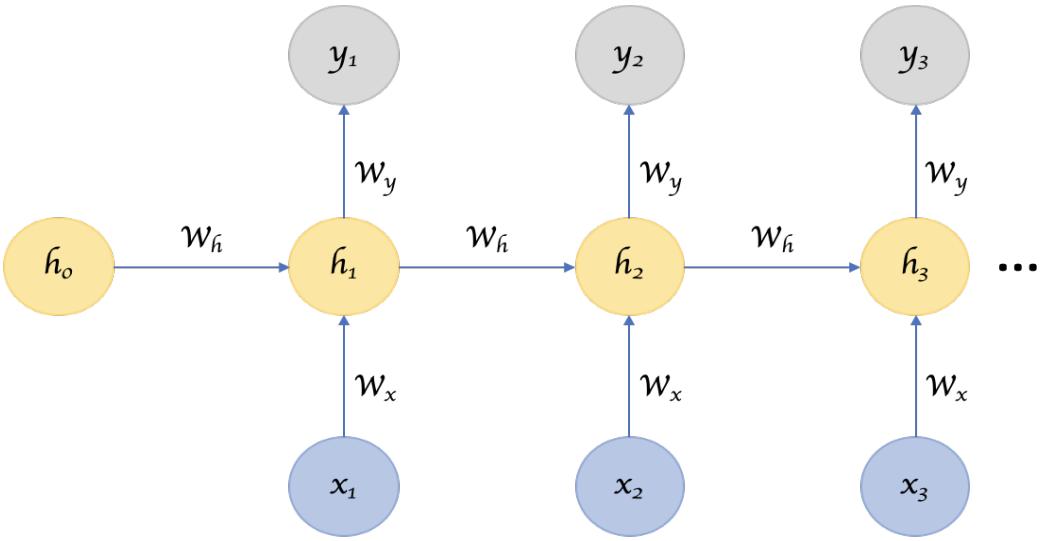


Figure 4: A Recurrent Neural Network, with a hidden state that is meant to carry pertinent information from one input item in the series to others.

Bidirectional RNNs

Sometimes it's not just about learning from the past to predict the future, but we also need to look into the future to fix the past. In speech recognition and handwriting recognition tasks, where there could be considerable ambiguity given just one part of the input, we often need to know what's coming next to better understand the context and detect the present.

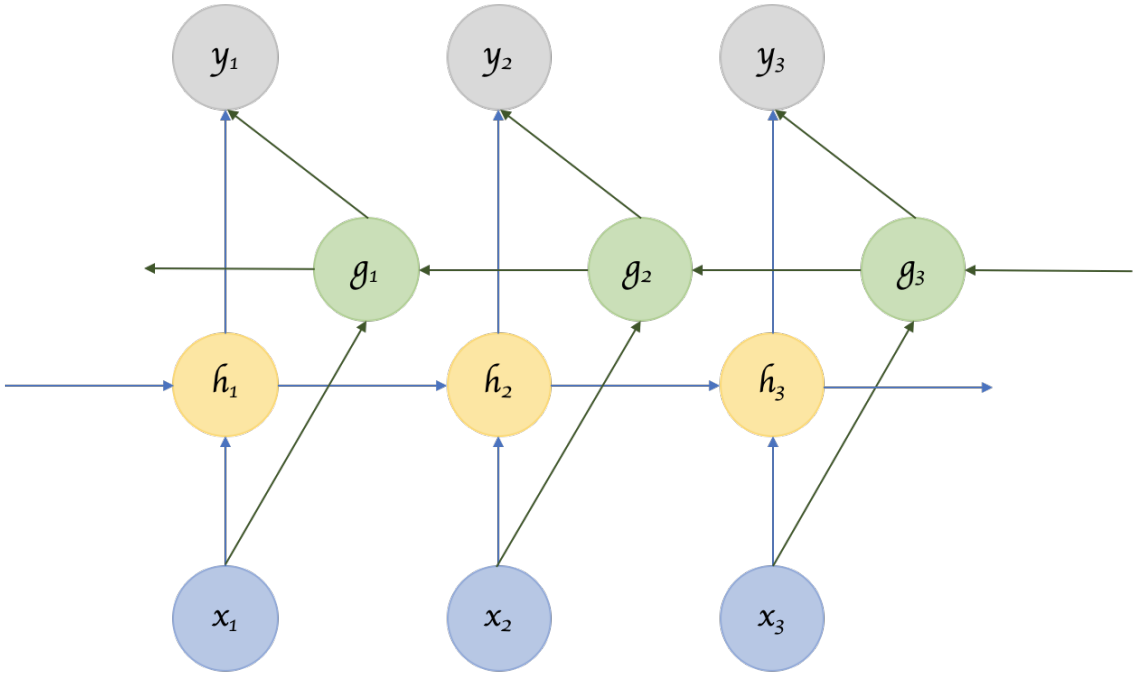


Figure 5: Bidirectional RNN

What we have seen here so far are only the simple architecture and some additional well known variants. But knowing about RNNs and the related variants has made it more clear that the trick to designing a good architecture is to get a sense of the different architectural variations, understand what benefit each of the changes bring to the table, and apply that knowledge to the problem at hand appropriately.

Chapter 2

LITERATURE REVIEW

2.1 Convolutional Neural Networks

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users' photos.

Let us consider the use of CNN for image classification in more detail. The main task of image classification is acceptance of the input image and the following definition of its class. This is a skill that people learn from their birth and are able to easily determine that the image in the picture is an elephant. But the computer sees the pictures quite differently:

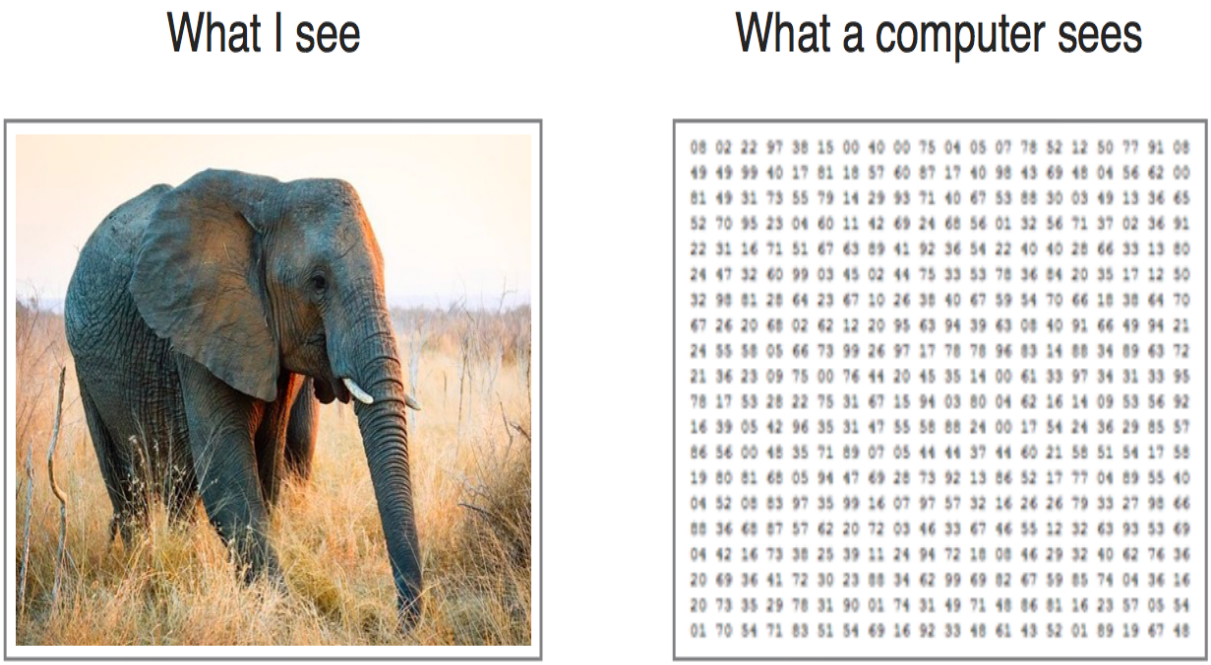


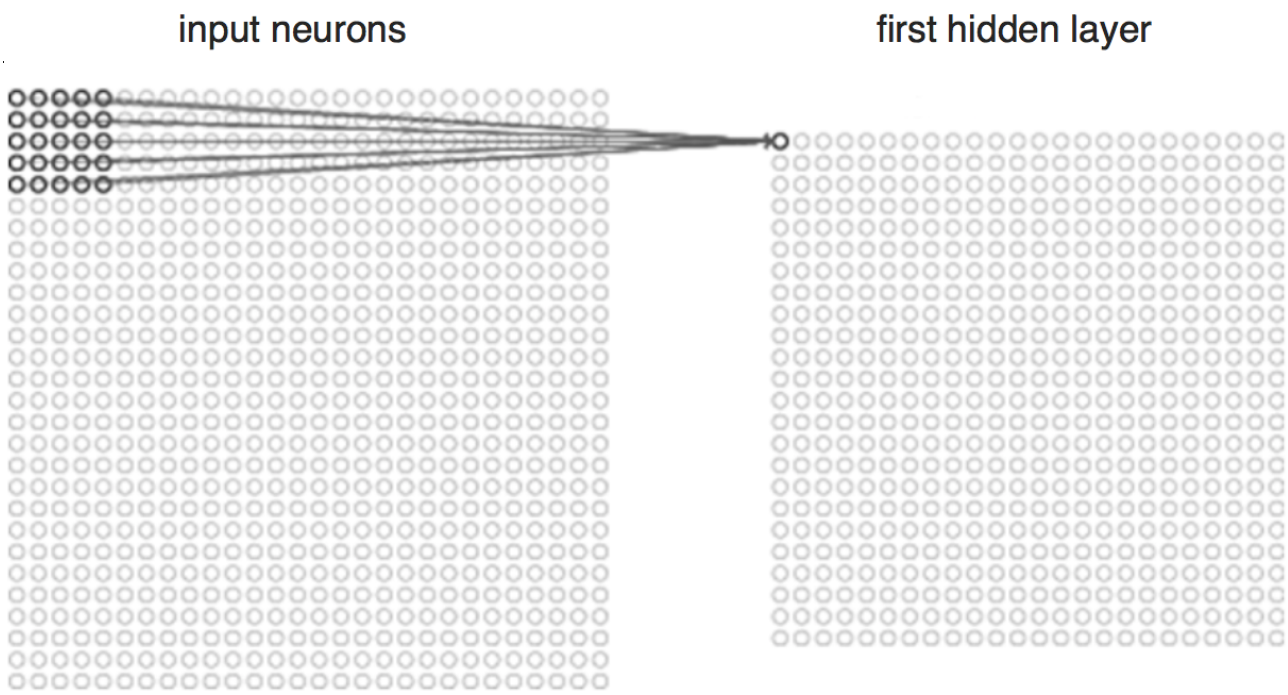
Figure No. 5:
Matrix View of an image

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

To solve this problem the computer looks for the characteristics of the base level. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.

In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

The Convolution layer is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a **filter** (or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter's task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then a input matrix.



First Layer

The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

The nonlinear layer is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a downsampling operation on them. As a result the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, than a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach **a fully connected layer**. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class.

A fragment of the code of this model written in Python will be considered further in the practical part.

2.2 Long Short Term Memory:

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on your iPhone's keyboard.
With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. The purpose of this article is to explain LSTM and enable you to use it in real life problems.

Limitations of RNNs

Recurrent Neural Networks work just fine when we are dealing with short-term dependencies. That is when applied to problems like:

The colour of the sky is _____.

RNNs turn out to be quite effective. This is because this problem has nothing to do with the context of the statement. The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue. Thus the prediction would be:

The colour of the sky is blue.

However, vanilla RNNs fail to understand the context behind an input. Something that was said long before, cannot be recalled when making predictions in the present. Let's understand this as an example:

I spent 20 long years working for the under-privileged kids in Spain. I then moved to Africa.

.....

I can speak fluent _____.

Here, we can understand that since the author has worked in Spain for 20 years, it is very likely that he may possess a good command over Spanish. But, to make a proper prediction, the RNN needs to remember this context. The relevant information may be separated from the point where it is needed, by a huge load of irrelevant data. This is where a Recurrent Neural Network fails!

The reason behind this is the problem of **Vanishing Gradient.** In order to understand this, you'll need to have some knowledge about how a feed-forward neural network learns. We know that for a conventional feed-forward neural network, the weight updating that is applied on a particular layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a particular layer is somewhere a product of all previous layers' errors. When dealing with activation functions like the sigmoid function, the small values of its derivatives (occurring in the error function) gets multiplied multiple times as we move towards the starting layers. As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

A similar case is observed in Recurrent Neural Networks. RNN remembers things for just small durations of time, i.e. if we need the information after a small time it may be reproducible, but once a lot of words are fed in, this information gets lost somewhere. This issue can be resolved by applying a slightly tweaked version of RNNs – the Long Short-Term Memory Networks.

Improvement over RNN: LSTM (Long Short-Term Memory) Networks

When we arrange our calendar for the day, we prioritize our appointments right? If in case we need to make some space for anything important we know which meeting could be canceled to accommodate a possible meeting.

Turns out that an RNN doesn't do so. In order to add a new information, it transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for '*important'* information and '*not so important'* information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

Example

We'll visualize this with an example. Let's take the example of predicting stock prices for a particular stock. **The stock price of today will depend upon:**

1. The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.
2. The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.
3. The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

These dependencies can be generalized to any problem as:

1. The previous cell state *(i.e. the information that was present in the memory after the previous time step)*
2. The previous hidden state *(i.e. this is the same as the output of the previous cell)*
3. The input at the current time step *(i.e. the new information that is being fed in at that moment)*

Another important feature of LSTM is its analogy with conveyor belts!

Architecture of LSTMs

The functioning of LSTM can be visualized by understanding the functioning of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.

Let's say, we were assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? You immediately **forget** the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You **input** this information into your news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and **output** the relevant things to your audience. Maybe in the form of "*XYZ turns out to be the prime suspect.*".

Now let's get into the details of the architecture of LSTM network:

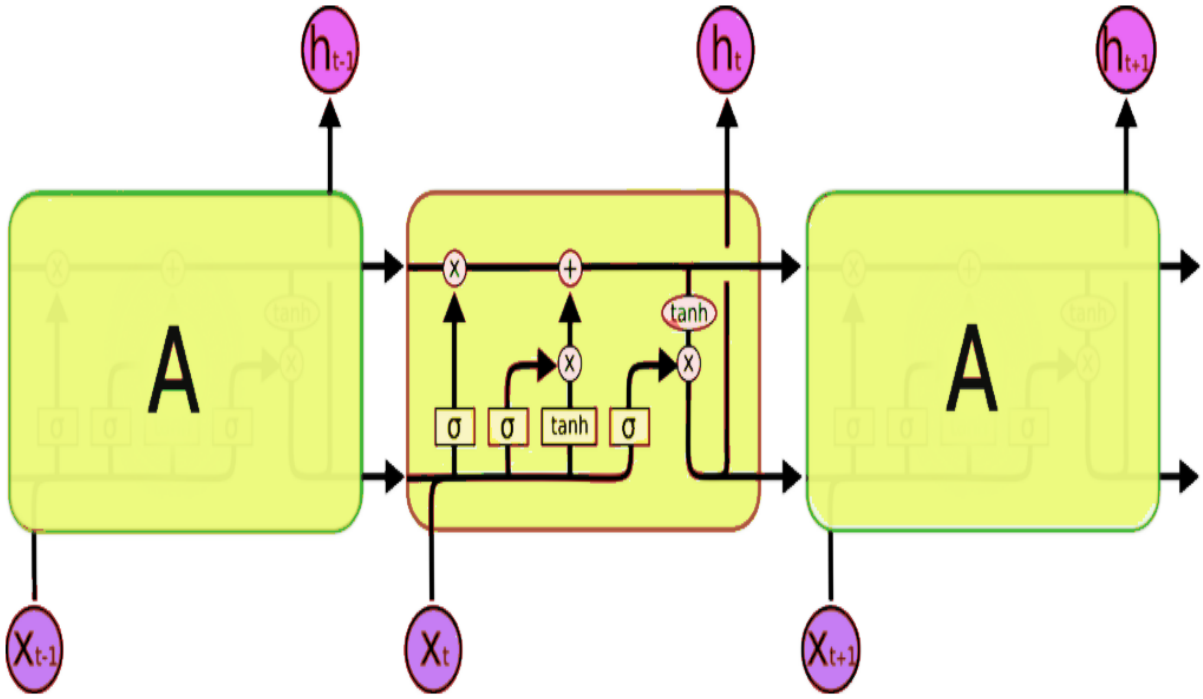


Figure no Architecture Of LSTM

Now, this is nowhere close to the simplified version which we saw before, but let me walk you through it. A typical LSTM network is comprised of different memory blocks called **cells** (the rectangles that we see in the image)**.** There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called **gates.** Each of them is being discussed below.

This is the diagram of a LSTM building block.

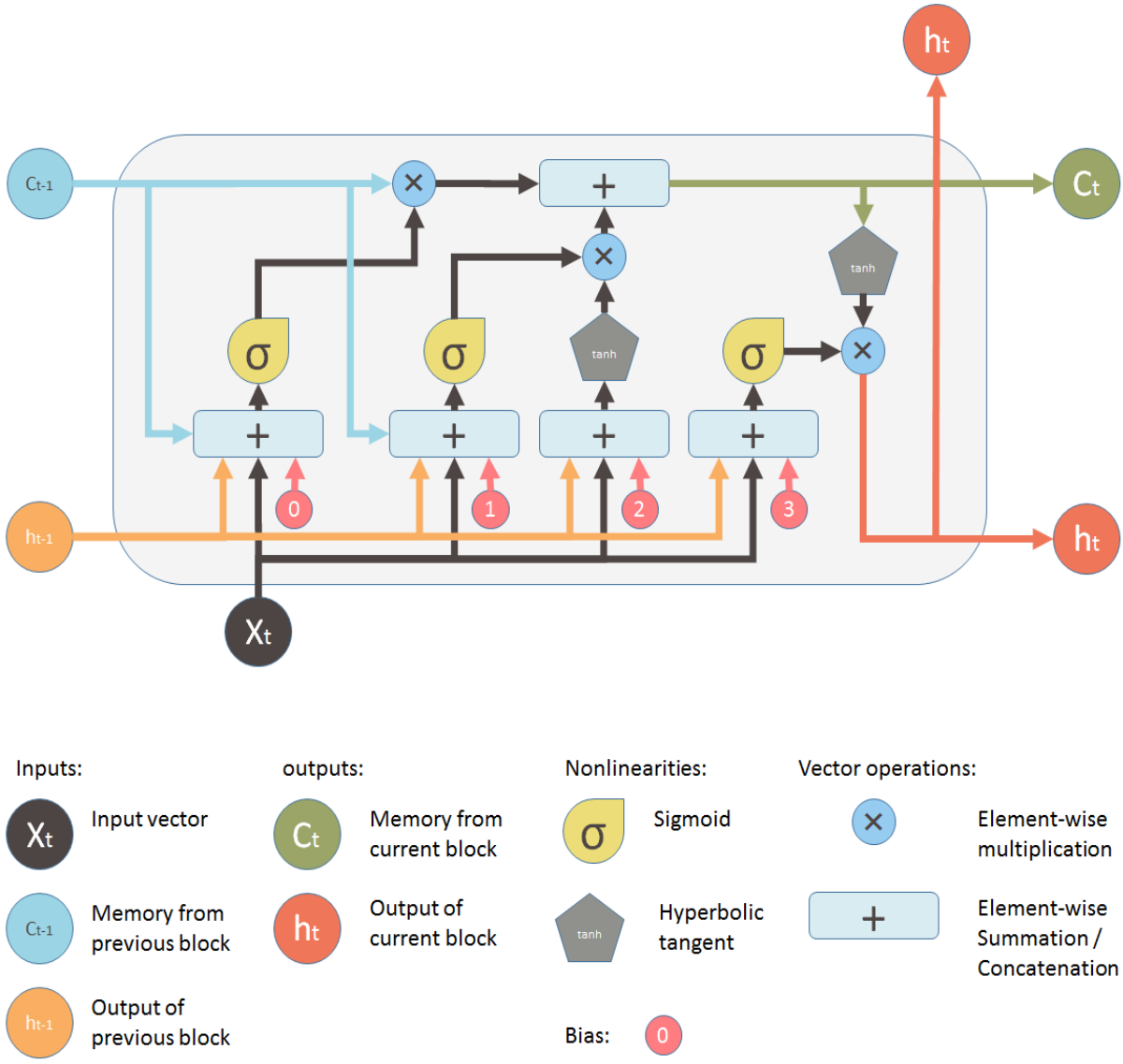


Figure: LSTM Building Block

2.3 Multilayer Perceptron (MLP)

Brief History of Perceptrons

The perceptron, that neural network whose name evokes how the future looked from the perspective of the 1950s, is a simple algorithm intended to perform binary classification; i.e. it predicts whether input belongs to a certain category of interest or not: `fraud` or `not_fraud`, `cat` or `not_cat`.

The perceptron holds a special place in the history of neural networks and artificial intelligence, because the initial hype about its performance led to a rebuttal by Minsky and Papert, and wider spread backlash that cast a pall on neural network research for decades, a neural net winter that wholly thawed only with Geoff Hinton's research in the 2000s, the results of which have since swept the machine-learning community.

Frank Rosenblatt, godfather of the perceptron, popularized it as a device rather than an algorithm. The perceptron first entered the world as hardware.[1] Rosenblatt, a psychologist who studied and later lectured at Cornell University, received funding from the U.S. Office of Naval Research to build a machine that could learn.

A perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line. Input is typically a feature vector `x` multiplied by weights `w` and added to a bias `b`: `y = w * x + b`.

A perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights (and sometimes passing the output through a nonlinear activation function). Here's how you can write that in math:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

where **w** denotes the vector of weights, **x** is the vector of inputs, **b** is the bias and phi is the non-linear activation function.

Rosenblatt built a single-layer perceptron. That is, his hardware-algorithm did not include multiple layers, which allow neural networks to model a feature hierarchy. It was, therefore, a shallow neural network, which prevented his perceptron from performing non-linear classification, such as the XOR function (an XOR operator trigger when input exhibits either one trait or another, but not both; it stands for "exclusive OR"), as Minsky and Papert showed in their book.

XOR

Input x_1	Input x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Multilayer Perceptrons (MLP)

Subsequent work with multilayer perceptrons has shown that they are capable of approximating an XOR operator as well as many other non-linear functions.

The multilayer perceptron is the hello world of deep learning: a good place to start when you are learning about deep learning.

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function.

Multilayer perceptrons are often applied to supervised learning problems[3]: they train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimize error. Backpropagation is used to make those weigh and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error (RMSE).

Feedforward networks such as MLPs are like tennis, or ping pong. They are mainly involved in two motions, a constant back and forth. You can think of this ping pong of guesses and answers as a kind of accelerated science, since each guess is a test of what we think we know, and each response is feedback letting us know how wrong we are.

In the *forward pass*, the signal flow moves from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels.

In the *backward pass*, using backpropagation and the chain rule of calculus, partial derivatives of the error function w.r.t. the various weights and biases are back-propagated through the MLP. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters may be adjusted as they move the MLP one step closer to the error minimum. This can be done with any gradient-based optimisation algorithm such as stochastic gradient descent. The network keeps playing that game of tennis until the error can go no lower. This state is known as *convergence*.

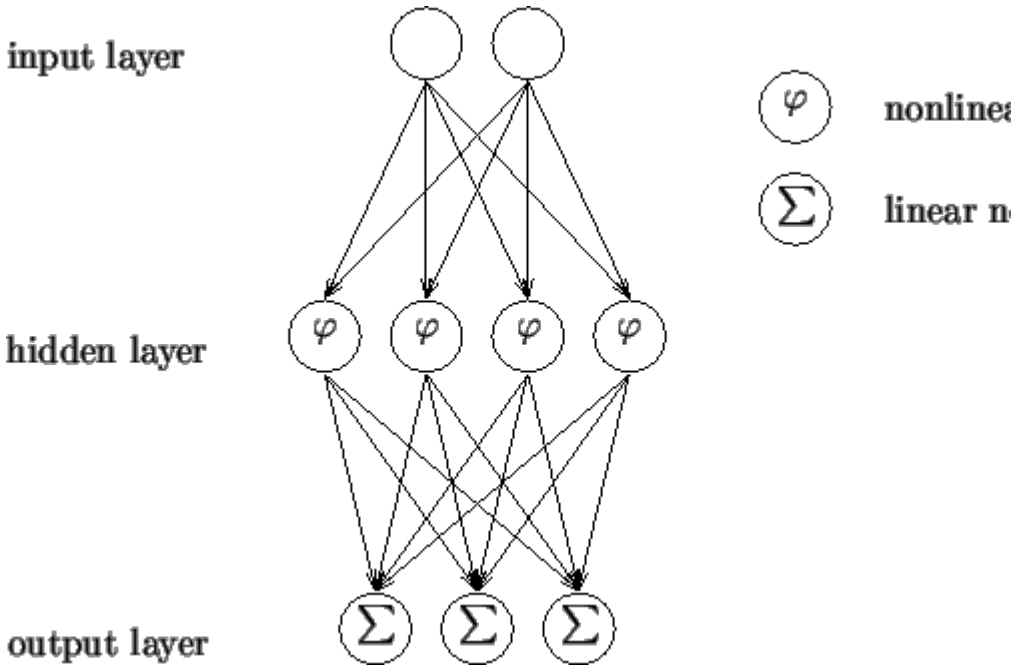


Figure No. MLP

Feed Forward Network, is the most typical neural network model. Its goal is to approximate some function f (). Given, for example, a classifier **y = f □ (x)** that maps an input x to an output class y, the MLP find the best approximation to that classifier by defining a mapping, y = f(x; θ) and learning the best parameters θ for it. The MLP networks are composed of many functions that are chained together. A network with three functions or layers would form f(x) = f (3)(f (2)(f (1)(x))). Each of these layers is composed of units that perform an affine transformation of a linear sum of inputs. Each layer is represented as y = f(WxT + b). Where f is the activation function (covered

below), W is the set of parameter, or weights, in the layer, x is the input vector, which can also be the output of the previous layer, and b is the bias vector.

The layers of an MLP consists of several fully connected layers because each unit in a layer is connected to all the units in the previous layer. In a fully connected layer, the parameters of each unit are independent of the rest of the units in the layer, that means each unit possess a unique set of weights.

In a supervised classification system, each input vector is associated with a label, or ground truth, defining its class or class label is given with the data. The output of the network gives a class score, or prediction, for each input. To measure the performance of the classifier, the loss function is defined. The loss will be high if the predicted class does not correspond to the true class, it will be low otherwise. Sometimes the problem of overfitting and underfitting occurs at the time of training the model. In this case, Our model performs very well on training data but not on testing data. In order to train the network, an optimization procedure is required for this we need loss function and an optimizer. This procedure will find the values for the set of weights, W that minimizes the loss function.

A popular strategy is to initialize the weights to random values and refine them iteratively to get a lower loss. This refinement is achieved by moving on the direction defined by the gradient of the loss function. And it is important to set a learning rate defining the amount in which the algorithm is moving in every iteration.

Activation function:

Activation functions also known non- linearity, describe the input-output relations in a non-linear way. This gives the model power to be more flexible in describing arbitrary relations. Here are popular activation functions. Sigmoid, Relu, Tanh and many more. I will describe these in my next blog.

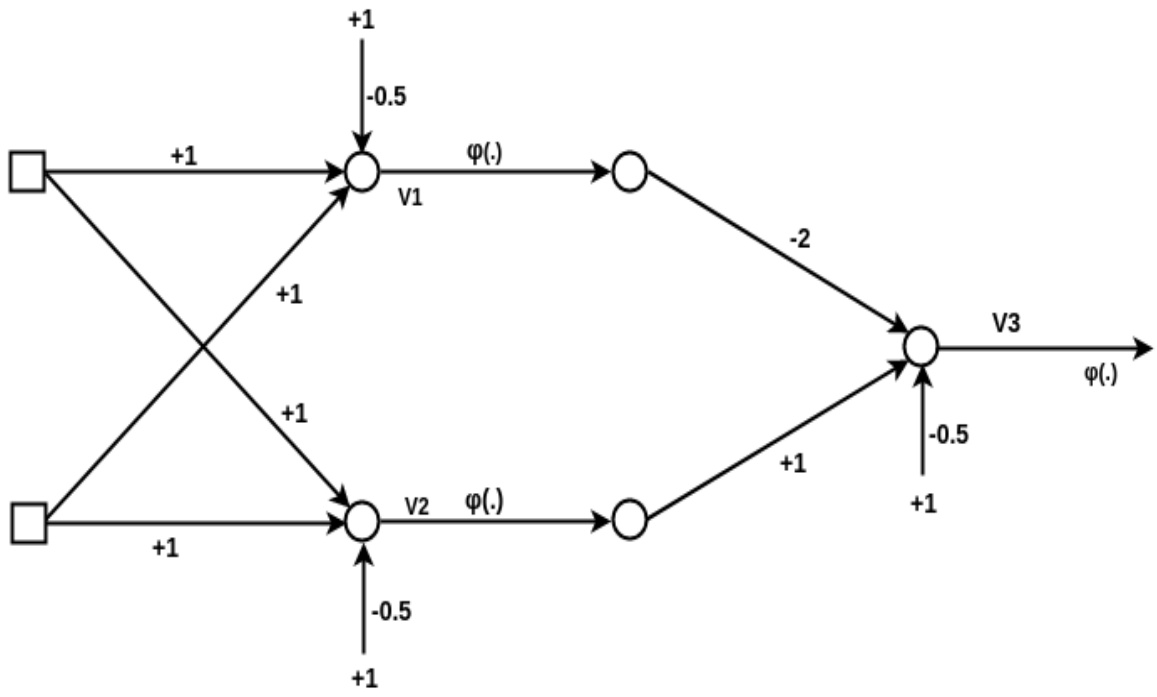
Training the Model-

There are basically three steps in the training of the model.

1. Forward pass
2. Calculate error or loss
3. Backward pass

1. Forward pass

In this step of training the model, we just pass the input to model and multiply with weights and add bias at every layer and find the calculated output of the model.



2. Loss Calculate

When we pass the data instance(or one example) we will get some output from the model that is called Predicted output(predout) and we have the label with the data that is real output or expected output(Expect_out). Based upon these both we calculate the loss that we have to backpropagate (using Back-propagation algorithm). There is various Loss Function that we use based on our output and requirement.

3. Backward Pass

After calculating the loss, we backpropagate the loss and updates the weights of the model by using gradient. This is the main step in the training of the model. In this step, weights will adjust according to the gradient flow in that direction.

2.4 Research Gap

[1] In recent years, many clustering algorithms for big data have been proposed which are based on distributed and parallel computation. Mac Queen in 1967 firstly proposed this technique. The standard algorithm was first proposed by Stuart Lloydin 1957 as a technique for pulsecode modulation. Sometimes it is referred as Lloyd-Forgy because in 1965, E.W.Forgy published essentially the same method.

[2] A more sophisticated baseline, widely used in VQA, consists on training a linear classifier or a multilayer perceptron using vectors representing a combination of the features as input. This combination can be a simple concatenation or a element wise sum or product of the features. The previous cited work experiments with two models: a multi-layer perceptron (MLP) neural network classifier with two hidden layers and 1000 hidden units (dropout 0.5) in each layer with tanh non-linearity, and an LSTM model followed by a softmax layer to generate the answer. In the first case, for the textual features they use a BOW approach, using the top 1,000 words in the questions and the 1,000 most popular words in the captions to compute them. For the image features they use the last hidden layer of VGGNet. As for the LSTM model, they use a one-hot encoding for the questions, and the same image features as above followed by a linear transformation to transform the image features to 1024 dimensions to match the LSTM encoding of the question. If the models are trained only on textual features, the accuracy is 48.09%, whereas if they are trained only on visual features it goes down to 28.13%. Their best model, a LSTM trained on both kind of features, has an accuracy of 53.74%.

[3] In Kafle and Kanan (2016), for example, the authors model the probability of image features given the question features and the type of the answer. They do so because they observe that given a question, the type of answer can be frequently predicted. For example, “How many players are in the image?” is a “how many” question, that needs a number as an answer. To model the probabilities they combine a Bayesian model with a discriminative model. Regarding the features, they use ResNet for the images and skip-thought vectors for text.

[4] The WUPS measure, proposed by Malinowski and Fritz in 2014, and based on the WUP measure by Wu and Palmer back in 1994, estimates the semantic distance between an answer and the ground truth, that is a value between 0 and 1. They rely on WordNet to compute similarity using the distance, in the semantic tree, of the terms contained both in the answer and the ground truth.

[5] In the evolving line of convolutional neural networks (CNNs) proposed for image feature extraction, LeNet, AlexNet, GoogLeNet VGG-Net and ResNet are representative and widely adopted networks. LeNet is a simple network structure with two convolutional layers and used to solved simple tasks such as handwritten digit recognition. AlexNet, a deeper network with 5 convolutional layers, was the first deep network to boost the classification accuracy by a significant stride and won the championship of ILSVRC2012 (ImageNet Large Scale Visual Recognition Challenge). It used ReLu (Rectified Linear Unit) as the activation function which can be trained much faster than sigmoid or tanh. Dropout was also adopted to avoid overfitting.

[6] Besides RNNs, convolutional neural networks (CNNs) were occasionally applied for feature extraction in VQA models, such as SANn, CNN-QAand MAVQA . As pointed out in , a pioneering work of applying CNN for sentence classification, CNN model is simple, easy to train and can even achieve promising performance. In the works of CNN-QA and MAVQA, CNN was also empirically compared with RNN in terms of textual feature extraction and demonstrated superior performance. Alternatively, we can also observe in several NLP tasks, RNN models were reported to be better than CNN. There also exist works, such as CRAN, which attempted to leverage the advantages of CNN and RNN and apply both of them for textual feature extraction.

2.5 Dataset

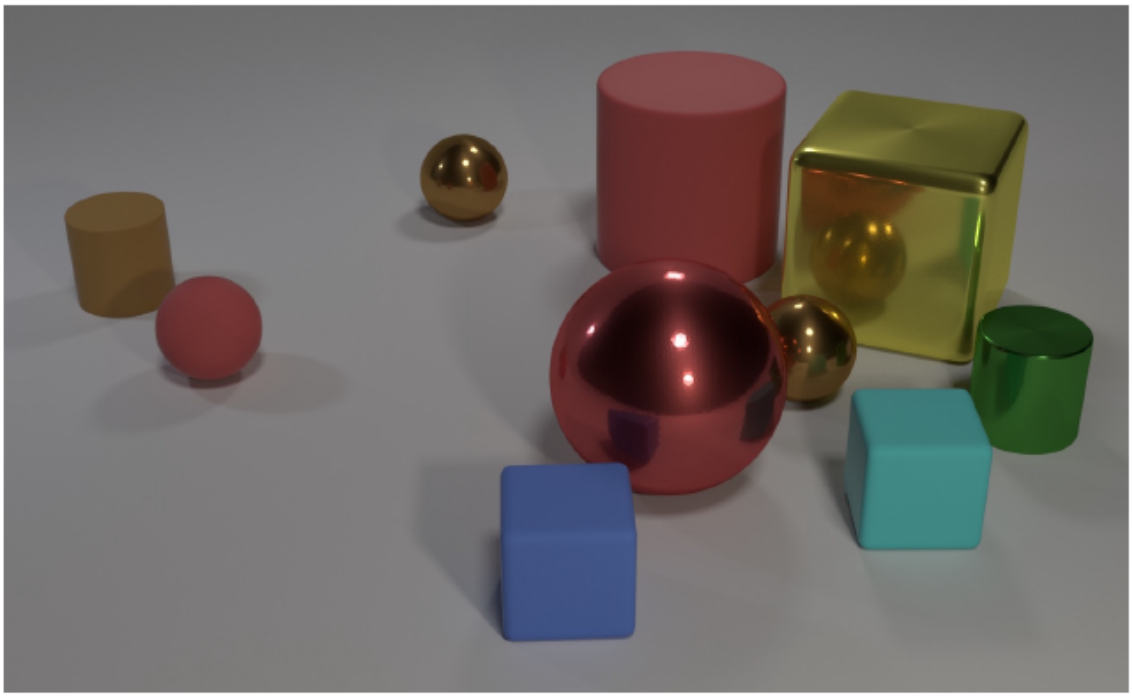
Visual Question Answering is a research area about building a computer system to answer questions presented in an image and a natural language. First of all, let's examine three datasets in **Visual Question Answering**.

VQA Dataset

Figure 6: VQA Dataset

In **VQA Dataset** , the computer system needs to address issues, such as, a binary classification problem (Is the umbrella upside down?), a counting problem (How many children are in the bed?), or an open-ended question (Who is wearing glasses? Where is the child setting?)

CLEVR Dataset

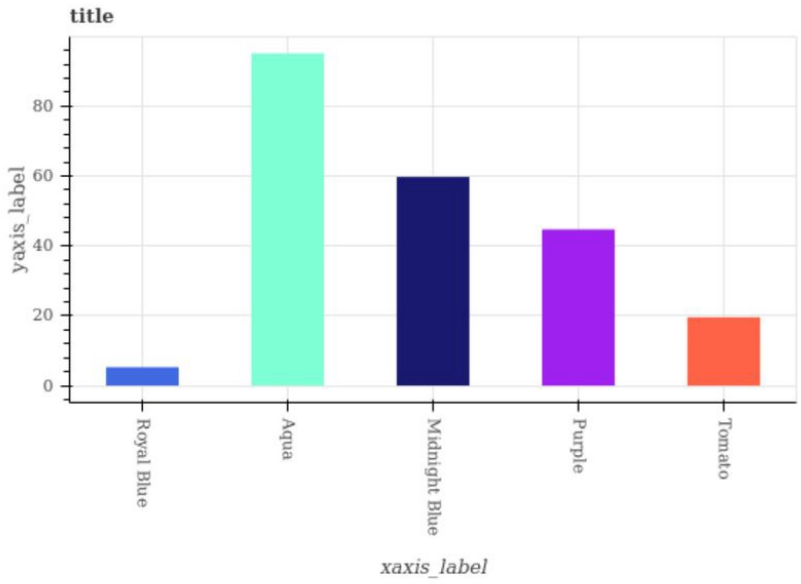


Q: Are there an equal number of large things and metal spheres?
Q: What size is the cylinder that is left of the brown metal thing that is left of the big sphere? **Q:** There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere?
Q: How many objects are either small cylinders or metal things?

Figure 7: CLEVR Dataset

In **CLEVR Dataset from Stanford**, the computer system needs to answer questions about the shape/color/size/material of the objects, and its spatial/logical relationship.

FigureQA Dataset



Q: Is Aqua the maximum?
A: Yes

Q: Is Midnight Blue greater than Aqua?
A: No

Q: Is Midnight Blue less than Aqua?
A: Yes

Q: Is Purple the high median?
A: Yes

Q: Is Tomato the low median?
A: No

Figure 8: FigureQA Dataset

In **FigureQA Dataset from Maluuba**, the computer system needs to answer questions presented by bar charts, pie charts, or line plots.

Chapter 3 PROPOSED SYSTEM

3.1 Problem statement

"To find the correct answer to a question posed based on an image using the technique of combination of language and vision via Keras, long short term memory, convolutional neural network and multilayer perceptron."

3.2 Problem Elaboration

This has been observed that finding the answers to questions based on an image correctly without inherent statistical bias on the dataset is a bit difficult. This leads to
answers based on the dataset bias, which give quite accurate results, but without considering the features of the image. Its solution can be addressed with the help of methodologies that include image feature extraction usingCNN and question semantic recognition and
understanding using LSTMs.

"To carry out the process of finding the correct answer to a question posed based on an image, we implement a python script using Keras, that encodes the question and image into vectors and then concatenates the two using MLP."

3.4 Objectives

- Understand the basics of the Visual Question Answering as they relate to image captioning; including natural language processing.
- To learn and group labeled data points together and discover underlying patterns.
- To derive efficient techniques for supervised learning on data. Understand CNN, RNN and Element wise summation.

3.5 Significance

- Some special type of spectacles can be made which will make use of computer vision (image captioning) along with neural networks to provide a virtual interface to answer questions based on image.
- Increasing Accuracy for answering various types of questions.
- Study and improve Deep learning techniques techniques in data mining algorithm.

Chapter 4 METHODOLOGY

4.1 Proposed Method

There are different methods in the language+vision domain to find the answer to the question posed based on the input image. But each of the methodologies has their pros and cons. To work effectively with the proposed framework and after an exhaustive comprehension of the given writing review, the proposed approach appears to be a reasonable fit for accomplishing best in class exactnesses.

To use the multilayer perceptron model with respect to visual question answering, all the input questions need to be transformed into image feature vectors of a fixed length and similarly for the question posed. Then we combine both these results and apply element wise addition in the multilayer perceptron to get the final answer result. The following steps are proposed:

1. The first step will be the word transformation. For the question, we will convert each word to its word vector, and then sum up all the vectors. We have to make sure that the length of this feature vector matches the length of a single word vector, and these word vectors are also called embeddings.

2. In the next step, these word vectors will be sent sequentially to the long short-term memory network, respective to the tokens in the question. The representation of the input question is actually the vector coming from the output gate of the long short-term memory.

3. Coming to the image, it is sent through a Deep Convolutional Neural Network (from the well-known VGG Architecture), and the image features are extracted from the activation of the second last layer (that is, the layer before the softmax function).

4. To combine the features from the image and the word vector, we use a multilayer perceptron consisting of fully connected layers. The final layer consists of the Softmax activation function, and thus we get a probability distribution over all the possible outputs. The output with the highest probability is our answer to the question posed based on the image.

Steps Followed in given Architecture:

- **Step 1**: import Keras
- **Step 2**: implement CNN for image recognition
- **Step 3**: implement RNN for natural language processing
- **Step 4**: combine the results from CNN and RNN to deliver the final answer.

Then the element-wise summation of these vectors results in the answer to the given example i.e. number of horses in the image.

Approaches based on attention

The goal of the attention-based approaches is to set the focus of the algorithm on the most relevant parts of the input. For example, if the question is "What color is the ball?", the image region containing the ball is more relevant than the others. In the same way, "color" and "ball" are more informative that the rest of the words.

The most common choice in VQA is to use spatial attention to generate region specific features to train CNNs. There are two common methods to obtain the spatial regions of an image. First, by projecting a grid over the image.

4.2 Proposed System Architecture

The proposed system workflow is as given as :

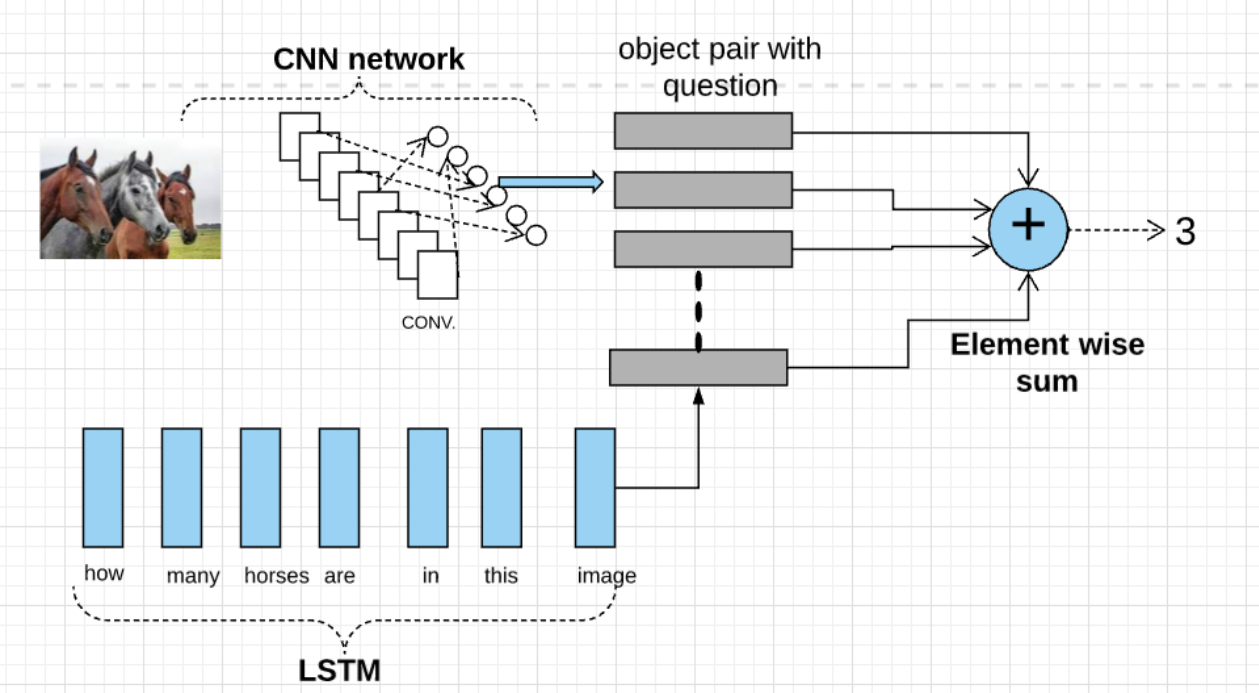


Figure 9: Work Flow

The above block diagram illustrates the architecture that we propose for the Visual Question Answering system. Here we import the **Keras** library to create a Convolutional Network layer for particular image and extract the required image features. LSTM as a part of RNN is used for natural language processing to convert the question into word vector, understanding its semantics. We merge the extracted image features and word vector and using MLP we create an (object,question) pair. Then the element-wise summation of these vectors results in the answer to the given example i.e. number of horses in the image.

Chapter 5

Implementation

In this chapter, the algorithm is selected, implemented and reviewed the analysis, in order to achieve the proposed system's objectives.

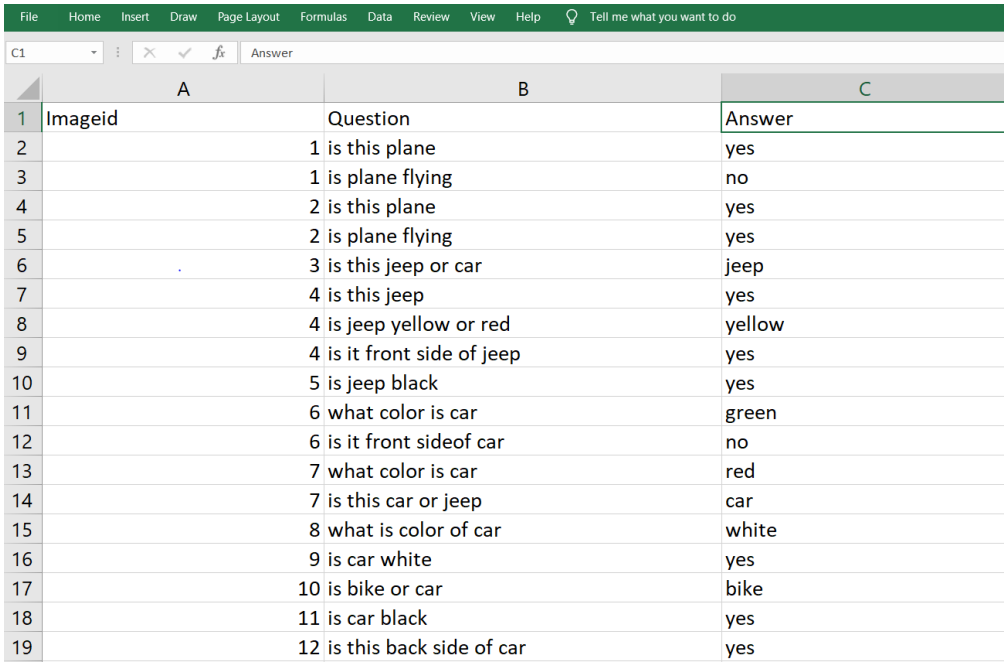
Requirements for implementations:
Libraries: Flask, os, werkzeug.utils
.
Dataset: Natural images

5.1 Building dataset

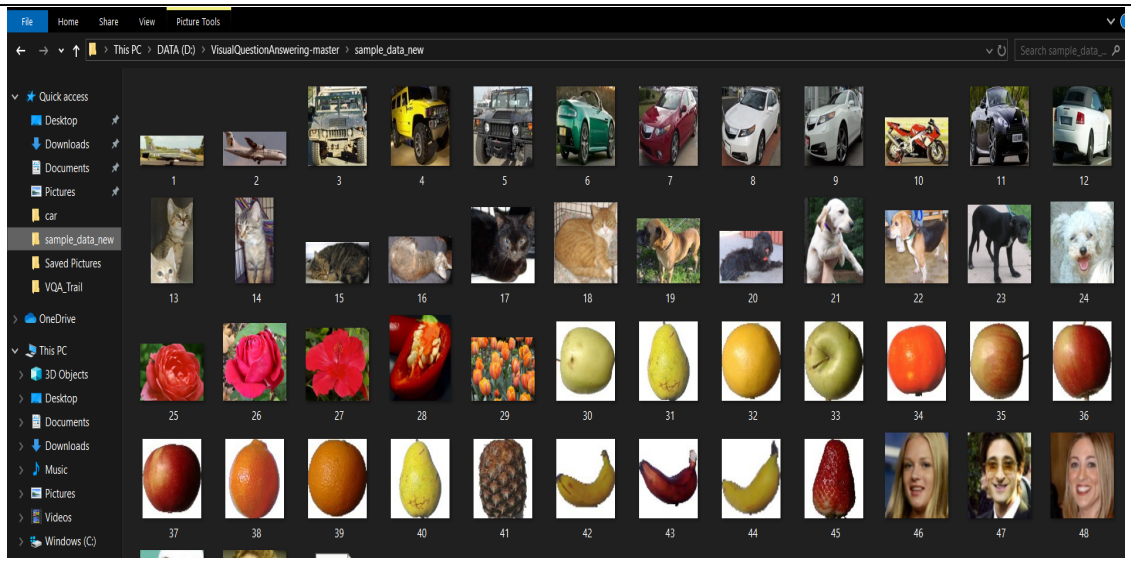
There are some dataset already existing on Kaggle but it would certainly be nice to construct our personal ones to test our own ideas and find the limits of what proposed system can and cannot achieve.

List of the steps required to get dataset

1. Download and collect set of natural images from various repositories
2. Make sure images have same extension(.jpg or .png)
3. Create training.csv for all types of natural images you want to select (Eg: Cars,Fruits,Flowers,etc)
4. Attributes such as IMG_ID, Question and answer mentioned in excel or .CSV file.



	A	B	C
1	Imageid	Question	Answer
2	1	is this plane	yes
3	1	is plane flying	no
4	2	is this plane	yes
5	2	is plane flying	yes
6	3	is this jeep or car	jeep
7	4	is this jeep	yes
8	4	is jeep yellow or red	yellow
9	4	is it front side of jeep	yes
10	5	is jeep black	yes
11	6	what color is car	green
12	6	is it front sideof car	no
13	7	what color is car	red
14	7	is this car or jeep	car
15	8	what is color of car	white
16	9	is car white	yes
17	10	is bike or car	bike
18	11	is car black	yes
19	12	is this back side of car	yes



5.2 Algorithm

Libraries: keras=2.1.0, Tensorflow=1.15, os, sklearn, cv2, spacy, numpy, vgg16

Algorithms:

1. Take pre-processed VGG-16 dataset weights, which categorize objects into 1000 categories. It has 16 layers and we pop the last 2 layers for use in our image object classification.
2. The image is converted to a corresponding (1, 4096) dimension vector by the VGG-16 Model
3. We use the spacy dataset for word-embeddings of our question tokens and convert it to a question tensor.
4. We have {{22}} trainY labels, so we convert it to categorical variables using to_categorical function.
5. Our final model has the following layers:

Layer (type)	Output Shape	Param #
merge_2 (Merge)	(None, 4608)	0
dense_5 (Dense)	(None, 1024)	4719616
activation_5 (Activation)	(None, 1024)	0
dropout_4 (Dropout)	(None, 1024)	0
dense_6 (Dense)	(None, 1024)	1049600
activation_6 (Activation)	(None, 1024)	0
dropout_5 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 1024)	1049600
activation_7 (Activation)	(None, 1024)	0
dropout_6 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 22)	22550

Total params: 12,704,790
Trainable params: 12,704,790
Non-trainable params: 0

6. We train the data by creating image features and question features by passing it through model.fit() function.

7. Later, we save the model architecture in a JSON file and the model weights in a H5 file.
8. In our working application, we load the architecture and weights using these saved files, and input image and corresponding question to get result.
9. The result is probabilistic values of the categories and we output the category with the maximum probability.

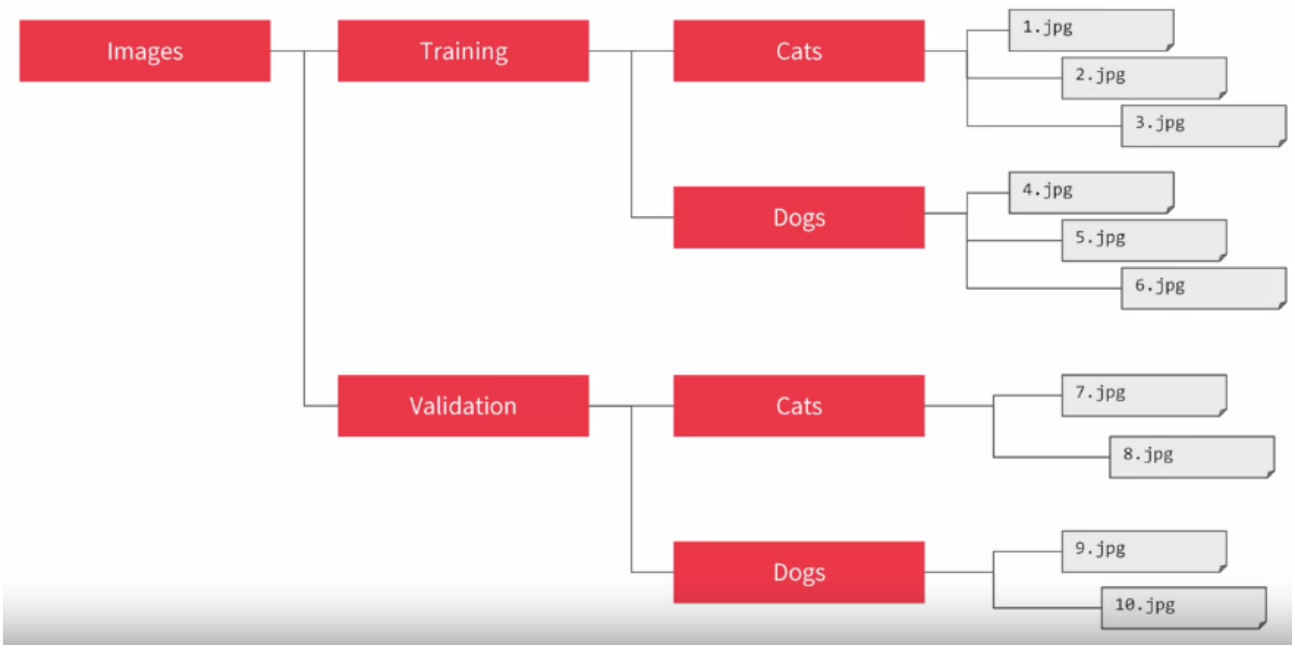
5.3 Experimental Implementation

VGG16

VGG16 is a convolution neural net (CNN) architecture. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC (fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx.) parameters.

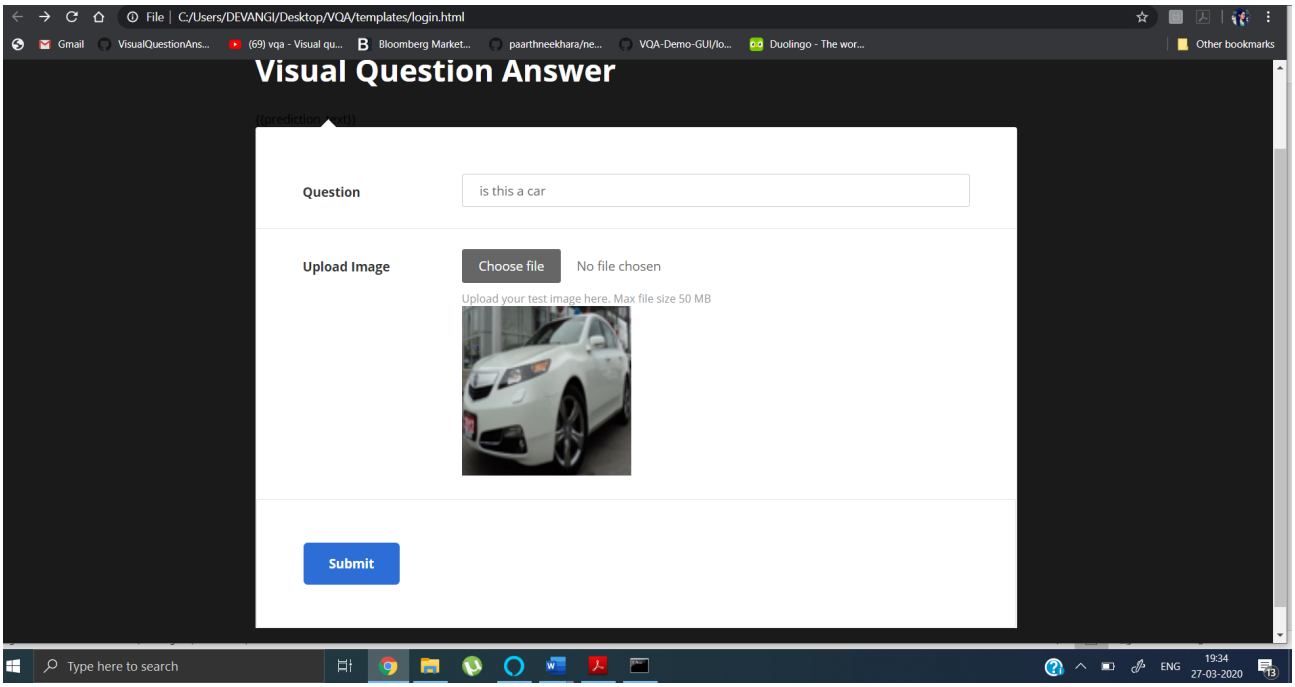
SpaCy

SpaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. It has the facilities for pre-trained word vector generation and integration with deep learning networks. SpaCy supports deep learning workflows that allow connecting statistical models trained by machine learning libraries like TensorFlow, Keras, Scikit-learn or PyTorch, few of which we are using in our implementation.

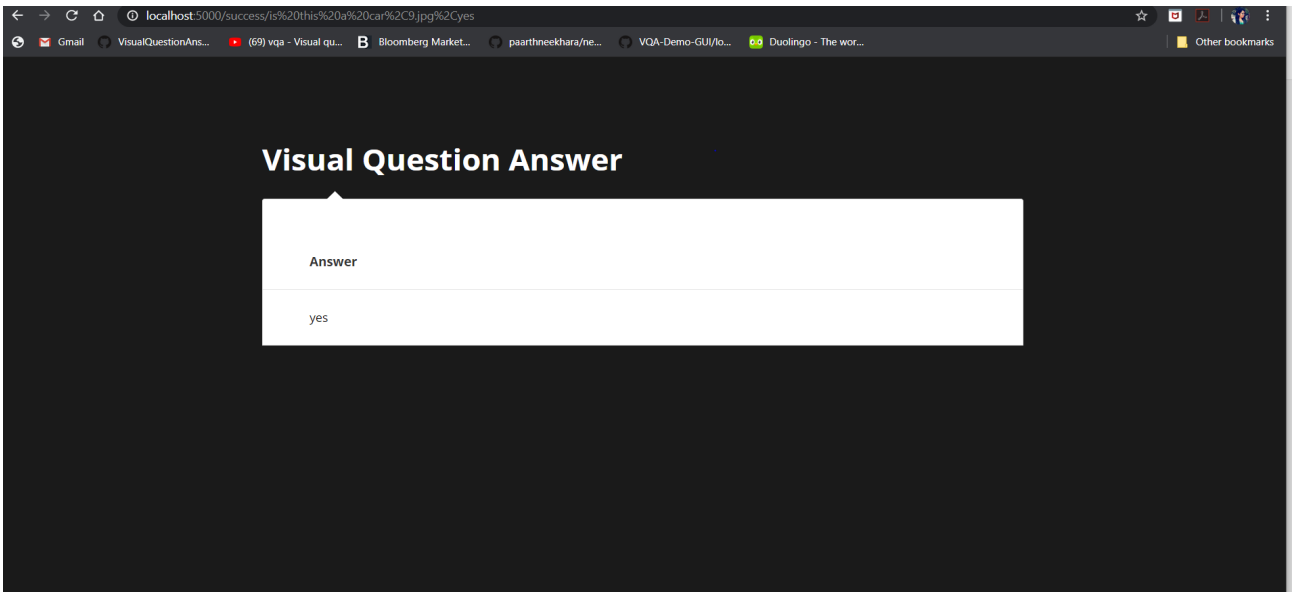


Screenshots:

Ask Question with selected image

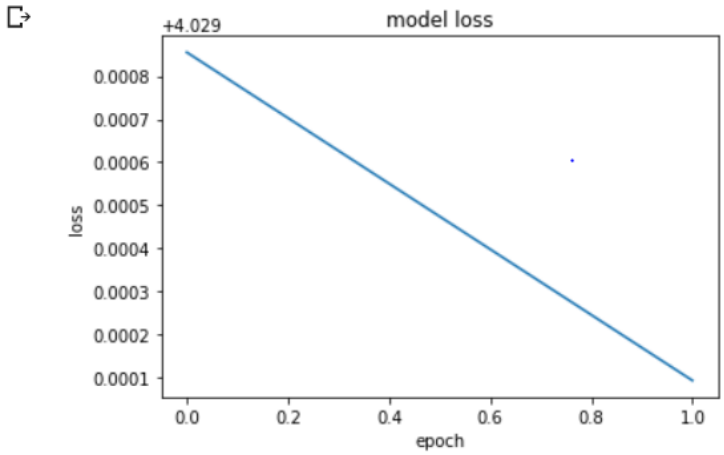


Answer using trained model:



Error Rate:

```
[41] import matplotlib.pyplot as plt
     # summarize history for accuracy
     plt.plot(history.history['loss'])
     plt.title('model loss')
     plt.ylabel('loss')
     plt.xlabel('epoch')
     plt.show()
```



Chapter 6

Future Scope:

Future datasets need to be larger. While VQA datasets have been growing in size and diversity, algorithms do not have enough data for training and evaluation. We did a small experiment where we trained a simple MLP baseline model for VQA using ResNet-152 image features and skip-thought features for the questions, and we assessed performance as a function of the amount of training data available on COCO-VQA. The results are shown where it is clear that the curve has not started to approach an asymptote. This suggests that even on datasets that are biased, increasing the size of the dataset could significantly improve accuracy. However, this does not mean that increasing the size of the dataset is su□cient to turn it into a good benchmark, because humans tend to create questions with strong biases. Future datasets need to be less biased. We have repeatedly discussed the problem of bias in existing VQA datasets in this paper, and pointed out the kinds of problems these biases cause for truly evaluating a VQA algorithm.

Chapter 7 CONCLUSIONS

In this paper, we have implemented methodologies for visual question answering using the pre-processed VGG-16 weights for image feature extraction and the Spacy word embeddings for question vector creation. We have implemented the techniques using Python Keras framework using Tensorflow as the backend. For the application part of the project, we have used the light-weight Flask web-framework. The validation set of the dataset exhibits a peak accuracy of 94 percent. This is better than the traditional models not using the convolutional and recurrent neural network deep learning techniques. However, there is a good amount of future scope to improve the accuracy on real-time data, using larger datasets and a wide range of questions along with attention modelling.

REFERENCES:

[1] Fang, Hao, Gupta, Saurabh, Iandola, Forrest, Srivastava, Rupesh K., Deng, Li, Dollar, Piotr, Gao, Jianfeng, He, Xiaodong, Mitchell, Margaret, Platt, John C., Lawrence Zitnick, C., and Zweig, Geoffrey. From captions to visual concepts and back. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[2] Antol, Stanislaw, Agrawal, Aishwarya, Lu, Jiasen, Mitchell, Margaret, Batra, Dhruv, Zitnick, C. Lawrence, and Parikh, Devi. Vqa: Visual question answering. In International Conference on Computer Vision (ICCV), 2015

[3] Chen, Xinlei, Fang, Hao, Lin, Tsung-Yi, Vedantam, Ramakrishna, Gupta, Saurabh, Dollar, Piotr, and Zitnick, C Lawrence. Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325, 2015.

[4] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[5] Yin and Yang: Balancing and Answering Binary Visual Questions (CVPR 2016)

[6] VQA: Visual Question Answering by Aishwarya Agrawal , Jiasen Lu , Stanislaw Antol ,Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh

[7] Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, Devi Parikh, Virginia Tech, Army Research Laboratory ,Georgia Institute of Technology.

[8]https://towardsdatascience.com/deep-learning-and-visual-question-answering-c8c8093941bc

[9]https://visualqa.org/

[10] https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering/

[11] Image Captioning and Visual Question Answering Based on Attributes and External Knowledge Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, Anton van den Henge

[12] Learning to Reason: End-to-End Module Networks for Visual Question Answering ICCV 2017 Ronghang Hu Jacob Andreas Marcus Rohrbach Trevor Darrell Kate Saenko

[13] Graph-Structured Representations for Visual Question Answering CVPR 2017 Damien Teney, Lingqiao Liu, Anton van den Hengel
[14]BLOG from Andrej Karpathy **here.**

[15] https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f

[16] http://www.datamind.cz/cz/vam-na-miru/umela-inteligence-a-strojove-uceni-ai-machine-learning

[17] https://en.wikipedia.org/wiki/Artificial_neural_network

[18] https://en.wikipedia.org/wiki/Deep_learning

[19] https://en.wikipedia.org/wiki/Convolutional_neural_network

[20]https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[21] https://www.lynda.com/Google-TensorFlow-tutorials/Building-Deep-Learning-Applications-Keras-2-0/601801-2.html

[22] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html https://keras.io