

# **Further Analysis of Accident Detection**

Project report submitted in partial fulfillment of the requirement for the degree  
of Bachelor of Technology

in

**Computer Science and Engineering/Information Technology**

By

Prattyush Shanker Sinha (161471)

Under the supervision of

Dr.Rakesh Kanji

to



Department of Computer Science & Engineering and Information Technology  
**Jaypee University of Information Technology Wanknaghat, Solan-173234,**  
**Himachal Pradesh**

**Certificate**

## **Candidate's Declaration**

I hereby declare that the work presented in this report entitled “ **Further Analysis of Accident Detection**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from May 2020 to June 2020 under the supervision of **Dr. Rakesh Kanji** , Assistant Professor, Department of Computer Science and Information Technology. The matter embodied in the report has not been submitted for the award of any other degree or diploma.



(Student Signature)

Prattyush Shanker Sinha, 161471.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



(Supervisor Signature)

Dr. Rakesh Kanji

Assistant Professor

Department of Computer Science and Information Technology

Dated:

## **ACKNOWLEDGEMENT**

I would like to thank my guide Dr. Rakesh Kanji for providing me with necessary inputs and guidance for further exploring the techniques useful in Accident Detection. He was consistent in supporting me in my ideas and filling the voids existing previously in my project and its overall applications which may have arisen previously. This not only resulted in better project outcomes but also helped me in my knowledge and deep understanding in the vast areas of machine learning and its real world usage.

<b>TABLE OF CONTENT</b>	<b>Page no.</b>
Chapter-1 Introduction	<b>1-11</b>
Chapter-2 Literature Survey	<b>12-14</b>
Chapter-3 System Development	<b>15-31</b>
Chapter-4 Performance Analysis	<b>32-29</b>
Chapter-5 Conclusion	<b>40-42</b>

<b>List of Figures</b>	<b>Page No.</b>
1.1.3 Ministry of Road Transport data	3
1.2.1 Seat Belt System	5
1.2.2 SOS calls system in cars	5
1.2.3 News about a road accident	6
1.2.4 Prompt arrival of help	7
1.4.1.1 Video library	8
1.4.1.2 Size of video footages	9
1.4.1.3 Uniform dataset	9
1.4.2 Array of pixels in frame	10
2.1 ML algorithms for prediction	13
2.2 Vanishing gradient defect	14
3.1 A standard RNN	15
3.2 LSTM to remove vanishing gradient	16
3.1.1 Gates used in LSTM	16
3.1.3.1 Learn Gate	18
3.1.3.2 forget gate	19
3.1.3.3 Remember Gate	20
3.1.3.4 use gate	21
3.2 Steps in LSTM	22
3.2 Gated RNN	23
3.2.1 RNN with GRU	24
3.2.1.2 Reset Gate	25
3.2.1.3 Current memory	26
3.2.1.4.1 final GRU output	27
3.2.1.4.2 final memory of GRU	28
4.1 RNN accuracy output	32
4.2 RNN epoch	33
4.2.1 dataset division	34
4.2.2 dataset subdivision	34
4.2.3 dataset path in system	35
4.2.4 dataset path in code	35
4.3.1 training of LSTM	37
4.4.1 accuracy using LSTM	38
4.4.2 accident occurring screen grab	39
5 Highway pileup	42

<b>List Of Graphs</b>	<b>Page No.</b>
1.1.1 Death by accidents.	1
1.1.2 World deaths from road accidents	2
1.1.4 Road accidents in India	4

## **ABSTRACT**

After my detailed analysis in the Project of ‘Accident Detection’, I needed a different approach which would discover ways better at handling and detecting the occurrence of accidents and improve the overall accuracy of the project, thereby producing better results.

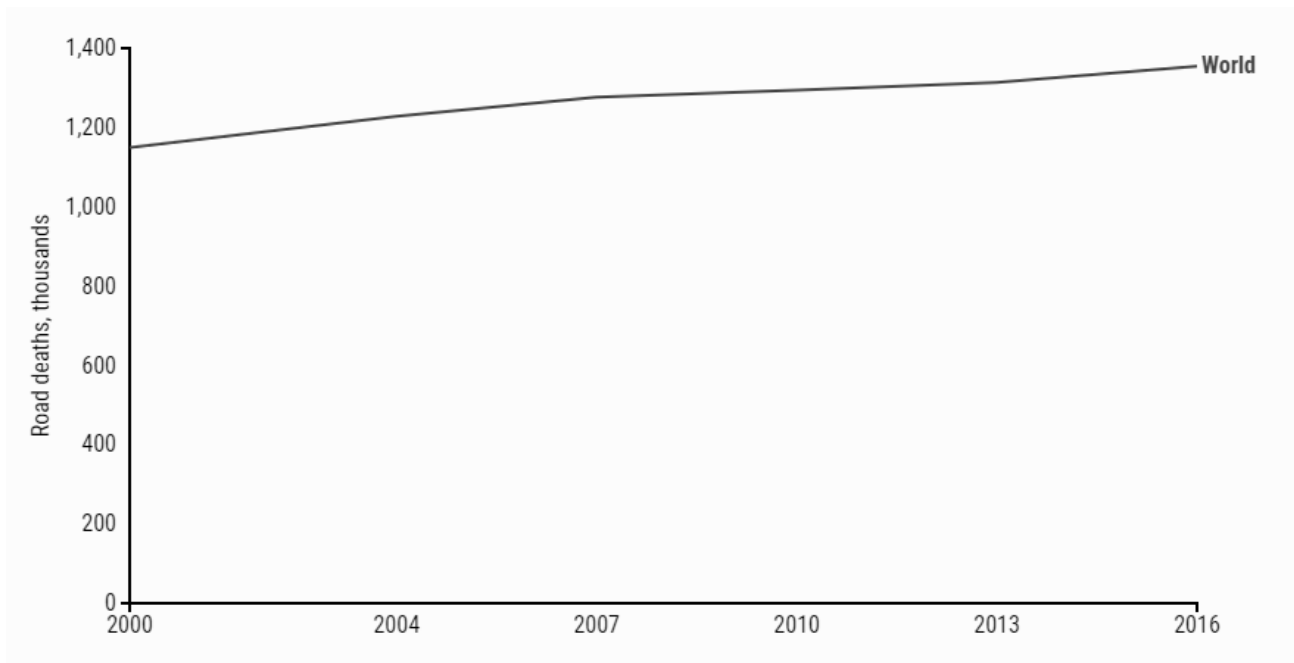
This project extends the research and studies carried out in my previous project of Accident Detection using algorithms and techniques different from those which I applied earlier for detecting the occurrence of accidents. This analysis will help us in understanding the differences in the results the overall model generates while using newer algorithms and help us decide why and which algorithm proves to be efficient for this model in total.

## Chapter-1

### INTRODUCTION

#### 1.1 Introduction

According to WHO, the cost arising from the road crashes in any country approximately costs it around 3% of its Gross Domestic Product (GDP). No wonder why all of the world nations are at the moment infested with finding out the concrete methodologies for effectively reducing the accidental damages costing people their life and destruction of properties both public as well as private. Here's a graph showing the casualties in road accidents that have taken place globally till 2016.

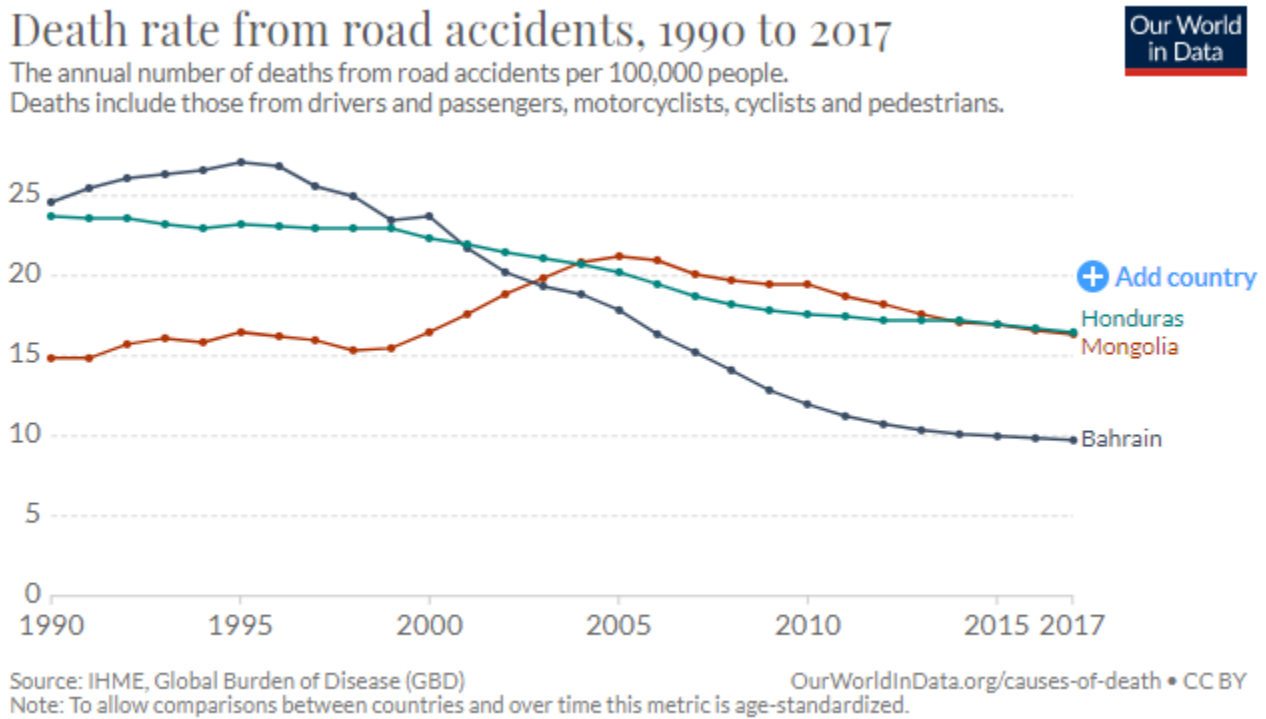


**Fig.1.1.1** the global cases of death by road accidents have only gone up in the last few years.

Humans, the most advanced living creatures have managed to find cures of even the deadliest of diseases but have not been able to check the damage caused by road accidents.

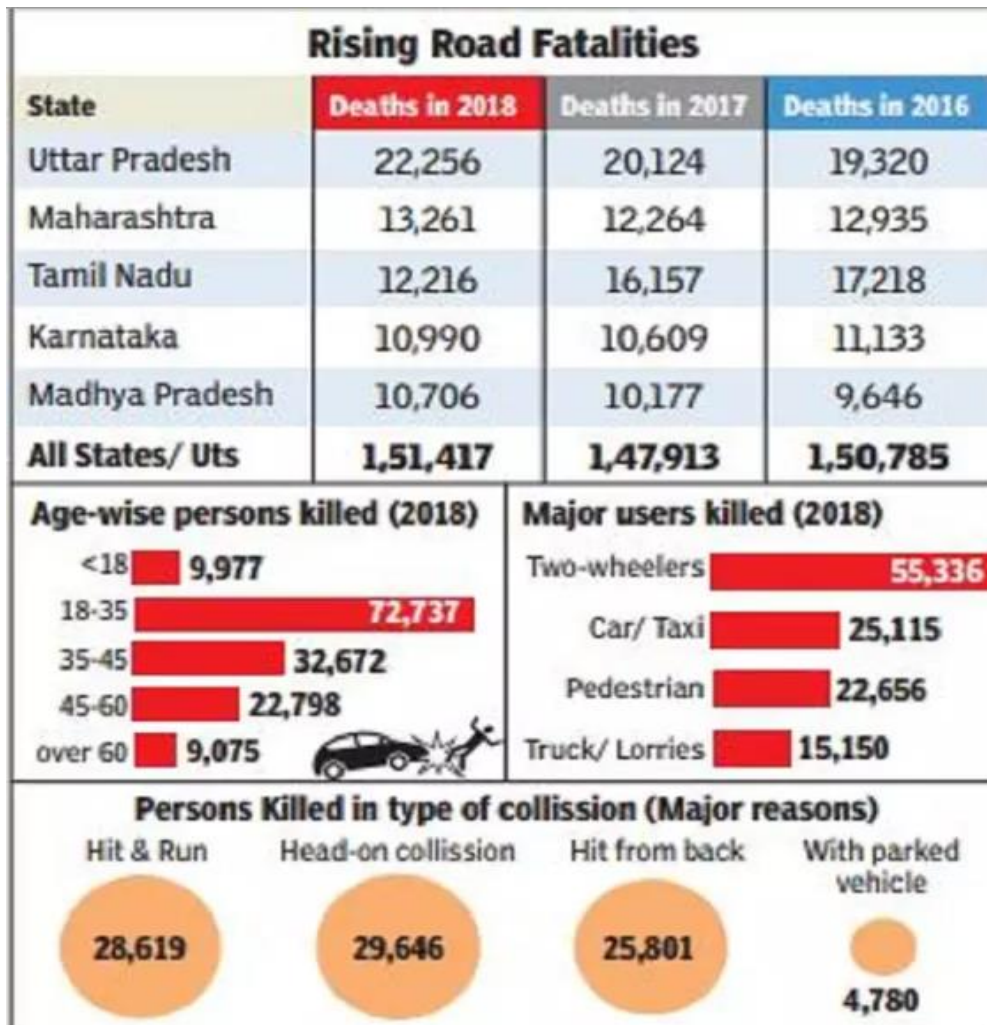


The following graph in the figure demonstrates the global statistics depicting death rate per 100,000 persons as a result of road accidents taking place from 1990 to 2017.



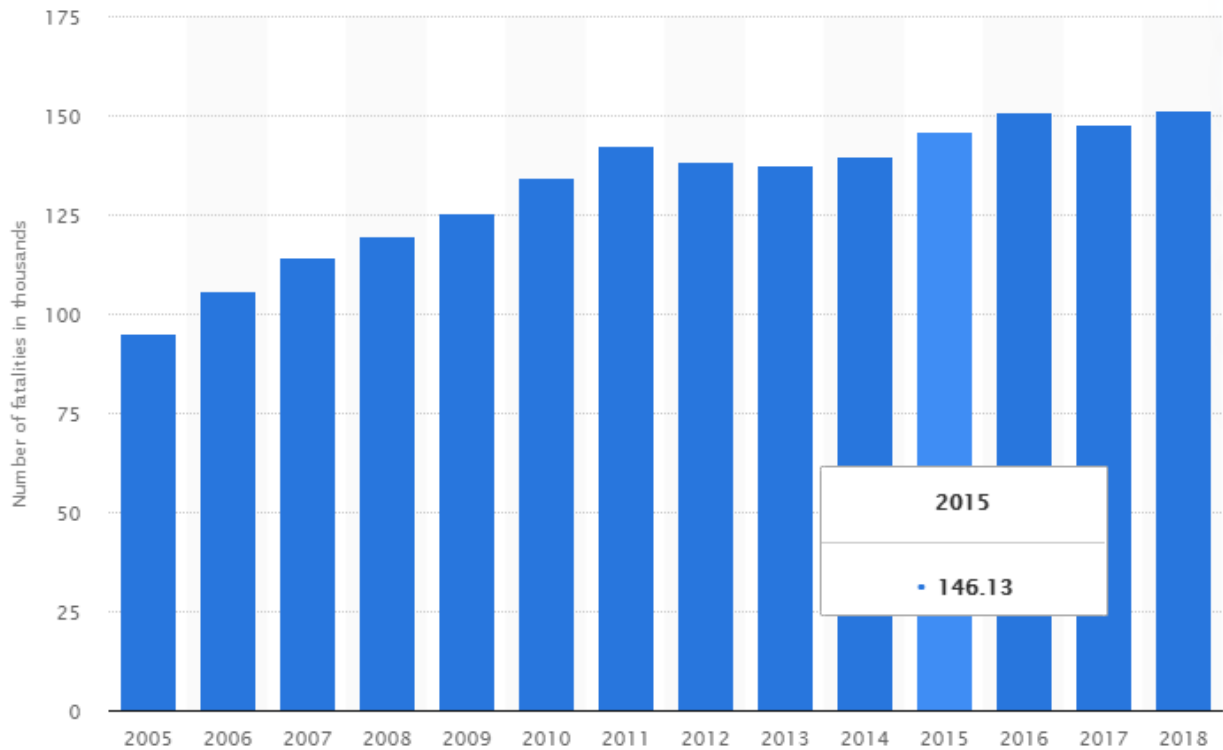
**Fig 1.1.2** World statistics showing death from road accidents over a course of 13 years.

According to the Ministry of Road Transport, the rate of deaths caused in India in 2018 alone from road accidents has seen a drastic surge compared to the previous year, i.e. a sharp increase of 2.4% in deaths from road rage with the state of Uttar Pradesh topping the list.



**Fig.1.1.3** Ministry of Road Transport statistics showing the death toll of road related cases.

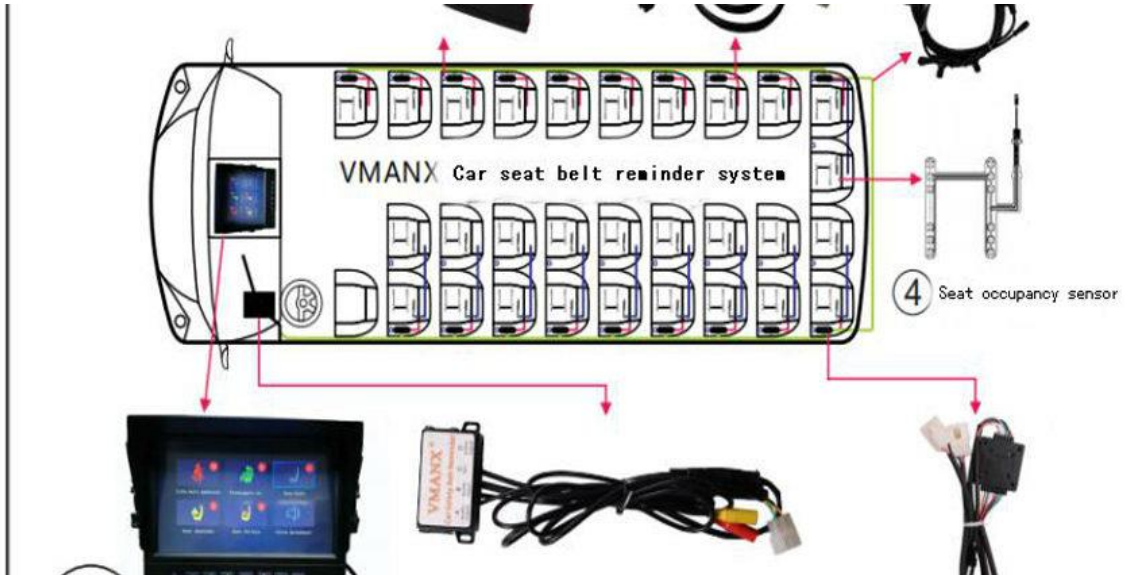
Inspite of strict laws,traffic rules and regulations in our nation,there has been no control on the number of registered accident cases over the last 13 years.With the increasing traffic,this number will only go up.This is shown in the figure below,



**Fig 1.1.4** Road accidents over the years in India

## 1.2 Problem Statement

In spite of the various methods devised by the government to ensure road safety, there has yet not been a straightforward approach that could curb the growing rates of road rage cases. Whenever an accident takes place, it takes a while for the report to reach the concerned authorities. In Europe General Safety Regulations were implemented in the year 2014 where a advanced system of emergency brakes were used in the vehicles in which the emergency brakes would themselves spring into action if any vehicle would come in the path of the moving car. Though this system was fruitful in reducing the road accidents by some margin, nonetheless it never put a full stop to this problem completely. Apart from this several other preventive measures such as the driver seat belt reminder systems had also been put to use.



**Fig 1.2.1** Seat belt Reminder system in European countries

In Australia, the automatic electronic call system sends an SOS message to the concerned emergency services in case the vehicle meets with a road accident. Such alarm based systems are installed into the vehicles to reduce the time in the arrival of the emergency services.



**Fig 1.2.2** SOS calls system in cars.

One of the major reasons why these solutions could not curb the road casualties was because these systems were entirely dependent on the individual vehicle in which they were deployed. There is massive need of a universal unified monitoring system that can function intact for all the passing vehicles, pedestrians and traffic overall and itself bring out a solution in case a mishap occurs.

Thus, we thought of proposing a method that would be universally applicable and useful that can reduce the time between the tragedy and the instant at which relief is provided for the same.



**Fig 1.2.3** Often the news about a road accident reaches late to the authorities

The delay between the instant the accident has occurred and the time at which the emergency services arrive on the spot has caused the loss of lives which could have otherwise been avoided. This project works on understanding models for detecting accidents and reporting them to the concerned authorities and emergency providing services.



**Fig 1.2.4** Prompt arrival of help can reduce loss of life.

### 1.3 Objectives

The objective of our project is to make sure that the occurrence of an accident is duly reported from the video footage available. In our previous analysis, we had used RNN algorithm for the model being implemented and were somewhat successful in achieving our task. This time we decided to use a different algorithm to report the accident taking place and improve our results. The outcome will finally help in minimizing the time it takes to report a road tragedy to the authorities' responsible and timely arrival of aid.

### 1.4 Methodology

The internal working of the project involves a solid methodology and its understanding. This can further be simplified into four major subparts,

#### 1.4.1 Dataset

The dataset consists of video footage of traffic captured from different CCTV cameras from various parts of the world. Our search for a video dataset which comprised of all kinds of traffic footage useful for our model was fulfilled by a website VSlabs.

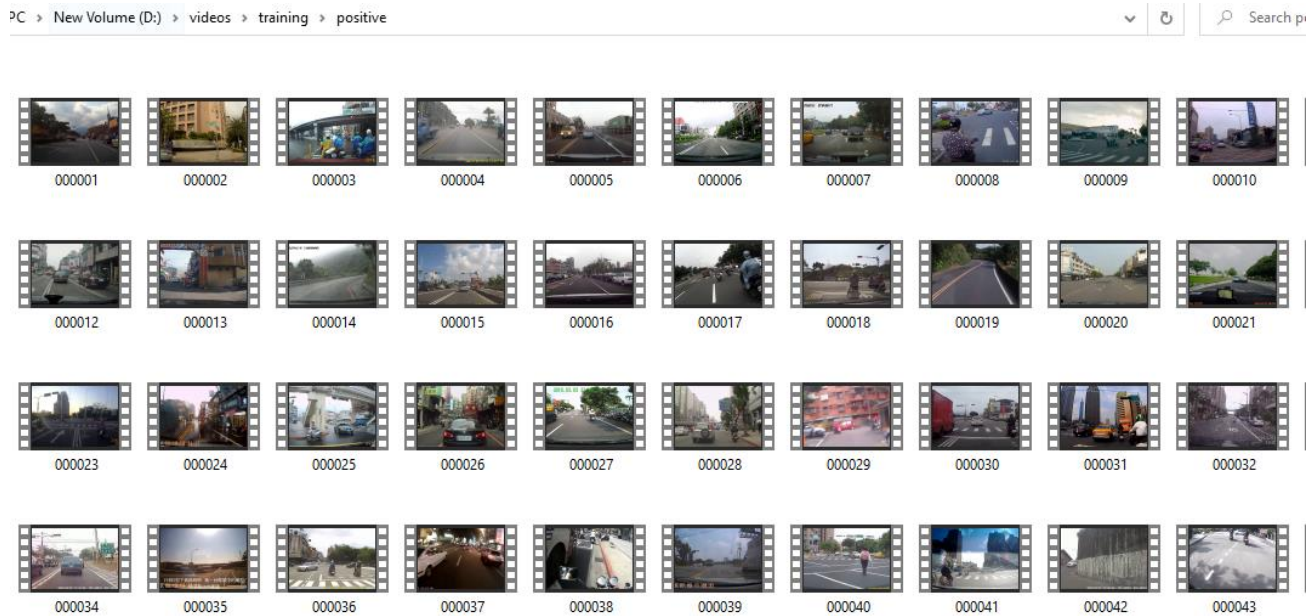
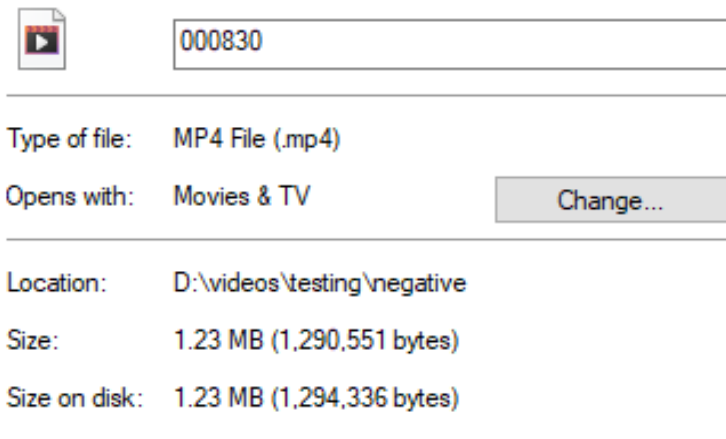
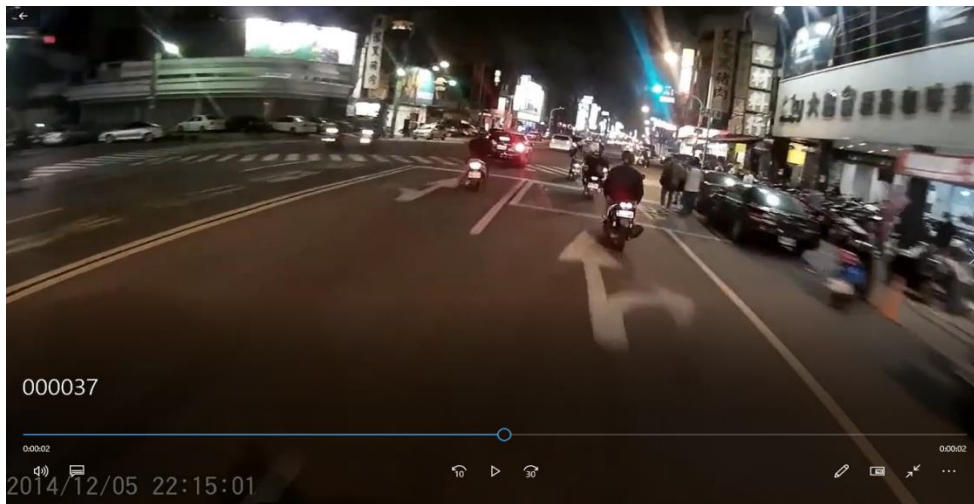


Fig 1.4.1.1 Video dataset library

Here, we were able to collect videos almost of the same size. We further created a smaller dataset which contained clips of traffic with a few similarities which would create data uniformity and easy analysis.



**Fig 1.4.1.2** Duration and size of video clips used was same



**Fig 1.4.1.3** Uniform dataset was created



## 1.4.2 Preprocessing

As done previously, the preprocessing here remained the same. A video is made up of several images and each image is made up of frames. We extracted each image and broke it down to a total of 99 frames. Now we simply downscaled the frame to five and stored them into a list of 3 dimensions. We repeat this for all 99 frames of the image and obtain an array of pixels where each frame is itself an image of 255\*144 sizes.

```
[ ] import keras
    from keras.models import Model
    from keras.layers import Input, Dense, TimeDistributed
    from keras.layers import LSTM
    import random
    from keras.layers import Conv2D, Flatten
    ### set hyper-parameters
    batch_size = 10
    num_classes = 2
    epochs = 10

    ### number of hidden layers in each NN
    row_hidden = 20
    col_hidden = 20

    ### print basic info
    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_test.shape[0], 'test samples')

    frame, row, col = (99, 144, 256)
    x = Input(shape=(frame, row, col))
    conv2d=(Conv2D(32, (3,3), strides=(1, 1), activation="relu"))(x)
    encoded_rows = TimeDistributed(LSTM(row_hidden))(conv2d)
    encoded_columns = LSTM(col_hidden)(encoded_rows)

    ### set up prediction and compile the model
    prediction = Dense(num_classes, activation='softmax')(encoded_columns)
    model = Model(x, prediction)
```

**Fig 1.4.2** Array of pixels for each frame in an image

### **1.4.3 Training and Testing**

Retaining the original protocols we had two datasets, the training and the testing. Earlier we had searched for algorithms that would maintain time dependency, an extremely crucial aspect of our model. Thus, we came across RNN and had analyzed our project based on that algorithm. This time we worked with a better and improved algorithm of LSTM and gated RNN to produce solid results and increase our model's efficiency.

## Chapter-2

### LITERATURE REVIEW

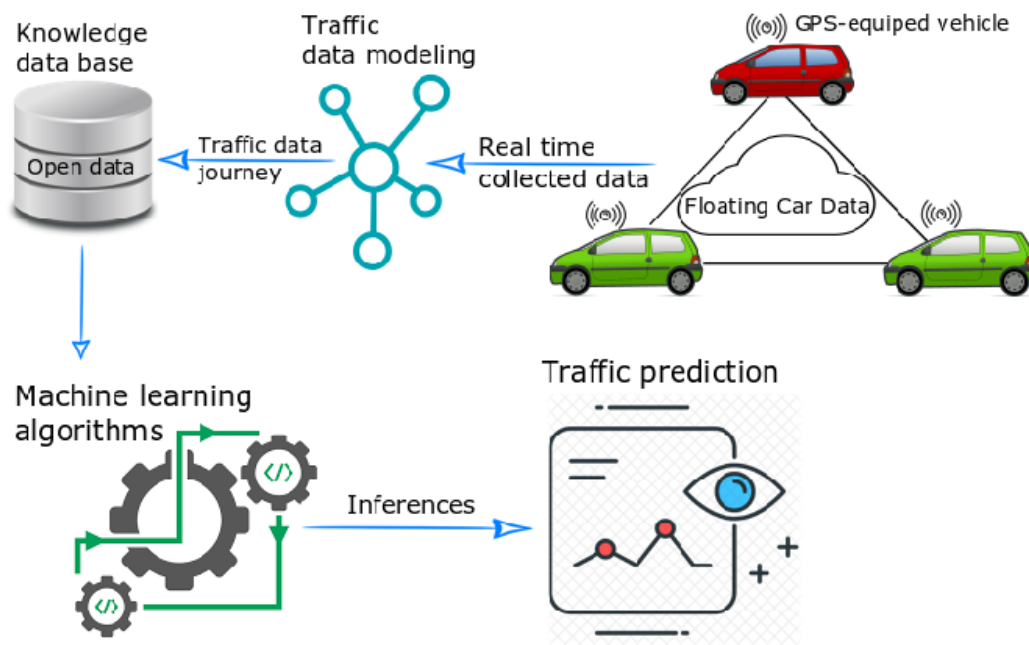
#### **1. Deep bidirectional unidirectional LSTM Recurrent Neural Network for Network Wide traffic speed Prediction**

This paper by Zhiyong Cui, Ruimin Ke, Ziyuan Pu ,Yinhai Wang proposes both unidirectional and bidirectional LSTM in which there can be both forward and backward dependencies which can be used to predict traffic speeds. These few years have seen a major scope of deep learning methods such as LSTM in traffic forecasting project.

In spite of the recent developments there has yet not fully been an exploration of the potential of these methods of deep learning in terms of their model and their scale area of prediction. A bidirectional LSTM layer is exploited to capture bidirectional temporal dependencies from historical data available.

In this research the authors have applied a bidirectional Long Short Term Memory algorithm that can even compensate for the missing input values by a masking mechanism.

This entire model is repeated with unidirectional and bidirectional LSTM thereby cementing the fact that it yields a superior prediction performance compared to its previous deep learning algorithms.



**Fig 2.1** Machine learning algorithms can be used in prediction based models

## 2. The vanishing gradient problem during learning recurrent neural nets and problem solutions-Sepp Hochreiter

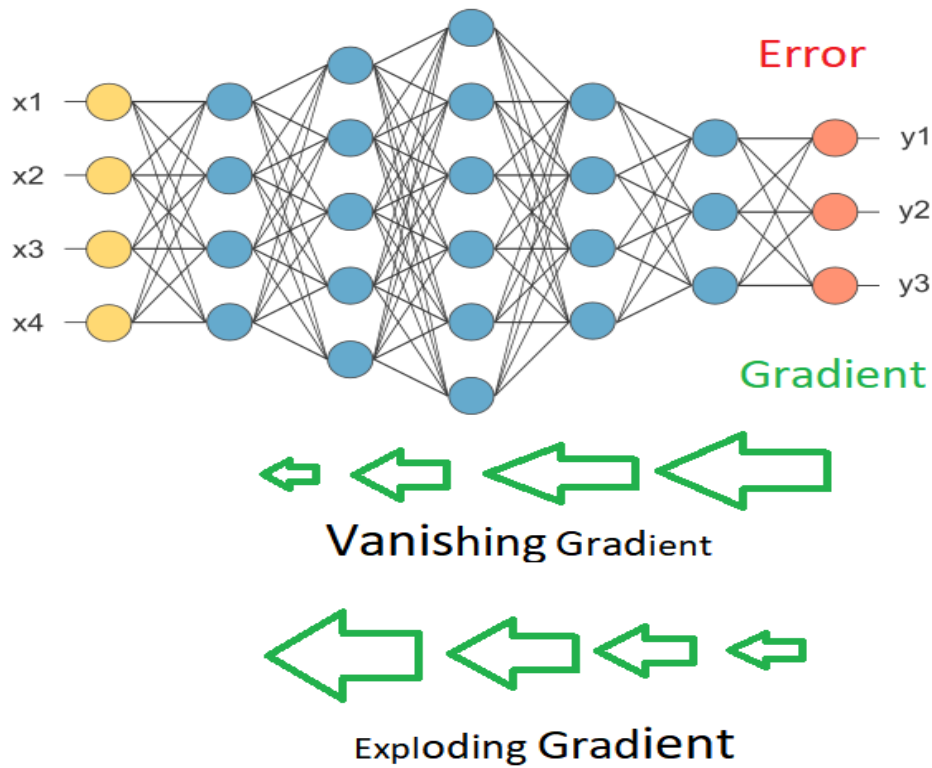
RNNs are capable of storing previous inputs to predict the outputs or produce the current outputs. This is the major reason why recurrent networks are used in time series prediction and process control. Real world applications involve several time steps or temporal dependencies.

The error vanishes as it gets propagated back which leads to an increased learning time of the model. The ability to extract temporal dependencies is a key feature of recurrent neural networks.

The same reason why recurrent networks are used in applications that involve the temporal delay of relevant signals. Some of these applications are speech or language processing, analysis of time series, process control and much more.

In some cases of standard recurrent neural networks, the long term dependencies are hard to learn

due to insufficient weight changes.



**Fig 2.2** Vanishing gradient defect has to be taken care of in the model.

### 3. Gate Variants of Gated Recurrent Unit (GRU) Neural Networks-Michigan State University

This paper by Rahul Dey and Fathi M.Salem, the focus is on three new type of gate variants with parameterization of the gates much reduced. The gating mechanism involved in gated recurrent unit is simply a replica of the normal or standard recurrent neural networks. Each of the gates have a certain weight associated with them. Thus, information regarding the present current input and the hidden previous state are reflected in the latest variable.

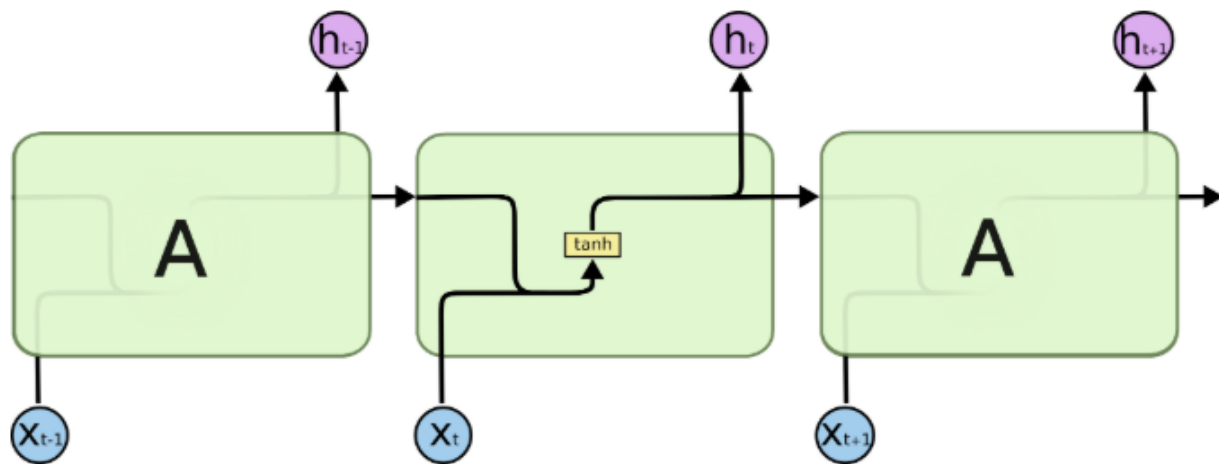
## Chapter-3

### System Development

#### 3.1 Long Short Term Memory (LSTM):

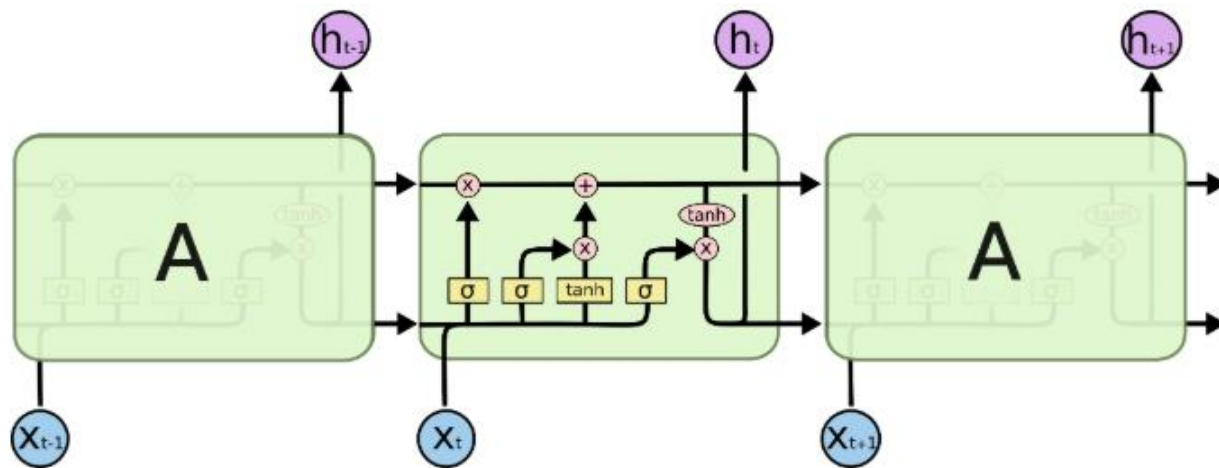
We often come across situation wherein our neural network need switching between an older memory or/and a newer or latest ones. These patterns of older and newer memories are maintained simultaneously by two memories, long-term and short-term memory. Since we had earlier faced the problem of long term dependency, LSTM are explicitly designed to overcome them. Long short term memories are simply a special version of RNN capable of working upon long term dependencies.

The chains of which standard RNNs are built are a single layer of **tanh**. Contrary to RNNs, the LSTMs have a long connected multiple layers where they can gather information with the help of gates. Gates in LSTM provide the option of letting the relevant information inside as per the model being used.



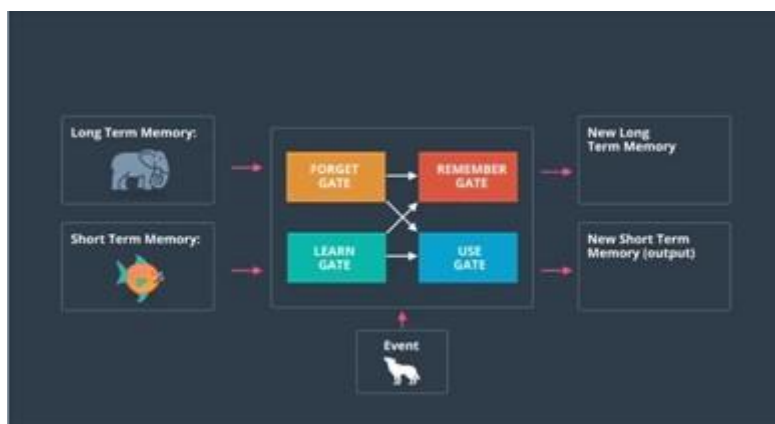
**Fig 3.1** A standard RNN contains a single layer

The task of deciding, whether a given piece of information is to be stored in a cell or not is completely decided by the sigmoid layer, details about which are covered under the mathematics section.



**Fig 3.2** The vanishing gradient problem of RNN is overcome here by LSTM.

### 3.1.1 Structure of LSTM:



**Fig 3.1.1** The gates used in LSTM.

As per **Fig 3.1.1** there are 4 gates that help retaining all the memories in neural network:

1. Forget Gate
2. Learn Gate
3. Remember Gate
4. Use Gate

The **Forget Gate** gets the long term memory and forgets irrelevant part of that memory.

The **Learn Gate** takes up the short term memory and the event from which it collects the information acquired and discards useless content keeping only the relevant information.

Now we have a recent memory and a long term memory which has been retained. The **Remember Gate** combines both of these long term memory and the recent memory, to in turn produce a new and updated long term memory. After that, the decision of using which of the memory, recent or the older one is of the **use gate**, hence the final output.

### **3.1.2 Advantage of LSTM over RNNs**

The vanishing gradient descent problem is solved by LSTM:

### **3.1.3 Vanishing Gradient Descent:**

When we increase the layers using the activation function in RNN, the gradient of the loss function starts diminishing (or we can say vanishing) and tends to 0, creating difficulty in training the model as the input and output are interdependent.



## Mathematics involved in LSTM:

### 3.1.3.1 Learn gate:

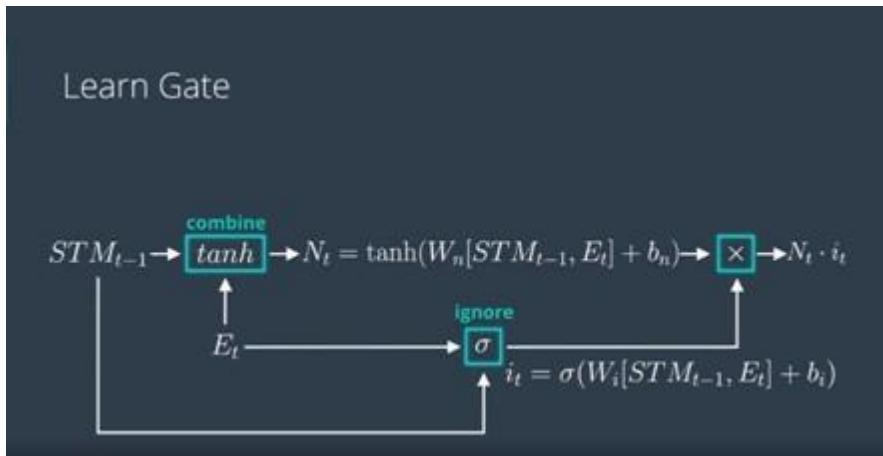


Fig 3.1.3.1

$STM_{t-1}$  = short term memory;  $E_t$  = event. These are connected using a linear function joining the vectors, multiplied by a weight matrix, as well as addition of a bias. At the end the result is passed through the tanh function.

Equation eq1 represents it as;

$$N_t = \tanh(W_n[STM_{t-1}, E_t] + b_n)$$

$b_n$  = bias.

We multiply it with an ignore factor  $i_t$  to ignore part of it.

$i_t$  can be calculated as:

A small neural network with inputs  $STM_{t-1}$  and  $E_t$  is created. This small neural network is similarly created in a linear function as above, multiplied with a weight, added with a bias and passed through the sigmoid function.

Eq2:

$$i_t = (\mathbf{W}_i[\mathbf{STM}_{t-1}, \mathbf{E}_t] + \mathbf{b}_i)$$

$\sigma$ : sigmoid activation function().

$\mathbf{b}_i$  is bias.

$\mathbf{N}_t$  and  $\mathbf{i}_t$  are multiplied to get the final result.

### 3.1.3.2 Forget Gate

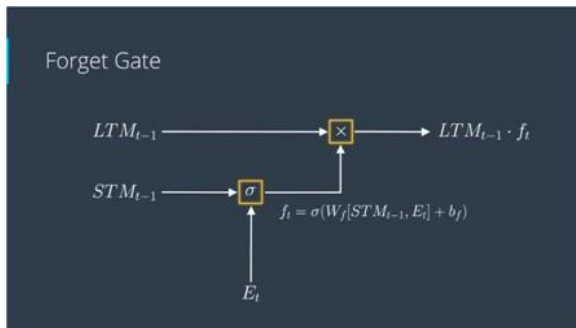


Fig 3.1.3.2

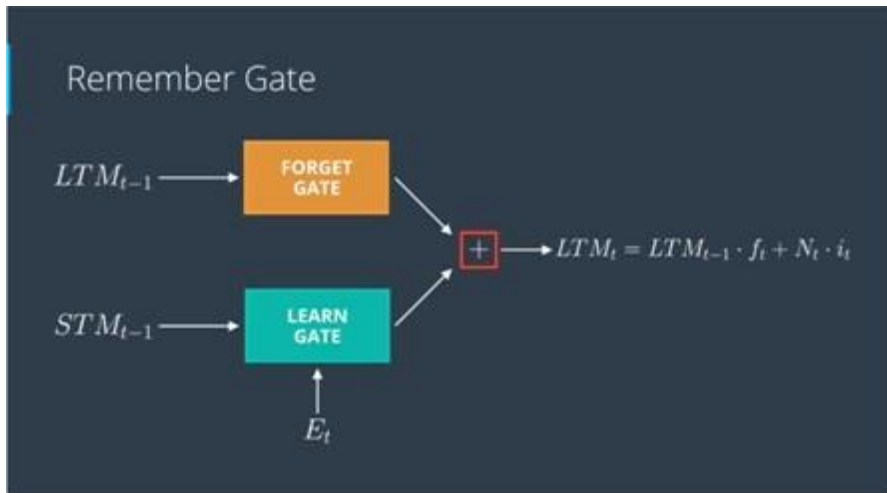
Short term memory  $STM_{t-1}$  and an event  $E_t$  are combined by a linear function, multiplied by weight, added with a bias and passed from sigmoid function.

This can be represented as equation 3.

$$f_t = (W_f[STM_{t-1}, E_t] + b_f)$$

Then,  $f_t$  and  $LTM_{t-1}$  (Long term memory) are multiplied.

### 3.1.3.3 Remember Gate



**Fig 3.1.3.3**

Coming to the Remember Gate, we just combine the outputs coming from learn gate and forget gate.

Output coming from forget gate:

$$op1 = LTM_{t-1} \cdot f_t$$

Where

$$f_t: (W_f[STM_{t-1}, E_t] + b_f)$$

Output coming from

Learn gate:  $op2 = N_t \cdot i$

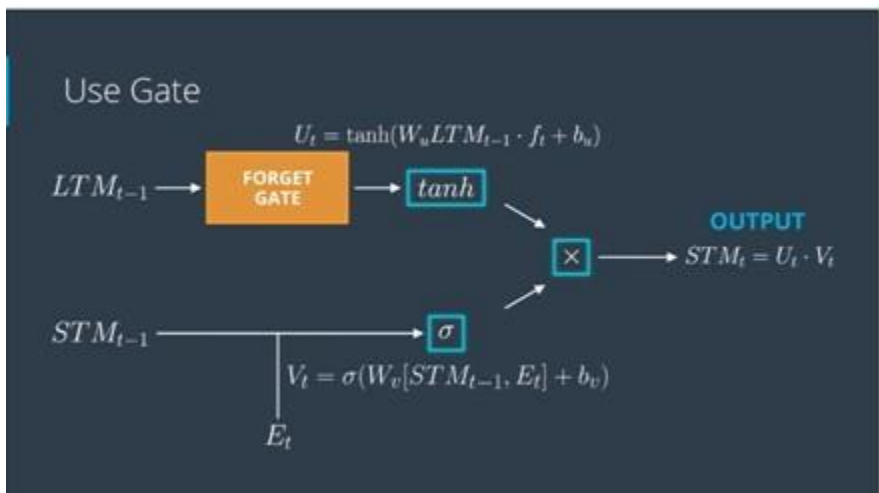
Where,

$N_t: \tanh(W_n[STM_{t-1}, E_t] + b_n)$

$i_t: (W_i[STM_{t-1}, E_t] + b_i)$

Final output is  $op1 + op2$ .

### 3.1.3.4 Use Gate



**Fig 3.1.3.4**

The output from forget gate is combined with a small neural network having a weight and added a bias, and then squished through the function **tanh**.

This can be represented by eq4.

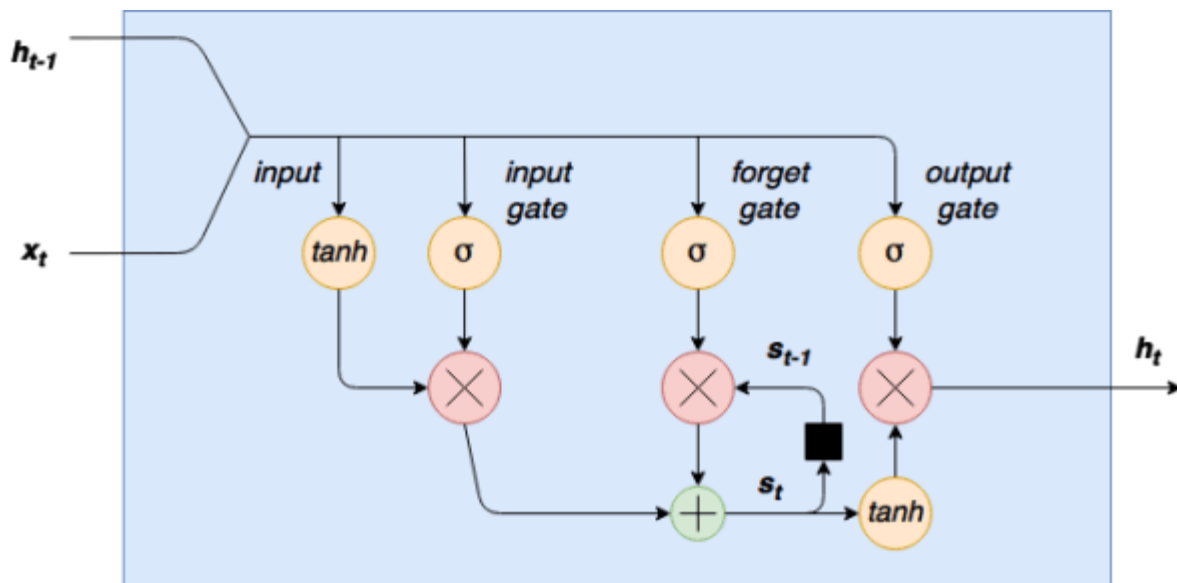
$$U_t = \tanh(W_u LTM_{t-1} \cdot f_t + b_u)$$

Again we use a short term memory  $STM_{t-1}$  and event  $E_t$  and combine it with a small neural network with  $W_v$  weight and bias  $b_v$ .

Eq5 represents this:

$$\mathbf{V}_t = (\mathbf{W}_v [\mathbf{STM}_{t-1}, \mathbf{E}_t + \mathbf{b}_v])$$

$\mathbf{U}_t \cdot \mathbf{V}_t$  is the output of the Use Gate.



**Fig 3.2** Steps involved in the long short term Memory algorithm.

### 3.2 Gated Recurrent Unit (GRU):

In order to get rid of the vanishing-exploding gradient, the major drawback encountered in recurrent neural networks, several variations were developed. One of the lesser discussed and not much well known method is the gated recurrent unit or GRUs which is equally effective as the long short term memory algorithm discussed earlier.

Both the Gated Recurrent Units as well as the standard Recurrent Neural Network Units are similar in their functioning as except for the fact that gated recurrent units use specially functioned gates for each recurrent unit that deal with the current input and keep a track record of the previous hidden states encountered in the model.

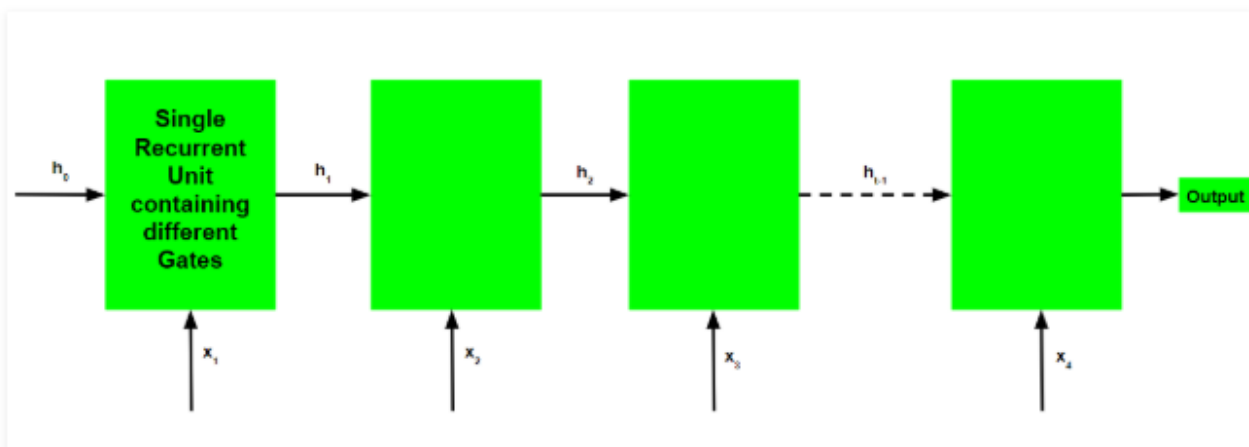


Fig 3.2

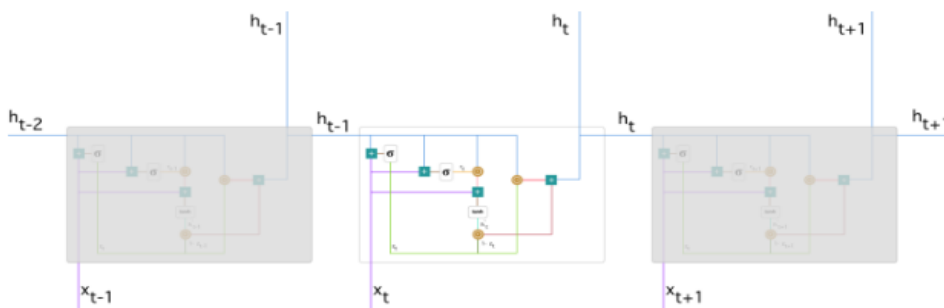
As we know, the improved version of the standardized RNN is called the GRU. Here we will study what makes them so valuable and effective in their working. Just like LSTM, the GRUs also solves the problem of the vanishing gradient, but in a different manner. Here, GRU uses two types of gates:

1. Update gate
2. Reset gate

These two are actually vectors and the information that is passed to the output is decided by them. Unlike, the LSTM these gates can be used to keep track of information from very long age, even without losing it with the time or removing any part of it which is unnecessary for the prediction.

### 3.2.1 Mathematics behind GRUs:

In order to understand the process taking place we will analyze a single unit from the given RNN:



**Fig 3.2.1** RNN with Gated Recurrent Unit

#### 3.2.1.1 Update gate

Firstly, we calculate the update  $\mathbf{z}_t$  for time step  $t$  with the formula:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

The expression  $\mathbf{x}_t$  is multiplied by its own weight  $\mathbf{W}(\mathbf{z})$  as soon as it is plugged into the network unit. Similarly,  $\mathbf{h}(t-1)$  is also multiplied by its own weight  $\mathbf{U}(\mathbf{z})$  and it also contains information of  $t-1$  units. Adding both the results together, we squash the output from sigmoid activation function to make result value between 0 and 1.

Thus, the update gate helps the given model to understand the size of the past memory from the earlier time steps that needs to be passed to the future. Thus, eliminating the chances of vanishing

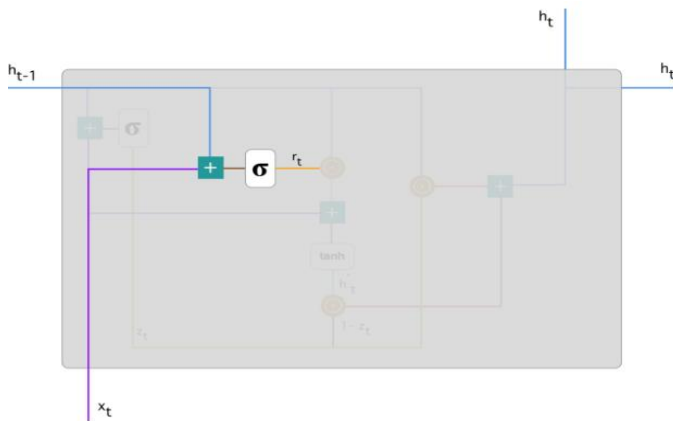
gradient.

### 3.2.1.2 Reset gate

The reset gate is useful in deciding the amount of past information which has to be forgotten. We use the following to calculate it:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

The only difference this formula has from update gate is that of weights and gate's usage.



**Fig 3.2.1.2** Reset Gate

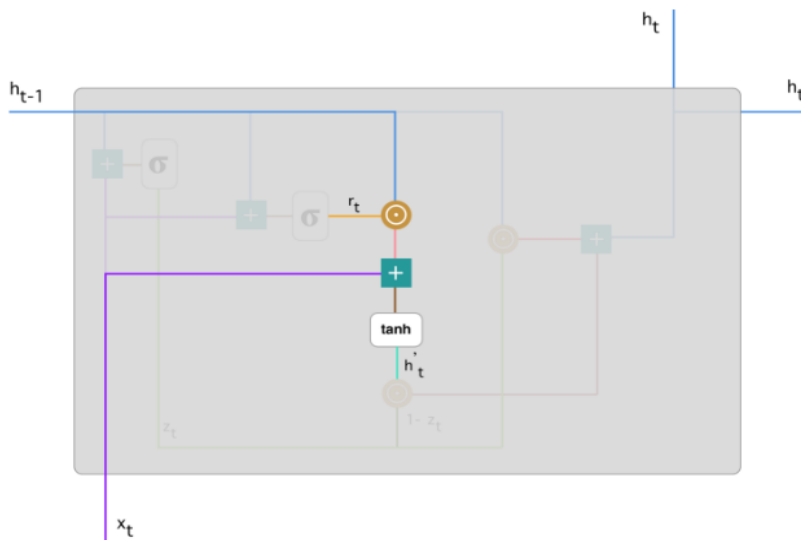
### 3.2.1.3 Current memory content

We will now understand how the gates will affect the final outcome. Starting with the reset gate, a new memory content will be introduced which will use this gate to store information from the past which is actually relevant. It can be calculated:



$$h'_t = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

Input  $x_t$  is multiplied with weight  $W$  and  $h_{(t-1)}$  with weight  $U$ . After that the reset gate  $r_t$  is multiplied by  $Uh_{(t-1)}$  which gives the Hadamard product which will help determining what to exclude from the older time steps. Now we apply  $\tanh$  function on the sum of the previous both steps.



We do an element-wise multiplication of  $h_{(t-1)}$  — blue line and  $r_t$  — orange line and then sum the result — pink line with the input  $x_t$  — purple line. Finally,  $\tanh$  is used to produce  $h'_t$  — bright green line.

**Fig 3.2.1.3** Current memory content

### 3.2.1.4 Final memory at current time step

In the final step of the calculation our network has to find  $h_t$  vector that holds the current unit information and does the task of sending it down. This requires the update gate which manages the collection from current memory as well as previous steps

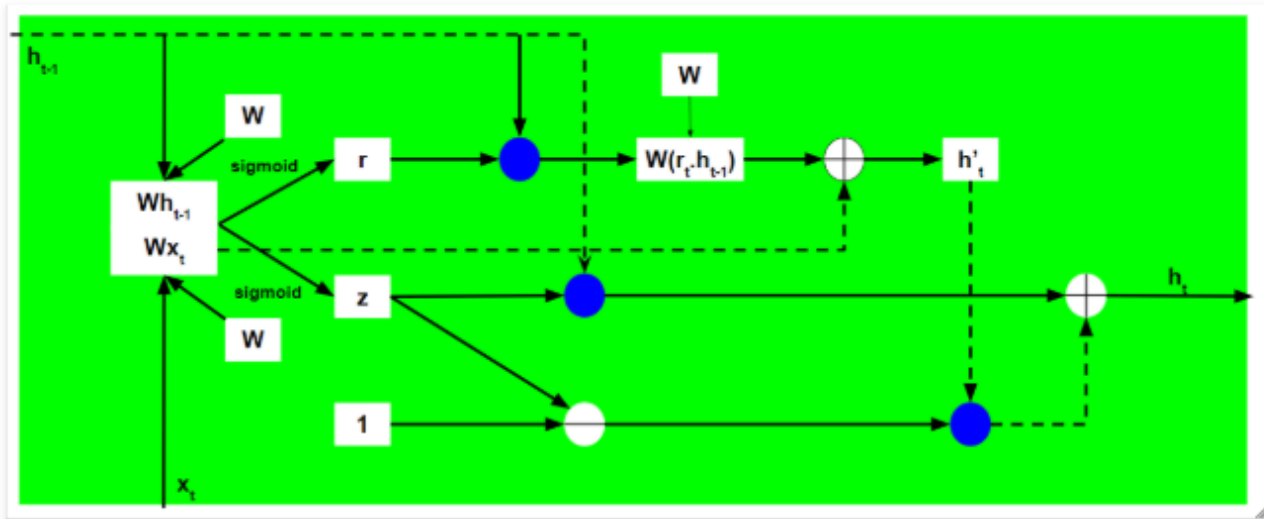
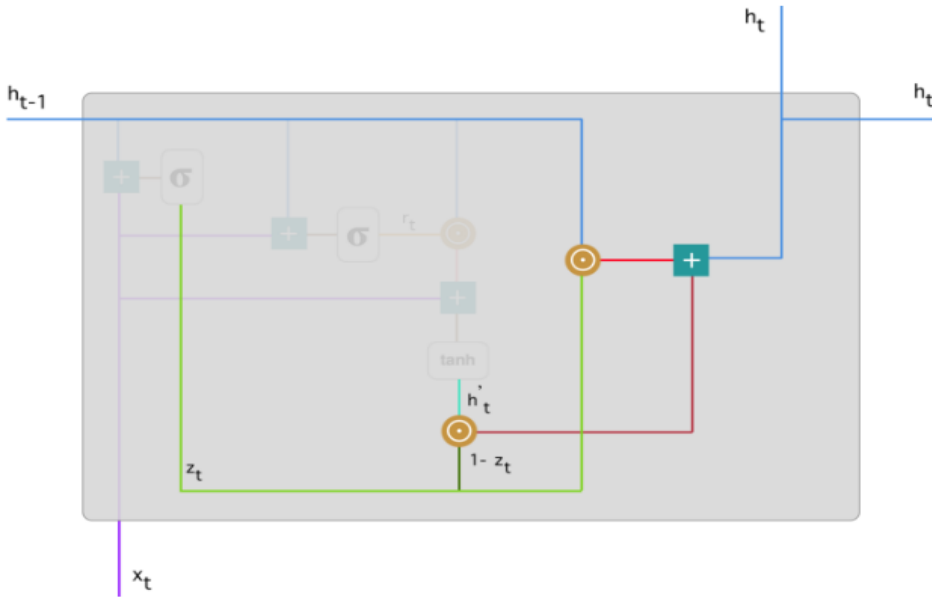


Fig 3.2.1.4.1

Now, the update gate  $z_t$  and  $h_{t-1}$  are multiplied element wise. Similarly apply it to  $(1 - z_t)$  and  $h'_t$  and add both the steps performed.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$

Hence the equation of GRU is obtained,



Following through, you can see how  $z_t$  — green line is used to calculate  $1-z_t$  which, combined with  $h'_t$  — bright green line, produces a result in the dark red line.  $z_t$  is also used with  $h_{(t-1)}$  — blue line in an element-wise multiplication. Finally,  $h_t$  — blue line is a result of the summation of the outputs corresponding to the bright and dark red lines.

**Fig 3.2.1.4.2** Final memory content

Hence, we have now understood how Gated Recurrent units are efficient in storing and filtering information using their gates which are update and reset. This in turn puts an end to the problem of vanishing gradient as this particular model is not discarding the new input each time but instead stores the important information and makes sure it reaches further to the future time steps of the neural network being used.

The back propagation through time here in Gated Recurrent Unit is similar to that of Long Short Term Memory and the distinction lies in its differential chain formation.

Let the predicted output at each time step be  $\bar{y}_t$ , and be the actual output at each time step be  $y_t$ . Now the error at each time step be determined as,

$$E_t = -y_t \log(\bar{y}_t)$$

Summation of all time steps will give us the total error,

$$E = \sum_t E_t$$

This can further be simplified as,

$$\Rightarrow E = \sum_t -y_t \log(\bar{y}_t)$$

Summation of gradients at every time step can be defined by  $\frac{\partial E}{\partial W}$

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

Using the chain rule along with the fact that  $\bar{y}_t$  is a function of  $h_t$ , which we already are aware of  $\bar{h}_t$ , we derive the following expression,

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \bar{y}_t} \frac{\partial \bar{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots \frac{\partial h_0}{\partial W}$$

Thus, summation will give us the total error

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial \bar{y}_t} \frac{\partial \bar{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots \frac{\partial h_0}{\partial W}$$

### **How do GRUs take care of the problem of Vanishing Gradient?,**

A chain of derivatives exist that controls the value of the gradients. This chain of derivatives begins from

$$\frac{\partial h_t}{\partial h_{t-1}}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \bar{h}_t$$

Above is the expression for  $h_t$ .

Using this expression, the value for  $\frac{\partial h_t}{\partial h_{t-1}}$  can be calculated as,

$$\frac{\partial h_t}{\partial h_{t-1}} = z + (1 - z) \frac{\partial \bar{h}_t}{\partial h_{t-1}}$$

Using the expression found for,

We can calculate the value of  $\frac{\partial \bar{h}_t}{\partial h_{t-1}}$

$$\begin{aligned} \frac{\partial \bar{h}_t}{\partial h_{t-1}} &= \frac{\partial(\tanh(W \odot x_t + W \odot (r_t \odot h_{t-1})))}{\partial h_{t-1}} \\ \Rightarrow \frac{\partial \bar{h}_t}{\partial h_{t-1}} &= (1 - \bar{h}_t^2)(W \odot r) \end{aligned}$$

Both the gates i.e. the update and reset gate use the sigmoid function as their activation function therefore they can possess the values 0 or 1.

We shall look at all the cases corresponding to the output of sigmoid function,

### Case 1, when z=1:

Here, it doesn't matter what the value of r is, the term  $\frac{\partial \bar{h}_t}{\partial h_{t-1}}$  is equal to z which in turn is equal to 1.

### Case 2A (z=0 and r=0):

For this particular case, the term  $\frac{\partial \bar{h}_t}{\partial h_{t-1}}$  is equal to 0.

**Case 2B (z=0 and r=1):**

The term  $\frac{\partial \bar{h}_t}{\partial h_{t-1}}$  value equals  $(1 - \bar{h}_t^2)(W)$  which is controlled by the weight matrix. The weight matrix is trainable and our network learns to adjust the weight matrix in a way that the term starts becoming closer to 1 in value.

From our study about Gated Recurrent Units we have learnt that the back-propagation through time algorithm adjusts their respective weights in such a way that the value of the chain of the derivatives ultimately comes as close to unity as it can.

## Chapter-4

### Performance Analysis

#### 1. Recurrent Neural Network (RNNs)

Using RNN, the accuracy we received as output was 33.33%. This was when RNN was implemented in Keras library.

```
Epoch 00006: val_acc did not improve from 0.66667
Epoch 7/10
8/8 [=====] - 6s 712ms/step - loss: 0.0052 - acc: 1.0000 - val_loss: 3.6165 - val_acc: 0.3333

Epoch 00007: val_acc did not improve from 0.66667
Epoch 8/10
8/8 [=====] - 6s 688ms/step - loss: 0.0045 - acc: 1.0000 - val_loss: 3.7097 - val_acc: 0.3333

Epoch 00008: val_acc did not improve from 0.66667
Epoch 9/10
8/8 [=====] - 5s 653ms/step - loss: 0.0039 - acc: 1.0000 - val_loss: 3.7967 - val_acc: 0.3333

Epoch 00009: val_acc did not improve from 0.66667
Epoch 10/10
8/8 [=====] - 5s 649ms/step - loss: 0.0034 - acc: 1.0000 - val_loss: 3.8793 - val_acc: 0.3333

Epoch 00010: val_acc did not improve from 0.66667
6/6 [=====] - 1s 227ms/step
Test loss: 3.879347085952759
Test accuracy: 0.3333333432674408
```

**Fig 4.1**

Our primary target was to improve the accuracy of accident detection in our model. As stated above, the accuracy which we obtained through RNN was 33.33%. We employed several techniques such as changing the epoch layers but that hardly produced any major variation in the output in overall. As we were trying different approaches we suddenly discovered a major checkpoint. Accuracy of the model was affected by the contents of the training dataset. Earlier we were inputting the dataset in a mixed format i.e. a combination of videos where there was footage of accident in some of them and normal vehicular movement in the others.

This time we arranged our dataset in a precise format where there was an equal weight age of videos with normal traffic and collision of vehicles. Say, if we utilized twenty video clips, ten of them would have regular vehicular traffic and the other ten accidents.

```
Epoch 00002: val_acc did not improve from 0.66667
Epoch 3/6
8/8 [=====] - 4s 541ms/step - loss: 0.2885 - acc: 0.8750 - val_loss: 1.2867 - val_acc: 0.3333

Epoch 00003: val_acc did not improve from 0.66667
Epoch 4/6
8/8 [=====] - 6s 730ms/step - loss: 0.2364 - acc: 0.8750 - val_loss: 0.6670 - val_acc: 0.6667

Epoch 00004: val_acc did not improve from 0.66667
Epoch 5/6
8/8 [=====] - 4s 494ms/step - loss: 0.2420 - acc: 1.0000 - val_loss: 1.9859 - val_acc: 0.3333

Epoch 00005: val_acc did not improve from 0.66667
Epoch 6/6
8/8 [=====] - 5s 639ms/step - loss: 0.3704 - acc: 0.8750 - val_loss: 0.6278 - val_acc: 0.5000

Epoch 00006: val_acc did not improve from 0.66667
6/6 [=====] - 1s 109ms/step
Test loss: 0.6278041005134583
Test accuracy: 0.5
```

**Fig 4.2**

## Using LSTM

As we had already implemented our model and calculated for its accuracy using RNN, it was now time to implement the exact same concept using LSTM. The effect of the vanishing gradient issue was already evident in our previous results and this had to gotten rid of. In LSTM, our problem was finally solved as the overall output was significantly improved thereby getting rid of the shortcomings we had encountered during the use of the RNN model



## 4.2 Dataset

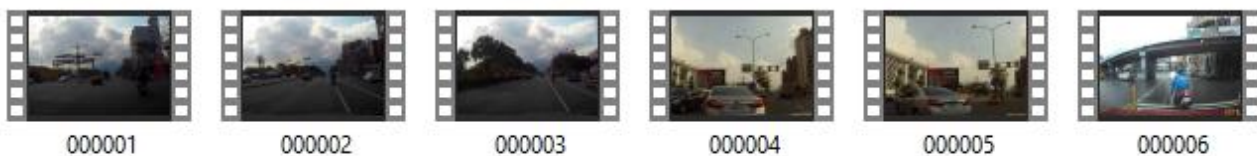
In order to create our dataset, we used the content library present on VSlabs. There were thousands of video clips and footages present which contained the occurrences of accidents and also had footage of normal traffic conditions. We divided our dataset into two subparts, the positive and negative. The positive folder contained the footage of accidents taking place while the negative folder had clips of normal moving traffic as shown in Fig 4.2.



**Fig 4.2.1**

Since it was not possible to train our model using such a large volume of videos as it would have been an extremely time taking process. In order to reduce the size of our dataset, we subdivided the training set of data into further two parts that is, positive 1 and negative 1. This is shown below in figure 4.2.2 and figure 4.2.3.

PC > New Volume (D:) > videos > training > negative1



**Fig 4.2.2** Subdivision of dataset.

```
0 C:/Users/Dell/Videos/training/positive1\000004.mp4
1 C:/Users/Dell/Videos/training/positive1\000008.mp4
2 C:/Users/Dell/Videos/training/positive1\000003.mp4
3 C:/Users/Dell/Videos/training/positive1\000002.mp4
```

**Fig 4.2.3**

```
img_filepath = 'C:/Users/Dell/Videos/training/'
neg_all = glob.glob(img_filepath + 'negative1/*.mp4')
pos_2 = glob.glob(img_filepath + 'positive1/*.mp4')
pos_1 = glob.glob(img_filepath + '../YTpickles/*.pkl')
```

**Fig 4.2.4** Assigning path to the datasets.

## 4.3 Training

Initially, we had trained our model with RNN and received 50 % accuracy.

This time, we used two different algorithms for our model and received accuracy as:

### 4.3.1 Using LSTM:

With the help of Keras library in Python, we implemented our algorithm of LSTM. We also had to avoid over fitting of our model as an over fitted model performs decently in training phase but produces unsatisfactory outcome in the testing phase at the end. At the training phase the outputs are as shown below,

```

Epoch 1/20
12/12 [=====] - 99s 8s/step - loss: 0.7059 - acc: 0.4167 - val_loss: 0.6750 - val_acc: 0.5833

Epoch 00001: val_acc improved from -inf to 0.58333, saving model to model1.hdf5
Epoch 2/20
12/12 [=====] - 23s 2s/step - loss: 0.6680 - acc: 0.5833 - val_loss: 0.6536 - val_acc: 0.5833

Epoch 00002: val_acc did not improve from 0.58333
Epoch 3/20
12/12 [=====] - 5s 415ms/step - loss: 0.6390 - acc: 0.5833 - val_loss: 0.6418 - val_acc: 0.5833

Epoch 00003: val_acc did not improve from 0.58333
Epoch 4/20
12/12 [=====] - 4s 355ms/step - loss: 0.6076 - acc: 0.6667 - val_loss: 0.6246 - val_acc: 0.5833

Epoch 00004: val_acc did not improve from 0.58333
Epoch 5/20
12/12 [=====] - 4s 361ms/step - loss: 0.5422 - acc: 0.6667 - val_loss: 0.7306 - val_acc: 0.5000

Epoch 00005: val_acc did not improve from 0.58333
Epoch 6/20
12/12 [=====] - 4s 334ms/step - loss: 0.6649 - acc: 0.5833 - val_loss: 0.6267 - val_acc: 0.5833

Epoch 00006: val_acc did not improve from 0.58333
Epoch 7/20
12/12 [=====] - 4s 334ms/step - loss: 0.5820 - acc: 0.6667 - val_loss: 0.6201 - val_acc: 0.5833

Epoch 00007: val_acc did not improve from 0.58333
Epoch 8/20
12/12 [=====] - 4s 325ms/step - loss: 0.5521 - acc: 0.8333 - val_loss: 0.6599 - val_acc: 0.5833

Epoch 00008: val_acc did not improve from 0.58333

Epoch 9/20
12/12 [=====] - 4s 343ms/step - loss: 0.6066 - acc: 0.8333 - val_loss: 0.6619 - val_acc: 0.5833

Epoch 00009: val_acc did not improve from 0.58333
Epoch 10/20
12/12 [=====] - 4s 329ms/step - loss: 0.5700 - acc: 0.7500 - val_loss: 0.7159 - val_acc: 0.5833

Epoch 00010: val_acc did not improve from 0.58333
Epoch 11/20
12/12 [=====] - 4s 337ms/step - loss: 0.5631 - acc: 0.7500 - val_loss: 0.6169 - val_acc: 0.6667

Epoch 00011: val_acc improved from 0.58333 to 0.66667, saving model to model1.hdf5
Epoch 12/20
12/12 [=====] - 4s 338ms/step - loss: 0.4632 - acc: 0.9167 - val_loss: 0.6197 - val_acc: 0.6667

Epoch 00012: val_acc did not improve from 0.66667
Epoch 13/20
12/12 [=====] - 4s 313ms/step - loss: 0.4147 - acc: 0.9167 - val_loss: 0.6908 - val_acc: 0.6667

Epoch 00013: val_acc did not improve from 0.66667
Epoch 14/20
12/12 [=====] - 4s 310ms/step - loss: 0.6274 - acc: 0.6667 - val_loss: 0.6363 - val_acc: 0.5833

Epoch 00014: val_acc did not improve from 0.66667
Epoch 15/20
12/12 [=====] - 4s 307ms/step - loss: 0.4128 - acc: 0.8333 - val_loss: 0.6072 - val_acc: 0.5833

Epoch 00015: val_acc did not improve from 0.66667
Epoch 16/20
12/12 [=====] - 4s 310ms/step - loss: 0.3663 - acc: 0.9167 - val_loss: 0.6813 - val_acc: 0.7500

Epoch 00016: val_acc improved from 0.66667 to 0.75000, saving model to model1.hdf5
Epoch 17/20
12/12 [=====] - 4s 306ms/step - loss: 0.3650 - acc: 0.9167 - val_loss: 0.6655 - val_acc: 0.6667

```

Activate V  
Go to Setting

```

Epoch 18/20
12/12 [=====] - 4s 325ms/step - loss: 0.6071 - acc: 0.6667 - val_loss: 0.6644 - val_acc: 0.5833

Epoch 00018: val_acc did not improve from 0.75000
Epoch 19/20
12/12 [=====] - 4s 374ms/step - loss: 0.5119 - acc: 0.6667 - val_loss: 0.6630 - val_acc: 0.6667

Epoch 00019: val_acc did not improve from 0.75000
Epoch 20/20
12/12 [=====] - 4s 316ms/step - loss: 0.3632 - acc: 0.9167 - val_loss: 0.9495 - val_acc: 0.5833

Epoch 00020: val_acc did not improve from 0.75000

<keras.callbacks.History at 0x153654a83c8>

```

**Fig 4.3.1** Training phase of LSTM

During the training phase we obtained an accuracy of 75 %

#### 4.4 Testing

After the training phase, it was time to for the testing phase of our model. In the testing phase, our model performed good and yielded results better than the previous. We calculated the accuracy of our model by,

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

Where,

$T_p$  =True positive

$T_n$ =True negative

$F_p$ = False positive

$F_n$ =False negative

The matrix containing these four values is called a confusion matrix.

Understanding the meaning of these attributes,

A true positive signifies that an accident had taken place and it was correctly identified by our model.

A true negative implies that an accident did not occur and our model also states that it did not take place.

A false positive means that an accident had not occurred and but our model said it took place.

A false negative means that an accident actually took place but our model did not identify it.

In our analysis using LSTM, we had a total of 14 videos identified as true positive, 7 as true negative, and 3 as false positive while none as false negative.

```
0 E:/videos/training/negative1\000009.mp4
0 E:/videos/training/negative1\000005.mp4
0 E:/videos/training/positive1\000003.mp4
0 E:/videos/training/negative1\000016.mp4
0 E:/videos/training/positive1\000019.mp4
0 E:/videos/training/positive1\000069.mp4
0 E:/videos/training/negative1\000018.mp4
0 E:/videos/training/negative1\000010.mp4
0 E:/videos/training/positive1\000016.mp4
0 E:/videos/training/positive1\000009.mp4
0 E:/videos/training/positive1\000018.mp4
0 E:/videos/training/positive1\000001.mp4
0 E:/videos/training/negative1\000002.mp4
0 E:/videos/training/negative1\000020.mp4
0 E:/videos/training/positive1\000008.mp4
0 E:/videos/training/positive1\000012.mp4
0 E:/videos/training/positive1\000017.mp4
0 E:/videos/training/positive1\000013.mp4
0 E:/videos/training/negative1\000019.mp4
0 E:/videos/training/negative1\000004.mp4
0 E:/videos/training/positive1\000010.mp4
0 E:/videos/training/positive1\000060.mp4
0 E:/videos/training/positive1\000015.mp4
0 E:/videos/training/negative1\000012.mp4
Train
Area under curve 0.9285714285714285
Accuracy 0.875
```

**Fig 4.4.1** Accuracy achieved using LSTM

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$= (14+7) / (14+7+3+0) = 0.875$$

This proves that we have successfully implemented our model using LSTM with better and improved performance in terms of detection. This has produced an accuracy of 87.5 which is higher than our previous RNN algorithm.



**Fig 4.4.2** A video used in dataset depicting an accident taking place

## Chapter 5

### Conclusion

#### Conclusion

The final outcome of our project is the detection of the occurrence of the accident in the given dataset. We used RNN algorithm as the first machine learning algorithm to begin implementing our idea in the project. RNN worked on a video dataset of several videos in which was a combination of both the cases of an accident occurring and not occurring. The algorithm produced the output by detecting the patterns from the given video based on memories from the previous inputs. The output produced stood at 44%.

However, the accuracy of the results was not up to the mark. On further studies and analysis we found that the problem occurred was the effect of the vanishing gradient.

We then decided to re implement the complete model using the Long Short Term Memory which effectively deals with the issue of vanishing gradient.

As previously stated we divided our whole dataset of videos into subdivision of training, testing and validation datasets. In each of this dataset subdivision we had mixed video footages of positive, where accidents are occurring and negative, devoid of any sort of road rage. During the training phase of our model, the accuracy we obtained was 75%, which is 31% more than the final outcome when recurrent neural network was used in our model.

After wrapping up the training phase, we moved towards the testing phase of our model. Testing our model using LSTM, the accuracy resulted in 87.5 %.

Now we can conclude that the model has improved in its working from the previous time. The problem of vanishing gradient has been fixed and the accuracy or the output has drastically increased when the entire steps were thoroughly carried out using long short Term Memory algorithm.

## **Future Scope**

Unlike recurrent neural network where output is produced based upon the current inputs, this time, we equipped our project with the usage of the long short term memory or the LSTM which not only works upon the current inputs being transmitted but also the past inputs using two different memories, one long and the other short term. Both of these memories work parallel in dealing with the input data of the present and using the relevant past information to reach a decision.

Since we have reduced the vanishing gradient defect using the Long Short Term Memory, there are possibilities that we can test the model using a different algorithm which could produce accuracy somewhere near to LSTM. A similar algorithm is that of Gated Recurrent Unit or GRUs. With the help of GRU we can work all over again on the dataset and produce the outputs having values close to the ones produced during LSTM or maybe even higher. The gated recurrent units are somewhat similar in structure to standard recurrent neural networks with the addition of update gate and reset gate.

Using LSTM worked on the dataset to produce a better accuracy in detecting accidents and furthermore we can employ Gated Recurrent Unit to achieve results in the future. In addition to this, we can also implement Convolution Neural Network or CNN along with LSTM in the project and analyze the results produced thereon. In case we try training our model in such a way such that it only detects accidents of high intensity i.e., those in which the amount of damage taking place was above a certain threshold, we can generate results specially for cases where the level of damage or collision has been drastically higher than normal cases.

## **Applications**

We can connect our model directly to live CCTV footage from cameras installed at places to monitor traffic. The footage will constantly be analyzed by our model and in case if any accident or collision takes place it will easily be identified and be quickly dealt with. Usually under cold weather conditions and fog, car wreckage on a busy expressway causes the trailing vehicles to collide in succession resulting in a vehicular pileup. This pileup has



become frequent over the past few years and certainly need a solution. As soon as a vehicle breaks down on a highway or meets with an accident we can alert the other cars travelling towards that spot beforehand in order to prevent happenings of such highway pileups.



**Fig 5** Highway pileups can be avoided using the model.

## REFERENCES

- [1] Road Traffic Injuries-who.int
- [2] Article about road traffic accidents-timesofindia.indiatimes.com
- [3] Understanding GRU networks-towardsdatascience.com
- [4] The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions- Hochreiter, S. (1998).
- [5] Gate-RNN variations-Rahul Dey and Fathi M. Salem,Michigan State University
- [6] Introduction to Machine Learning book by Nils.J.Nilsson., Informing Science and Technology Volume 11, 2015 , Cornell University, USA.
- [7] Road Accidents in India-statista.com
- [8] Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction-Zhiyong Cui, Ruimin Ke, Ziyuan Pu, Yinhai Wang

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**

**PLAGIARISM VERIFICATION REPORT**

Date: ...15-07-2020..... ✓

Type of Document (Tick):  **PhD Thesis** |  **M.Tech Dissertation/ Report** |  **B.Tech Project Report** |  **Paper**

Name: PRATTYUSH SHANKER SINHA Department: COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY(CSE &IT) Enrolment No 161471

Contact No. 9794123914 E-mail. sinha.prattyush@gmail.com

Name of the Supervisor: DR.RAKESH KANJI

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): FURTHER ANALYSIS OF ACCIDENT DETECTION

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

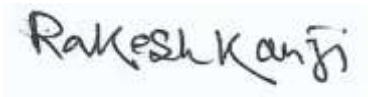
- Total No. of Pages = 50
- Total No. of Preliminary pages = 7
  
- Total No. of pages accommodate bibliography/references = 1



(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at ...12.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.



(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

.....

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

---