

**Image Captioning using Convolutional Neural Network  
and  
Long Short-Term Memory**

Project submitted to

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**JADAVPUR UNIVERSITY**

In partial fulfillment of the requirement for the Degree of

**MASTER OF COMPUTER APPLICATIONS, 2019**

By

**ANKIT KUMAR MANDAL**

Registration Number. : 137331 of 2016-2017

Examination Roll No. : MCA196020

Under the Guidance of

**Prof. Susmita Ghosh**

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2019

Department of Computer Science and Engineering Faculty

Faculty of Engineering and Technology

Jadavpur University

**To whom it may concern**

This is to certify that the work in this project report entitled as “**Image Captioning using Convolutional Neural Network and Long Short-Term Memory**“ has been satisfactorily completed by **Ankit Kumar Mandal**. It is a bona-fide piece of work for the fulfillment of the requirements for the degree of Master of Computer Application, of the Department of Computer Science & Engineering, Faculty of Engineering & Technology, Jadavpur University, during the academic year 2018-19.

---

**Prof. Susmita Ghosh**

*(Supervisor)*

*Professor*

*Department of Computer Science and Engineering*

*Jadavpur University*

---

**Prof. Mahantapas Kundu**

*(Head of the Department)*

*Professor*

*Department of Computer Science and Engineering*

*Jadavpur University*

---

**Prof. Chiranjib Bhattacharjee**

*(Dean)*

*Professor*

*Department of Computer Science and Engineering*

*Jadavpur University*

# DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC PROJECT

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work as part of my MCA studies and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Name : **Ankit Kumar Mandal**

Registration No. : 137331 of 2016-2017

Class Roll No. : 001610503024

Examination Roll No. : MCA196020

Project Report Title : **Image Captioning using Convolutional Neural Network and Long Short-Term Memory**

---

**Ankit Kumar Mandal**

**Department of Computer Science and Engineering Faculty**

**Faculty of Engineering and Technology**

**Jadavpur University**

**Certificate of Approval**

This is to certify that the project titled “**Image Captioning using Convolutional Neural Network and Long Short-Term Memory** “ is a bona-fide record of work carried out by **Ankit Kumar Mandal** in partial fulfillments for the award of the degree of Master of Computer Application, of the Department of Computer Science & Technology, Jadavpur University during the period January 2019 to May 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis only for the purpose for which it is submitted.

\_\_\_\_\_  
Signature of the Examiner

Date: \_\_\_\_\_

\_\_\_\_\_  
Signature of the Examiner

Date: \_\_\_\_\_

# Acknowledgements

First and foremost, I would like to pay my heartiest thanks to my respected guide **Prof. Susmita Ghosh**, Department of Computer and Engineering, Jadavpur University, for her keen interest, invaluable suggestions and guidance rendered to me during the course of my project work.

I would also like to convey my sincere gratitude to all my respected teachers and faculty members in this department for their invaluable suggestions and kind cooperation.

I would also like to express my sincere gratitude to all of them, who directly and indirectly helped me for the successful completion of this work.

Last but not the least, I would like to convey thanks to my mother buchchi devi, and my elder sister Sangeeta devi for their moral support throughout the course.

Regards,

**Ankit kumar Mandal**

Examination Roll No. : MCA196020

Master of Computer Application

Jadavpur University

# Tables of Contents

<b>1. Introduction</b>	.....	<b>1</b>
1.1. Image captioning	.....	<b>1</b>
1.2. Application of Image captioning	.....	<b>2</b>
1.3. Algorithm used for Image captioning	.....	<b>2</b>
1.3.1. Convolution Neural Network	.....	<b>2</b>
1.3.2. Recurrent Neural Network	.....	<b>2</b>
1.3.3. Long Short Term Memory	.....	<b>2</b>
1.4. Why are CNN and RNN used for image captioning ?	.....	<b>3</b>
1.5. Process of Image Captioning	.....	<b>3</b>
1.6. Scope of the project	.....	<b>4</b>
1.7. Organization of the project	.....	<b>4</b>
<b>2. Artificial Neural Network's</b>	.....	<b>5</b>
2.1. Introduction to Artificial Neural Network	.....	<b>5</b>
2.2. Artificial Neural Network Architecture	.....	<b>7</b>
2.3. Activation function:	.....	<b>9</b>
2.4. Loss function	.....	<b>10</b>
2.5. Optimizer	.....	<b>10</b>
<b>3. Recurrent Neural Network</b>		
3.1. Introduction to recurrent neural network	.....	<b>11</b>
3.2. Initialization	.....	<b>12</b>
3.3. Forward propagation	.....	<b>14</b>
3.4. Error calculating	.....	<b>15</b>
3.5. Back Propagation Through Time	.....	<b>15</b>

3.6. Application of recurrent neural network	23
3.7. Two issues of recurrent neural network	24
3.8. Long Short Term Memory	25
<b>4. Convolution Neural Network</b>	<b>27</b>
4.1. Convolution	27
4.2. ReLu	28
4.3. Pooling	29
4.4. Max-pooling	29
4.5. Fully connected layer	29
<b>5. Image captioning</b>	<b>30</b>
5.1. Introduction to image captioning	30
5.2. Architecture	31
5.3. Methodology to solve the task	33
<b>6. Implementation and results</b>	<b>35</b>
6.1. Implementation mechanism	35
6.2. Dataset	35
6.3. Implementation process	35
6.4. Results	41
<b>7. Conclusion and future work</b>	<b>44</b>
<b>8. Bibliography</b>	<b>45</b>

# Chapter 1

## Introduction

### 1.1 Image captioning :

Caption generation is a challenging artificial intelligence problem where a textual description is generated for a given photograph. It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem [ ]. What is most impressive about these methods is a single end-to-end model can be defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

### 1.2 Application of image captioning

- **Self driving cars:** Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self driving system [5].
- **Aid to the blind:** We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep Learning[5 ]
- CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accident[5].
- Automatic captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.



## 1.3 Algorithm used for Image captioning

There are several algorithm exist in literature which generate captions of a given image. They are mainly

### 1.3.1 Convolution neural network

A convolution neural network (CNN) [17 ] is one of the most popular algorithm for deep learning, a type of machine learning in which a model learns to perform classification tasks directly from images, video, or text. CNNs are particularly useful for finding patterns in images to recognize object, faces and scenes. They learn directly from image data, using patterns to classify images and eliminating the need for manual feature extraction. CNNs provide an optimal architecture for image recognition and pattern detection [17]. Combined with advances in GPUs and parallel computing, CNNs are a key technology underlying new developments in automated driving and facial recognition. For example, deep learning applications use CNNs to examine thousands of pathology reports to visually detect cancer cells.[5] CNNs also enable self-driving cars [ 5] to detect objects and learn to tell the difference between a street sign and a pedestrian. We have discussed this in details in Chapter 4.

### 1.3.2 Recurrent neural network :

Recurrent neural network is one of the most popular neural networks for language modeling based on existing words to predict next word (based on existing characters to predict next character) [26]. The logic behind a recurrent neural network is to consider the sequence of the input. For us to predict the next word in the sentence we need to remember which word appeared in the previous time step. These neural networks are called **Recurrent** because this step is carried out for every input. As these neural network consider the previous word during predicting, it acts like a memory storage unit which stores it for a short period of time. We have discussed this in details in RNN Chapter 3.

### 1.3.3 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.[11 ] This work tremendously well on

a large variety of problems, and are now widely used [12]. LSTMs are explicitly designed to avoid the long-term dependency problem. It overcome the problems associated with Recurrent Neural Network are :-

- Exploding Gradients( discussed this in RNN Chapter 3.6)
- Vanishing Gradients ( discussed this in RNN Chapter 3.6)

#### 1.4 Why are CNN and RNN used for image captioning ?

A captioning model relies on two main components, a CNN and an RNN. Captioning is all about merging the two to combine their most powerful attributes [ 1].

1. As we know that CNN is used to classify the images and extract the features from the image so we first extract the features from the all the images and these features used as input of previous state of the LSTM.
2. RNNs work well with any kind of sequential data, such as generating a sequence of words. So by merging the two, you can get a model that can find patterns and images, and then use that information to help generate a description of those images.

#### 1.5 Process of Image captioning

The image captioning process consists of several stages

- Prepare photo and text data for training a deep learning model.
- Design and train a deep learning caption generation model.
- Evaluate a train caption generation model and use it to generate caption for entirely new photographs.

## 1.6. Scope of the project

The automatic image caption generation, proposed in this project, uses various neural network techniques. Several such techniques for image captioning are present in literature. In this present study we have used convolutional neural network and Long Short Term Memory. Convolution neural network is used to extract the feature of images and recurrent neural network used to generate text.

## 1.7. Organization of the project

The thesis is organized as follows: chapter 2 contains introduction to Artificial Neural Network's. Chapter 3 includes discussion on Recurrent Neural Network and Long Short Term Memory. Chapter 4 contains discussion on Convolutional Neural Network. Chapter 5. Contains image captioning. Chapter 6. Includes implementation and results. Finally the conclusion and future work are put in Chapter 7.

# Artificial Neural Network

## 2.1. Introduction to Artificial Neural Network :

Artificial Neural Network (ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problem [20]. Let us start by first understanding how our brain processes information:

In our brain, there are billions of cells called neurons, which processes information in the form of electric signals. External information is received by the dendrites of the neuron, processed in the neuron cell body, converted to an output and passed through the Axon to the next neuron. The next neuron can choose to either accept it or reject it depending on the strength of the signal [21].

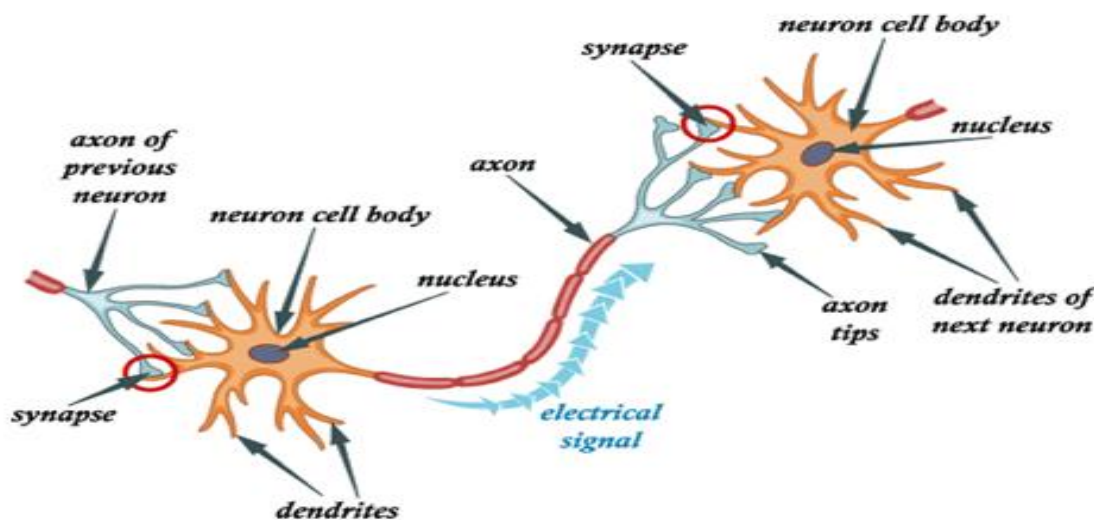


Fig 2.1: To understand the Biological Neuron Structure[21]

The Following steps are performed in a Biological Neuron network

**Step1:** External signal received by dendrites

**Step2:** External signal processed in the neuron cell body

**Step3:** Processed signal converted to an output signal and transmitted through the Axon

**Step 4:** Output signal received by the dendrites of the next neuron through the synapse.

Now, let us try to understand how a ANN works:

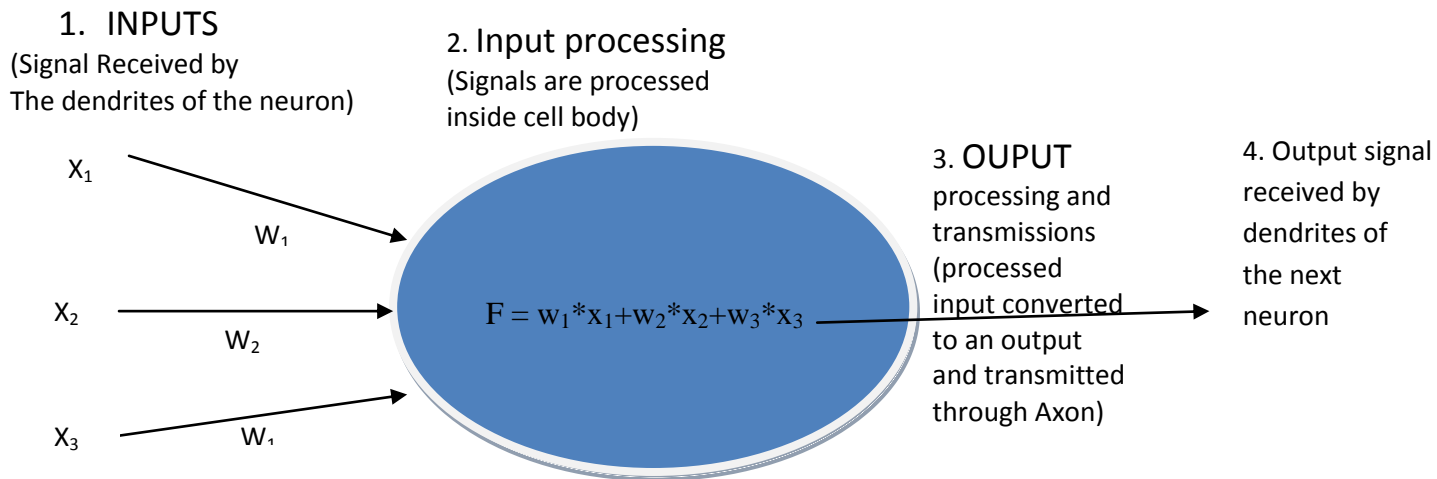


Fig 2.2 Represents the general model of ANN

Here,  $w_1$ ,  $w_2$ ,  $w_3$  gives the strength of the input signals

As we can see from the above, an ANN is a very simplistic representation of a how a brain neuron works. An Artificial Neural Network(ANN) is composed of four principal objects:

- **Layers:** all the learning occurs in the layers. There are 3 layers:
  - 1) Input (discussed next section of this Chapter)
  - 2) Hidden (discussed next section of this Chapter)
  - and 3) Output (discussed next section of this Chapter)
- **Feature and label:** Input data to the network is called features and output from the network is called labels)
- **Loss function:** Metric used to estimate the performance of the learning phase.
- **Optimizer:** Improve the learning by updating the knowledge in the network

A neural network takes the inputs data and feed them into next layer. The network needs to evaluate its performance with a loss function. The loss function gives to the network an idea

of the path it needs to take before it masters the knowledge. The network needs to improve its knowledge with the help of an optimizer.

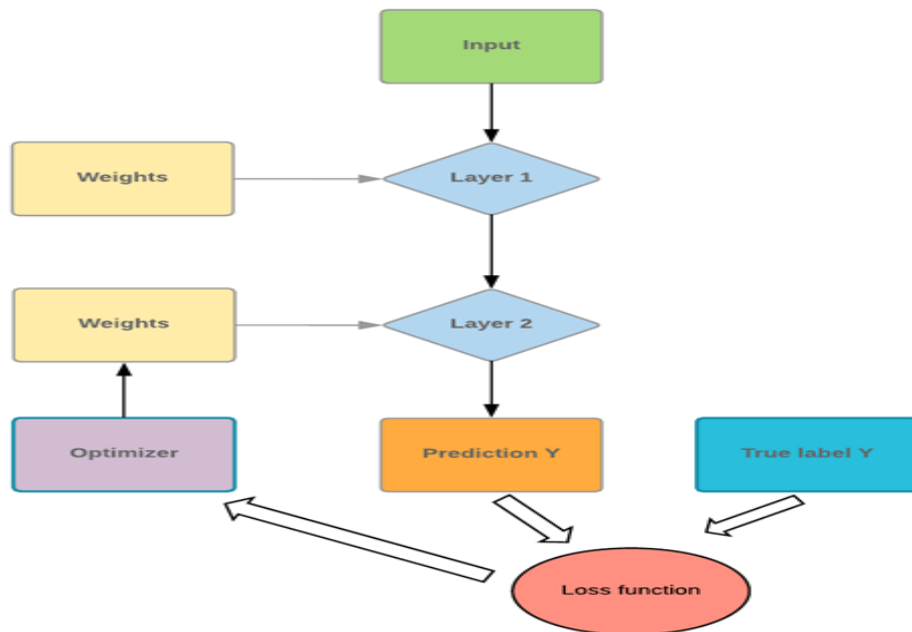


Fig 2.3. ANN Underlying Mechanism [ 3]

The program takes some input values and feed them into two fully connected layers.

To improve its knowledge, the network uses an optimizer, updates its knowledge, and tests its new knowledge to check how much it still needs to learn. The program will repeat this step until it makes the lowest error. If the error is far from 100%, but the curve is flat, it means with the current architecture. It cannot learn anything else. The network has to be better optimized to improve the knowledge.

## 2.2 Neural Network Architecture:

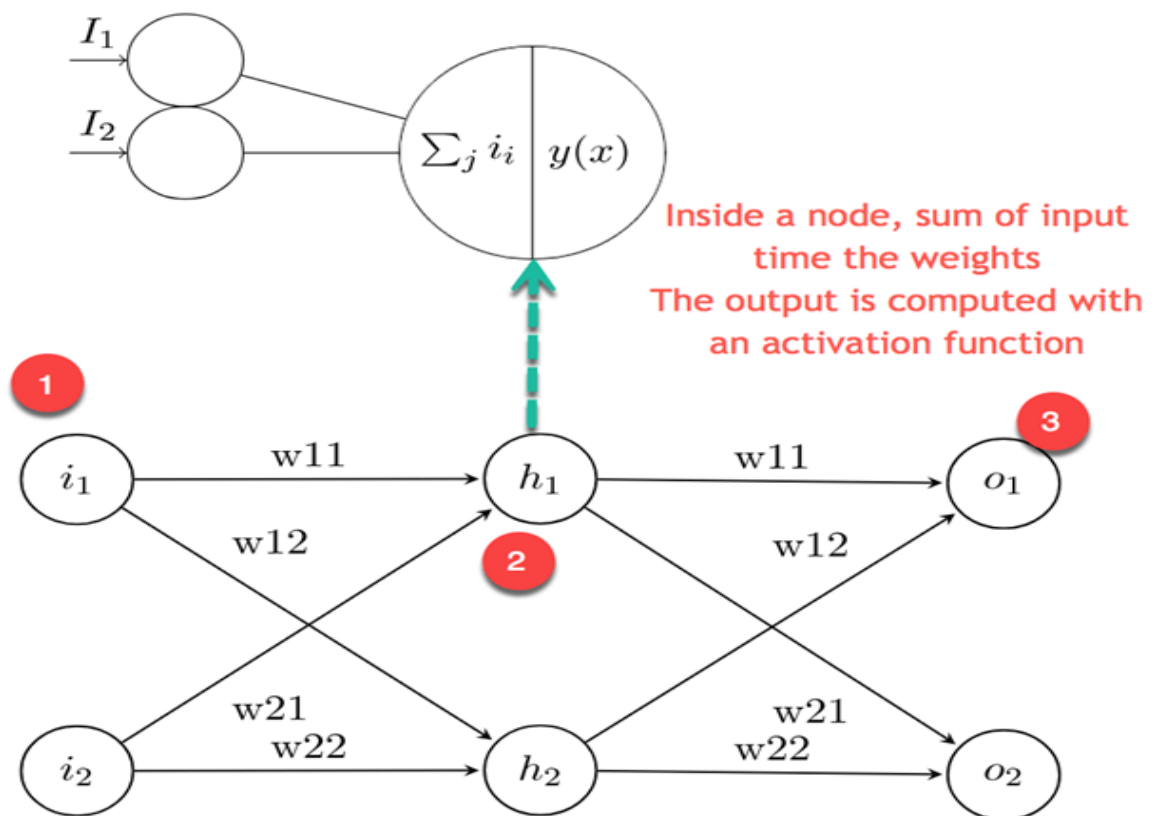
The ANN architecture typically consists of the following layers.

- **Input layer** – This layer provides the input to the ANN system.
- **Hidden layer** – Hidden layer in ANN is a layer of neurons, which are flanked on the input side by neurons in the input layer and on the output side by neurons belonging to the output layer. There may be more than one hidden layer in a system, none of

which are visible as a network output. The hidden layer transforms the inputs into signals that the output layer can use.

- **Output layer** – This layer consists of the output units whose values are dependent on the hidden units and the weights between the hidden and output units. The output units decide the class label.

A layer is where all the learning takes place. Inside a layer, there are an infinite amount of weights (neurons). A typical neural network is often processed by densely connected layers (also called fully connected layers). It means all the inputs are connected to the output. A typical neural network takes a vector of input and a scalar that contains the labels. The most comfortable set up is a binary classification with only two classes: 0 and 1. The network takes an input, sends it to all connected nodes and computes the signal with an **activation** function.



**Fig. 2.3** plots the following ideas [21]

. The first layer is the input values for the second layer, called the hidden layer, receives the weighted input from the previous layer

1. The first node is the input values
2. The neuron is decomposed into the input part and the activation function. The left part receives all the input from the previous layer. The right part is the sum of the input passes into an activation function.
3. Output value computed from the hidden layers and used to make a prediction. For classification, it is equal to the number of class.

## 2.1 Activation function:

### 1.3.1 ReLU :

The activation function of a node defines the output given a set of inputs. You need an activation function to allow the network to learn non-linear pattern.

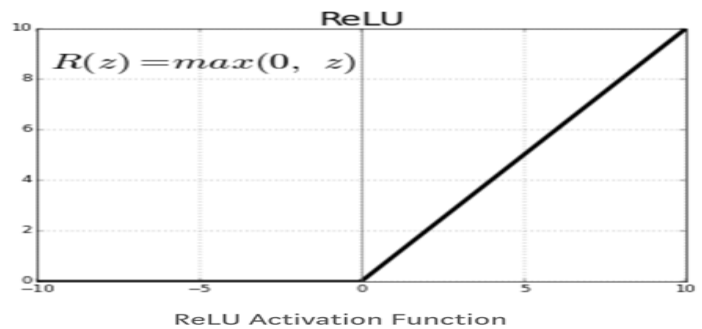


Fig.2.4 ReLU activation function [22].

A common activation function is a **Relu, Rectified linear unit**. The function gives a zero for all negative values.

### 1.3.2. Sigmoid:

A sigmoid function produces a curve with an “S” shape. The example sigmoid function shown on the left is a special case of the logistic function, which models the growth of some set.

$$Sig(x) = \frac{1}{1 + e^{-x}}$$

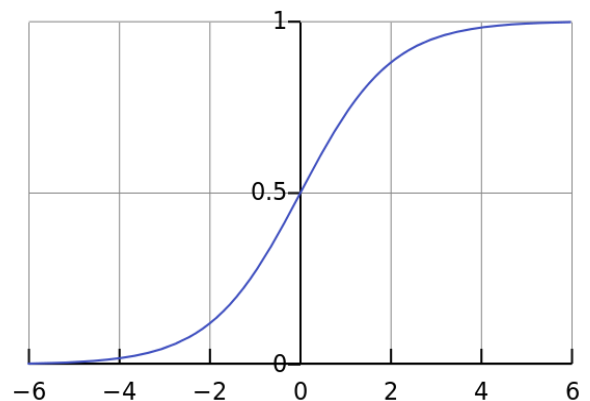
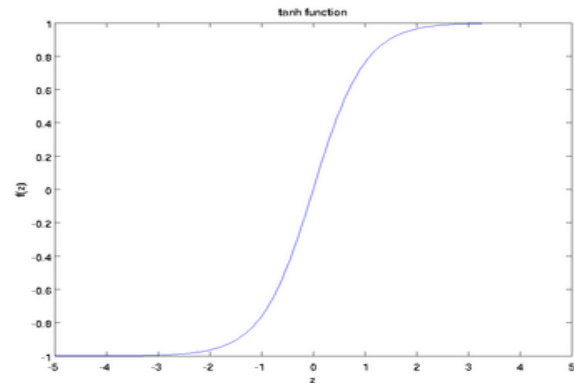


Fig.2.5 Sigmoid function [22].



### 1.3.3. Tanh:

This has characteristics similar to **sigmoid** that we discussed above. It is nonlinear in nature, its values are lies between (-1,1) .



**Fig.2.6** Tanh activation function [22] .

Deciding between the **sigmoid** or **tanh** ,it depends on our requirement of gradient strength. Like sigmoid, **tanh** also has the vanishing gradient problem. **Tanh** is also a very popular and widely used activation function.

$$f(x) = \tanh(x) = \frac{1}{1 + e^{-2x}} - 1$$

## 1.4. Loss function

After we have defined the hidden layers and the activation function, you need to specify the loss function and the optimizer.

For binary classification, it is common to use a binary cross entropy loss function. In the linear regression, we use the mean square error(discussed in Chapter 3).

The loss function is an important metric to estimate the performance of the optimizer. During the training, this metric be minimized. we need to select this quantity carefully depending on the type of problem we are dealing with.

## 1.5. Optimizer

The loss function is a measure of the model's performance. The optimizer helps improve the weights of the network in order to decrease the loss. There are different optimizers available, but the most common one is the Stochastic Gradient Descent. We have used Adam optimization to optimize the weights.

## Recurrent Neural Network

## 3.1 Recurrent neural network

Recurrent neural network is one of the most popular neural networks for language modeling based on existing words to predict next word (based on existing characters to predict next character)[26]. The logic behind a recurrent neural network is to consider the sequence of the input. For us to predict the next word in the sentence we need to remember which word appeared in the previous time step. These neural networks are called **Recurrent** because this step is carried out for every input. As these neural network consider the previous word during predicting, it acts like a memory storage unit which stores it for a short period of time.

.For example, when we build a RNN for a language, that means: the training data is a list of sentences. Each sentence is a series of words (tokenized words). For each sentence, from the first word, we will predict the second word. From the first and the second word, we will predict the third word, etc.

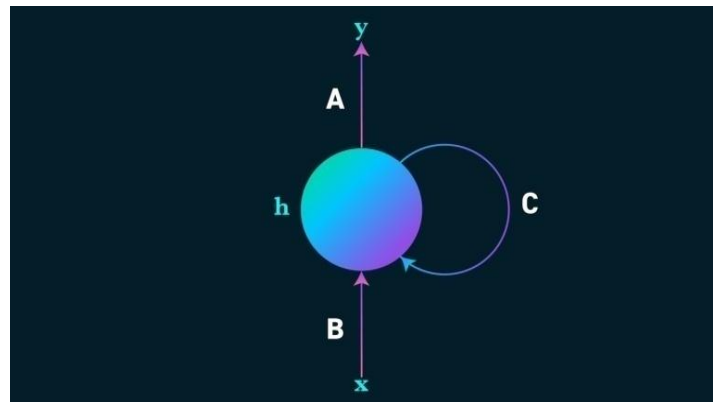


Fig. 3.1 Recurrent Neural Network [18]

Recurrent neural network means when it predict time order  $t$ , it will remember the information from time order 0 to time order  $t$ .

Let's denote the sentence having  $t+1$  words as

$x = [x_0, x_1, \dots, x_t]$ . we start from  $x_0$  to status

$S_0 = \tanh(Ux_0 + Ws_0)$ , where  $s_{-1}$  is the initialization of status initialization as 0.

The output  $o_0 = \text{softmax}(Vs_0)$ . Then when we go to next word  $x_1$  we will have updated status

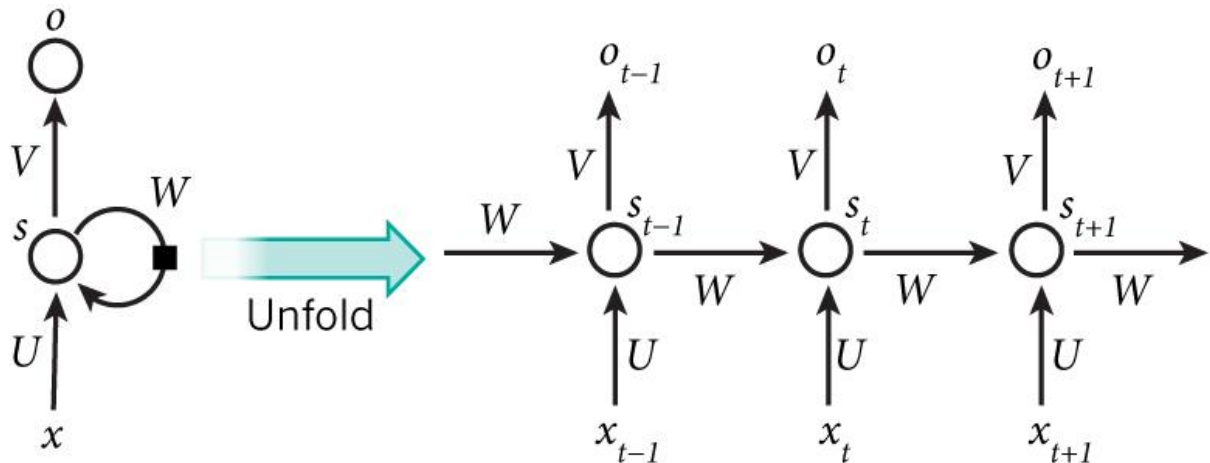
$S_1 = \tanh(Ux_1 + Ws_0)$  and the corresponding output

$O_1 = \text{softmax}(Vs_1)$ . We will see at time order  $t=1$  it not only depends on input  $x_1$  but also depends on the previous status  $s_0$ . The equation we have used here are :

$$S_t = \tanh(Ux_t + Ws_{t-1})$$

$$O_t = \text{softmax}(Vs_t)$$

If we plot the logic of RNN and the corresponding forward propagation, it is like



**Fig.3.2:** This illustrates an unrolled RNN

### 3.2. Initialization

We simplify the above architecture into given below for example all input nodes as output layer, all hidden nodes as hidden layer and all output nodes as output layer .

Here,

$X_0$ =input at time  $t=0$  ,  $X_1$ =input at time  $t=1$  ,  $X_2$ =input at time  $t=2$

$W_1$ = weight of neuron from hidden layer to hidden layer  $S_0$  to  $S_1$

$W_2$ = weight of neuron from hidden layer to hidden layer  $S_1$  to  $S_2$

$V_0$ = weight of neuron from hidden layer to output layer  $S_0$  to  $O_0$

$V_1$ = weight of neuron from hidden layer to output layer  $S_1$  to  $O_1$

$V_2$ = weight of neuron from hidden layer to output layer  $S_2$  to  $O_2$

$U_0$ = weight of neuron from input layer to hidden layer  $X_0$  to  $S_0$

$U_1$ = weight of neuron from input layer to hidden layer  $X_1$  to  $S_1$

$U_2$ = weight of neuron from input layer to hidden layer  $X_2$  to  $S_2$

$$V_0=0.5, V_1=0.5, V_2=0.5, W_{01}=0.5, W_{12}=0.5$$

$$X_0=0.2, X_1=0.2, X_2=0.2, O_0=0.2, O_1=0.2, O_2=0.$$

Every node of neuron has performed two operation :

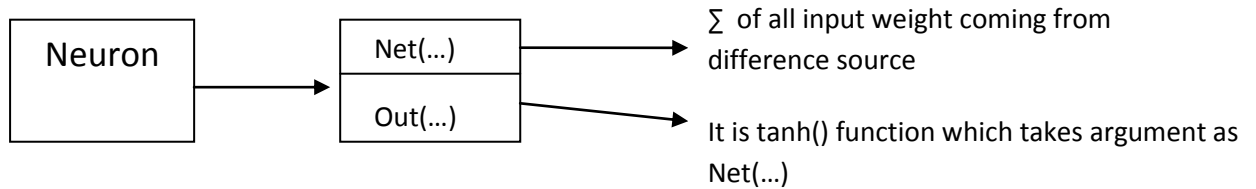
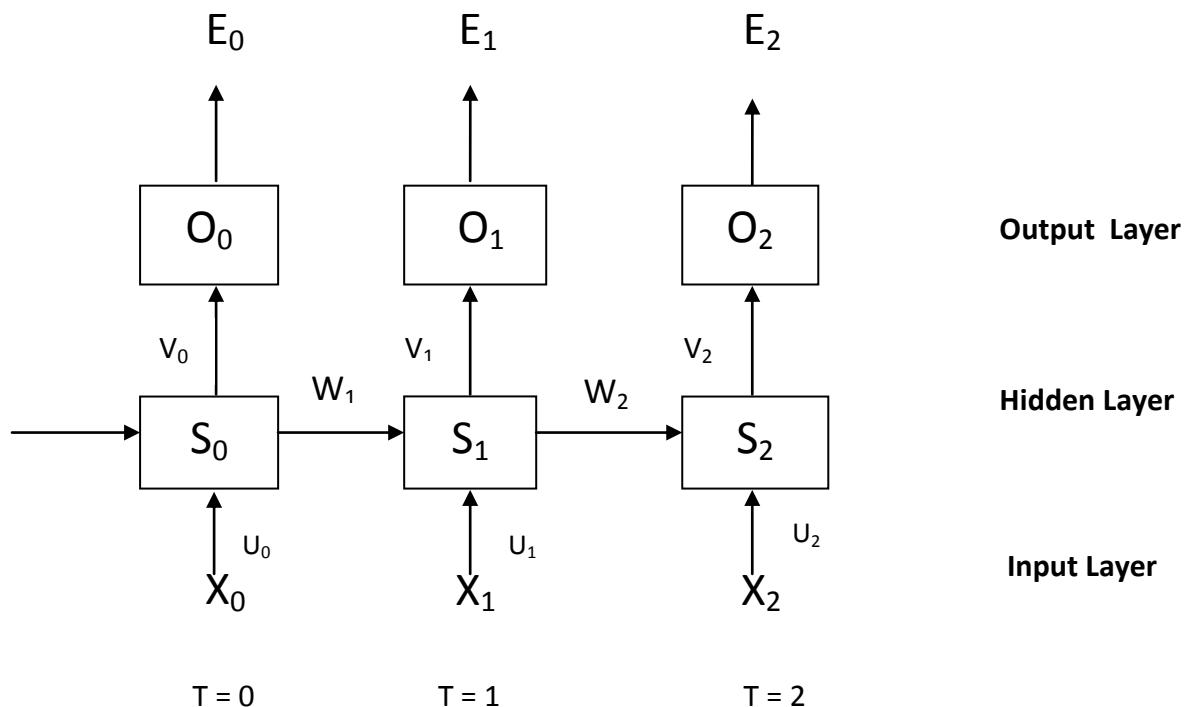


Fig.3.3. it shows that every neuron perform two operation

**Net ( ... )** : It is summation of all input weight coming from different source and

**OUT( ... )** : It is tanh() activation function which takes arguments as result of Net(...)



**Fig. 3.4.** All three layer clearly shown and how parameter pass to one layer to another.

### 3.3. Forward Propagation

In RNN, forward propagation is same as MLP (multilayer perceptron) forward propagation but difference is that in RNN, there is one more input of the previous hidden layer output which is fed to next hidden layer as input. In below figure there is input from  $S_0$  to  $S_1$  and  $S_1$  to  $S_2$  and having corresponding weight  $W_1$  and  $W_2$ .

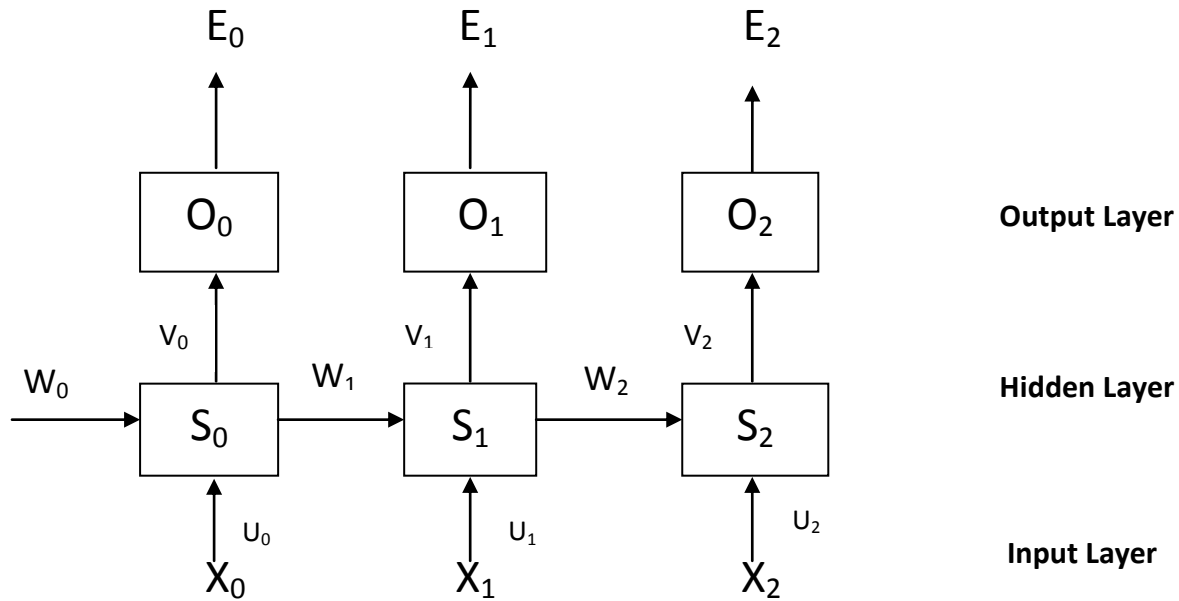


Fig 3.5. This diagram helps to better understand how forward propagation perform.

In forward propagation,

**$S_0$ :**

$$\text{Net}(s_0) = (V_0 X_0) = 0.2 \quad (1)$$

$$\text{Out}(S_0) = \sum (\text{Net}(s_0)) = 0.5498 \quad (2)$$

$$\text{Net}(O_0) = V_0 \text{Out}(S_0) = 0.2749 \quad (3)$$

$$\text{Out}(O_0) = \sum \text{Net}(O_0) = 0.5683 \quad (4)$$

**$S_1$ :**

$$\text{Net}(s_1) = (V_1 X_1 + w_1 \text{Out}(S_0)) = 0.3 \times 0.5 + 0.5498 = 0.5749 \quad (5)$$

$$\text{Out}(S_1) = \sum (\text{Net}(s_1)) = 0.6399 \quad (6)$$

$$\text{Net}(O_1) = V_1 \text{ Out}(S_1) = (0.5 \times 0.6399) = 0.3199 \quad (7)$$

$$\text{Out}(O_1) = \sum \text{Net}(O_1) = 0.5793 \quad (8)$$

**S<sub>2</sub>:**

$$\text{Net}(s_2) = (V_2 X_2 + w_2 \text{ Out}(S_1)) = (0.2 \times 0.5 \times 0.6399) \quad (9)$$

$$\text{Out}(S_2) = \sum (\text{Net}(s_2)) = 0.627 \quad (10)$$

$$\text{Net}(O_2) = V_1 \text{ Out}(S_2) = (0.5 \times 0.6271) = 0.3135 \quad (11)$$

$$\text{Out}(O_2) = \sum \text{Net}(O_2) = 0.5777 \quad (12)$$

### 3.4. Error calculating

Error calculating using Squard error function :-

$$E_{\text{total}} = \sum \frac{1}{2} (\text{Given\_output} - \text{Actual\_output})^2$$

$$E_2 = \frac{1}{2} (\text{Given}(O_2) - \text{Out}(O_2))^2 = 0.0158 \quad (13)$$

$$E_1 = \frac{1}{2} (\text{Given}(O_1) - \text{Out}(O_1))^2 = 0.0390 \quad (14)$$

$$E_0 = \frac{1}{2} (\text{Given}(O_0) - \text{Out}(O_0))^2 = 0.0678 \quad (15)$$

$$E_{\text{Total}} = E_0 + E_1 + E_2 = (0.0158 + 0.0390 + 0.0678) \quad (16)$$

### 3.5. Back Propagation Through Time (BPTT)

In neural networks, we do Forward-Propagation to get the output of our model and check if this output is correct or incorrect, to get the error. Which is nothing but going backwards through our neural network to find the partial derivatives of the error with respect to the weights, which enables us to subtract this value from the weights. Those derivatives are then used by Gradient Descent, an algorithm that is used to iteratively minimize a given function. Then it adjusts the weights up or down, depending on which decreases the error. That is exactly how a Neural Network learns during the training process. So applying Back Propagation, we try to improve weights and minimize the loss, while training.

Within BPTT the error is back-propagated from the last to the first time step, while unrolling all the time steps. This allows calculating the error for each time step, which allows updating

the weights. Note that BPTT can be computationally expensive when you have a high number of time steps.

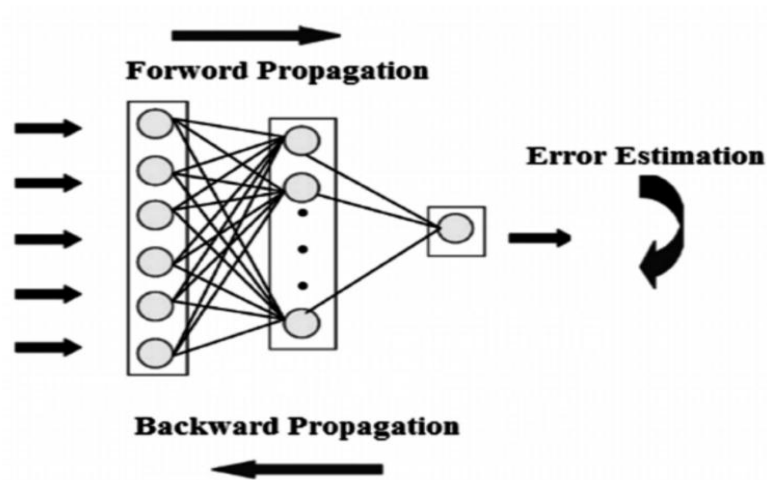


Fig. 3.5 Illustration the concept of Forward Propagation and Backward Propagation perfectly at the example of a Feed Forward Neural Network[ 18 ].

### Back propagation through time: t=2

Unfolding over(t=2):

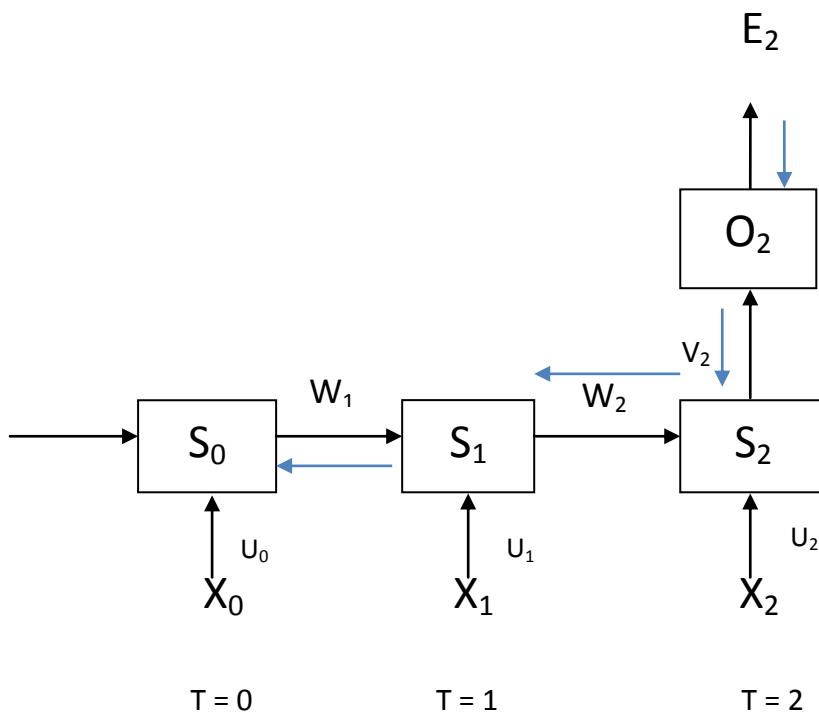


Fig. 3.6 This diagram represents Back propagation when unfolding over t=2

Calculating Gradient of  $E_2$  with respect to  $V_2$  (we have to use chain rule to find derivatives)

Since,

$$\frac{\partial E_2}{\partial V_2} = \frac{\partial E_2}{\partial(\text{out}(O_2))} \times \frac{\partial(\text{out}(O_2))}{\partial(\text{net}(O_2))} \times \frac{\partial(\text{net}(O_2))}{dv_2}$$

(17)

From , First part of the equation (17) .

$$\frac{\partial E_2}{\partial(\text{out}(O_2))} = \frac{d(\frac{1}{2}(\text{Given}(o_2) - \text{Out}(o_2))^2)}{\partial(\text{out}(O_2))} = -(\text{Given}(o_2) - \text{Out}(o_2)) = 0.4271$$

(18)

Second part of the equation (17)

$$\frac{\partial(\text{out}(O_2))}{\partial(\text{net}(O_2))} = \text{out}(O_2)(1 - \text{out}(O_2)) = 0.2338$$

(19)

Third part of the equation (17)

$$\frac{\partial(\text{net}(O_2))}{dv_2} = \text{out}(s_2) = 0.6271$$

(20)

Finally, substituting the value of equations 18,19,20 in equation 17

$$\frac{\partial E_2}{\partial V_2} = \frac{\partial E_2}{\partial(\text{out}(O_2))} \times \frac{\partial(\text{out}(O_2))}{\partial(\text{net}(O_2))} \times \frac{\partial(\text{net}(O_2))}{dv_2} = (0.4271 + 0.2338 + 0.6271) = 0.0626$$

Derivatives of  $E_2$  with respect to  $W_{12}$

$$\frac{\partial E_2}{\partial W} = \frac{\partial E_2}{\partial W_{12}} + \frac{\partial E_2}{\partial w_{01}}$$

(21)

First part of the equation (21), finding derivatives of  $E_2$  w.r.t to  $W_{12}$

$$\frac{\partial E_2}{\partial w_{12}} = \frac{\partial E_2}{\partial \text{out}(O_2)} \times \frac{\partial \text{out}(O_2)}{\partial \text{net}(O_2)} \times \frac{\partial \text{net}(O_2)}{\partial \text{out}(s_2)} \times \frac{\partial \text{out}(s_2)}{\partial \text{net}(s_2)} \times \frac{\partial \text{net}(s_2)}{\partial (w_{12})}$$

(22)

And now from first of equation (22)



$$\frac{\partial E_2}{\partial(out(O_2))} = 0.4271 \quad (23)$$

Now, taking 2<sup>nd</sup> part of equation (22)

$$\frac{\partial(out(O_2))}{\partial(net(O_2))} = out(s_2)(1 - out(s_2)) = (0.5777) \times (1-0.5777) = 0.2439 \quad (24)$$

Taking 3<sup>rd</sup> part of equation (22)

$$\frac{\partial net(O_2)}{\partial out(s_2)} = \frac{\partial(V_2 out(s_2))}{\partial(out(s_2))} = V_2 = 0.5 \quad (25)$$

Taking fourth and fifth part of equation (22)

$$\frac{\partial out(s_2)}{\partial net(s_2)} = out(S_2) - (1 - out(S_2)) = (0.6271 \times (1-0.6271)) = 0.2338 \quad (26)$$

$$\frac{\partial net(s_2)}{\partial(w_{12})} = \frac{\partial(U_2 X_2 + W_{12} out(s_1))}{\partial(w_{12})} = out(S_1) = 0.6399 \quad (27)$$

Finally, Substituting the value of equations 23,24,25,26 and 27 in equations 22.

$$\frac{\partial E_2}{\partial w_{12}} = \frac{\partial E_2}{\partial out(O_2)} \times \frac{\partial out(O_2)}{\partial net(O_2)} \times \frac{\partial net(O_2)}{\partial out(s_2)} \times \frac{\partial out(s_2)}{\partial net(s_2)} \times \frac{\partial net(s_2)}{\partial(w_{12})}$$

$$\text{Or, } \frac{\partial E_2}{\partial w_{12}} = 0.4271 \times 0.2439 \times 0.5 \times 0.2338 \times 0.6395 = 0.0077 \quad (28)$$

Now, taking the Second part from equation 22.

$$\frac{\partial E_2}{\partial w_{01}} = \frac{\partial E_2}{\partial out(O_2)} \times \frac{\partial out(O_2)}{\partial net(O_2)} \times \frac{\partial net(O_2)}{\partial out(s_2)} \times \frac{\partial out(s_2)}{\partial net(s_2)} \times \frac{\partial net(s_2)}{\partial out(S_1)} \times \frac{\partial out(S_1)}{\partial net(S_1)} \times \frac{\partial net(S_1)}{\partial(w_{01})} \quad (29)$$

Taking fifth part of equation 29.

$$\frac{\partial net(s_2)}{\partial out(S_1)} = \frac{\partial(U_2 X_2 + W_{12} out(s_1))}{\partial(out(S_1))} = W_{12} = 0.5 \quad (30)$$

$$\frac{\partial(out(s_1))}{\partial(net(s_1))} = out(S_1) - (1 - out(S_1)) = 0.6399 \times (1 - 0.6399) = 0.2304 \quad (31)$$

$$\frac{\partial(net(s_1))}{\partial(w_{01})} = \frac{\partial(U_1 X_1 + W_{01} out(s_0))}{\partial(W_{01})} = out(S_0) = 0.5498 \quad (32)$$

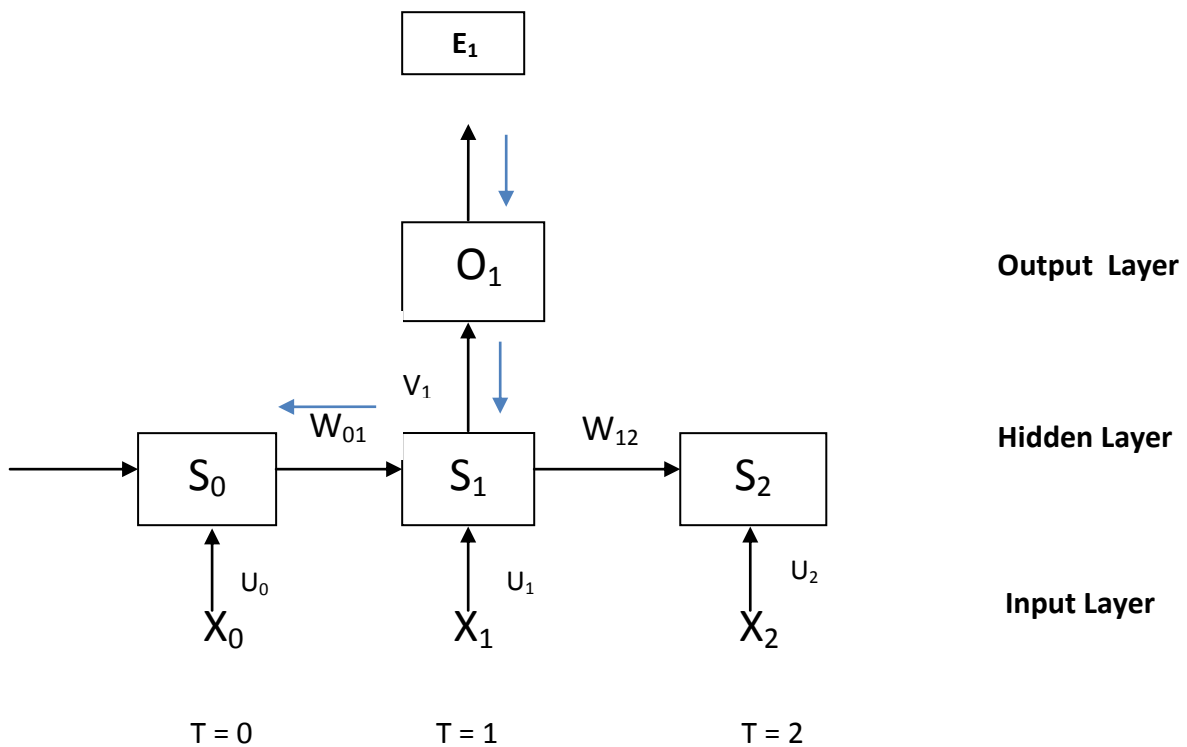
Finally,

$$\frac{\partial E_1}{\partial w_{01}} = \frac{\partial E_2}{\partial out(O_2)} \times \frac{\partial out(O_2)}{\partial net(O_2)} \times \frac{\partial net(O_2)}{\partial out(S_2)} \times \frac{\partial out(S_2)}{\partial net(S_2)} \times \frac{\partial net(S_2)}{\partial out(S_1)} \times \frac{\partial out(S_1)}{\partial net(S_1)} \times \frac{\partial net(S_1)}{\partial (W_{01})}$$

$$\text{Or, } \frac{\partial E_1}{\partial w_{01}} = 0.4271 \times 0.2439 \times 0.5 \times 0.2338 \times 0.5 \times 0.2304 \times 0.5498 = 0.0007 \quad (33)$$

### Back propagation through time: t=1

Unfolding over t=1



**Fig. 3.7** This diagram represents Back propagation when unfolding over t=1

Calculate gradient w.r.t  $V_1$

$$\frac{\partial E_1}{\partial v_1} = \frac{\partial E_1}{\partial(out(O_1))} \times \frac{\partial(out(O_1))}{\partial(net(O_1))} \times \frac{\partial(net(O_1))}{\partial(v_1)} \quad (34)$$

Now, taking first part of the equation 34.

$$\frac{\partial E_1}{\partial(out(O_1))} = \frac{d(\frac{1}{2}(Given(o_1) - Out(o_1))^2)}{\partial(out(O_1))} = -(Given(o_1) - Out(o_1)) = -(0.3-0.5793)=0.2437 \quad (35)$$

Now, taking second part of the equation (34).

$$\frac{\partial(out(O_1))}{\partial(net(O_1))} = out(S_1) - (1 - out(S_1)) = (0.5793)(1-0.5793)=0.2437 \quad (36)$$

Now, taking third part of the equation (34).

$$\frac{\partial(net(O_1))}{\partial(v_1)} = \frac{\partial(v_1 out(S_1))}{\partial(v_1)} = out(S_1) = 0.6399 \quad (37)$$

$$\frac{\partial E_1}{\partial v_1} = \frac{\partial E_1}{\partial(out(O_1))} \times \frac{\partial(out(O_1))}{\partial(net(O_1))} \times \frac{\partial(net(O_1))}{\partial(v_1)} = 0.2793 \times 0.2437 \times 0.6399 = 0.0435$$

Calculating gradient w.r.t.,  $W_{01}$

$$\frac{\partial E_1}{\partial w_{01}} = \frac{\partial E_1}{\partial(out(O_1))} \times \frac{\partial(out(O_1))}{\partial(net(O_1))} \times \frac{\partial(net(O_1))}{\partial(out(S_1))} \times \frac{\partial(out(S_1))}{\partial(net(S_1))} \times \frac{\partial(net(S_1))}{\partial w_{01}} \quad (38)$$

Now, taking first part of the equation (38)

$$\frac{\partial E_1}{\partial(out(O_1))} = \frac{d(\frac{1}{2}(Given(o_1) - Out(o_1))^2)}{\partial(out(O_1))} = -(Given(o_1) - Out(o_1)) = -(0.2 - 0.5793)=0.3793 \quad (39)$$

Now, taking second part of the equation (38)

$$\frac{\partial(out(O_1))}{\partial(net(O_1))} = out(O_1) - (1 - out(O_1)) = (0.5793 \times (1-0.5793))=0.2437 \quad (40)$$

Now, taking third part of the equation (38)

$$\frac{\partial(\text{net}(O_1))}{\partial(\text{out}(S_1))} = \frac{\partial(V_1 \text{net}(S_1))}{\partial(\text{out}(S_1))} = v_1 = 0.5 \quad (41)$$

Now, taking fourth part of the equation (38)

$$\frac{\partial(\text{out}(S_1))}{\partial(\text{net}(S_1))} = \text{out}(S_1) - (1 - \text{out}(S_1)) = (0.6399 \times (1 - 0.6399)) = 0.2304 \quad (42)$$

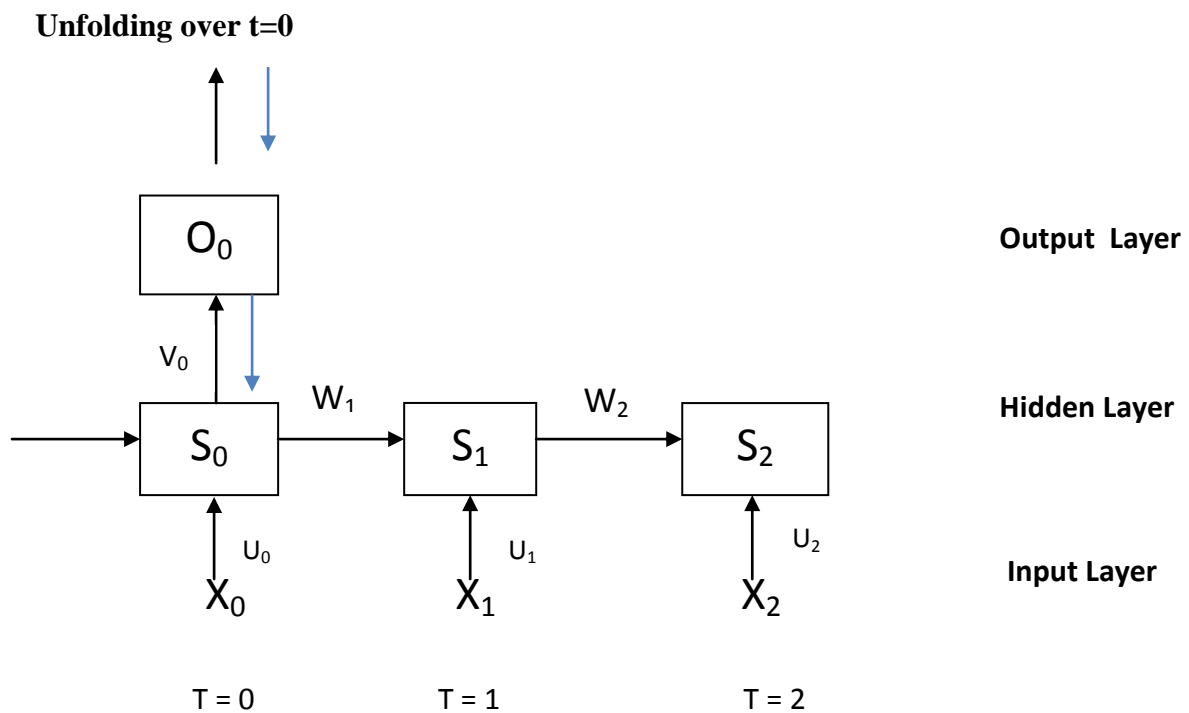
Now, taking fifth part of the equation (38)

$$\frac{\partial(\text{net}(S_1))}{\partial w_{01}} = \frac{\partial(U_1 X_1 + W_{01} \text{out}(S_0))}{\partial(W_{01})} = \text{out}(S_0) = 0.5498 \quad (43)$$

Finally, 
$$\frac{\partial E_1}{\partial w_{01}} = \frac{\partial E_1}{\partial(\text{out}(O_1))} \times \frac{\partial(\text{out}(O_1))}{\partial(\text{net}(O_1))} \times \frac{\partial(\text{net}(O_1))}{\partial(\text{out}(S_1))} \times \frac{\partial(\text{out}(S_1))}{\partial(\text{net}(S_1))} \times \frac{\partial(\text{net}(S_1))}{\partial w_{01}}$$

Or, 
$$\frac{\partial E_1}{\partial w_{01}} = 0.3793 \times 0.2437 \times 0.5 \times 0.2304 \times 0.5498 = 0.0058 \quad (44)$$

### Back propagation through time: t=0



**Fig. 3.8** This diagram represents Back propagation when unfolding over t=0

Calculating gradient with respect to  $V_0$

$$\frac{\partial E_0}{\partial V_0} = \frac{\partial E_0}{\partial(\text{out}(O_0))} \times \frac{\partial(\text{out}(O_0))}{\partial(\text{net}(O_0))} \times \frac{\partial(\text{net}(O_0))}{\partial(V_0)} \quad (45)$$

Now, taking first part of the equation (45)

$$\frac{\partial E_0}{\partial(\text{out}(O_0))} = \frac{d(\frac{1}{2}(\text{Given}(O_0) - \text{Out}(O_0))^2)}{\partial(\text{out}(O_1))} = -(\text{Given}(O_0) - \text{Out}(O_0)) = -(0.2 - 0.5683) = 0.3683 \quad (46)$$

Now, taking third part of the equation

$$\frac{\partial(\text{out}(O_0))}{\partial(\text{net}(O_0))} = \frac{\partial(V_0 \text{out}(S_0))}{\partial(V_0)} = V_0 = 0.5683 \times (1 - 0.5683) = 0.2453 \quad (47)$$

Finally,

$$\frac{\partial E_0}{\partial V_0} = \frac{\partial E_0}{\partial(\text{out}(O_0))} \times \frac{\partial(\text{out}(O_0))}{\partial(\text{net}(O_0))} \times \frac{\partial(\text{net}(O_0))}{\partial(V_0)}$$

$$\text{Or, } \frac{\partial E_0}{\partial V_0} = 0.3683 \times 0.2453 \times 0.5498 = 0.0496$$

(48)

**Weight updating for  $V_2$ .**

$$\frac{\partial E_2}{\partial V_2} = 0.0626 \quad [ \text{ From equation (17) } ]$$

$$V_2' = V_2 - \eta \left[ \frac{\partial E_2}{\partial V_2} \right] = (0.5 - (0.5 \times 0.0626)) = 0.4687$$

(49)

**Weight updating for  $V_1$**

$$\frac{\partial E_1}{\partial V_1} = 0.0435$$

$$V'_1 = V_1 - n \left[ \frac{\partial E_1}{\partial V_1} \right] = (0.5 - (0.5 \times 0.0435)) = 0.4782$$

### Weight updating for $V_0$

$$\frac{\partial E_0}{\partial V_0} = 0.0435$$

$$V'_0 = V_0 - n \left[ \frac{\partial E_0}{\partial V_0} \right] = 0.5 - (0.5 \times 0.0496) = 0.4752$$

### Weight updating for $W_{12}$

$$\frac{\partial E_2}{\partial W_{12}} = 0.0435$$

$$W'_{12} = W_{12} - n \left[ \frac{\partial E_{12}}{\partial W_{12}} \right] = 0.5 - (0.5 \times 0.0077) = 0.4961$$

### Weight updating for $W_{01}$

$$\frac{\partial E_2}{\partial W_{01}} = 0.0007$$

$$\frac{\partial E_1}{\partial W_{01}} = 0.0058$$

$$W'_{01} = W_{01} - n \left[ \frac{\partial E_2}{\partial W_{01}} + \frac{\partial E_1}{\partial W_{01}} \right] = 0.5 - (0.5 \times (0.0007 + 0.0058)) = 0.4967$$

After updating these weights, repeat the same procedure same again and again minimize the error and improve the accuracy.

## 3.6 Applications of RNN

- Recurrent neural network has many uses, specially when it comes to predicting the future. For example In the financial industry, it can be useful to predicting stock prices or the sign of the stock market direction.

- It is widely used in text analysis, image captioning, sentiment analysis and machine translation.

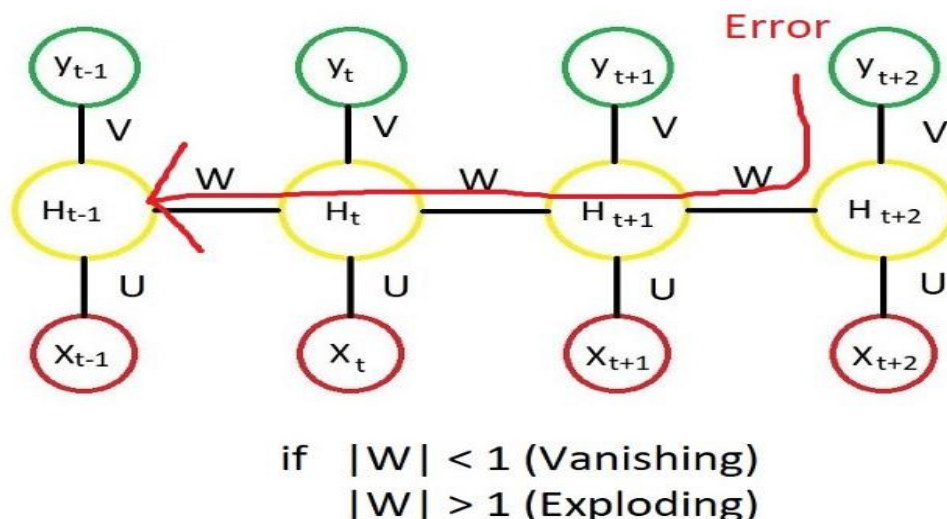
Other area where data comes in sequential fashion.

### 3.7 Issues of standard RNN's

As we have seen above recurrent neural network uses an algorithm **Back Propagation through time** to update the weights of the networks. In which it first calculate gradients from the error using the chain rule in Calculus, then it updates the weights(Gradient descent). since the BPTT starts from the output layer to all the way back to input layer , As we go back with gradients, it is possible that the values get either smaller exponentially or larger exponentially.

**3.7.1. Vanishing Gradients :** When values get smaller exponentially which causes Vanishing Gradients. As we can see in Fig.3.9 when  $|W| < 1$  then it is called vanishing gradient.

**3.7.2. Exploding Gradients :** When values get larger exponentially which causes Exploding Gradients. As we can see in Fig. when  $|W| > 1$  then it is called Exploding Gradient.



**Fig. 3.9** This diagram shows Vanishing and Exploding Gradient [19].

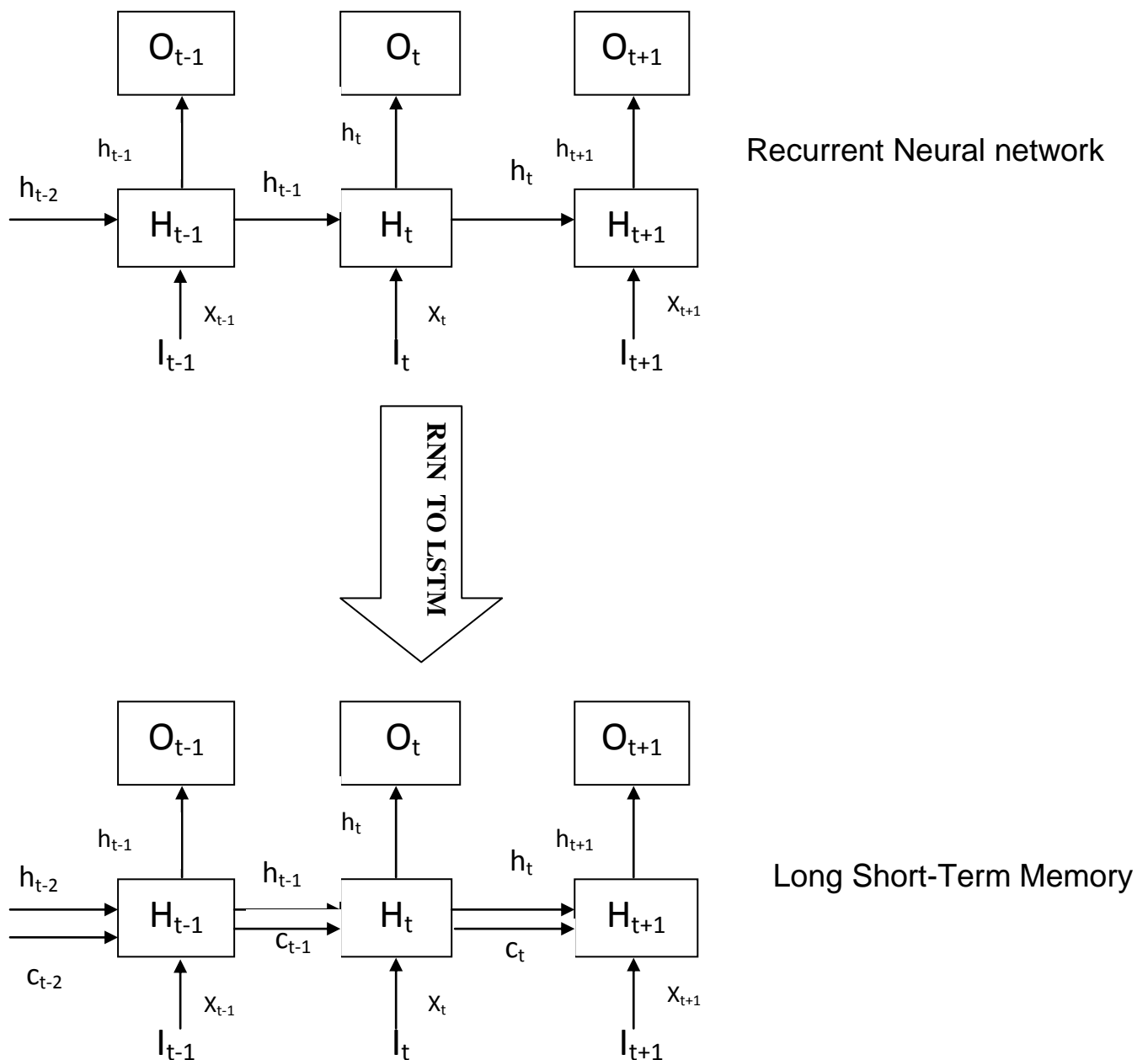
### 3.8. LSTM (Long Short Term Memory)

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. Long Short Term Memory is introduced to overcome the above limitations of Recurrent Neural Network.

**Recurrent Neural Network to Long Short-term Memory:** RNN has limitations

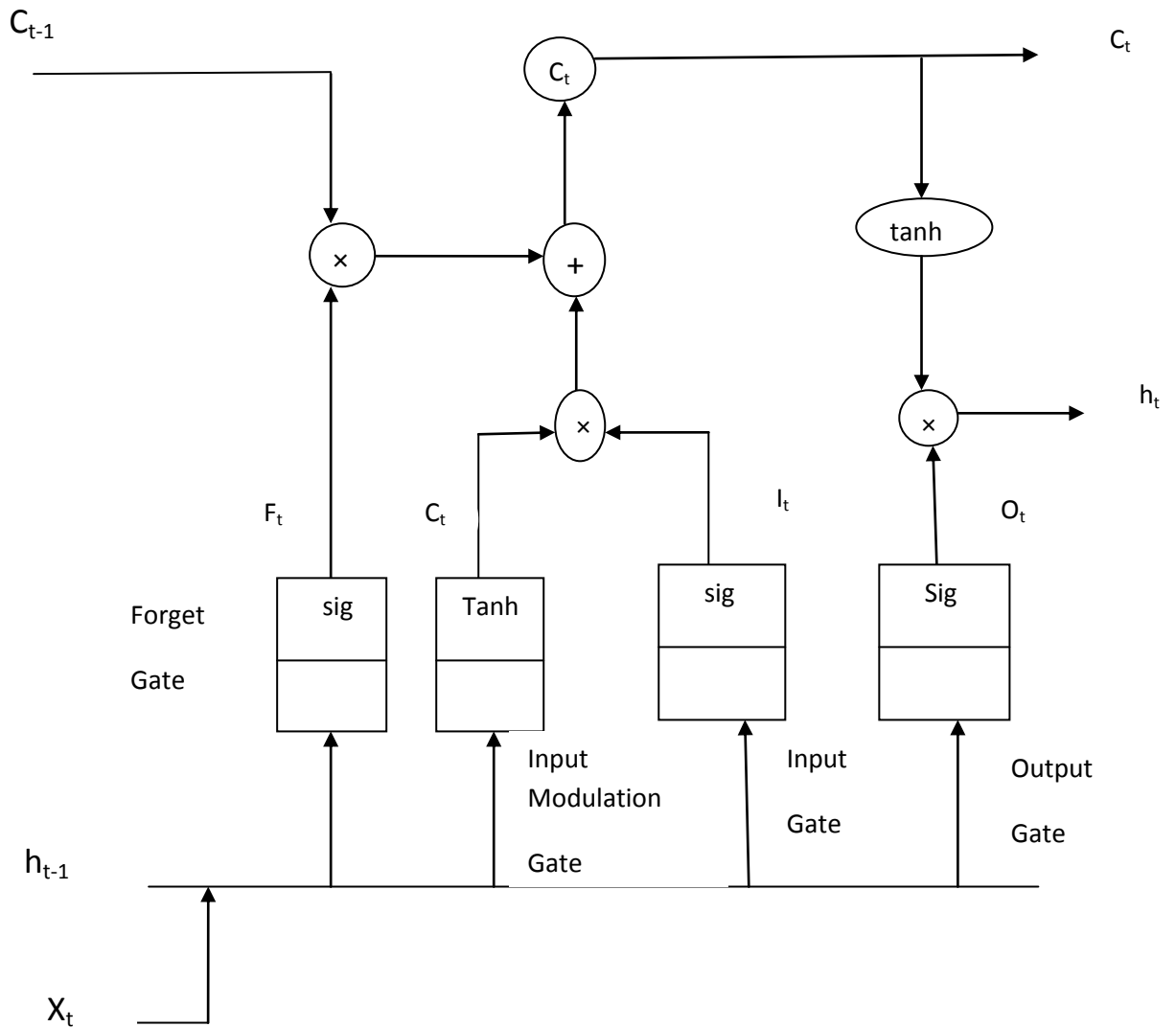
- Due to the vanishing gradient problem, RNN's effectiveness is limited when it needs to go back deep in to the context.
- There is no finer control over which part of the context needs to be carried forward and how much of the past needs to be "forgotten".

LSTM were designed to resolve vanishing gradients through a gating mechanism.





At hidden layer ( $H_t$ ) :



Working of gates:

$$C_t = \tanh(W_c X^t + U_c h^{t-1} + b_c)$$

$$I^t = \text{sig}(W_i X^t + U_i h^{t-1} + b_i)$$

$$F^t = \text{sig}(W_f x^t + U_f h^{t-1} + b_f)$$

$$O^t = \text{sig}(W_o X^t + U_o h^{t-1} + b_o)$$

## Convolutional neural network

### 4.1 Introduction to Convolutional Neural Network

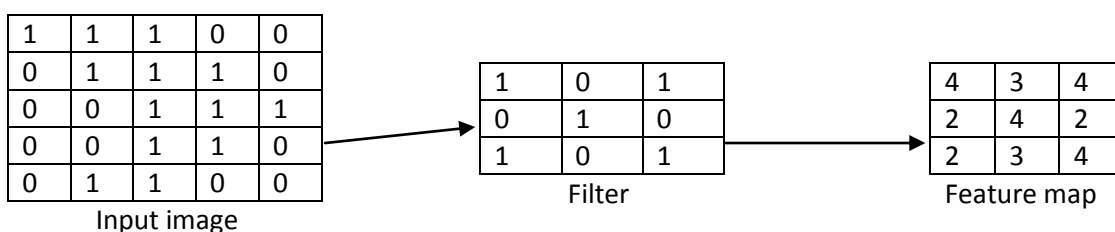
A convolution neural network (CNN)[ 23] is one of the most popular algorithm for deep learning, a type of machine learning in which a model learns to perform classification tasks directly from images, video, or text. CNNs are particularly useful for finding patterns in images to recognize object, faces and scenes. They learn directly from image data, using patterns to classify images and eliminating the need for manual feature extraction. CNNs provide an optimal architecture for image recognition and pattern detection[ 17 ]. Combined with advances in GPUs and parallel computing, CNNs are a key technology underlying new developments in automated driving and facial recognition. For example, deep learning applications use CNNs to examine thousands of pathology reports to visually detect cancer cells.[ 5] CNNs also enable self-driving cars [5 ] to detect objects and learn to tell the difference between a street sign and a pedestrian.

There are four components of Convolutional neural network

1. Convolution
2. Non Linearity (ReLU)
3. Pooling
4. Classification (Fully Connected Layer)

#### 1.2.1 Convolution:

The convolution is used to extract the features of the object on the image i.e it learns specific patterns within the picture. Convolution is an element-wise multiplication. The computer scans a part of the image, usually with a dimension of 3x3 and multiplies it to a filter. The output of the element-wise multiplication is called a feature map. This step is repeated until all the image is scanned. After the convolution, the size of the image is reduced.



There are numerous channels available. In Fig.4.1 it has different kernel for different operation, Kernel is a synonym of the filter.





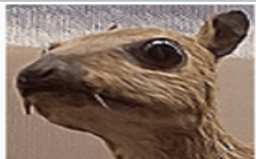
Operation	Kernel	Image result
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Edge detection</b>	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Sharpen</b>	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Fig. 4.1 different types of kernel with their uses and corresponding matrix [23].

### 1.2.2. Non Linearity (ReLU) :

The activation function of a node defines the output given a set of inputs. You need an activation function to allow the network to learn non-linear pattern. The usual activation function for convolutional neural network is the Relu.

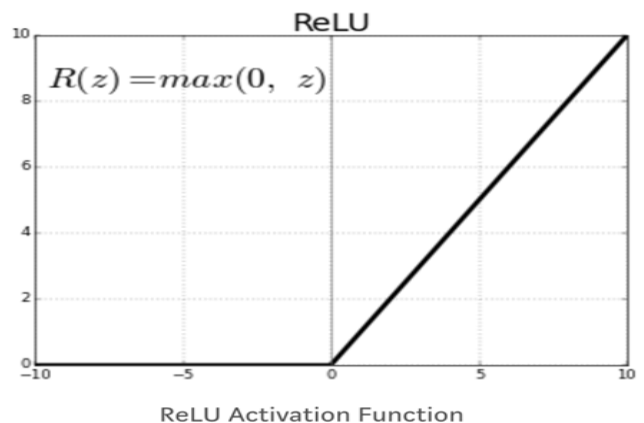


Fig.1.4 ReLU activation function [22]

At the end of the convolution operation, the output passes to **Relu** activation function to allow non-linearity. All the pixel with a negative value will be replaced by zero.

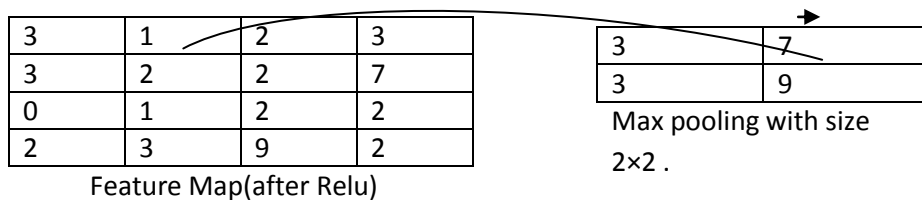
### 1.2.3.Pooling

The pooling is used to reduce the dimensionality of the input image. A convolution neural network uses of pooling layer provides following advantages.

- By having less spatial information, computation performance is increased
- Less spatial information also means less parameters, so less chance to overfit.
- Networks gets some translation invariance.

#### 1.2.3.1: Max-pooling

It passes the maximum value among the pixels of the given patch area. As we have seen in Fig.4.1, feature map matrix obtained after applying convolution onto image pixel matrix and when we apply max pooling into feature map (after Relu ) then it returns maximum value.



### 1.2.4 Fully connected layers

This is similar to traditional neural network as we have already discussed in Chapter 2. We connect all neurons from the previous layer to the next layer. We use a softmax activation function [discussed in Chapter 3 ] to classify input image. Basically it is used when we have to classifying a image but in case of my image captioning this layer is not required because we only need a features of each image which can be obtained using convolution operation.

## Image Captioning

### 3.1.1 Introduction to Image captioning

Image captioning is the process of generating textual description from an image, based on the objects and actions in the image.

For example:

This process has many potential applications in real life. A interesting one would be to save the captions of an image so that it can be retrieved easily at a later.



Fig. 3.1 images description is shown in corresponding image.[ 18 ]

### 3.1. 2 The Image Captioning Problem entail

The first thing that comes to our mind is an image can be describe many ways

Here are a few sentences that described Fig.3.2.

- i) A man and girl are sitting on the ground and eating
- ii) A man and a little girl are sitting on a sidewalk near a blue bag and eating.



Fig. 3.2 This picture descried in three different ways

- iii) A man wearing a black shirt and a little girl wearing an orange dress share a treat.

## 3.2. Architectures:

**3.4.1. Word embeddings:** The Embedding layer is used to create word vectors for incoming words. It sits between the input (matrix of word's indices in the vocabulary) and the LSTM layer, i.e. the output of the Embedding layer is the input to the LSTM layer.

**3.4.2. Recurrent neural network:** In our model RNN is to take a prefix of embedded words and produce a single vector that represents the sequence. Long Short Term Memory was used in our experiments for the simple reason that it is a powerful RNN that has only one hidden state vector (discussed in Chapter 3).

**3.4.3. Image:** Before to training, all images were vectorised using the activation values of the fc7 ( Extracted featured at last layer of VGG model ) of the VGG OxfordNet 19-layer convolution neural network [ 17], which is trained to perform object recognition and returns a 4096-element vector. The convolutional neural network is not influenced by the caption generation training. During training, a feed forward layer of the neural network compresses this vector into a smaller vector.

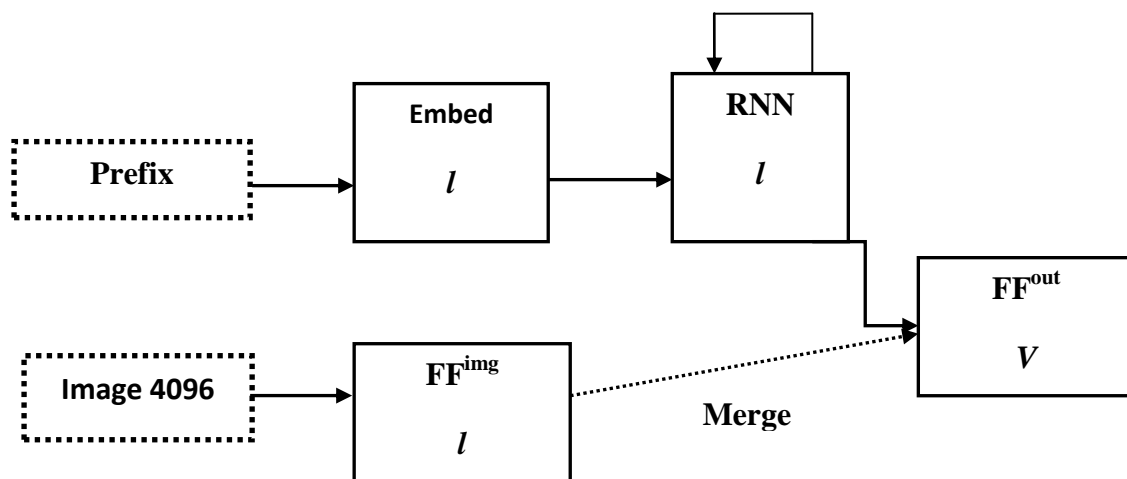


Fig. 3.3 An illustration of the merge architecture  
Of our model. [1]

`FF' – fully connected feed forward layer

`FF<sup>img</sup>' - layer projecting the image vector

`FF<sup>out</sup>' - layer projecting into the softmax output;

'l' - the layer size (which is the same for three different layers);

'v' - the vocabulary size (which is different for different datasets).

**3.4.4. Output:** Once the image and the caption prefix have been vectorised and mixed into a single vector, the next step is to use them to predict the next word in the caption. This is done by passing the mixed vector through a feed-forward layer with a softmax activation function that outputs the probability of each possible next word in the vocabulary. Based on this distribution, the next word that comes after the prefix is selected.

The merge architecture had been applied in our model:

**Merge:** The image vector and caption prefix vector are concatenated into a single vector before being fed to the output layer.

We now discuss the architecture in a more formal notation. As a matter of notation, we treat vectors as horizontal.

The **LSTM** model (also discussed in Chapter 3) is defined as follows:

$$r_t = sig(x_t W_{xr} + S_{t-1} W_{sr} + b_r) \quad (1)$$

$$u_t = sig(x_t W_{xu} + S_{t-1} W_{su} + b_u) \quad (2)$$

$$C_t = tanh(x_t W_{xc} + (r \odot S_{t-1}) W_{sc} + b_c) \quad (3)$$

$$S_t = u_t \odot S_{t-1} + (1 - u_t) \odot C_t \quad (4)$$

where  $x_t$  is the  $t^{\text{th}}$  input,  $S_t$  is the hidden state vector after  $t$  inputs,  $r_t$  is the reset gate after  $t$  inputs,  $u_t$  is the update gate after  $t$  inputs,  $W_{\alpha\beta}$  is the weight matrix between  $\alpha$  and  $\beta$ ,  $b_{\alpha}$  is the bias vector for  $\alpha$ , and  $\odot$  is the element wise vector multiplication operator. In the above, 'sig' represents sigmoid function

which is defined as:

$$sig(x) = \frac{1}{1+e^{-x}} \quad (5)$$

The feed forward layers used for the image and output are defined as

$$z = xW + b \quad (6)$$

where  $z$  is the net vector,  $x$  is the input vector,  $W$  is the weight matrix, and  $b$  is the bias vector.

The net vector can then be passed through an activation function, such as the **softmax** function, which is defined as

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (7)$$

where  $\text{softmax}(z)_i$  refers to the  $i^{\text{th}}$  element of the new vector. Another activation function is the rectified linear unit function, or ReLU, which is defined as

$$\text{ReLU}(z)_i = \max(z_i, 0) \quad (8)$$

where  $\text{ReLU}(z)_i$  refers to the  $i^{\text{th}}$  element of the new vector.

### 3.5. Methodology to Solve the Task :

The task of image captioning is divided into two modules one is an **image based model** which extracts the features and the other one is a **language based model** which translates the features and objects given by our image based model to a natural sentence. For our image based model (encoder) we rely on a Convolutional neural network model and for our language based model (viz decoder) we rely on a Recurrent Neural Network [2].

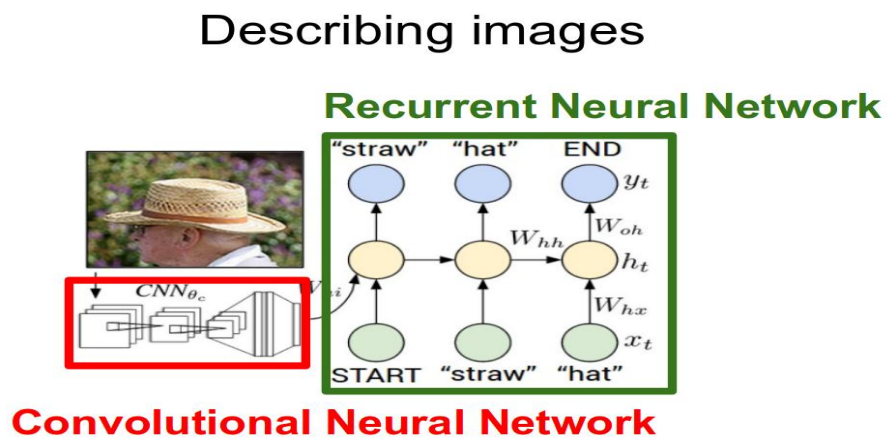


Fig3.4 The image below summarizes the approach given above. [2]

We have used VGG pre-trained CNN model to extract the features from our input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the LSTM network. This network is trained as a language model on our feature vector.



For training our LSTM model, we predefine our label and target text.

For example, if the caption is “A man and a girl sit on the ground and eat.”, our label and target would be as follows –

*Label* – [ <start>, A, man, and, a, girl, sit, on, the, ground, and, eat, . ]

*Target* – [ A, man, and, a, girl, sit, on, the, ground, and, eat, ., <end>

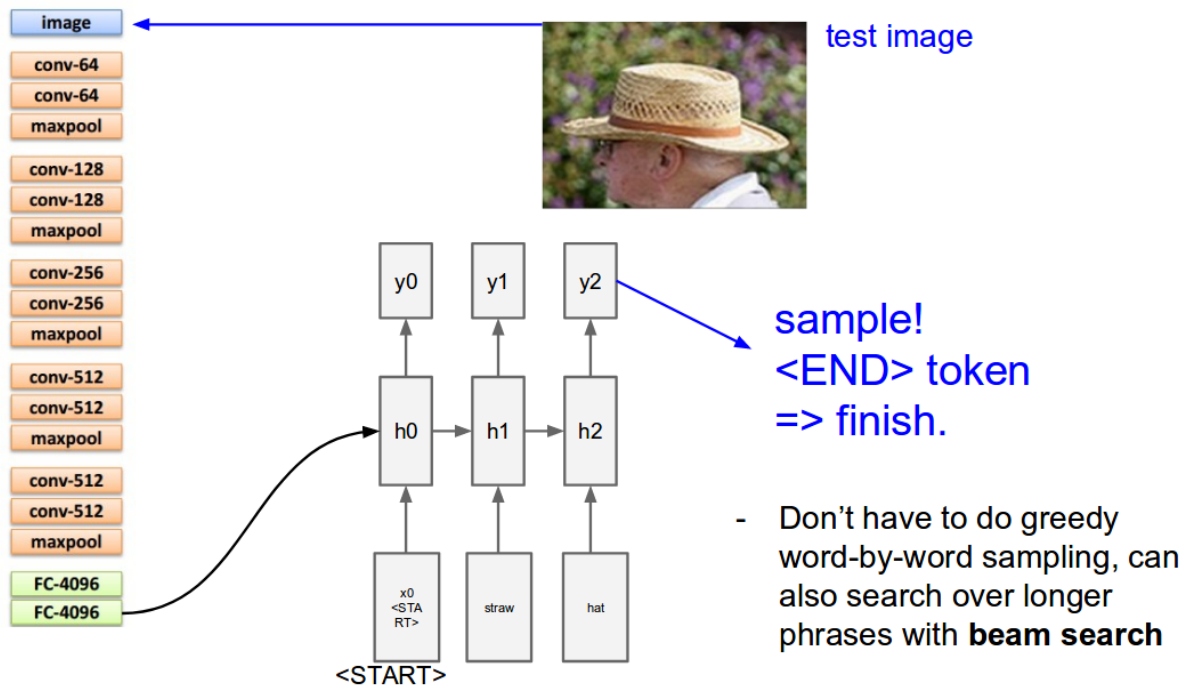


Fig 3.5 This is done so that our model understands the start and end of our labeled sequence.[2]

## 5.1 Implementation Mechanism

In this implementation, we used a pre-trained Visual Geometry Group model [24](discussed later in this Chapter) as encoder and LSTM (discussed later in this chapter) as decoder.

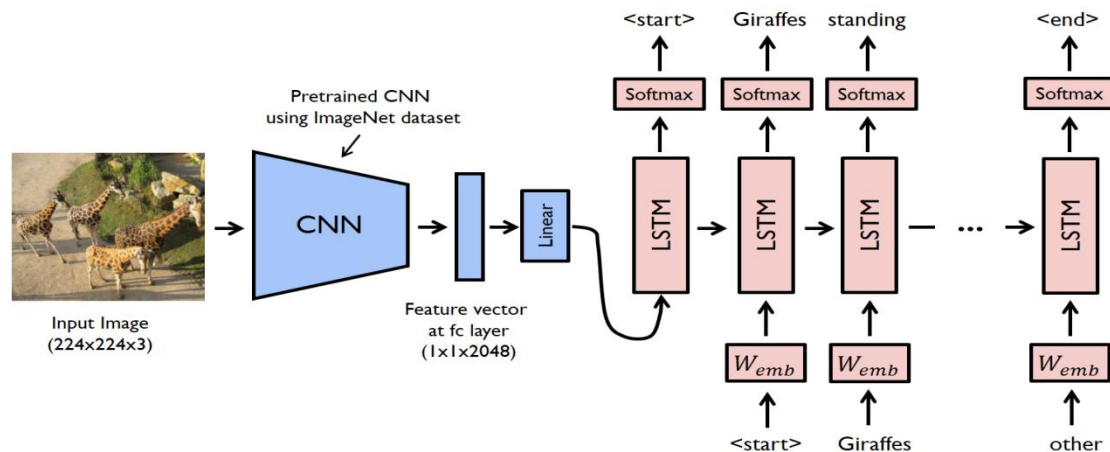


Fig.5.1 Image captioning implementation mechanism[2].

**5.2 Dataset :** We have used Flickr8k dataset [ ] which is small in size (about 1 GB) and it is convenient to process this dataset on personal computer. We have downloaded the dataset and unzip it into our current working directory. We had two directories:

- **Flicker8k\_Dataset:** Contains 8092 photographs in JPEG format.
- **Flicker8k\_text:** Consists of a number of files containing different sources of descriptions for the photographs.

The dataset had a pre-defined training dataset (6,000 images), development dataset (1,000 images), and test dataset (1,000 images).

The complete process consists of 6 phases; they are:

1. Preparation of image Data
2. Preparation of text Data
3. Develop Deep Learning Model
4. Train With Progressive Loading

5. Evaluate Model
6. Generate New Captions

### 5.3.2 Preparation Image Data :

We have been use Oxford Visual Geometry Group, (or VGG), model that won the ImageNet competition in 2014 [24]. Keras [25] provides this pre-trained model directly. ‘Image features’ were pre-computed using VGG model and saved them to file.VGG model was loaded using VGG class in keras and last layer was removed from the loaded model, because it was known to us that last layer is used to predict a classification of image. It is to be noted that we were not interested in classifying images, rather in the internal representation of the image exactly before a classification was made. These were the “features” that the model had extracted from the image.

A function named *extract\_features()* has been defined ,by given a directory name, which loaded each image and prepared it for VGG, and collected the predicted features from the VGG model.

The image features were an array of 4,096 elements for each image.This function was called to prepare the photo data for testing our models, then saved the results to a file named *‘features.pkl’*.

### 5.1.3 Preparing Text Data :

The dataset contains multiple descriptions for each image and the text of the descriptions required some minimal cleaning.The file containing all of the description has been loaded. Each image had a unique identifier. This identifier was used on the photo filename and in the text file of descriptions. Next, a function *load\_descriptions()* has been define that, given the loaded document text, returned a dictionary of image identifiers to descriptions. Each image identifier is mapped to a list of one or more textual descriptions. Afterwards, we were need to cleaned the description text. The descriptions were already tokenized and easy to work with. Text file has been cleaned in the following ways in order to reduced the size of the vocabulary of words:

- All words Converted to lowercase.
- All punctuation are remove
- Removed all words that are one character.
- Removed all words with numbers in them.

A function *clean\_descriptions()* has been defined that, given the dictionary of image identifiers to descriptions, steps through each description and cleaned the text.

A function *clean\_descriptions()* has been define by passing the dictionary of image identifiers to description as arguments, steps through each description and cleaned the text.

For reference, clean descriptions has been transferred into a set and print its size to get an idea of the size of our dataset vocabulary. Afterward, saved the dictionary of image identifiers and descriptions to a new file named *descriptions.txt*, with one image identifier and its description in per line. Then, A function *save\_descriptions()* has been define with given a dictionary containing the mapping of identifiers to descriptions and a filename, and saved the mapping to file. Finally, the cleaned descriptions were written to '*descriptions.txt*'.

## 5.1.4. Development of Deep Learning Model

In this section, we would define the deep learning model and how fitted it on the training dataset.

The development step consists of full phases.

1. Loading Data.
2. Defining the Model.
3. Fitting the Model.

### 5.1.4.1 Loading Data

Prepared photo and text data has been loaded so that it could be used to fit the model.

We trained the data on all of the photos and captions in the training dataset. While training, we were going to monitor the performance of the model on the development dataset and used that performance to decide when to save models to file.

The train and development dataset had been pre-defined in the *Flickr\_8k.trainimages.txt* and *Flickr\_8k.devImages.txt* respectively, that both contain lists of photo file names. From these file names, we extracted the photo identifiers and used identifiers to filter photos and descriptions for each set.

We defined a function *load\_set()* that loaded a pre-defined set of identifiers given the train or development sets filename.

Now, we loaded the photos and descriptions using the pre-defined set of train or development identifiers.

We then defined a function *load\_clean\_descriptions()* that loaded the cleaned text descriptions from '*descriptions.txt*' for a given set of identifiers and returned a dictionary of identifiers to lists of text descriptions.

The model we developed generated a caption given a photo, and the caption generated one word at a time. The sequence of previously generated words provided as input. Therefore, we

need ‘*first word*’ to start the generation process and a ‘*last word*’ to signal the end of the caption.

We used the strings ‘*startseq*’ and ‘*endseq*’ for this purpose. These tokens were added to the loaded descriptions.

Next, we loaded the photo features for a given dataset.

We defined a function named *load\_photo\_features()* that loaded the entire set of photo descriptions, then returned the subset of interest for a given set of photo identifiers.

#### 5.1.4.2 Defining the Model

We defined a deep learning based on the “*merge-model*” which have been discussed in Chapter 3.

We described the model in three parts:

- **Photo Feature Extractor:** This is a 16-layer VGG model pre-trained on the ImageNet dataset. We pre-processed the photos with the VGG model and used the extracted features predicted by this model as input[1].
- **Sequence Processor:** This is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer[1].
- **Decoder :** Both the feature extractor and sequence processor output a fixed-length vector. These were merged together and processed by a Dense layer to make a final prediction[1].

The Photo Feature Extractor model took input photo features of a vector of 4,096 elements. These were processed by a Dense layer to produce a 256 element representation of the photo.

The Sequence Processor model took input sequences with a pre-defined length (34 words) which were then fed into an Embedding layer that used a mask to ignore padded values. It was followed by an LSTM layer with 256 memory units. Both the input models produced a 256 element vector. The Decoder model merged the vectors from both input models using an addition operation. It was then fed to a Dense 256 neuron layer and then to a final output Dense layer that made a **Softmax** prediction over the entire output vocabulary for the next word in the sequence.

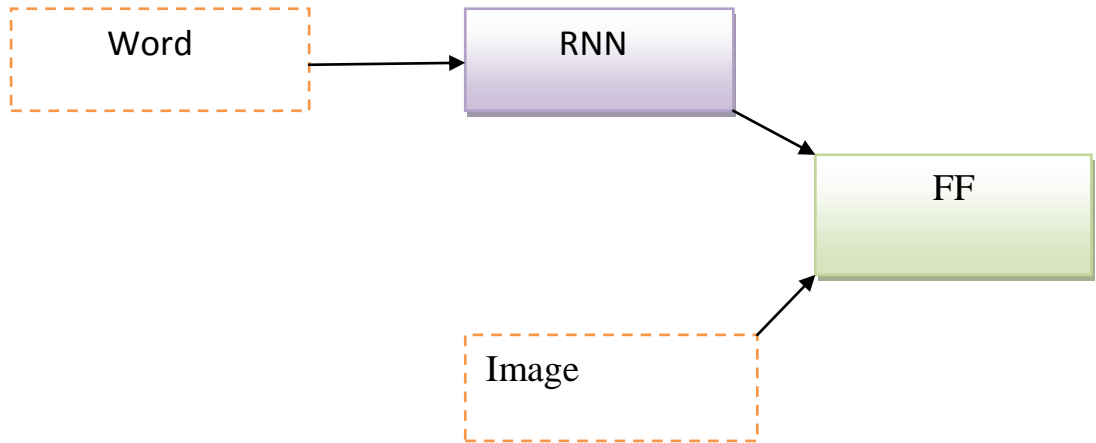


Fig. 5.2 Schematic of the Merge Model For Image Captioning[1]

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.

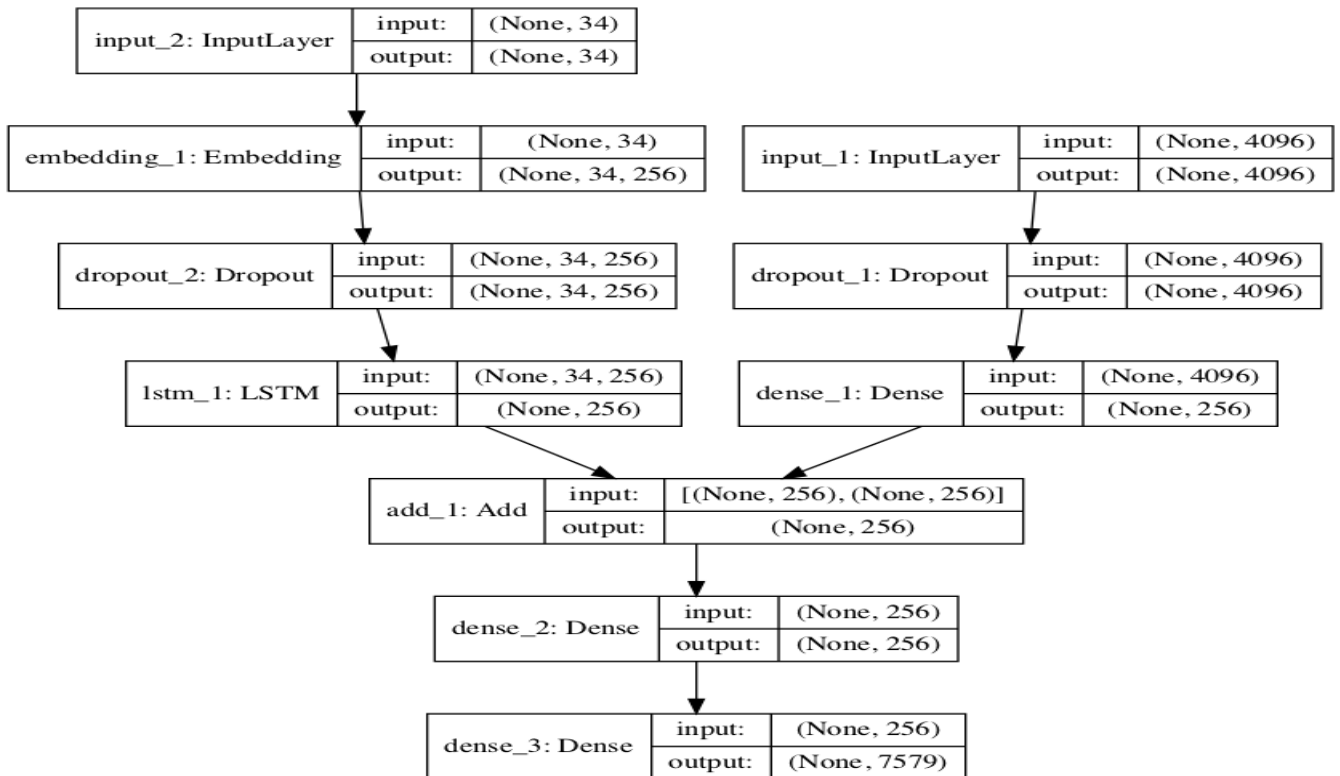


Fig.5.4 7 Plot of the Caption Generation deep Learning Model

### 5.1.4.3 Fitting the Model :

After defining the model, we fitted it on the training dataset which means adapt the weights on a training dataset. The model learnt fast and quickly overfitted the training dataset. For this reason, we monitored the skill of the trained model on the holdout development dataset. When the skill of the model on the development dataset improved at the end of an epoch, we

saved the whole model to file. At the end of the process, we then used the saved model with the best skill on the training dataset as our final model. We did this by defining a *ModelCheckpoint* in Keras and specifying it to monitor the minimum loss on the validation dataset and saved the model to a file that had both the training and validation loss in the filename.

### 5.1.2. Evaluating model

Once the model was fit, we evaluated the skill of its predictions on the holdout test dataset.

We evaluated a model by generating descriptions for all photos in the test dataset and evaluated those predictions with a standard cost function.

After that, we were able to generate a description for a photo using a trained model.

This involved passing in the start description token '*startseq*', generating one word, then calling the model recursively with generated words as input until the end of sequence token was reached '*endseq*'.

We defined a function named *generate\_desc()* implements this behavior and generates a textual description given a trained model, and a given prepared photo as input. It calls the function *word\_for\_id()* in order to map an integer prediction back to a word.

We will generate predictions for all photos in the test dataset and in the train dataset.

We defined a function named *evaluate\_model()* evaluated a trained model against a given dataset of photo descriptions and photo features. The actual and predicted descriptions were collected and evaluated collectively using the corpus BLEU score that summarizes how close the generated text is to the expected text.

### 5.1.3 Generate New Captions :

We need the Tokenizer (every word mapped to unique integer) for encoding generated words for the model while generating a sequence, and the maximum length of input sequences, used when we defined the model (e.g. 34).

We hard coded the maximum sequence length. With the encoding of text, we created the tokenizer and saved it to a file so that later we would be able to load it whenever we need it without needing the entire Flickr8K dataset..

We created the Tokenizer and saved it as a pickle file *tokenizer.pkl* and after that we loaded the tokenizer whenever we need it without loading the entire training dataset.

## 5.2 Results:

Some results of our model gives appropriate caption and some results are not good.

File name: Example1.jpg



**Fig. 5.4: startseq dog is running through the grass endseq** (This caption predicted by our model)

File name: Example2.jpg



**Fig 5.5 startseq two man are standing in front of the water endseq** (This caption predicted by our model)



File name: Example3.jpg



**Fig 5.6 Caption:** startseq two dogs are running through the grass endseq

This picture has grass in background and a running dog our model predicted background and foreground object correctly. Only mistake is the in the quantity of the foreground object which should be one dog but our model predicted two dog.

File name: Example4.jpg



**Fig. 5.7 Caption:** startseq man in red shirt is its bike on the catch endseq (This caption predicted by our model)

In fig. 5.6 have car in foreground and wood in background and this picture had caption “moving car in the forest” but our model is not able to predict object correctly and give totally wrong caption.

File name: Example5.jpg



**Fig. 5.8** Caption : startseq dog is playing in the water endseq

### Conclusion and Future work

For the task of image captioning, in this project, a deep learning model has been built that generates image captions. This model is based on a Convolution neural network and Long Short Term Memory. A pre-trained image-model (VGG) is used to generate a "features-vector" of what the image contains, and Long Short Term Memory is used to train the model to map this "features-vector" to a sequence of words. It is important to understand that this model does not have a human-like understanding of what the images contain. If it gives an image of a dog and correctly produces a caption then, it does not mean that the model has a deep understanding of what a giraffe is.

In future, this concept can be used to work in text-to-speech conversion, so that the generated descriptions are automatically read out loudly. In addition, static images can only provide blind people with information about one specific time instant, while video caption generation could provide blind people with continuous real time information; in such cases, the above mentioned method will be useful.

## Bibliography

[1]. Where to put the Image in an Image Caption. Generator, Marc Tanti Albert Gatt Institute of Linguistics and Language Technology University of Malta marc.tanti.06@um.edu.mt albert.gatt@um.edu.mt Kenneth P. Camilleri Department of Systems and Control Engineering University of Malta kenneth.camilleri@um.edu.mt 12-Mar-2018

[2].<https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/>

[3].[https://www.tutorialspoint.com/artificial\\_neural\\_network/artificial\\_neural\\_network\\_basic\\_concepts.htm](https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm)

[4]. Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. JAIR,

[5]. Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, 1409.1556.

[6]. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In Proc. CVPR'09.

- [7]. Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2016). Selfcriticalsequence training for image captioning. CoRR, 1612.00563.
- [8]. Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2015a). Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). Proc. ICLR'15.
- [9]. P. Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, 1412.6980.
- [10]. Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A methodfor automatic evaluation of machine translation. In Proc. ACL'02, pages 311.
- [11]. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll\_ar,P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In Proc. ECCV'14, pages 740{755.
- [12]. Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proc. Workshop on Intrinsic and extrinsic evaluation measures for machine translation and/or summarization, volume 29, pages 65-72.
- [13]. Vedantam, R., Zitnick, C. L., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In Proc. CVPR'15.

[14]. <https://www.studocu.com/en/u/751952>

[15]. Generator, Marc Tanti Albert Gatt Institute of Linguistics and Language Technology University of Malta marc.tanti.06@um.edu.mt albert.gatt@um.edu.mt Kenneth P. Camilleri Department of Systems and Control Engineering University of Malta kenneth.camilleri@um.edu.mt 12-Mar-2018.

[16]. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. CoRR, 1301.3781.

[17]. Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, 1409.1556.

[18] <https://www.analyticsindiamag.com/wp-content/uploads/2018/01/neural-network-04-768x432.jpg>

[19]. <https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235>

[20]. [https://www.tutorialspoint.com/artificial\\_intelligence/artificial\\_intelligence\\_neural\\_networks.htm](https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm)

[21]. [https://www.tutorialspoint.com/artificial\\_neural\\_network/artificial\\_neural\\_network\\_building\\_blocks.htm](https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_building_blocks.htm)

[22]. [https://cdn-images-1.medium.com/max/800/0\\*vGJq0cIuvTB9dvf5](https://cdn-images-1.medium.com/max/800/0*vGJq0cIuvTB9dvf5).

[23]. <https://www.guru99.com/convnet-tensorflow-image-classification.html>

[24]. <https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>

[25]. [https://keras.io/examples/cifar10\\_cnn/](https://keras.io/examples/cifar10_cnn/)

[26]. <https://ayearofai.com/rohan-lenny-3-recurrent-neural-networks10300100899b>