

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

Chandrashekhar Choudhary (161472)

Under the supervision

of Dr.Hemraj Saini

to



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

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled “ 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 August 2019 to December 2019 under the supervision of Dr. Hemraj Saini , Associate 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)

Chandrashekhar Choudhary, 161472.

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



(Supervisor Signature)

Dr. Hemraj Saini

Associate Professor

Department of Computer Science and Information Technology

ACKNOWLEDGEMENT

I would like to extend our sincere thanks to Dr. Hemraj Saini, our project guide for providing us with his valuable feedback and suggestions from time to time for our project 'Accident Detection' which helped us in improving the entire work in its crucial areas, especially the weaker ones. During our work on the project development, I inculcated new subject matter and algorithms related to the area of machine learning and sciences which made us familiar with the domain of machine learning. I thank him for his consistent support.

Table of content

Content	Page No
Chapter-1 Introduction	1
Chapter-2 Literature Survey	6
Chapter-3 System Development	11
Chapter-4 Performance Analysis	24
Chapter-5 Conclusion	35

List of Figures	Page No.
1.2.1 Emergency services	3
1.4.1.1 Video dataset from CCTV footage	4
1.4.3.1 Analysis by OpenCV	5
2.1.1 Points on the surface representing function	6
2.2.1 Display of computer vision	7
2.3.1 Example of a neural network	8
2.4.1 MLP with its inputs	9
3.1.1.1 Folded model of RNN	11
3.1.1.2 Unfolded model of RNN	12
3.2.1.1 Gates Involved in LSTM	15
3.2.2.1 Memories	16
3.2.2.2 Long Term Memory	16
3.2.2.3 Short Term Memory	17
3.2.3.1 Learn Gate	19
3.2.3.2 Forget Gate	20
3.2.3.3 Remember Gate	20
3.2.3.4 Use Gate	21
3.3.1 Flow Diagram	22
3.4 System Architecture	23
4.1.1.1 Folder containing Dataset	24
4.1.1.2 Video Dataset	24
4.1.1.3 Path of Dataset	25
4.1.2.1 Different types of accidents	25
4.1.2.3 Shape of x_test	27
4.1.2.3 Numpy array containing Image	28
4.1.2.4 Label showing one-hot encoding	28
4.1.3.1 RNN accuracy	28
4.1.3.2 RNN accuracy after arranging proper videos	29
4.1.3.3 Confusion Matrix	29
4.1.3.4 LSTM accuracy	31
4.1.3.5 Confusion Matrix for LSTM	32
4.2.1 Sample of video input	33

List Of Graphs	Page No.
1.1.1 Registered vehicles over the years.	1
1.1.2 WHO Data	2
3.2.2.3 Sigmoid function	18

ABSTRACT

With the ever-growing number of vehicles every day on road, it has become impossible to evict the dangers posed due to the risk of accidents caused due to road rage and rash driving. In spite of strict laws and rules towards safety the people have been inclining towards reckless driving habits. Often, in developing countries like India, many accident cases are reported late and hence the delay in the arrival of emergency services. This has caused several lives in the past.

This project is aimed at prompting messages to the concerned authorities for providing to reach the affected spot on time. A message for the drivers of the vehicles approaching towards the accident area can be stopped to prevent a vehicle pile-up. This model can be used effectively by several authorities and civilians alike.

Chapter-1

INTRODUCTION

1.1 Introduction

Vehicular traffic, both in India and abroad has been growing at an alarming rate. So have been the cases of accidents caused due to road rage. The total number of registered vehicles in the national Capital region, which forms one of the world's largest urban clusters, increased from 1.42 million in 1988 to 11.2 million in 2016.

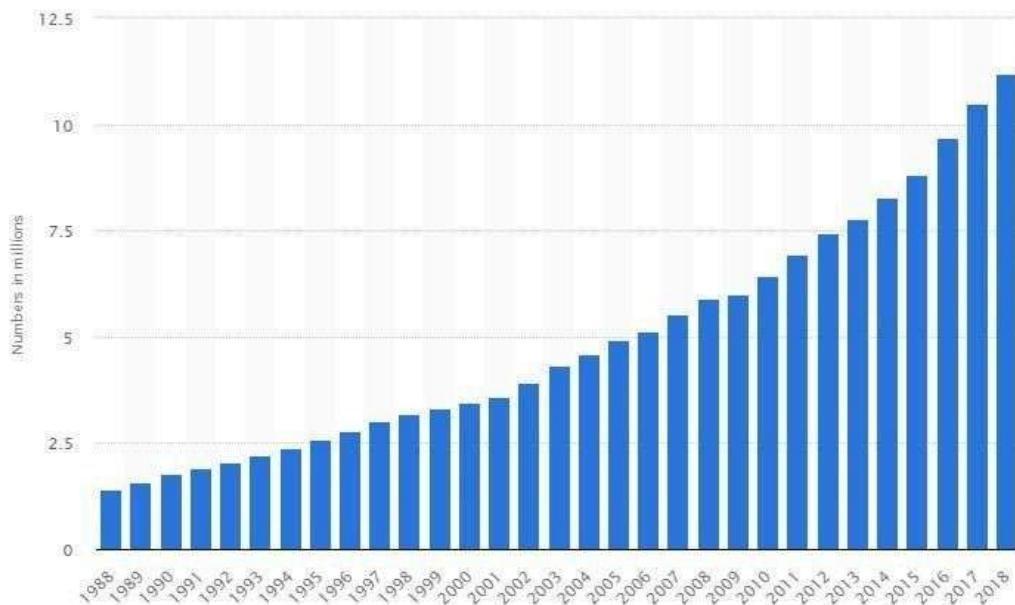


Fig.1.1.1 The number of registered vehicles in Delhi has skyrocketed over the years.

Road accidents are a neglected threat to humans that demands immediate action to deteriorate their growing stake in human casualties and personal and/or public losses in life

or property. As per a report published by World health organization in 2016, an estimated 1.35 million people are dying due to road accidents each year on the world roads.

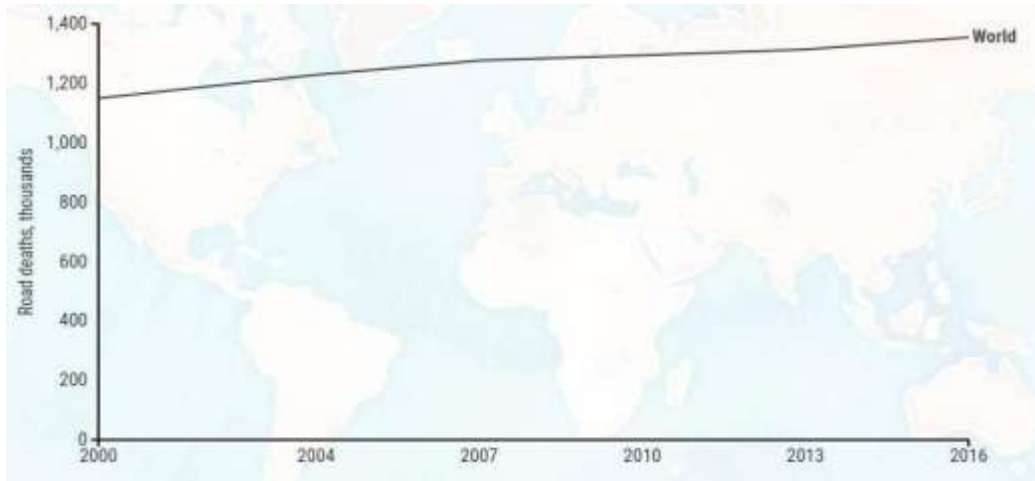


Fig.1.1.2 WHO statistics indicating the death toll of road related cases.

1.2 Problem Statement

There has been no concrete system of reporting the occurred road accidents to the concerned authorities with immediate effect. As per the present scenario, the highway patrolling vehicles or the passerby's have to report the matter or call the ambulance for an action. This entire process is elaborated to an extent that it has cost several lives; especially in countries where the emergency services to arrive on spot have not been quick. An information retrieval system is the need of the present scenario which can analyze the collision and produce information which can detect the occurrence of an accident, thereby triggering an immediate response which in turn will alert the local emergency services .This would help the authorities in reaching the affected spots timely for help.



Fig 1.2.1 Emergency services often arrive late.

1.3 Objectives

The main objective of our project is to reduce the intermediate time between the occurrence of an accident and the arrival of help thereafter. Also if the previous if fully implemented with the desired results, we can also decide the intensity of the damage or loss incurred in an accident, thereby triggering the message to the corresponding services needed as per the situation. This will greatly help in reducing the number of human casualties and the loss of life.

1.4 Methodology

In order to implement this project, we need to follow a proper analysis based methodology. The methodology involves four major steps,

1.4.1 Dataset

Firstly, we will be working on the collection of our dataset. The primary task involves finding the real footages of accidents occurred which we can use to work upon. The dataset will include footage from CCTV cameras across the globe. To collect these videos, we first browsed through the video library on the social networking video sharing site, YouTube. Despite the content being available, we were unable to use them because of their length and a few quality issues. In order to work with the dataset, we needed a uniform set of videos where the length of footage was more or less, same. Finally, we came across a website,

VSlab.Videos of optimum length and quality were found there.The length of most videos were short,they were about 4 to 5 seconds.Still,the dataset had too many variations,i.e.the types of accidents occurring.The task was going to b e difficult.We segregated the dataset into a su b dataset having similar kind of video footage.



Fig.1.4.1.1 The video dataset was an assortment of CCTV footages

1.4.3 Preprocessing

Now coming to the preprocessing part.Priorly,we had to store the dataset.We created a list containing paths of the video data which we would be using. We further used opencv and extracted 99 frames of each image in the video footage and downscaled them by 5 and stored it in the 3 dimensional list. Now I have a dataset which is a 3 dimensional list containing 99 frames and each frame is an image in itself. The size of each image is in terms of an array of pixels,255*144.



fig.1.4.3.1 OpenCV analyses the still image from a video.

1.4.4 Training and Testing

We spliced our dataset into training and testing sets separately. Now the major challenge was how to maintain the time dependencies while choosing the appropriate model. After going through several algorithms for this task, we stumbled upon RNN. This algorithm helped us in maintaining time dependencies.

Chapter-2
LITERATURE REVIEW

1. Learning Input-Output Functions-Introduction to Machine Learning

The book by Nils.J.Nilsson stresses out the terminology applied while elaborating the problem of learning a function. Let's consider a function, f , and we are given with the job of guessing its actual description. Let the hypothesis we are learning about the function, h . Consider, h being implemented by a machine with input as X and output as $h(X)$. now let's assume a priory that h , which is the hypothesized function being selected from within a class of H functions. If we that f belongs to class H or to one of its subsets, we take h based on a training set E with input vector examples 'm'. A lot of essential details depend upon nature of assumption we have made about these entities given.

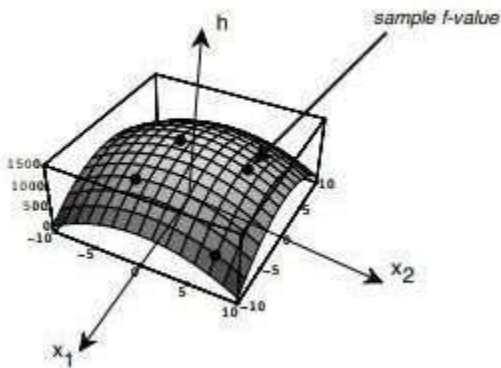


Fig.2.1.1 Four points on the surface

2. Computer Vision-Learning OpenCV.

This text matter given in the book by Gary Bradski and Adrian Miller explains vividly the conflict between man and machine visions. Computer vision actually means the transformation being carried out by a computer when it visualizes a given image, a still from a video footage, or simply a scenery. This transformation may include the conversion of

coloured imagery to gray-scale for analyses purposes. Computer vision varies drastically from human vision. We, being visual creature a habitual of recognizing things using our brain whenever we encounter an imagery in real world or a screen grab from a song or movie. Our brain analyzes the given picture by only stressing or focusing on the crucial aspect of the photograph. Say, if we are to analyze the picture of a car, our brain senses that it is a car by the signal it receives using the memory we have establishes as human beings. This memory is based upon the visualization we have seen over the years living in this world. This is the manner in which we are able to associate things or relate to what we see through our eyes. On the other hand, a computer on encountering an image only perceives it as a huge grid of numbers. Any particular number Plus any number given on the grid gives the computer much less information and has an extra noise component which has to be gotten rid of. We therefore get a bigger task at hand, to transform all these noisy numbers into a perception.

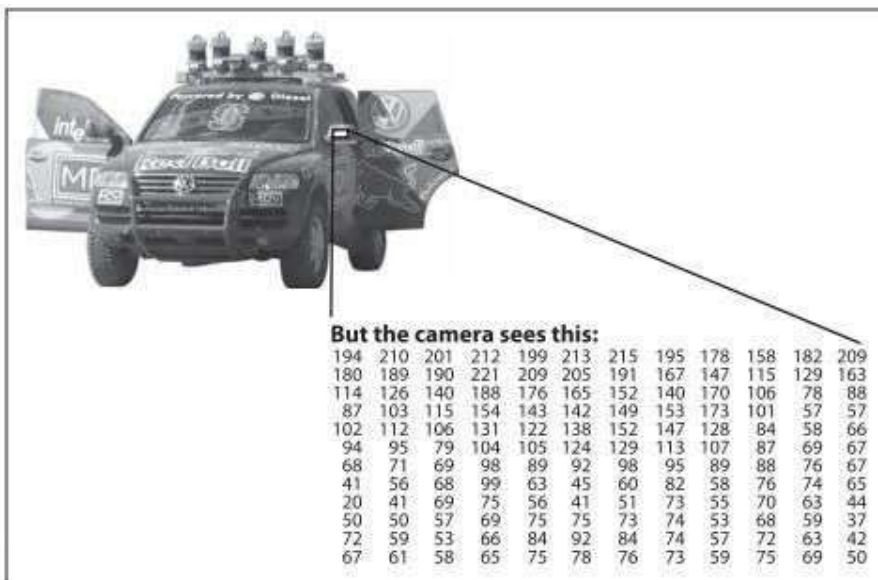


Fig.2.2.1 A computer vision sees only a grid of numbers

3. Neural networks-Learning OpenCV

Neural networks are a combination of non linear elements interconnected through weights adjustable by nature.

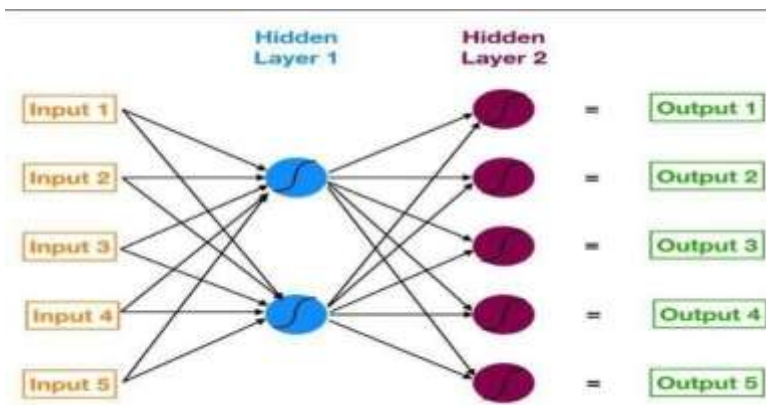


Fig.2.3.1 An example of a neural network

These networks are called neural networks because the inputs of non-linear elements are actually the weighted sum of outputs produced by other elements, in a fashion similar to that of neurons present in biological beings. These neural networks play a prominent role in machine learning.

4. Recurrent Neural Network Architecture-Neural Computation, University of Birmingham

The fundamental property of a recurrent neural network is that it contains at least one feed-back connection so as to facilitate the flow of activations around the loop. This technique helps the recurrent network to perform temporal processing and learn sequences, for example, performing sequence reproduction or recognition or temporal association and prediction.

Learning in architectures which are simple and deterministic activation functions can be achieved with the help of similar descent procedures to those leading to back-propagation algorithm or feed-forward networks.

A multi-layer perception with previous set of hidden unit activations feeding back into the network along with the inputs is the simplest form of fully recurrent neural network.

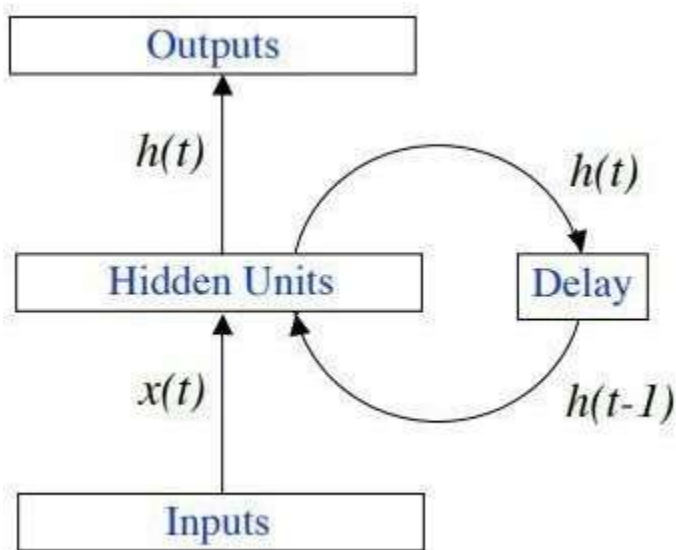


Fig.2.4.1 An MLP along with its inputs.

5. Why Gradients Explode or Vanish, Roger Grosse-University of Toronto

Consider the architecture of an encoder-decoder machine. Let us say, that the task given to it is the conversion of a given English sentence to French. A normal sentence consists of about 20 words. Now the prediction of the output based on only the last word of the English sentence is not possible and will produce an inaccurate output. This is because the memory of the machine has not retained the initial words of the sentence or say, the initial inputs. This long distance between the first and the last input to the machine causes the false prediction. This is known as vanishing gradient problem.

Chapter-3 System Development

3.1 Recurrent Neural Networks (RNNs):

Many applications in our day to day involves reviewing our previous tasks or in other words they require timely dependent. When we deal with time dependencies, we require memory which can store previous data. Dealing with video dataset are same. Videos are collection of images that are timely dependent. To deal with these type of time dependencies we require memory and RNNs comes with this advantage. RNNs have memory which can be used to save previous outputs , they have state which can be used to store previous states and reviewing these states we can predict next output.

3.1.1 Structure of RNNs:

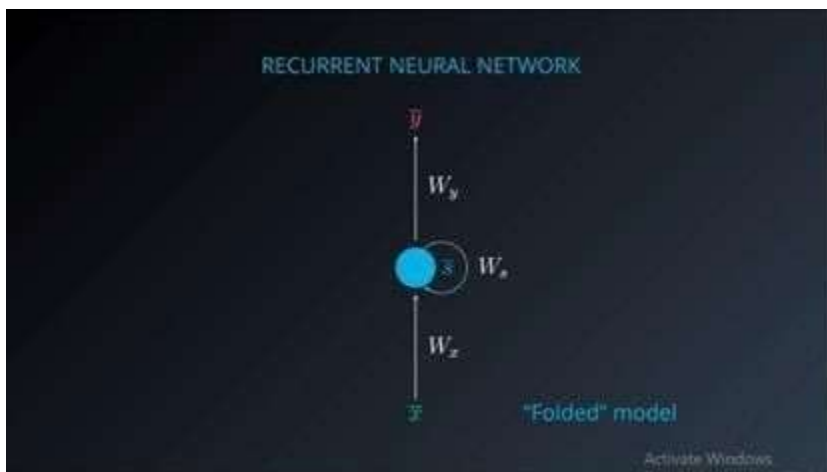


Fig 3.1.1.1 folded model of RNN

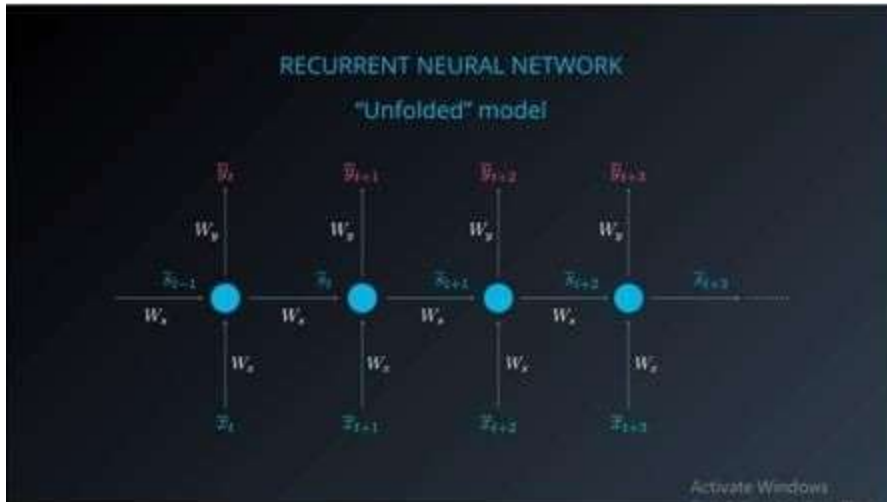


Fig 3.1.1.2 Unfolded model of RNN

In the figure Fig 3.1.1.1, x represents the input vector, y represents the output vector and s denotes the state vector.

W_x is the weight matrix connecting the inputs of the state layer.

W_y is the weight matrix that is connecting the state layer to the output layer.

W_s represents the weight matrix connecting the state from the previous timestep to the state from the previous timestep to the state in the current timestep.

The model can be unfolded in time. The unfolded model is usually what we use when working with RNNs as represented in Fig 3.1.1.2.

In both the folded and unfolded models shown in Fig 3.1.1.1 and Fig 3.1.1.2 the same notation is used.

3.1.2 Theory of RNNs:

Recurrent means occurring often or repeatedly. In RNNs we use sequences as input and have memory element. The mathematical calculations needed for training RNN systems are fascinating. To deeply understand the process, we first need to feel confident with the vanilla

FFNN system. We need to thoroughly understand the feedforward process, as well as the backpropagation process used in the training phases of such systems.

We will address the feedforward process as well as backpropagation, using specific examples. These examples will serve as extra content to help further understand RNNs. As mentioned before, when working with neural networks we have 2 primary phases:

1. Training
2. Evaluation

During the training phase, we take the data set (also called the training set), which includes many pairs of inputs and their corresponding targets (outputs). Our goal is to find a set of weights that would best map the inputs to the desired outputs.

In the evaluation phase, we use the network that was created in the training phase, apply our new inputs and expect to obtain the desired outputs. The training phase will include two steps:

1. Feedforward
2. Backpropagation

We will repeat these steps as many times as we need until we decide that our system has reached the best set of weights, giving us the best possible outputs.

RNNs are based on the same principles as those behind Feedforward Neural Networks (FFNNs), which is why we spend so much time reminding ourselves of the feedforward and backpropagation steps that are used in the training phase.

There are two main differences between FFNNs and RNNs. The Recurrent Neural Network uses:

sequences as inputs in the training phase, and
memory elements

Memory is defined as the output of the hidden layer neurons, which will serve as additional input to the network during the next training step.

3.2 Long Short Term Memory (LSTM):

LSTM or Long Short Term Memory Networks, and are quite useful when our neural network needs to switch between remembering recent things, and things from a long time ago. It maintains two memories one is long term memory and the other is short term memory , these memories helps to retain patterns. LSTM deals with the disadvantage of RNNs i.e Vanishing Gradient Descent.

3.2.1 Structure of LSTM:

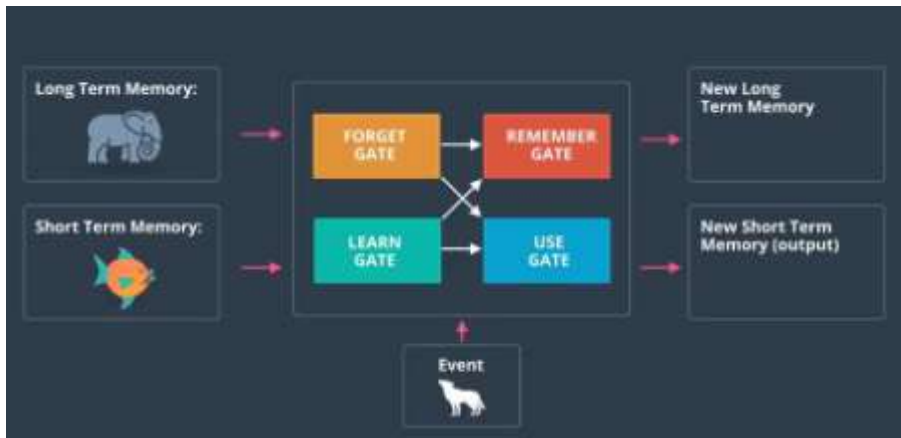


Fig 3.2.1.1 Gates involved in LSTM.

3.2.2 Theory of Long Short Term Memory (LSTM)

To explain LSTM let's take an example We have a tv show of nature and science. We have seen a lot of forest animals. We recently have seen Squirrels and trees. Our job is to predict whether given animal is a dog or wolf.

So what LSTM does?

It maintains three memories, short term, long term memory, event. In long term memory we have the idea that the show is about science and nature , forest animals. In short term memory we have recently seen squirrels and trees. We also maintain event which is to find whether the animal is a dog or wolf.

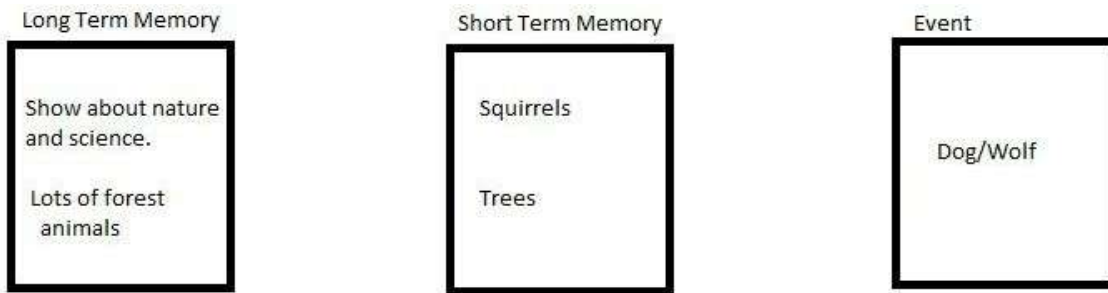


Fig 3.2.2.1

Now we will use all three memories to update the long term memory. Long term memory will be updated as:



Fig 3.2.2.2

We forget about science and since we recently saw a tree so we remember a tree and our long term memory is updated.

Again we use those three memories to update Short term memory. Short Term memory is updated as:



Fig 3.2.2.3

We forget about trees and update short term memory with event.

As Shown in Fig 3.2.1.1 we have four gates which helps in updating the memories.

These gates are:

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

Mathematics involved is already discussed in previous chapter. Let's see how these gates help to given task.

The long term memory goes to the Forget Gate where it forgets the things which are not needed.

The short term memory and event are joined together in the Learn Gate where it collects the information we have recently learnt and removes unnecessary information. The Long term memory that we have not forgotten yet and the new information we have recently learnt are joined together in the Remember Gate. This information becomes the new updated long term memory. Finally the Use Gate is the one that decides which information will be used from previously known and recently learnt data to make the prediction. The output from this gate becomes the prediction and the new short term memory.

3.2.1 Advantage of LSTM over RNNs

LSTM deals with the vanishing gradient descent of RNNs.

Vanishing Gradient Descent:

As we increase the numbers of layers using some activation functions in RNNs the gradient of the loss function tends to zero and this makes hard to train the model because our inputs and its output are dependent on one another.

Why this problem occurs?

Few activation functions such as sigmoid function, squishes larger data into range of 0-1 and hence if we make a large change in the input of sigmoid function the output will be changed in small quantity. Therefore, the derivative becomes smaller.

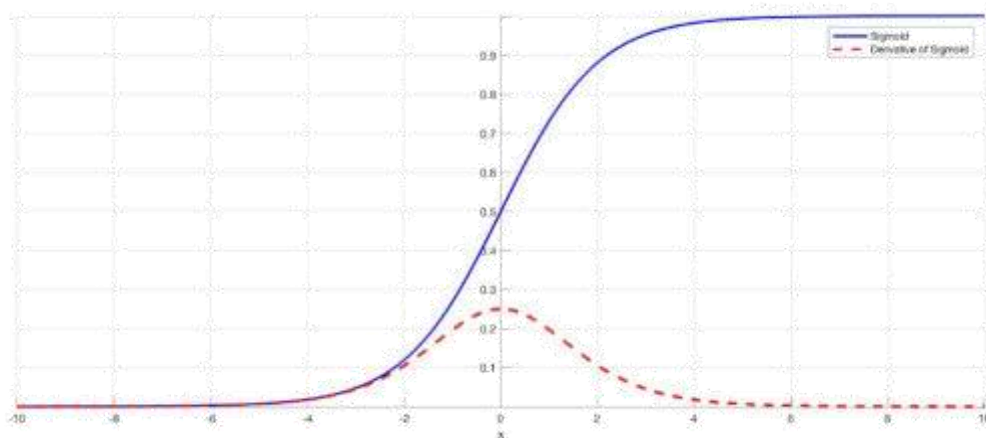


Fig 3.2.2.3

The above figure shows the sigmoid function and its derivative. Notice how depending on inputs of sigmoid function the derivative becomes closer to zero.

3.2.3 Gates involved in LSTM:

3.2.3.1 Learn gate:

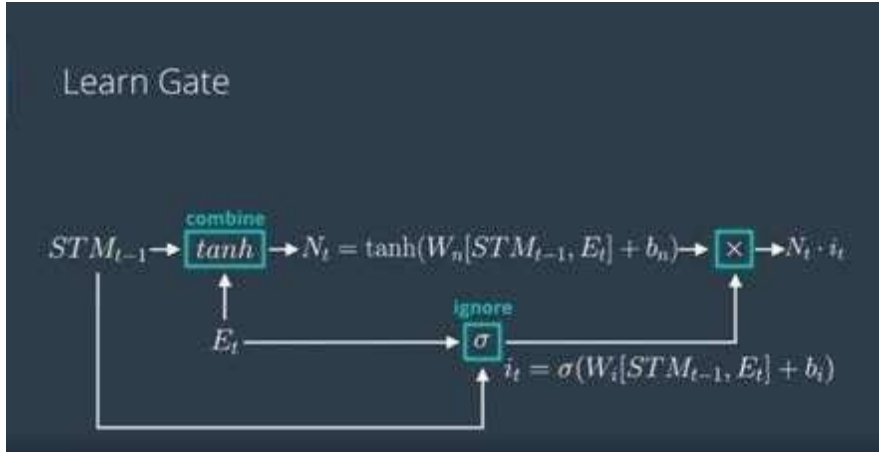


Fig 3.2.3.1

We have a short term memory STM_{t-1} and an event E_t which is combined through a linear function joining the vectors, multiplied by a weight matrix, adding a bias and finally squishing result through tanh function.

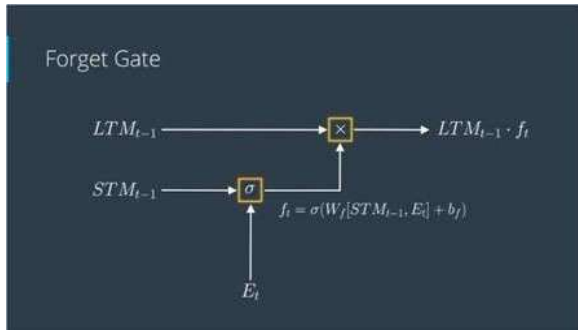
Now how do we ignore part of i_t ?

We do it by multiplying it with an ignore factor i_t .

We create a small neural network whose inputs are STM_{t-1} and E_t which is again combined through a linear function joining the vectors, multiplied by new weights, added a new bias and finally squishing result through sigmoid function(N_t).

The final result is obtained by multiplying N_t and i_t .

3.2.3.2 Forget Gate



We have a short term memory STM_{t-1} and an event E_t which is combined through a linear function joining the vectors, multiplied by a weight matrix W_f , adding a bias b_f and finally squishing result through sigmoid function (σ) to obtain f_t .

We then multiply this f_t and LTM_{t-1} (Long term memory)

3.2.3.3 Remember Gate

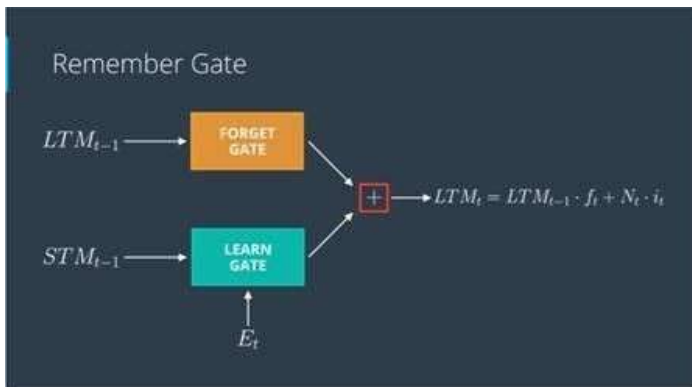


Fig 3.2.3.3

We simply combine the outputs coming from forget gate and learn gate in Remember gate.

Output from forget gate:

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

Where

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

Output from Learn gate:

$$op2 = N_t \cdot i_t$$

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 $op1 + op2$.

3.2.3.4 Use Gate

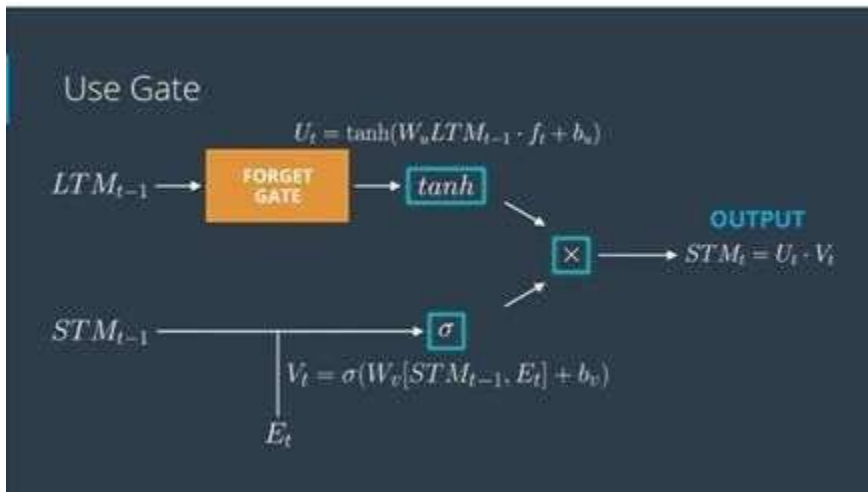


Fig 3.2.3.4

We combine output coming from forget gate with a small neural network having weight matrix W_u and bias of b_u and squishing it through tanh activation function.

We can represent this equation with eq4.

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

Next we combine short memory STM_{t-1} and event E_t with a small neural network having weight matrix W_v and bias of b_v and squishing it through sigmoid activation function. We can represent this equation with eq5.

$$V_t = (W_v[STM_{t-1}, E_t] + b_v)$$

The output of the Use Gate is $U_t \cdot V_t$

3.3 Flow Diagram

The complete accident detection system involves various steps. Firstly, the videos dataset is pre-processed according to input criteria of the algorithms. And then the pre-processed data is passed further to the training step where our deep learning model will be trained to detect accidents in any given video. Also, the validation of the trained model is done adjacently. At the end we will get a trained deep learning model that would be capable to detect accident in any video. This pre-trained model could be used at various places for the accident detection purposes.

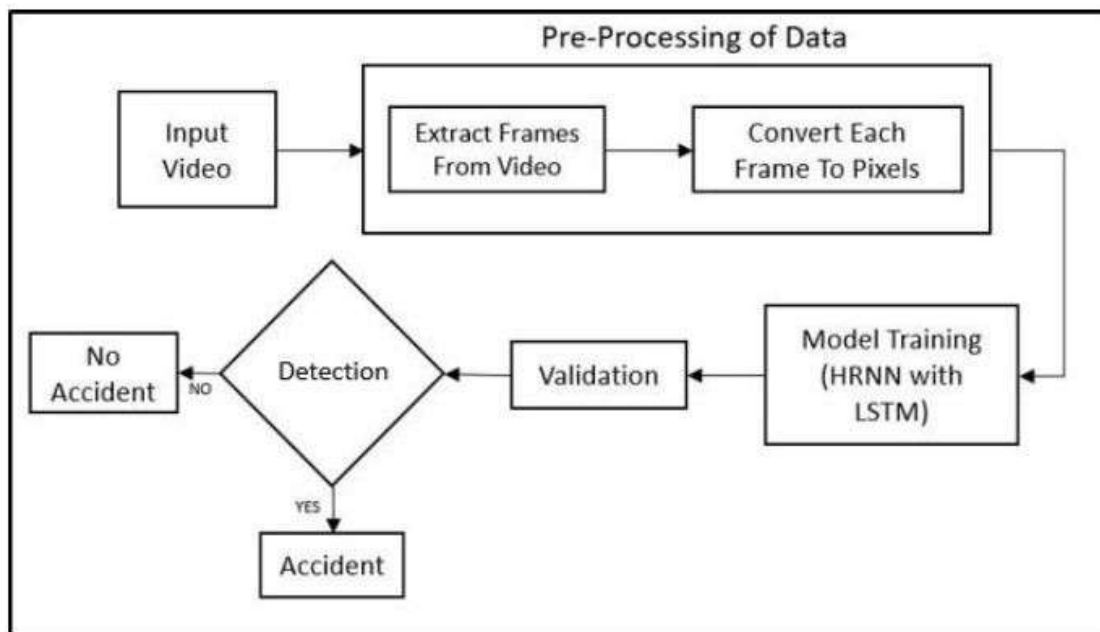


Fig 3.3.1 Flow Diagram

3.4 System Architecture:

The proposed system architecture for our project is shown in fig. 5.1. The initial step deals with the dataset. The data is first pre-processed and then the negative and positive both video data is passed through the neural network. At the end we get a trained deep learning model which would be capable to detect accidents in any video.

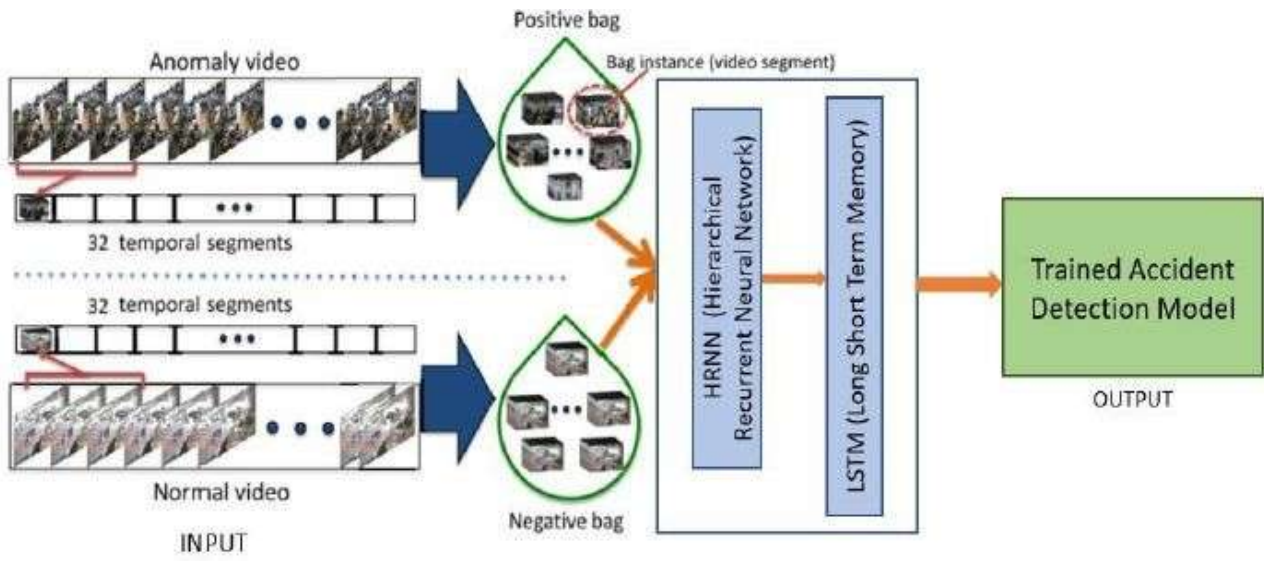


Fig. 3.4 System Architecture

Chapter-4

Performance Analysis

4.1 Output at various stages:

4.1.1 Creating Dataset

We obtained our dataset from VSLabs which contained 1000s of videos containing accident and normal traffic. Our dataset were contained in two different folders. First positive and another negative, positive folder contained videos of accidents and negative folder contained videos of normal traffic as shown in Fig 4.1.1.1, Fig 4.1.1.2. Training model with 1000s of videos was not possible so we created two different folders positive1, negative1 containing 10 videos each. Then I stored path of videos for training, testing and validating using glob function as shown in Fig 4.1.1.3.



Fig. 4.1.1.1

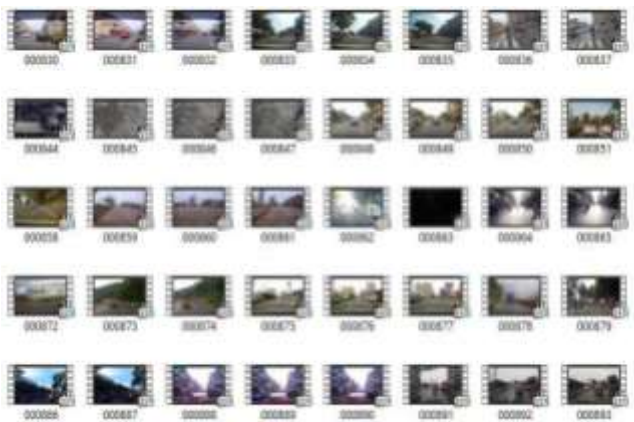


Fig 4.1.1.2


```
0 E:/videos/training/positive1\000004.mp4
1 E:/videos/training/positive1\000008.mp4
2 E:/videos/training/positive1\000003.mp4
3 E:/videos/training/positive1\000002.mp4
4 E:/videos/training/negative1\000001.mp4
5 E:/videos/training/negative1\000007.mp4
```

Fig 4.1.1.3

4.1.2 Preprocessing

Videos are collection of image frames and images are array of pixels with different numbers denoting different colors.

The major question was why preprocessing was required?

It was because of two major reasons:

1. Too much variation in dataset:

There were type of accidents car bike accidents, bike bike accidents, car car accident etc.

Also there were repeated videos in dataset.



Fig4.1.2.1: Showing different types of accidents.

2. Imbalanced dataset:

The positive video were too less than negative videos. When I say positive video it means video contains accidents and when I say negative it means video do not contain accident.

To deal with these two problems preprocessing was required.

Manual deletion of 1000+ videos was a tedious job. I came up with a better plan. I decided to train my model for 40 positive video and 40 negative video and when my model is prepared, I would save the model and feed it with the complete data.

Steps involved in preprocessing:

1. Since I was dealing with video data which contained thousands of image frames which corresponded to huge data. I decided to take 99 image frames of each video data and store it in a 3-dimensional array. 3-dimensional array, because an image comprises of 2-D array each cell corresponding to pixels. The 3rd dimension was introduced because of frame number since we have 99 image frames.
2. To create my dataset. I stored the path using glob function of glob library.

```
img_filepath = 'D:/videos/training/'  
neg_all = glob.glob(img_filepath + 'negative1/*.mp4')  
pos_2 = glob.glob(img_filepath + 'positive1/*.mp4')
```

neg_all: stored all the video paths without accident

pos_2: stored all the video paths containing accident

3. Now ready to create the dataset.

I defined a function `make_dataset` which would take path of a video as parameter and convert that video in 99 image frames (A 3-D array).

This `make_dataset` uses another function `load_set` which take path of the video as input parameter and returns an array containing pixels of downscaled image of size 144 x 255. This function loaded the video using `openCV` and converted the video into

image frames. Also using openCV I downscaled my image to decrease my processing time.

```
def make_dataset(rand):
    seq1 = numpy.zeros((len(rand), 99, 144, 256))
    for i, fi in enumerate(rand):
        print (i, fi)
        if fi[-4:] == '.mp4':
            t = load_set(fi)
            if numpy.array(t).shape==(99,144,256):
                seq1[i] = t
            else:# TypeError:
                'Image has shape ', numpy.array(t).shape, 'but needs to be shape', numpy.array(seq1[0]).shape
                pass
    return seq1
```

fig. Function to create dataset.

```
def load_set(videofile):
    vidcap = cv2.VideoCapture(videofile)
    success,image = vidcap.read()
    count = 0
    success = True
    img = []
    while success:
        success, image = vidcap.read()
        if success:
            dummy = numpy.zeros(image.shape)
            if (image-dummy).any()!=None:
                image = skimage.color.rgb2gray(image)
                image = skimage.transform.downscale_local_mean(image, (5,5))
                img.append(image)
            count += 1
    return img
```

fig. Function to create image frame for videos.

In short in this phase we extracted 99 frames of images, converted RGB to GrayScale and downgraded them by five using opencv and stored it in multidimensional numpy array. We can see the shape in Fig. 4.1.2.2 and the numpy array in Fig 4.1.2.3. Dealing with images in GrayScale is easier and faster than RGB. Now we are ready to perform operations on numpy array denoting the images and hence video. We one-hot encoded i.e created a label matrix of size 1X2 (1 row and 2 column). The columns denoted accident and not accident. Initially both column value is set to 0. If the video contained accident index 0 is set to 1 else index 1 is set to 1. The label matrix is shown in Fig 4.1.2.4.

```
In [22]: x_test.shape
Out[22]: (6, 99, 144, 256)
```

Fig 4.1.2.3

videos of accidents and other 5 videos of normal traffic. This change in arrangement resulted in accuracy of 50%.

```
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 4.1.3.2

```
Area under curve 0.6857142857142857
Accuracy 0.50
confusion matrix
[[ 5  5]
 [12 12]]
True negative: 5
False positive: 5
False negative: 12
True positive: 12
```

Fig 4.1.3.3 Confusion matrix:

Fig 4.1.3.3 Shows Confusion matrix. What is confusion matrix?

Confusion matrix is tool through which we can easily visualize the performance of our model. It is also called as error matrix. It shows the proper result depicting how much occur has been occurred and how much correct outputs are predicted by our model.

Confusion matrix has following attributes:

1. True Positive
2. True Negative
3. False Positive

4. False Negative

Now, what does these attributes signify?

True Negative: Our video didn't contain any accident and is correctly identified as normal traffic. In our case the number of videos that are correctly identified as normal traffic are 5.

False Positive: Our video didn't contain any accident and is wrongly identified as an accident. In our case the number of videos that are wrongly identified as accidents are 5.

False Negative: Our video contained any accident and is wrongly identified as normal traffic. This is the critical situation of our project as we can afford traffic to be identified as an accident but can't afford to wrongly identify accident as normal traffic. In our case the number of accidents wrongly identified as normal traffic are 12.

True Positive: Our video contained any accident and is correctly identified as an accident. In our case the number of videos that are correctly identified as normal traffic are 12.

Long Short Term Memory (LSTM)

We implemented LSTM using keras library. The test accuracy that we got was 75%.

```

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 model11.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 model11.hdf5
Epoch 17/20
12/12 [=====] - 4s 306ms/step - loss: 0.3650 - acc: 0.9167 - val_loss: 0.6655 - val_acc: 0.6667

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 4.1.3.4

We tried to increase our accuracy by increasing the number of epochs but the accuracy didn't improve. Then we tried to increase our accuracy by varying the number of layers but the result was the same. We never realized that training phase could be affected by the arrangement of video datasets. First we passed the videos containing accidents and normal traffic randomly, But then we passed evenly organized videos of accidents and normal traffic of equal amount. For instance, if we are passing 40 videos for training then, passing 20 videos of accidents and other 20 videos of normal traffic. This change in arrangement resulted in accuracy of 87.5%.

```
Area under curve 0.9285714285714285
Accuracy 0.875
confusion matrix
[[ 7  3]
 [ 0 14]]
True negative: 7
False positive: 3
False negative: 0
True positive: 14
```

Fig 4.1.3.5 Confusion Matrix

True Negative: Our video didn't contain any accident and is correctly identified as normal traffic. In our case the number of videos that are correctly identified as normal traffic are 5.

False Positive: Our video didn't contain any accident and is wrongly identified as an accident. In our case the number of videos that are wrongly identified as accidents are 5.

False Negative: Our video contained any accident and is wrongly identified as normal traffic. This is the critical situation of our project as we can afford traffic to be identified as an accident but can't afford to wrongly identify accident as normal traffic. In our case the number of accidents wrongly identified as normal traffic are 12.

True Positive: Our video contained any accident and is correctly identified as an accident. In our case the number of videos that are correctly identified as normal traffic are 12.

4.2 Outcome of the model

Model identified many videos correctly and many video wrong.

Correctly identified accident.



```
x=predic(model.predict(ta))
```

Accident Occurred



We got the result as an accident for the positive video. Now let's see the negative video. We got the result as No Accident for the negative video.

Chapter 5

Conclusion

5.1 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.

5.2 Future Scope

An algorithm that can reduce the vanishing gradient is Long Search Term Memory or LSTM in short. Using LSTM we can rework on the dataset hope to produce a better accuracy on detecting accidents. Instead of predicting outputs based simply on the most recent inputs, it works on two parallel memories, that is one a long term and other a short one. This increases the overall accuracy. Furthermore, we can implement CNN along with LSTM in the project. This may yield even better results. If we go on to train our model to detect only the accidents of very high intensity, we can only generate results for cases where the level of damage or collision has been excruciatingly high. In a further upgrade we can train the model to predict the percentage of collision in a particular accident and decide whether high level or low level emergency services are needed for that situation.

5.3 Applications

This model can be used to detect the occurrence of an accident using live data tracking of CCTV cameras. This would generate a message in case an accident has occurred. In foggy conditions, a highway pile up is a severe problem taking place. In case of an accident taking place on a highway, we can alert the trailing cars to prevent them from hitting the damaged vehicle thereby preventing a pile up.

REFERENCES

- [1] Introduction to Machine Learning book by Nils.J.Nilsson., Informing Science and Technology Volume 11, 2015 , Cornell University, USA.
- [2] Learning OpenCv by Gary Bradski and Adrian Miller, pp. 225–227.
- [3] Neural Network Computation – Lecture 12 University of Birmingham, pp. 21-22, 2013.
- [4] Roger Grose- University of Toronto, 2015.
- [5] Course for Data Visualization- [udemy.com](https://www.udemy.com/).
- [6] Introduction to Machine Learning-[udacity.com](https://www.udacity.com/).
- [7] Deep Learning-[udacity.com](https://www.udacity.com/).
- [8] Machine Learning-[coursera.com](https://www.coursera.com/).

Appendices

Source Code:

```
In [1]: %matplotlib inline
import numpy
import os
import re
import pickle
import timeit
import glob
import cv2
import matplotlib.pyplot as plt
import matplotlib

from skimage import transform
import skimage
from skimage import io

import sklearn
from sklearn.model_selection import train_test_split

import keras
from keras.preprocessing import image as image_utils
from keras.callbacks import ModelCheckpoint

plt.rcParams['font',family='Bitstream Vera Serif']
```

C:\Users\Chandrashekhar\Anaconda3\lib\site-packages\h5py__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == n

```
p.dtype(float).type`.  
from ._conv import register_converters as _register_converters  
Using TensorFlow backend.
```

```
In [116]: def load_set(videofile):  
    vidcap = cv2.VideoCapture(videofile)  
    success,image = vidcap.read()  
    count = 0  
    success = True  
    img = []  
    while success:  
        success, image = vidcap.read()  
        if success:  
            dummy = numpy.zeros(image.shape)  
            if (image-dummy).any!=None:  
                image = skimage.color.rgb2gray(image)  
                image = skimage.transform.downscale_local_mean(image, (5,5))  
                img.append(image)  
            count += 1  
    return img
```

```
In [117]: img_filepath = 'E:/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')
```

```
In [118]: pos_all = pos_1 + pos_2  
all_files = pos_all[:] + neg_all[:]  
print(len(all_files), len(pos_all), len(neg_all))
```

```
40 20 20
```

```
In [119]: def label_matrix(values):  
    '''transforms labels for videos to one-hot encoding/dummy variables'''  
    n_values = numpy.max(values) + 1  
    return numpy.eye(n_values)[values]  
labels = numpy.concatenate(([1]*len(pos_all[:]), [0]*len(neg_all[0:len(pos_all  
[:])]))))  
labels = label_matrix(labels)
```

```
In [120]: def make_dataset(rand):
seq1 = numpy.zeros((len(rand), 99, 144, 256))
for i,fi in enumerate(rand):
    print (i, fi)
    if fi[-4:] == '.mp4':
        t = load_set(fi)
    elif fi[-4:]==' .pkl':
        t = pickle.load(open(fi, 'rb'))
    if numpy.array(t).shape==(99,144,256):
        seq1[i] = t
    else:# TypeError:
        'Image has shape ', numpy.array(t).shape, 'but needs to be shape',
numpy.array(seq1[0]).shape
        pass
    return seq1
```

```
In [121]: import array
```

```
In [122]: x_train, x_t1, y_train, y_t1 = train_test_split(all_files, labels, test_size=
0.40, random_state=101,shuffle=True) ### split
```

```
In [123]: x_train = numpy.array(x_train)
y_train = numpy.array(y_train)
```

```
In [124]: x_testA = numpy.array(x_t1[len(x_t1)//2:])
y_testA = numpy.array(y_t1[len(y_t1)//2:]) ##### test set

### valid set for model
x_testB = numpy.array(x_t1[:len(x_t1)//2])
y_testB = numpy.array(y_t1[:len(y_t1)//2])
```



```
In [125]: y_testA
```

```
Out[125]: array([[1., 0.],  
                [1., 0.],  
                [0., 1.],  
                [1., 0.],  
                [1., 0.],  
                [0., 1.],  
                [0., 1.],  
                [1., 0.]])
```

```
In [126]: x_test = make_dataset(x_testB)
```

```
0 E:/videos/training/positive1\000061.mp4  
1 E:/videos/training/positive1\000020.mp4  
2 E:/videos/training/negative1\000017.mp4  
3 E:/videos/training/negative1\000006.mp4  
4 E:/videos/training/negative1\000007.mp4  
5 E:/videos/training/negative1\000011.mp4  
6 E:/videos/training/negative1\000001.mp4  
7 E:/videos/training/positive1\000002.mp4
```

```
In [128]: import keras  
          from keras.models import Model  
          from keras.layers import Input, Dense, TimeDistributed  
          from keras.layers import LSTM  
          import random  
  
          ### 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))  
  
          encoded_rows = TimeDistributed(LSTM(row_hidden))(x)  
          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)
model.compile(loss='categorical_crossentropy',
              optimizer='NAdam',
              metrics=['accuracy'])
i=0; filepath='modelextample.hdf5'
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best
_only=True, mode='max')
callbacks_list = [checkpoint]
numpy.random.seed(18247) ### set a random seed for repeatability
```

```
x_train shape: (24,)
24 train samples
8 test samples
```

```
In [129]: x_tr=make_dataset(x_train)
```

```
0 E:/videos/training/negative1\000009.mp4
1 E:/videos/training/negative1\000005.mp4
2 E:/videos/training/positive1\000003.mp4
3 E:/videos/training/negative1\000016.mp4
4 E:/videos/training/positive1\000019.mp4
5 E:/videos/training/positive1\000069.mp4
6 E:/videos/training/negative1\000018.mp4
7 E:/videos/training/negative1\000010.mp4
8 E:/videos/training/positive1\000016.mp4
9 E:/videos/training/positive1\000009.mp4
10 E:/videos/training/positive1\000018.mp4
11 E:/videos/training/positive1\000001.mp4
12 E:/videos/training/negative1\000002.mp4
13 E:/videos/training/negative1\000020.mp4
14 E:/videos/training/positive1\000008.mp4
15 E:/videos/training/positive1\000012.mp4
16 E:/videos/training/positive1\000017.mp4
17 E:/videos/training/positive1\000013.mp4
18 E:/videos/training/negative1\000019.mp4
19 E:/videos/training/negative1\000004.mp4
20 E:/videos/training/positive1\000010.mp4
21 E:/videos/training/positive1\000060.mp4
22 E:/videos/training/positive1\000015.mp4
23 E:/videos/training/negative1\000012.mp4
```

```
In [130]: y_train
```

```
Out[130]: array([[1., 0.],
                 [1., 0.],
                 [0., 1.],
                 [1., 0.],
                 [0., 1.],
                 [0., 1.],
                 [1., 0.],
                 [1., 0.],
                 [0., 1.],
                 [0., 1.],
                 [0., 1.],
                 [0., 1.],
                 [1., 0.],
                 [1., 0.],
                 [0., 1.],
                 [0., 1.],
                 [0., 1.],
                 [0., 1.],
                 [1., 0.],
                 [1., 0.],
                 [0., 1.],
                 [0., 1.],
                 [0., 1.],
                 [1., 0.]])
```

```
In [131]: Test=make_dataset(x_testA)
```

```
0 E:/videos/training/negative1\000013.mp4
1 E:/videos/training/negative1\000008.mp4
2 E:/videos/training/positive1\000011.mp4
3 E:/videos/training/negative1\000015.mp4
4 E:/videos/training/negative1\000003.mp4
5 E:/videos/training/positive1\000014.mp4
6 E:/videos/training/positive1\000004.mp4
7 E:/videos/training/negative1\000014.mp4
```

```
In [134]: model.fit(x_tr, y_train,
                   batch_size=10,
                   epochs=20,
                   validation_data=(Test, y_testA),
                   callbacks=callbacks_list, shuffle=True)
```

```
In [145]: # evaluate
scores = model.evaluate(x_test, y_testB, verbose=1)
print('Test loss:', scores[0])          ### test loss
print('Test accuracy:', scores[1])     ### test accuracy (ROC L
ater)
```

```
In [137]: model.load_weights("modelextample.hdf5")
model.compile(loss='binary_crossentropy', optimizer='Nadam', metrics=['accuracy'])
```

```
In [163]: plt.plot([0,1],[0,1], 'k:', alpha=0.5)
ys = [y_train, y_testA, y_testB]
labs = ['Train', 'Valid', 'Test']
col = ['#4881ea', 'darkgreen', 'maroon']
preds = []
for i,xset in enumerate([x_train, x_testA, x_testB]):
    if i==0:
        new_pred = []
        for k in xset:
            d = make_dataset([k])
            new_pred.append(model.predict(d))
        new_pred = numpy.array(new_pred).reshape((len(new_pred),2))
    else:
        d = make_dataset(xset)
        new_pred = model.predict(d)
    preds.append(new_pred)
    fpr, tpr, threshs = sklearn.metrics.roc_curve(ys[i][:,1], new_pred[:,1])
    plt.plot(fpr, tpr, '-', color=col[i], alpha=0.7, lw=1.5, label=labs[i])
```

```
print (labs[i])
print ('Area under curve',sklearn.metrics.auc(fpr, tpr) )
print ('Accuracy',sklearn.metrics.accuracy_score(ys[i][:,1], [round(j) for
j in new_pred[:,1]]))
print ('confusion matrix \n',sklearn.metrics.confusion_matrix(ys[i][:,1],
[round(j) for j in new_pred[:,1]]))
tn,fp,fn,tp=sklearn.metrics.confusion_matrix(ys[i][:,1], [round(j) for j i
n new_pred[:,1]]).ravel()
print('True negative:',tn,'\nFalse positive:',fp,'\nFalse negative:',fn,'
\nTrue positive:',tp)
plt.xlabel('False Positive Rate'); plt.ylabel('True Positive Rate')
plt.legend(fancybox=True, loc=4, prop={'size':10})
plt.show()
```

```
In [139]: def predic(x):
    if(x[0][1]<x[0][0]):
        print("Accident Occurred")
    else:
        print("Accident not Occurred")
```

```
In [160]: ta=make_dataset(glob.glob('E:/videos/training/000338.mp4'))
```

```
0 E:/videos/training/000338.mp4
```

fprj472

by Final Project

Submission date: 23-May-2020 09:47PM (UTC+0530)

Submission ID: 1330526357

File name: 161472.pdf (1.46M)

Word count: 5673

Character count: 27010

ORIGINALITY REPORT

9%

SIMILARITY INDEX

5%

INTERNET SOURCES

3%

PUBLICATIONS

8%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Jaypee University of Information Technology

Student Paper

4%

2

Submitted to University College London

Student Paper

1%

3

Marios Mourelatos, Christos Alexakos, Thomas Amorgianiotis, Spiridon Likothanassis.

"Financial Indices Modelling and Trading utilizing Deep Learning Techniques: The ATHENS SE FTSE/ASE Large Cap Use Case", 2018 Innovations in Intelligent Systems and Applications (INISTA), 2018

Publication

1%

4

pydeeplearning.weebly.com

Internet Source

<1%

5

krishikosh.egranth.ac.in

Internet Source

<1%

6

aludata.dk

Internet Source

<1%

7	ir.lib.hiroshima-u.ac.jp Internet Source	<1 %
8	engineering.purdue.edu Internet Source	<1 %
9	Submitted to placeholder Student Paper	<1 %
10	Submitted to Vels University Student Paper	<1 %
11	"Contemporary Issues in Soil Mechanics", Springer Science and Business Media LLC, 2019 Publication	<1 %
12	www.cksu.com Internet Source	<1 %
13	Submitted to Vietnam Maritime University Student Paper	<1 %
14	www.cs.bham.ac.uk Internet Source	<1 %
15	Submitted to University of Birmingham Student Paper	<1 %
16	arxiv.org Internet Source	<1 %
17	www.studymode.com Internet Source	<1 %

18

Submitted to Middle East Technical University

Student Paper

<1%

19

Submitted to University of Adelaide

Student Paper

<1%

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

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: Chandrashekhar Choudhary Department: IT Enrolment No 161472

Contact No. 8219853639 E-mail. Karan89009@gmail.com

Name of the Supervisor: Dr. Hemraj Saini


Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): 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 = 56
- Total No. of Preliminary pages = 8
- 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** at9.....(%). 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">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	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