**Attention Mechanism in Deep Learning**

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in** **COMPUTER SCIENCE AND ENGINEERING**

BY

**Abhishek Tyagi**

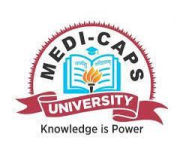
**Enrollment Number:**

EN18CS301009

**Under the Guidance of:**

**Dr. Sourabh Das (Assistant Professor IIT Indore)**

**Mr. Ashish Kumawat (Professor Medi-Caps Indore)**



**Department of Computer Science & Engineering Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**May 2022**

**Report Approval**

The project work “**Attention Mechanism in Deep Learning**” is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed, or conclusion drawn therein; but approve the “Project Report '' only for the purpose for which it has been submitted.

Internal Examiner

Name: Mr. Ashish Kumawat

Designation: Professor

Affiliation: Computer Science Engineering, Medi-Caps University, Indore

External Examiner

Name:

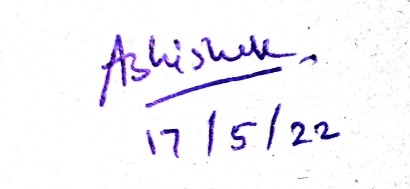
Designation

Affiliation

**Declaration**

I/We hereby declare that the project entitled **“Attention Mechanism in Deep Learning”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science Engineering’ completed under the supervision of **Mr.** **Ashish Kumawat, Professor,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.



Abhishek Tyagi

EN18CS301009

**Certificate**

I/We, **Dr. Sourabh Das (Assistant Professor IIT Indore), Mr. Ashish Kumawat (Professor Medi-Caps indore)** certify that the project entitled **“Attention Mechanism in Deep Learning”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology by **Abhishek Tyagi** istherecordcarried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Ashish Kumawat Dr. Sourabh Das

Computer Science Engineering Astronomy, Astrophysics and Space Engineering

Medi-Caps University, Indore IIT, Indore

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

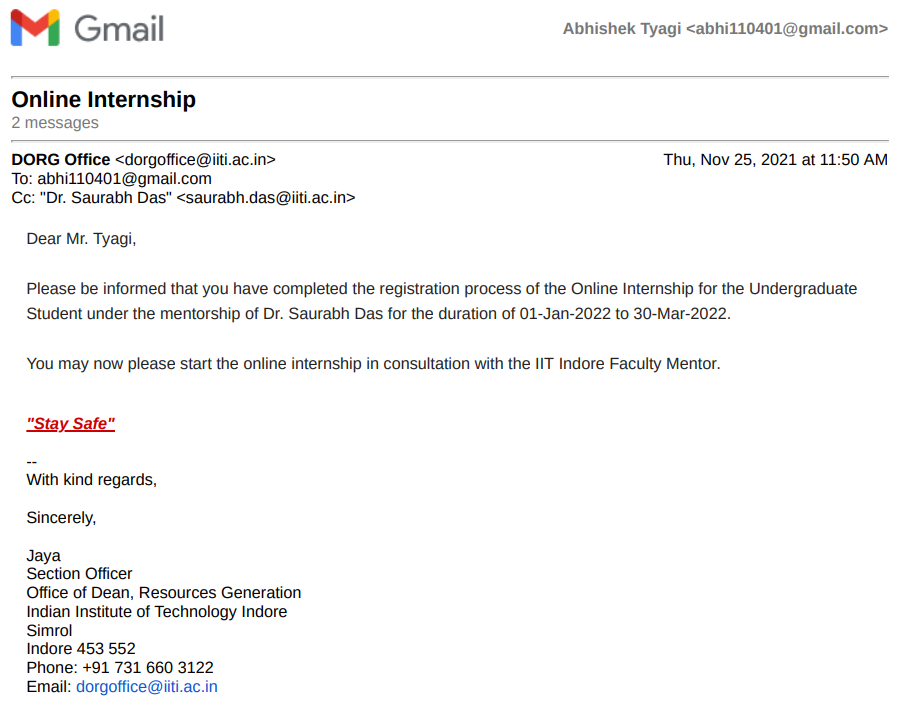
Dr. Pramod S. Nair

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

**Offer Letter of the Project work-II**



Offer letter of Abhishek Tyagi

**Completion Certificate**



Completion certificate of Abhishek Tyagi

**Acknowledgements**

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Dr. Dilip K. Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Dr. D.K. Panda,** Pro Vice Chancellor, **Dr. Suresh Jain,** DeanFaculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod S. Nair** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide, Dr. Sourabh Das (Assistant Professor IIT Indore)** as well as to my Internal Guide, **Mr. Ashish Kumawat (Professor Medi-Caps Indore)**, Dept. of Computer Science Engineering, MU without whose continuous help and support, this project would ever have reached its completion

I would also like to thank my team at Abhinav Mukerji, Abhishek Tyagi, Aadarsh Rawal who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report. Without their support this report would not have been possible.

**Abhishek Tyagi**

B.Tech. IV Year

Department of Computer Science & Engineering

Faculty of Engineering

Medi-Caps University, Indore

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S. NO.** | **CONTENTS** | **PAGE NO.** |
|  | Report Approval | ii |
|  | Declaration | iii |
|  | Certificate | iv |
|  | Offer Letter of the Project work-II | v |
|  | Completion certificate | vi |
|  | Acknowledgements | vii |
|  | Abbreviations | xi |
|  | Notations & Symbols | xiii |
|  | Abstract | xv |
| Chapter-1 | Introduction | 1 |
| 1.1 | Introduction | 1 |
|  | 1.1.1 Artificial Neural Network | 1 |
|  | 1.1.2 Convolutional Neural Network | 1 |
|  | 1.1.3 Recurrent Neural Network | 3 |
|  | 1.1.4 Long Short-Term Memory | 4 |
|  | 1.1.5 Gated Recurrent Unit | 5 |
| 1.2 | Literature Review | 7 |
| 1.3 | Objective | 8 |
| 1.4 | Significance | 9 |
| 1.5 | Research Design | 10 |
| Chapter-2 | Setup and the procedure adopted | 11 |
| 2.1 | Experimental Setup | 11 |
| 2.2 | Procedure Adopted | 12 |
| Chapter-3 | Attention Mechanism | 13 |
| 3.1 | Components of attention mechanism | 13 |
|  | 3.1.1 Convolutional Neural Network | 13 |
|  | 3.1.2 Long Short-Term Memory | 13 |
|  | 3.1.3 Gated Recurrent Unit | 14 |
| Chapter-4 | Implementation of Attention Mechanism | 16 |
| Chapter-5 | Result and Discussion | 20 |
| Chapter-6 | Summary and Conclusion | 22 |
| Chapter-7 | Future Scope | 23 |
|  | Appendix | 24 |
|  | Bibliography/References | 35 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **S. NO.** | **CONTENTS** | **PAGE NO.** |
| 1.1.1 | A Basic RNN | 1 |
| 1.1.2. a) | Working of CNN | 2 |
| 1.1.2. b) | Example of CNN | 3 |
| 1.1.3 | A common RNN network | 4 |
| 1.1.4. a) | An LSTM Representation | 5 |
| 1.1.4. b) | Internal representation of LSTM | 5 |
| 1.1.4. c) | Steps of LSTM | 5 |
| 1.1.5. a) | A GRU Representation | 6 |
| 1.1.5. b) | Notations in GRU | 6 |
| 1.5 | Timeline in Research and Design | 10 |
| 3.1.1 | Overview of CNN | 13 |
| 3.1.2 | LSTM | 14 |
| 3.1.3 | GRU | 14 |
| 4.1 | An Attention Mechanism | 16 |
| 4.2 | BLEU Score Observations | 16 |
| 4.3 | Representation of Attention Mechanism | 17 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **S. NO.** | **CONTENT** | **PAGE NO.** |
| 1. | Notations and Symbols | xv |
| 2. | Accuracy of different experiments | 20 |

**Abbreviations**

**Neuron-** Just like a neuron forms the basic element of our brain, a neuron forms the basic structure of a neural network. Just think of what we do when we get new information. When we get the information, we process it and then we generate an output. Similarly, in the case of a neural network, a neuron receives an input, processes it and generates an output which is either sent to other neurons for further processing or it is the final output.

**Weights** – When input enters the neuron, it is multiplied by a weight. For example, if a neuron has two inputs, then each input will have an associated weight assigned to it. We initialize the weights randomly and these weights are updated during the model training process.

**Bias** – In addition to the weights, another linear component is applied to the input, called the bias. It is added to the result of weight multiplication to the input. The bias is basically added to change the range of the weight multiplied input.

**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.

**RNN** - A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior.

**CNN** - A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

**Attention** - attention is the cognitive process of selectively concentrating on one or a few things while ignoring others.

**LSTM** - **Long short-term memory** (**LSTM**) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network can process not only single data points (such as images), but also entire sequences of data (such as speech or video).

**Epochs** – An epoch is defined as a single training iteration of all batches in both forward and back propagation. This means 1 epoch is a single forward and backward pass of the entire input data.

**Filters** – A filter in a CNN is like a weight matrix with which we multiply a part of the input image to generate a convoluted output. Let’s assume we have an image of size 28\*28. We randomly assign a filter of size 3\*3, which is then multiplied with different 3\*3 sections of the image to form what is known as a convoluted output.

**Pooling** – It is common to periodically introduce pooling layers in between the convolution layers. This is basically done to reduce a number of parameters and prevent over-fitting.

**Padding** – Padding refers to adding an extra layer of zeros across the images so that the output image has the same size as the input. This is known as Padding.

**Data Augmentation** – Data Augmentation refers to the addition of new data derived from the given data, which might prove to be beneficial for prediction.

**Gradient Descent** – Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimize a cost/loss function (e.g. in a linear regression).

**Vanishing Gradient Problem** – Vanishing gradient problem arises in cases where the gradient of the activation function is very small. During back propagation when the weights are multiplied with these low gradients, they tend to become very small and “vanish” as they go further deep in the network.

**Notations & Symbols**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Name** | **Notation/Symbol** |
| 1. | Neuron (Biological vs Computational) |  |
| 2. | Weights |  |
| 3. | Activation function |  |
| 4. | Input / Output / Hidden Layer |  |
| 5. | Gradient Descent |  |
| 6. | Recurrent Neuron |  |

Table 1. Notations and Symbols

**Abstract**

A neural network is considered to be an effort to mimic human brain actions in a simplified manner. Attention Mechanism is also an attempt to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks.

Understanding of Attention mechanism requires study of neural networks and its various sub-parts like Convolutional neural network, Recurrent neural network, and their various types. The Implementation of a basic neural network, a handwritten digit identifier, Implementation of Convolutional neural network for Image Classification, and finally the application of Attention Mechanism for Language Translation has been implemented.

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoders–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. Here, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-Bengali translation.

The results of respective code pieces have been illustrated for a better understanding of what “new” each progressive model is portraying. We have used Google Collab to perform the implementation.

**Chapter-1**

**1.1 Introduction**

**1.1.1 ANN (Artificial Neural Network)**

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems inspired by the [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network) that constitute animal [brains](https://en.wikipedia.org/wiki/Brain).

An ANN is based on a collection of connected units or nodes called [artificial neurons](https://en.wikipedia.org/wiki/Artificial_neuron), which loosely model the [neurons](https://en.wikipedia.org/wiki/Neuron) in a biological brain. Each connection, like the [synapses](https://en.wikipedia.org/wiki/Synapse) in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal then processes it and can signal neurons connected to it.

The "signal" at a connection is a [real number](https://en.wikipedia.org/wiki/Real_number), and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a [*weight*](https://en.wikipedia.org/wiki/Weighting) that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection.

Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.



Fig 1.1.1 A basic ANN

**1.1.2 CNN (Convolutional Neural Network)**

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network (CNN/ConvNet) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution.

Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

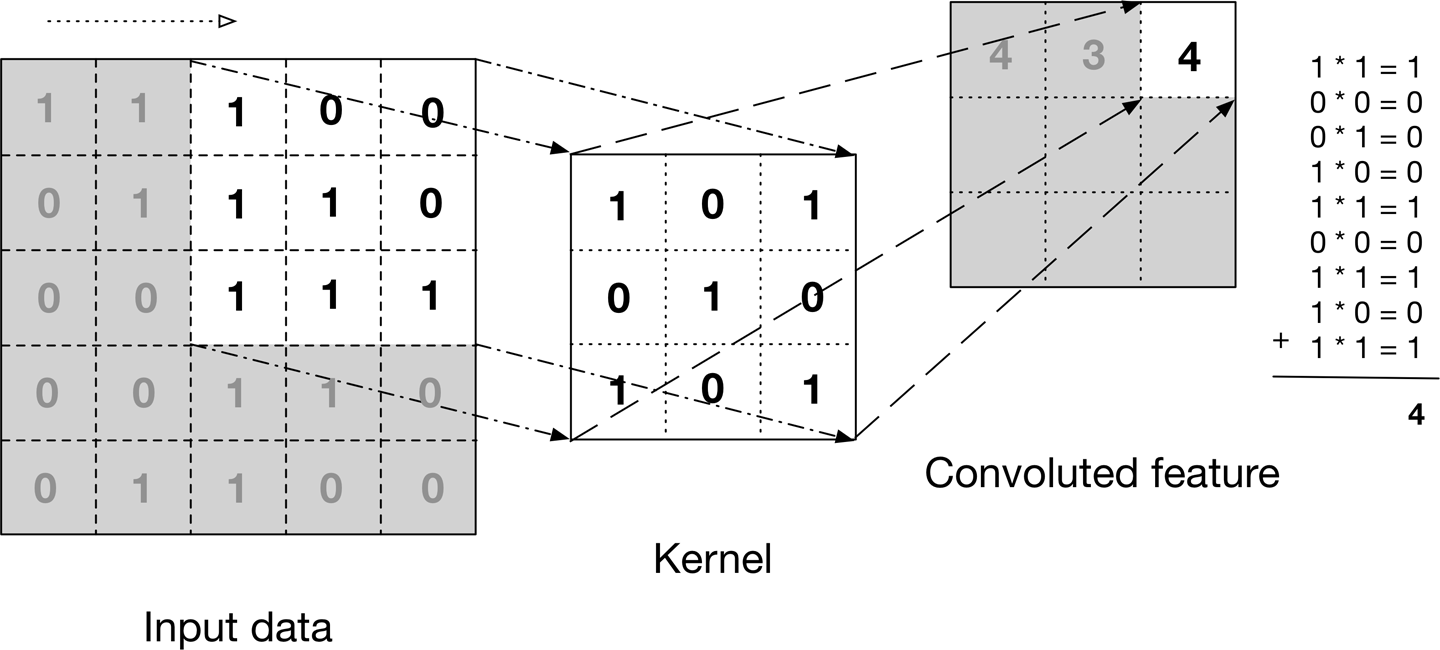
****

Fig 1.1.2.a) Working of CNN

The above image shows what a convolution is. We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value.

When you input an image in a ConvNet, each layer generates several activation functions that are passed onto the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

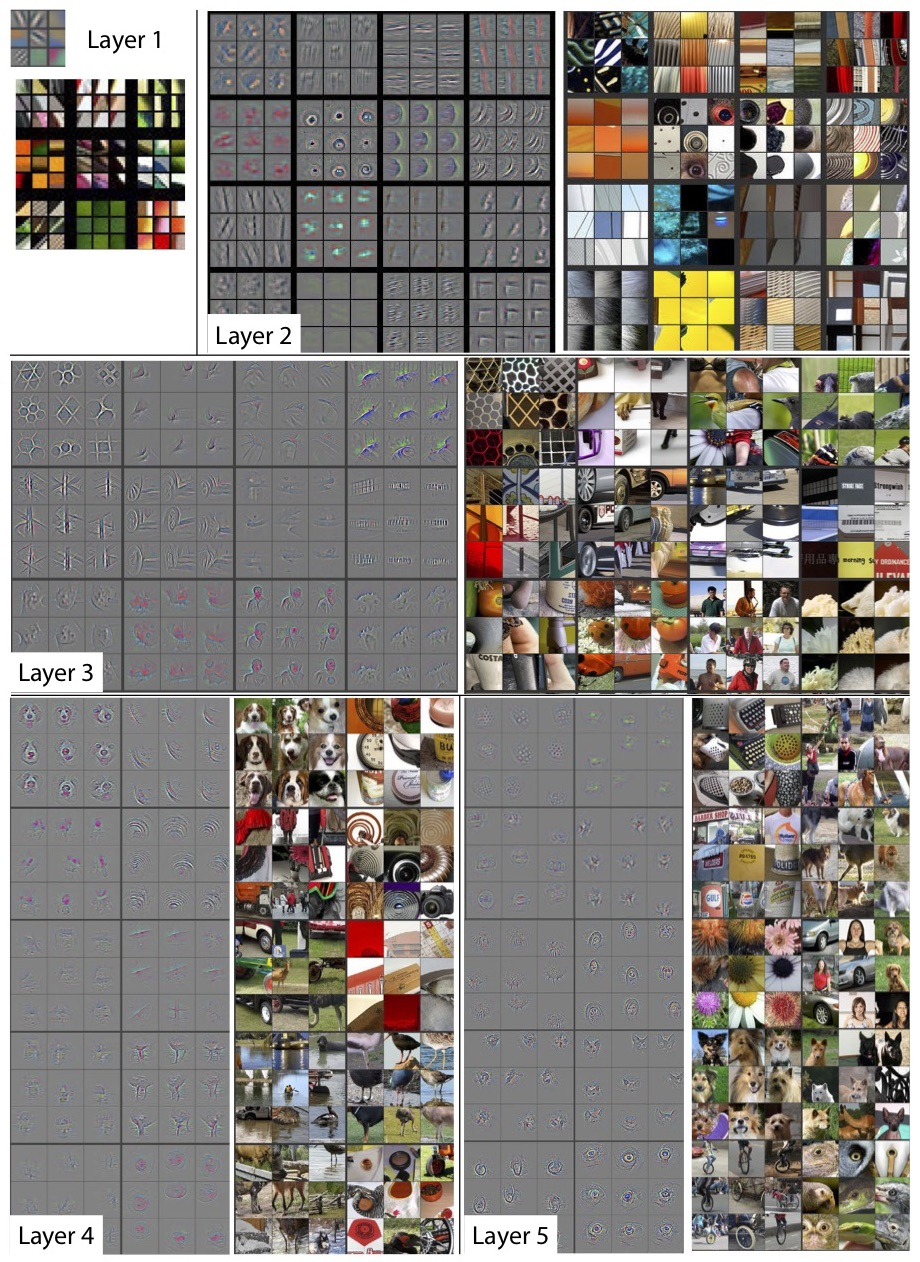
****

Fig 1.1.2.b) Example of CNN

**1.1.3 RNN (Recurrent Neural Network)**

A recurrent neural network (RNN) is a special type of an artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feed forward neural networks are only meant for data points, which are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points.

RNNs have the concept of ‘memory’ that helps them store the states or information of previous inputs to generate the next output of the sequence.

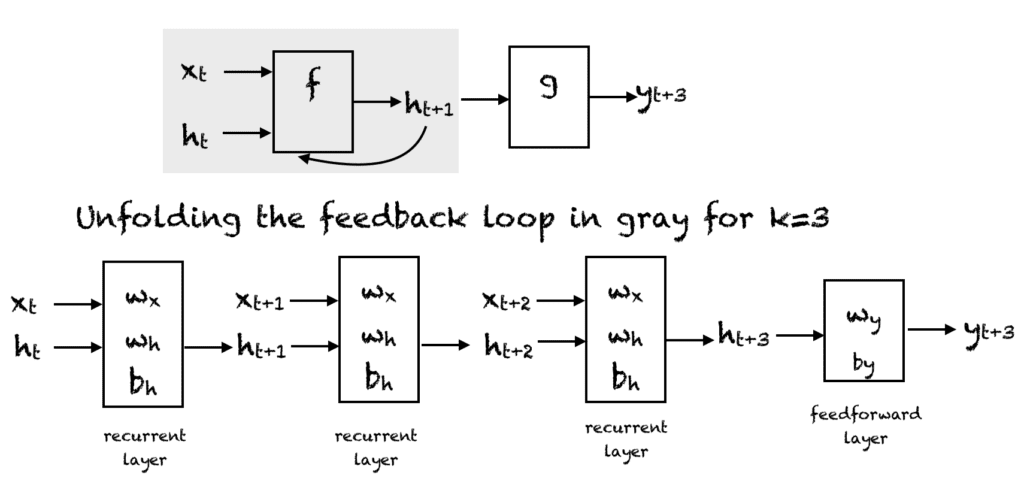


Fig 1.1.3 A common RNN network

A simple RNN has a feedback loop as shown in the first diagram of the above figure. The feedback loop shown in the gray rectangle can be unrolled in 3 time steps to produce the second network of the above figure.

Of course, you can vary the architecture so that the network unrolls k time steps.

**1.1.4 LSTM (Long Short-Term Memory)**

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.

They were introduced by [Hochreiter & Schmidhuber (1997)](http://www.bioinf.jku.at/publications/older/2604.pdf), and were refined and popularized by many people in following work.[1](https://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1) They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

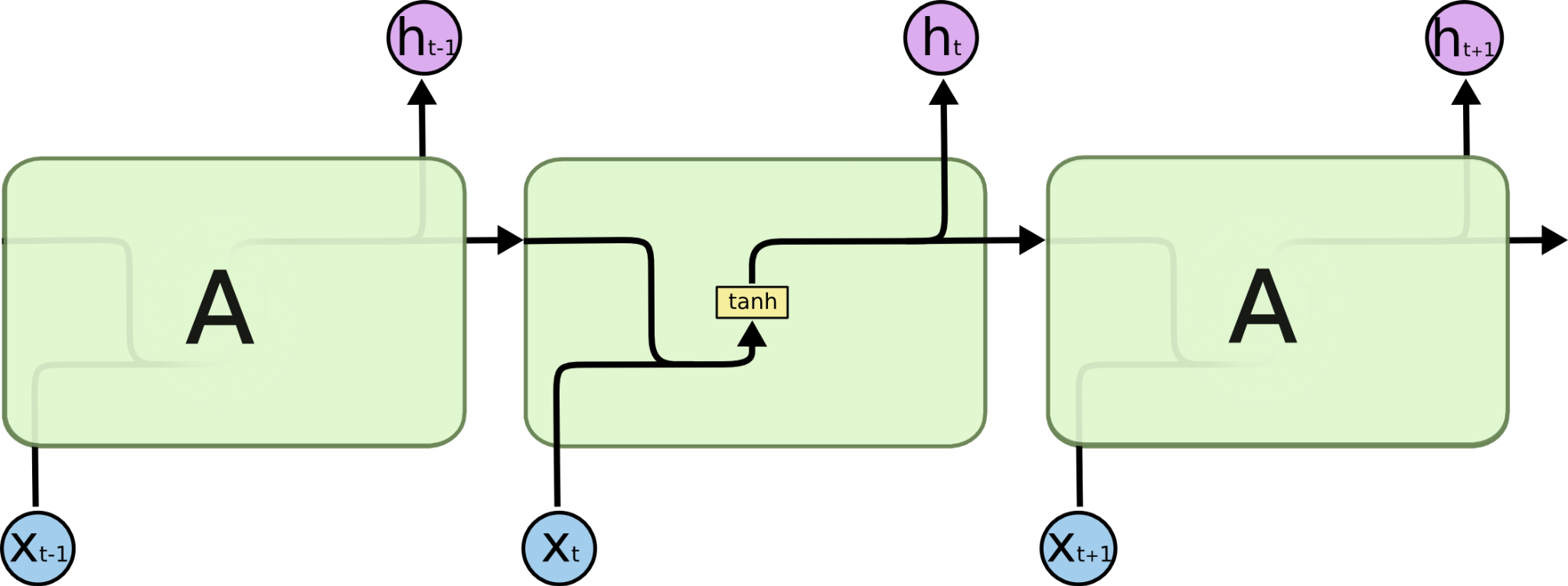


Fig 1.1.4.a) An LSTM representation

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way:

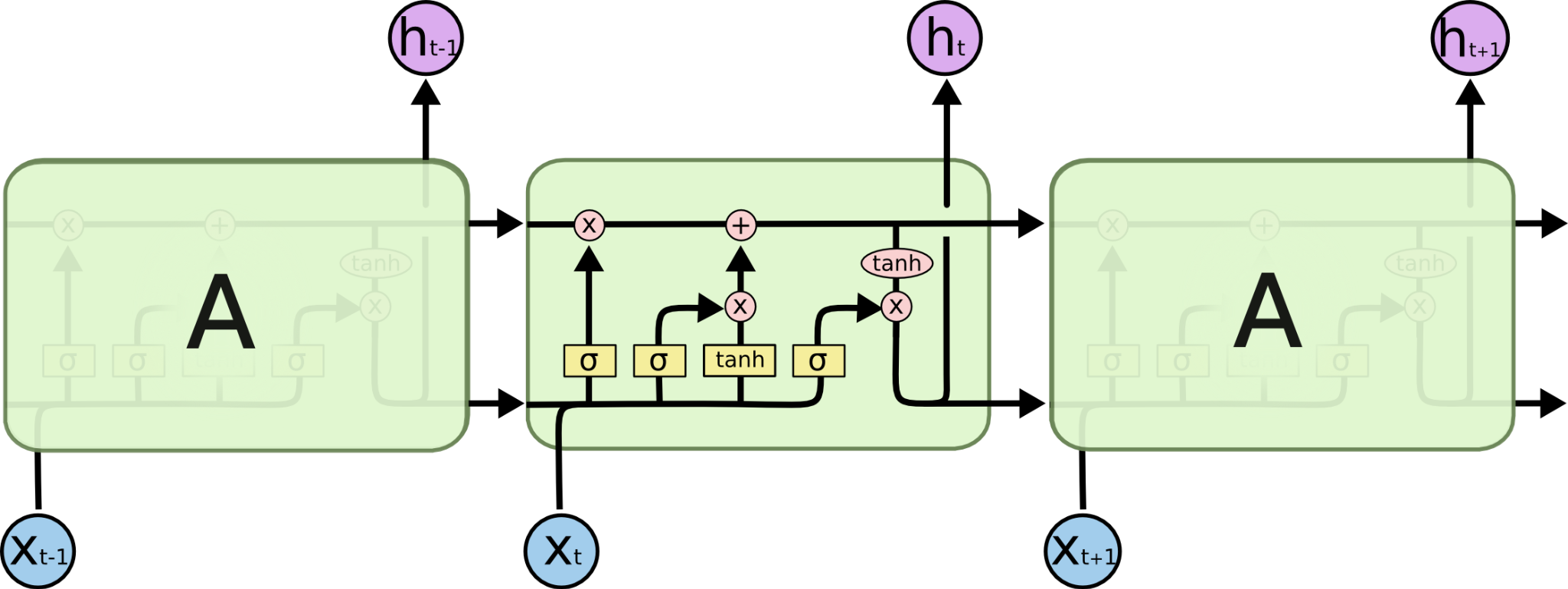


Fig 1.1.4.b) Internal representation of LSTM

Don’t worry about the details of what’s going on. We’ll walk through the LSTM diagram step by step later. For now, let’s just try to get comfortable with the notation we’ll be using.

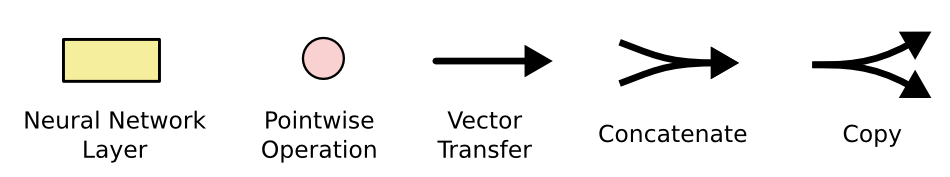


Fig 1.1.4.c) Steps of LSTM

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

**1.1.5 GRU (Gated Recurrent Unit)**

To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or removing information which is irrelevant to the prediction.

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.

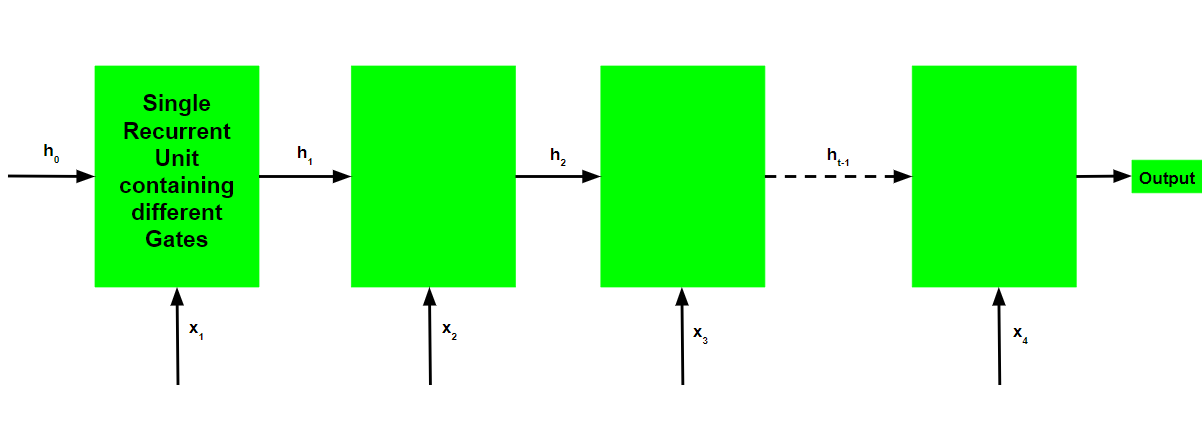


Fig 1.1.5.a) A GRU Representation

Introduction to the notations:

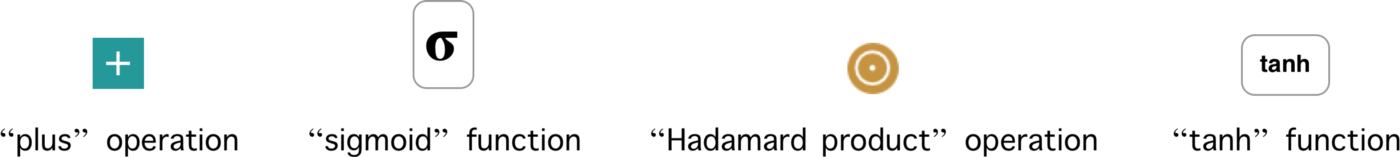


Fig 1.1.5.b) Notations in GRU

The different gates of a GRU are as described below: -

**Update Gate(z)**: It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit.

**Reset Gate(r)**: It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.

**Current Memory Gate**: It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some nonlinearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the futur

**1.2 Literature Review**

Attention Mechanism as a topic has gained more significance in the last 7-8 years, specially to sudden surge of growth in the Deep learning domain. Now, attention mechanism is not a standalone concept, but it is amalgamation of many smaller topics like RNN(Recurrent Neural Network), CNN(Convolutional Neural Network) ,LSTM(Long short-term memory) and GRU(gated Recurrent Unit). These concepts form the basis of attention mechanisms . This can be well understood after studying the research paper “Neural machine translation by jointly learning to align and translate”.

In neural machine translation, we fit a parameterized model to maximize the conditional probability of sentence pairs using a parallel training corpus. Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability

Our work is primarily based on the GRU as a basic unit, rather than using the LSTM unit. It is so because the LSTM units can be even more compressed to form the GRU unit, and GRU’s are much easier to handle. The research paper has worked with LSTM as a basic unit, but results will be more or less the same. In fact GRU seems to perform better.

Despite being a quite new approach, neural machine translation has already shown promising results. Sutskever et al. (2014) reported that the neural machine translation based on RNNs with long short term memory (LSTM) units achieves close to the state-of-the-art performance of the conventional phrase-based machine translation system on an English-to-bengali translation task.

**1.3 Objective**

The primary objective of this internship report is to successfully implement the “Attention Mechanism” and its underlying concepts like ANN(Artificial Neural Network), CNN(Convolutional Neural Network) and GRU(gated Recurrent Unit).

The motivation to understand attention mechanisms comes from the fact that it addresses the issue of not remembering the context properly in case of long sentences. This problem used to arise in the case of using traditional RNN (Recurrent neural network) units as a basic unit in language translation.

Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words - This forms the basis of Attention Mechanism.

**1.4 Significance**

As Attention Mechanism has been existing for quite some time now, we have tried to experiment with the basic components used in a traditional Attention Model. We have basically replaced the traditional LSTM (Long Short-Term Memory) with the GRU (Gated Recurrent Unit) in order to simplify the process of implementing Attention mechanism.

We have made a language translator from English to bengali language, as it is the need of the hour. Due to the increase in the number of people using the internet, and the ease with which people can access the data, the need for language translation is always on the rise,and it’s not going to stop anywhere soon. So we have tried to address a very small chunk of this problem with the help of the concept of “Attention Mechanism”.

We have used an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

**1.5 Research Design**

**Week 1 Week 2 Week 3 Week 4 Week 5 Week 6 Week 7 Week 7**

**Initial discussion of topic with mentor**

**Implementation of NN, and handwritten digit identifier using ANN.**

**Study of RNN,LSTM and GRU and their pros and cons.**

**Implementing Attention Mechanism for Language Translation.**

**Basic Information about Artificial neural network and study about weights,neurons and activation functions.**

**Study and Implementation of CNN for image classification**

**Understanding the architecture of Attention Mechanism.**

**Fig 1.5 Timeline of Research & Design**

Our Objective is to understand the concept of “Attention Mechanism” with the help of GRU, which is a relatively new phenomenon in the field of Deep Learning. The underlying concepts, and its functioning is our primary area of focus.

Understanding of topics like:

1. Neural network
2. RNN
3. CNN
4. CNN vs RNN
5. Feedforward vs RNN
6. What is Attention?
7. Application of CNN
8. Application of Attention Mechanism

We will implement the following:

1) Simple Artificial Neural Network

2)Implementing ANN for handwritten digit prediction.  
3) Image Classification using CNN

4) Understanding the working of Attention Mechanism.

**Chapter-2**

**2.1 Experimental Setup**

* **Setting up of Environment**

For the development environment we have the options of local compute power and online computer power systems both paid and free, local compute being a limitation we opted to go for the Online Computer System namely Google Colab also known as Google Colaboratory provided as a free and paid service differentiating in computation power.

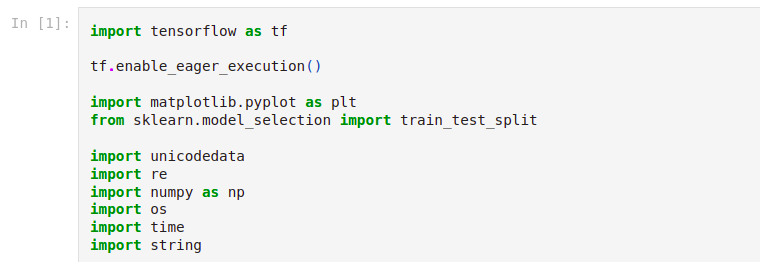
https://research.google.com/colaboratory/

For the setup of Python Environment follow these steps using conda package manager:

>> conda create -n <Name of Env> <Packages example:python=3.7>

Python libraries and packages required imported accordingly :

* tensorflow
* sklearn.model\_selection {for splitting test and training data}
* numpy
* string
* re {regex module }

****

**2.2 Procedure Adopted**

We have adopted Agile Methodology in our project, as it is one of the most widely used methods in the professional space. The meaning of Agile is swift or versatile. **“Agile process model**" refers to a software development approach based on iterative development.

Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

**1. Requirements gathering:** In this phase,we defined the requirements. We estimated the time and effort needed to build the project. Based on this information,we evaluated technical and economic feasibility.

**2. Design the requirements:** After identifying the project, we worked with stakeholders(the faculty) to define requirements, and started to study about the prerequisites.

**3. Construction/ iteration:** When the team defines the requirements, the work begins.The product went under various stages of improvement, such that it includes simple, minimal functionality.

**4. Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

**5. Feedback:** After making the final code, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

**Chapter-3**

**3.1 Components of Attention Mechanism**

Artificial neural networks (ANNs) are composed of a node layer, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

**3.1.1 Convolutional Neural Network**

In Deep Learning, Convolutional Neural Network (CNN) is a type of an Artificial Neural Network. CNN or ConvNet is a class of deep, feed-forward artificial neural systems, most normally connected to examining visual representations.

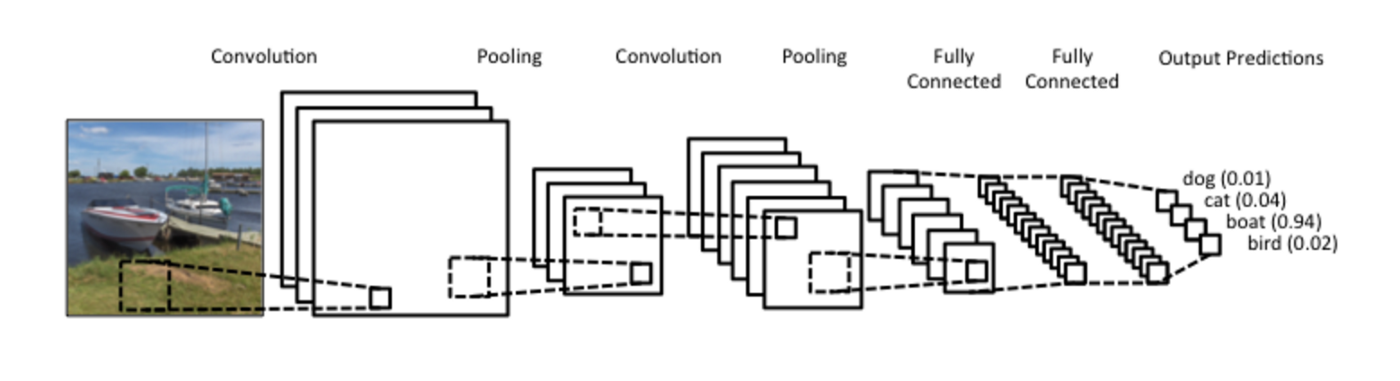


Fig. 3.1.1 Overview of CNN

CNN is a sequence of layers and every layer converts one volume of activations to another through a differentiable function. We use three main types of layers to build CNN architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer.**

**3.1.2** **Long Short-term Memory (LSTM)**

The data or information is not persistence for traditional neural networks but as they don’t have that capability of holding or remembering information but with Recurrent Neural Networks it’s possible as they are the networks which have loops in them and so they can loop back to get the information if the neural network has already processed such information.

LSTMS are a special kind of RNN which is capable of learning long-term dependencies. LSTM are designed to dodge long-term dependency problems as they are capable of remembering information for longer periods of time.

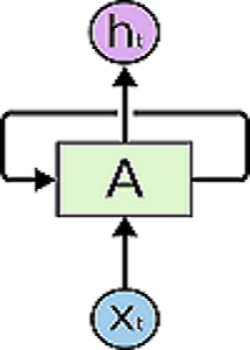


Fig. 3.1.2 LSTM

**3.1.3** **Gated recurrent unit (GRU)**

The Gated Recurrent Unit is a new gating mechanism introduced in 2014, it is a newer generation of RNN. GRU is similar to LSTM and has shown that it performs better on smaller datasets. Unlike LSTM, GRU has only two gates, a reset gate and an update gate and they lack output gate. GRU’s got itself free of the cell state and instead uses the hidden state to transfer information.

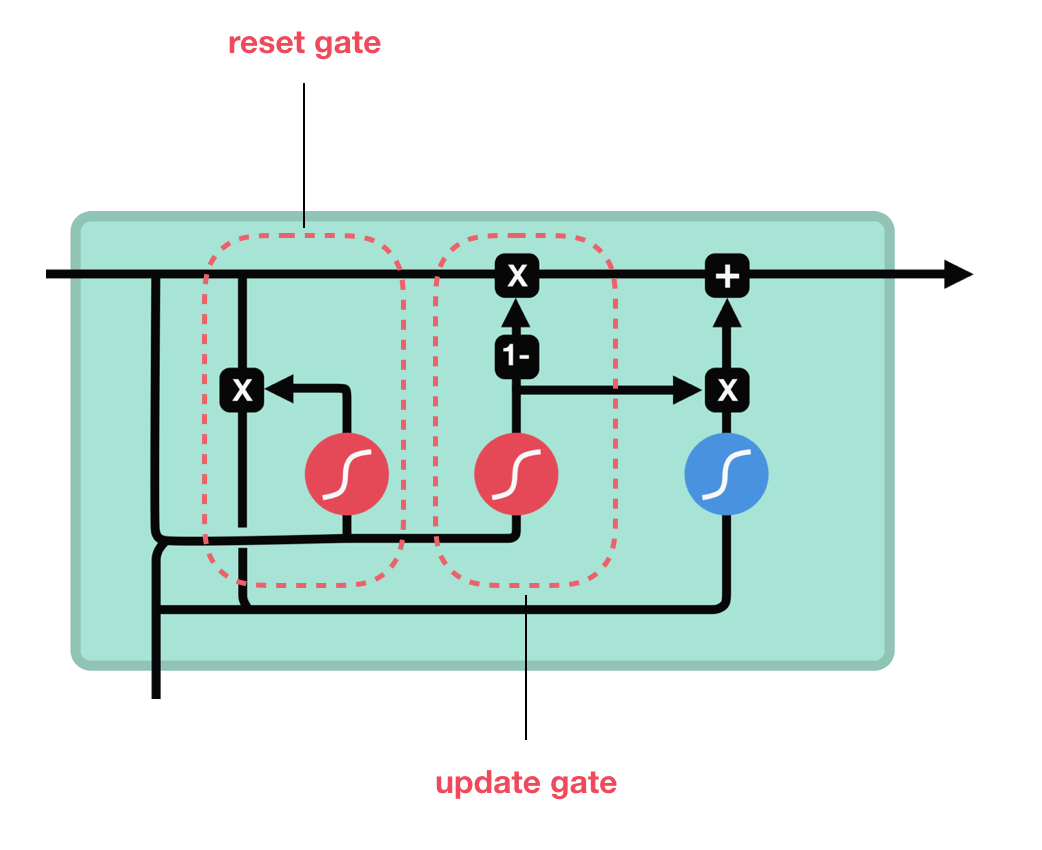


Fig. 3.1.3 GRU

The function of Update gate is similar to forget gate and input gate of LSTM, it decides what information to keep, add and let go.The Reset gate is used to decide how much of previous information to let go.The GRU has fewer operations compared to LSTM and hence they can be trained much faster than LSTMs.So, LSTMs and GRUs both were created as a solution to dodge short-term memory problems of the network using gates which regulates information throughout the sequence chain of the network.

**Chapter-4**

**Implementation of Attention Mechanism**

Attention is a mechanism that was developed to improve the performance of the Encoder-Decoder RNN on machine translation.

Attention was presented by Dzmitry Bahdanau, et al. in their paper “[Neural Machine Translation by Jointly Learning to Align and Translate](https://arxiv.org/abs/1409.0473)” that reads as a natural extension of their previous work on the Encoder-Decoder model.

Attention is proposed as a solution to the limitation of the Encoder-Decoder model encoding the input sequence to one fixed length vector from which to decode each output time step. This issue is believed to be more of a problem when decoding long sequences.

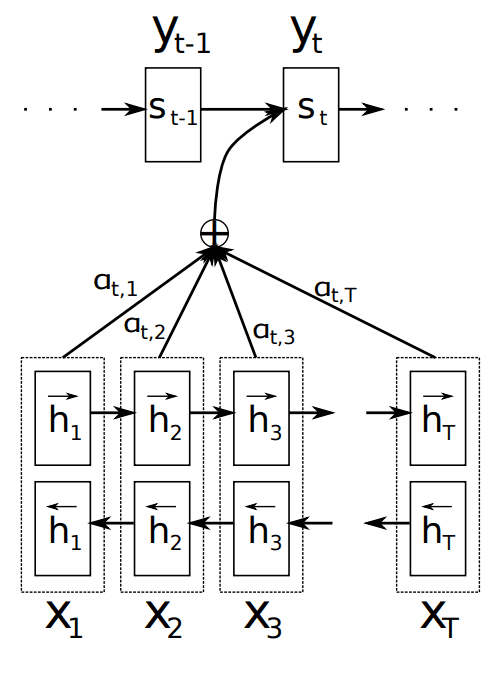


Fig. 4.1 An Attention Mechanism

Instead of encoding the input sequence into a single fixed context vector, the attention model develops a context vector that is filtered specifically for each output time step.

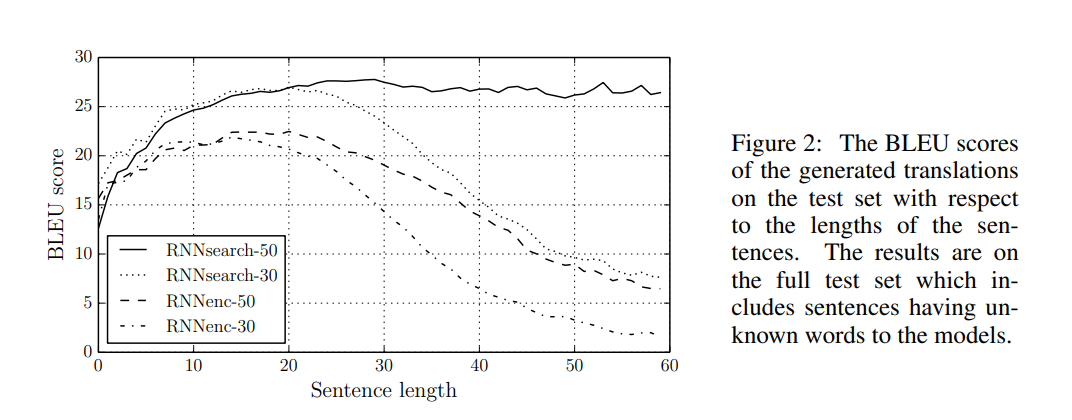
****

Fig. 4.2 BLEU score observation

Attention is proposed as a method to both align and translate.

Alignment is the problem in machine translation that identifies which parts of the input sequence are relevant to each word in the output, whereas translation is the process of using the relevant information to select the appropriate output.

Attention mechanism can further be visualized through image captioning systems

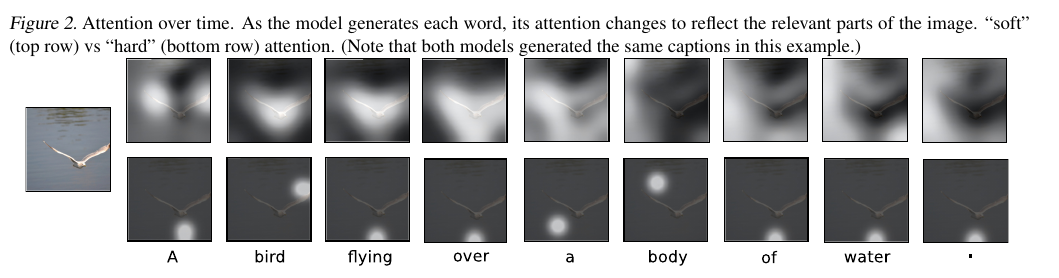


Fig. 4.3 Representation of Attention mechanism

Steps for attention mechanism:

4.1. **Compute a score each encoder state**

Since we are predicting the first word itself, the decoder does not have any current internal state. For this reason, we will consider the last state of the encoder (i.e. h5) as the previous decoder state.

Now using these two components (all the encoder states and the current state of the decoder), we will train a simple feed forward neural network.

4.2. **Compute the attention weights**

Once these scores are generated, we apply a softmax on these scores to produce the attention weights e1, e2, e3 ,e4 and e5 as shown above. The advantage of applying softmax is as below:

a) All the weights lie between 0 and 1, i.e., 0 ≤ e1, e2, e3, e4, e5 ≤ 1

b) All the weights sum to 1, i.e., e1+e2+3+e4+e5 = 1

Thus we get a nice probabilistic interpretation of the attention weights.

In our case we would expect values like below:

e1 = 0.75, e2 = 0.2, e3 = 0.02, e4 = 0.02, e5 = 0.01

4.3. **Compute the context vector**

Once we have computed the attention weights, we need to compute the context vector (thought vector) which will be used by the decoder in order to predict the next word in the sequence. Calculated as follows:

context\_vector = e1 \* h1 + e2 \* h2 + e3 \* h3 + e4 \* h4 + e5 \* h5

Clearly if the values of e1 and e2 are high and those of e3, e4 and e5 are low then the context vector will contain more information from the states h1 and h2 and relatively less information from the states h3, h4 and h5.

4.4. **Concatenate context vector with output of previous time step**

Finally the decoder uses the below two input vectors to generate the next word in the sequence

a) The context vector

b) The output word generated from the previous time step.

We simply concatenate these two vectors and feed the merged vector to the decoder. Note that for the first time step, since there is no output from the previous time step, we use a special <START> token for this purpose.

4.5. **Decoder Output**

The decoder then generates the next word in the sequence and along with the output, the decoder will also generate an internal hidden state, and let's call it “d1”.

4.6. Once the decoder outputs the **<END>** token, we stop the generation process.

Note that unlike the fixed context vector used for all the decoder time steps in case of the traditional Seq2Seq models, here in case of Attention, we compute a separate context vector for each time step by computing the attention weights every time.

Thus using this mechanism our model is able to find interesting mappings between different parts of the input sequence and corresponding parts of the output sequence.

Note that during the training of the network, we use teacher forcing in order to input the actual word rather than the predicted word from the previous time step.

As in case of any NLP task, after reading the input file, we perform the basic cleaning and preprocessing such as making all the alphabets in lower case, substituting a single apostrophe with space, removing the digits and then finally adding the <start> and the <end> tokens for identification, as we did in “**preprocess\_eng\_sentence”** and “**preprocess\_ben\_sentence”** code.

Create a class to map every word to an index and vice-versa for any given vocabulary. You can traverse words to index or vice versa for convenience sake. So it makes our traversal from word to index and from index to word very easy to handle. We did this by making **word2idx** and **idx2word** variables in our code.

Now we created our GRU encoder and decoder unit for addressing the problem of context remembrance and we will have to define the Optimizer, Loss Function and Checkpoints. We will create our own “**loss\_function**” method.

Now, we will train the model for 10 epochs, and we'll try to evaluate our predictions against the test data. We will make a random sentence generator which will be in “English” language, and it will be converted to “Bengali” language with the help of our Attention Model.

In the final output we can see the meaning of these two sentences is pretty much the same, and we have been able to successfully translate an English sentence to our target language i.e, bengali language.

If we had increased the number of epochs, the values predicted would’ve been more precise and accurate.

**Chapter-5**

**Results and Discussions**

After running the language translator using the attention mechanism for 10 epochs, we can clearly observe that there has been a successive decrease in the loss percentage of the process. This is because after each epoch, the precision of our model actually increases. Attention Mechanism has following components:

1. The encoder LSTM/GRU is used to process the entire input sentence and encode it into a context vector, which is the last hidden state of the LSTM/GRU. This is expected to be a good summary of the input sentence. All the intermediate states of the encoder are ignored, and the final state id supposed to be the initial hidden state of the decoder
2. The decoder LSTM or GRU units produce the words in a sentence one after another

This was quite a comprehensive look at the popular Attention mechanism and how it applies to deep learning. That’s why this technology has made quite a dent in the deep learning space. It is extraordinarily effective and has already penetrated multiple domains.

Had the number of epochs been increased, the precision had been even more , and the loss would’ve been a lot less. It takes a considerable amount of time for these epochs to get completed, and generate respective outputs.

This Attention mechanism has many more uses too like Image Generation, Image Captioning etc.

|  |  |  |
| --- | --- | --- |
| **S. NO.** | **NAME OF PRACTICAL** | **ACCURACY MEASURE** |
| 1 | Neural Network for handwritten digit | 95.125% (Avg) |
| 1.1 | With input and output layer only | 92.58% |
| 1.2 | With increased hidden layers | 97.67% |
| 2 | CNN Flower Image Classification | 67.97% (Avg) |
| 2.1 | With overfitted data | 65.57% |
| 2.2 | With augmented data | 70.37% |
| 3 | Language translation using attention mechanism | 85.48% |

Table 2. Accuracy of different experiments

**Chapter-6**

**Conclusions and Discussion**

The model is able to find the correct local mappings between the input and the output sequences which do match with our intuition.

Given more data and with more hyper parameter tuning, the results and mappings will definitely improve by a good margin. Using LSTM layers in place of GRU and adding Bidirectional wrapper on the encoder will also help in improved performance.

The conventional approach to neural machine translation, called an encoder–decoder approach, encodes a whole input sentence into a fixed-length vector from which a translation will be decoded. We conjectured that the use of a fixed-length context vector is problematic for translating long sentences, based on a recent empirical study.

Here, we proposed a novel architecture that addresses this issue. We extended the basic encoder–decoder by letting a model (soft-)search for a set of input words, or their annotations computed by an encoder, when generating each target word. This frees the model from having to encode a whole source sentence into a fixed-length vector, and also lets the model focus only on information relevant to the generation of the next target word.

Deep Learning models are generally considered as black boxes, meaning that they do not have the ability to explain their outputs. However, Attention is one of the successful methods that helps to make our model interpretable and explain why it does what it does.

Furthermore, it can be learned and concluded that the only disadvantage of the Attention mechanism is that it is a very time consuming and hard to parallelize system. To solve this problem, Google Brain came up with the “Transformer Model” which uses only Attention and gets rid of all the Convolutional and Recurrent Layers, thus making it highly parallelizable and compute efficient, however that is outside the scope of our current knowledge domain and Application.

**Chapter-7**

**Future Scope**

The model which we have made is able to find the correct local mappings between the input and the output sequences which do match with our intuition.

Given more data and with more hyper parameter tuning, the results and mappings will definitely improve by a good margin.

Using LSTM layers in place of GRU and adding Bidirectional wrapper on the encoder will also help in improved performance.

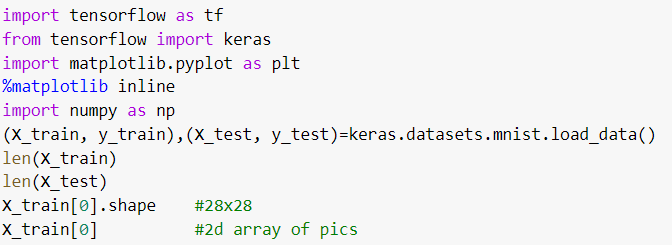
Deep Learning models are generally considered as black boxes, meaning that they do not have the ability to explain their outputs. However, Attention is one of the successful methods that helps to make our model interpretable and explain why it does what it does.

The only disadvantage of the Attention mechanism is that it is a very time consuming and hard to parallelize system. To solve this problem, Google Brain came up with the “Transformer Model” which uses only Attention and gets rid of all the Convolutional and Recurrent Layers, thus making it highly parallelizable and compute efficient.

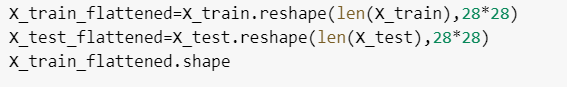
**Appendix**

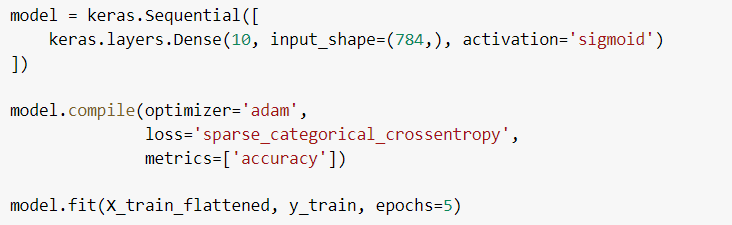
1.**ANN for Handwritten digit Classification:**

a) **With only input and output layer**

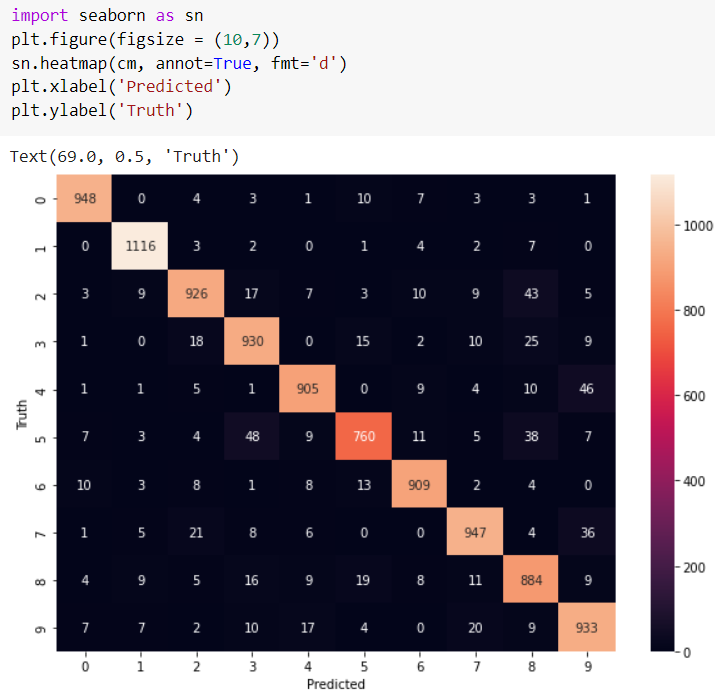




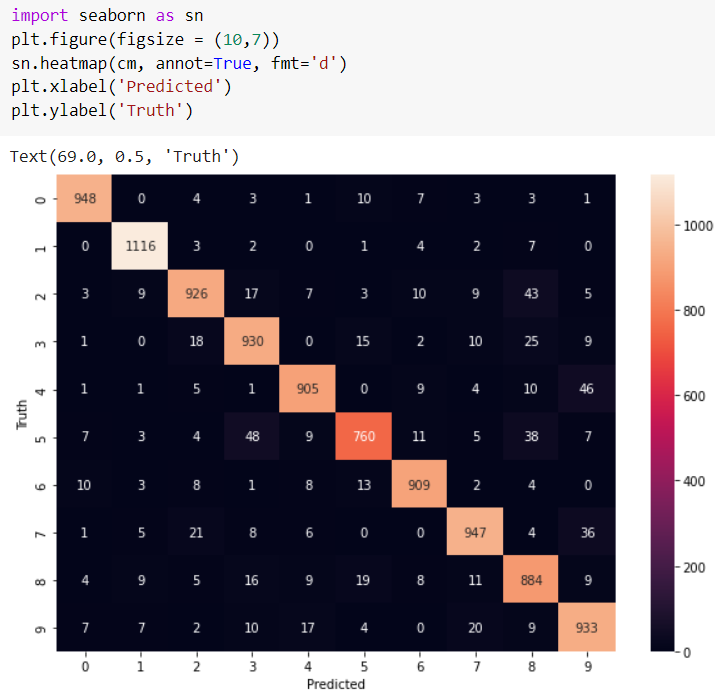




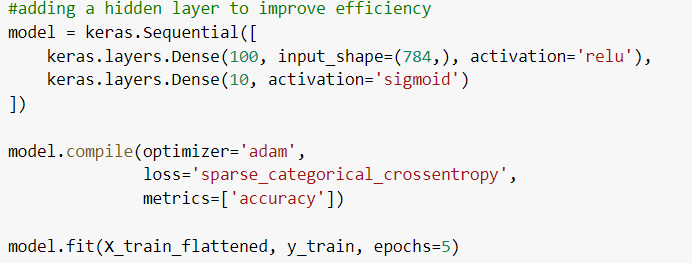


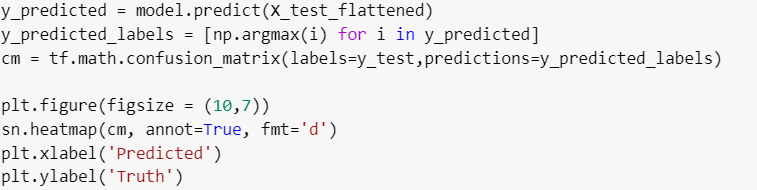


**Output:**

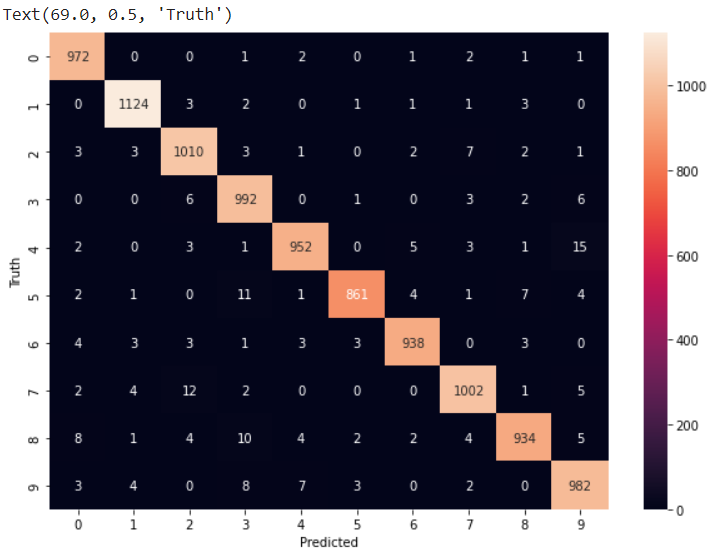


b) **With increased hidden layer**



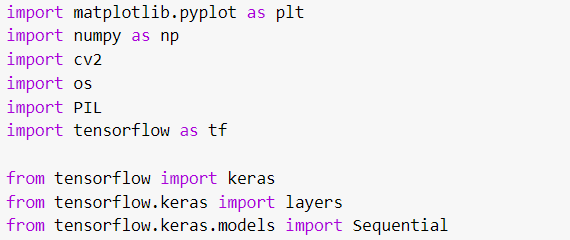


**Output:**

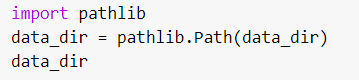


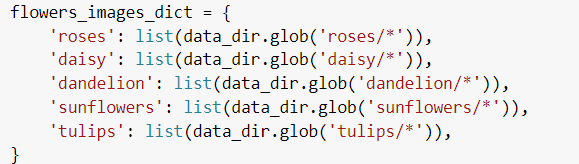
**2.CNN Flower Image Classification**

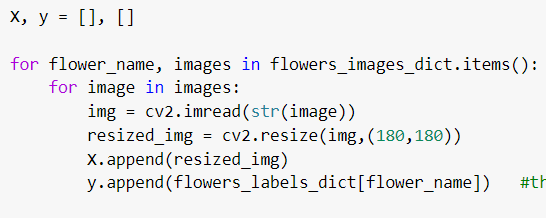
a) **With overfitted data**





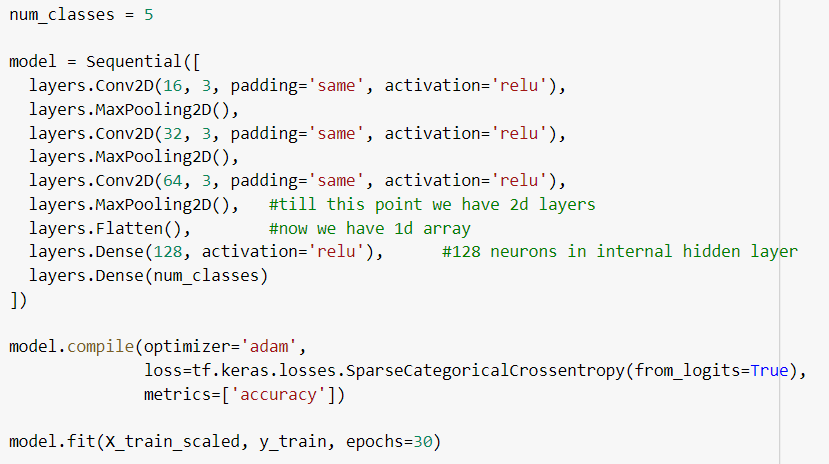






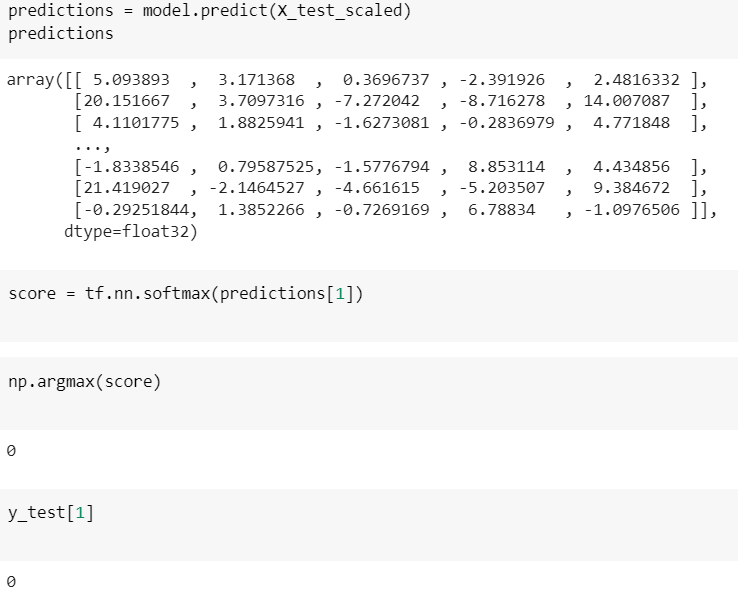




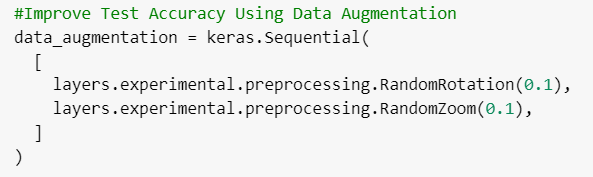




**Output:**



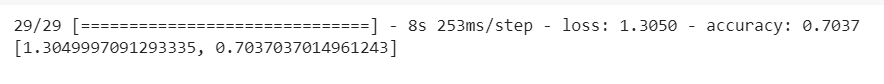
b) **With augmented data**



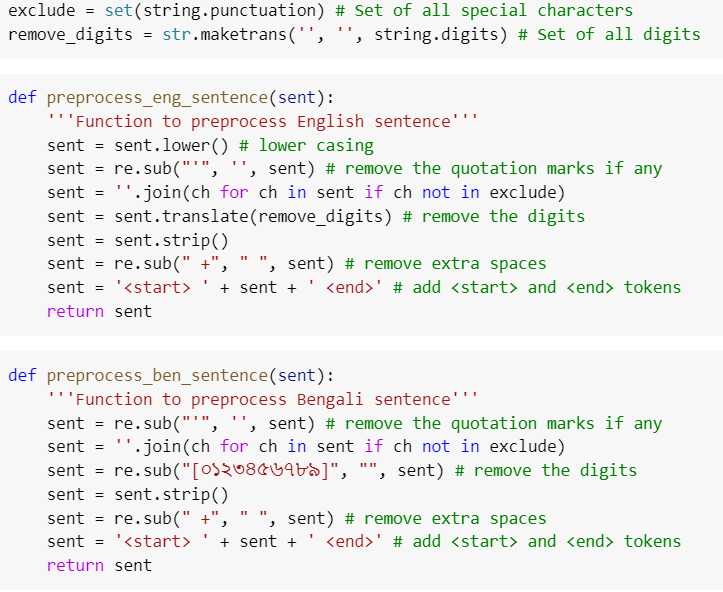
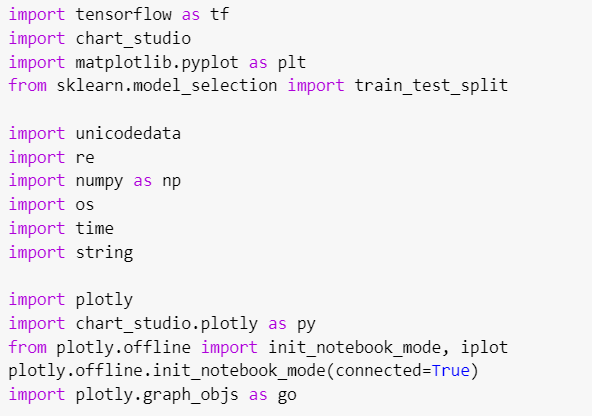


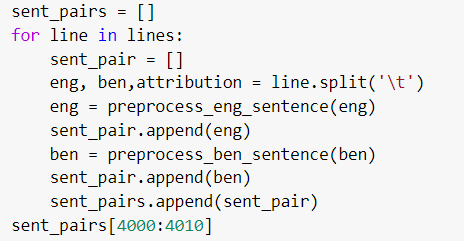


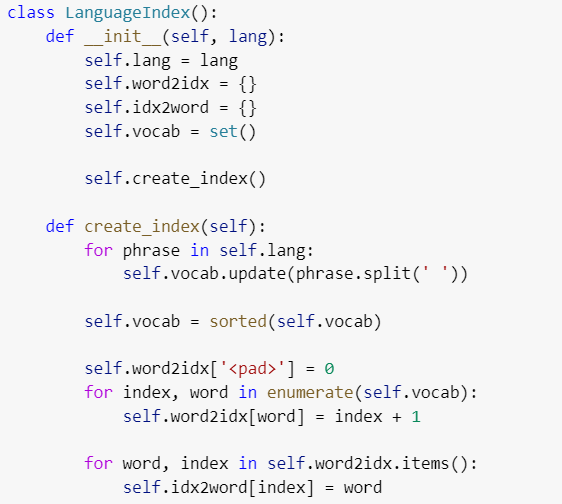
**Output:**

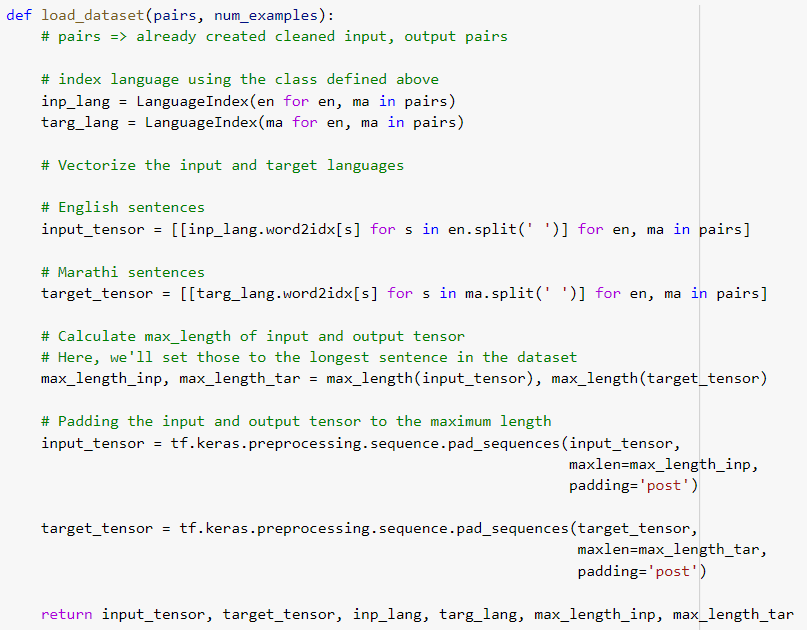


3.**Attention Mechanism for English to Bengali translation**



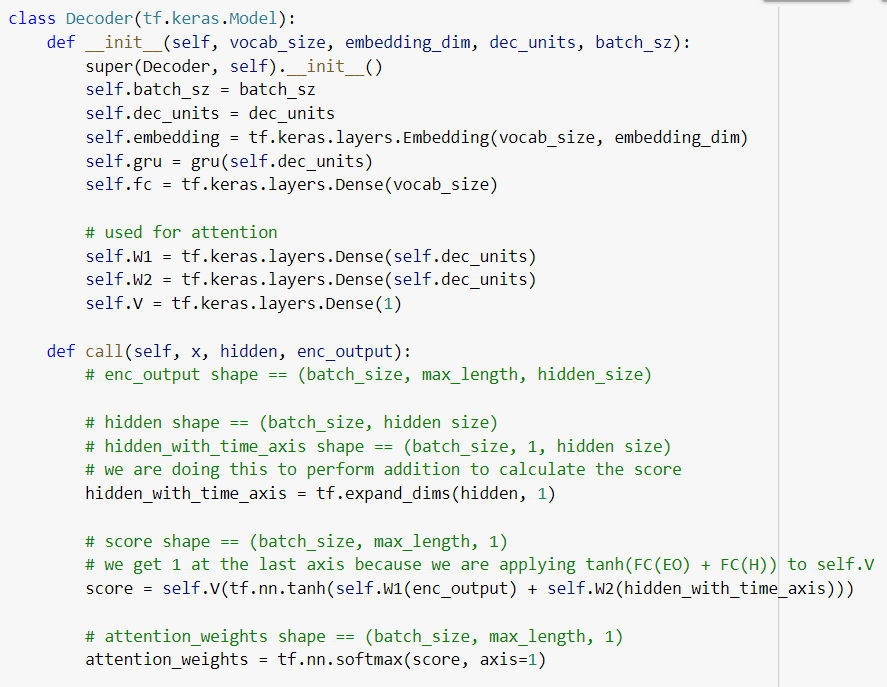


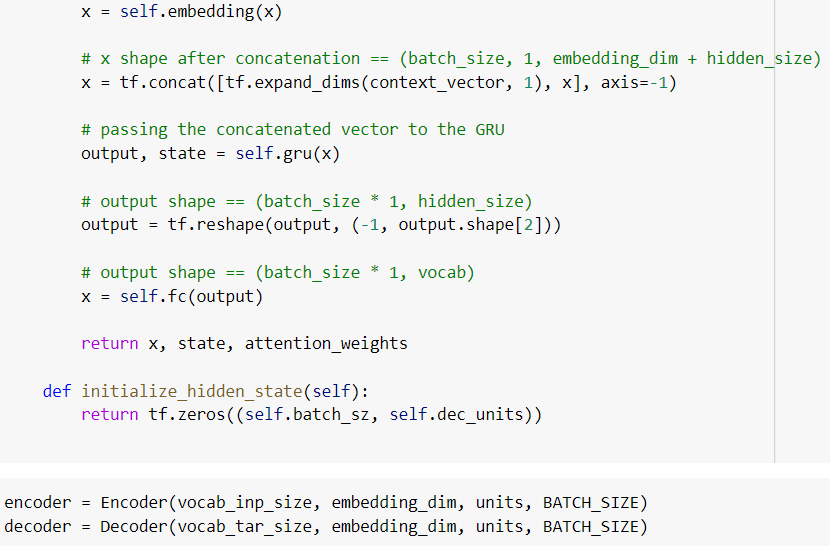


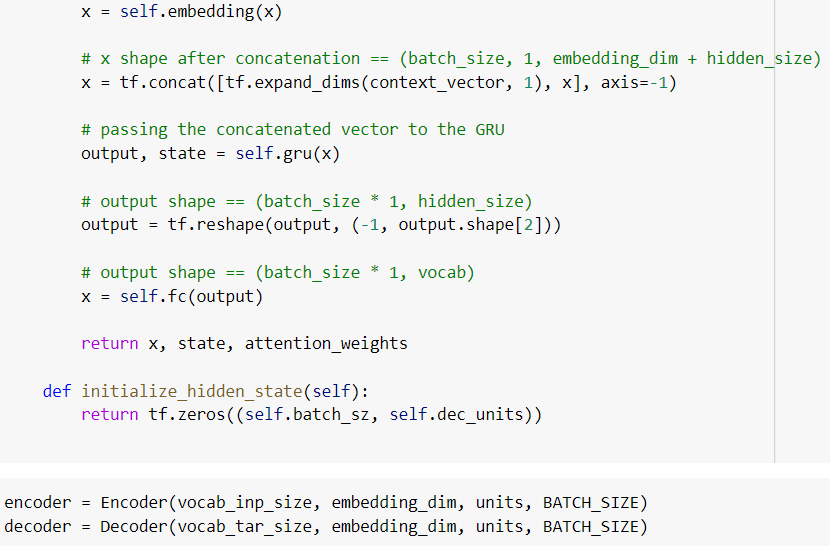




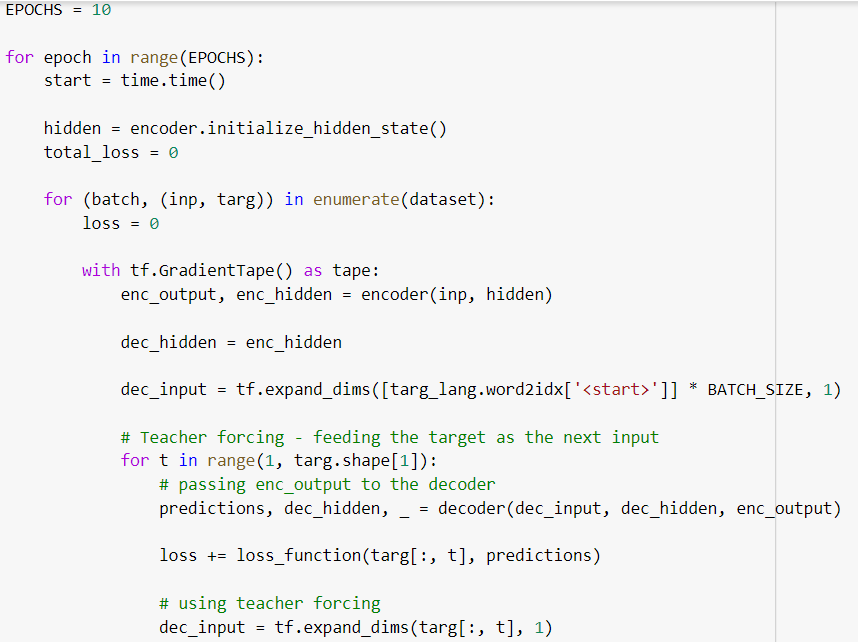


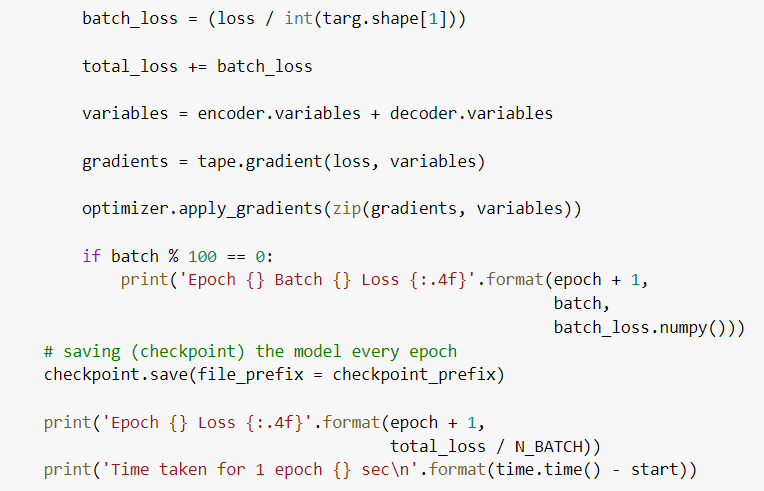


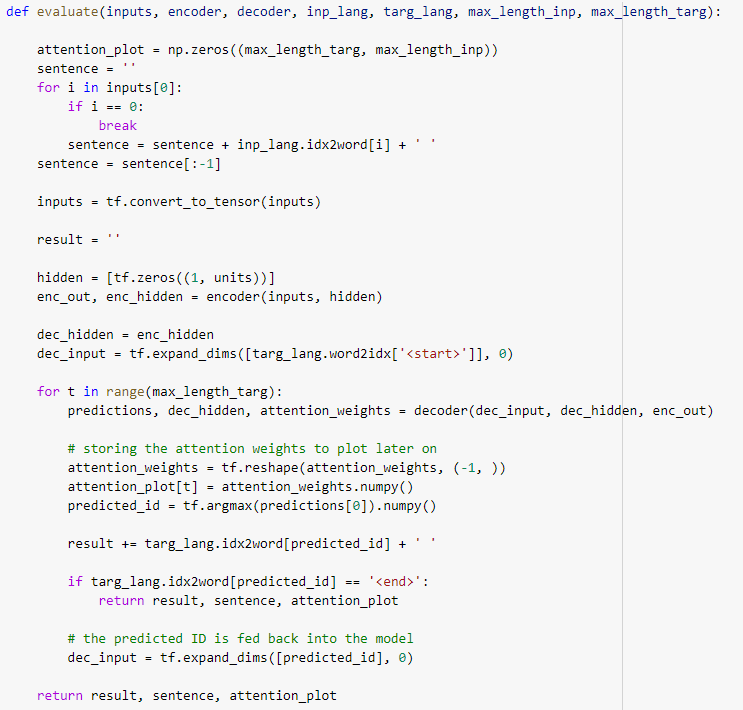








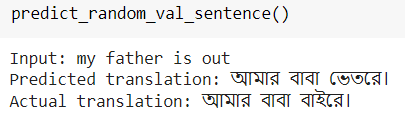




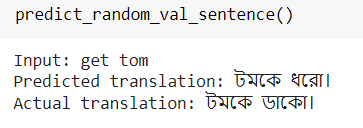


**Output:**

**Sentence-1:**



**Sentence-2:**



**Bibliography/References**

1. <https://arxiv.org/abs/1409.0473> (Original Paper)
2. <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb> (TensorFlow Implementation available on their official website as a tutorial)
3. <https://www.coursera.org/lecture/nlp-sequence-models/attention-model-lSwVa> (Andrew Ng’s Explanation on Attention)
4. <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>
5. <https://www.tensorflow.org/xla/broadcasting> (Broadcasting in TensorFlow)
6. Dataset: <http://www.manythings.org/anki/> (ben-eng.zip)