Estimating the Severity of Road Traffic Accidents by Deploying Satellite Images and Deep Learning

A report submitted for the course named Project IV (CS-410)

By

Rahul Ranjan

Bachelor of Technology, VIII Semester

Roll No. 16010121

Under the Supervision and Guidance of

Dr N.Kishorjit Singh

Department of Computer Science and Engineering

Indian Institute of Information Technology Manipur

June 11, 2020

Abstract

Road accidents have severe impact on society which cause lots of fatalities and injuries. Deep learning has great potential in learning patterns and in dealing with real life data particularly related to road traffic accidents. To address this issue we tried estimating the severity of road traffic accidents by deploying code-mixed data (textual data & satellite images) using deep learning. Through this project, we are aimed at improving the predicting capacity of deep learning models to predict the accident severity & locations. Later, visualizations have been used to estimate the root cause of road traffic accidents using code-mixed data. To implement this experiment, dataset was collected from road transport authority of the UK[1] which releases the open sourced data for research and analysis purposes. The open-sourced dataset contains the precise location of each every reported accidents in the form of latitudes and longitudes.The project has been classified into two sections based on types of data used for training the model. The model trained on code-mixed data outperformed the deep learning model implemented using text and image data alone. In this experiment, we have scraped the satellite images using Google Static Map API and haversine formula with grid-based searching approach using latitude and longitude of the reported accidents and the scraped images were classified into two classes safe (0) and danger (1). Further, based on the input images we have implemented various deep learning models such as Convolutional neural network (CNN), Bidirectional long short term memory (Bi-LSTM), Bidirectional GRU (Bi-GRU) and CUDA deep neural network. In order to obtain the state-of-the-art deep learning model, we have made a collation among our implemented models and pre-trained state-of-the-art existing model such as VGG-19. The deep learning models trained on code-mixed dataset outperformed the deep learning models trained on text and image dataset.On code-mixed input dataset, a hybrid architecture of CNN and Bi-LSTM with input-pipeline implemented using multi-layer perceptron (MLP) with 2 dense layers concatenated with CNN with 2 dense layers, 1 convolutional layer obtained

[1]https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data

a training accuracy of 93% and a testing accuracy of 86%. On same dataset, the CNN with 3 convolutional layers and 2 max-pooling layers obtained a training accuracy of 91% with testing accuracy of 89% which outperformed other models. The CNN model with 85% training accuracy and 76% testing accuracy, whereas when the same model was compared with pre-trained VGG-19 architecture, the VGG-19 obtained a training accuracy of 75% and a testing accuracy of 72%. We have obtained encouraging results, which are comparable to those from the state of the art systems.

Keyword - convolution neural network, deep learning, gated recurrent unit, long short term memory, road traffic accidents, satellite images

Declaration

I declare that this submission represents my idea in my own words and where others' idea or words have been included, I have adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/sources in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from proper permission has not been taken when needed.

(Signature)

(Rahul Ranjan)

Date: (16010121)

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

Dr. N.Kishorjit Singh Email: kishorjit@iiitmanipur.ac.in
Assistant Professor

To Whom It May Concern

This is to certify that the report entitled **"Estimating the Severity of Road Traffic Accidents by Deploying Satellite Images and Deep Learning"** submitted to by "Rahul Ranjan", has been carried out under my supervision and that this work has not been submitted elsewhere for a degree, diploma or a course.

Signature of Supervisor

(Dr. N.Kishorjit Singh)

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

Dr. N.Kishorjit Singh
Assistant Professor
Email: kishorjit@iiitmanipur.ac.in

To Whom It May Concern

This is to certify that the report entitled **"Estimating the Severity of Road Traffic Accidents by Deploying Satellite Images and Deep Learning"** submitted to by "Rahul Ranjan", has been carried out under my supervision and that this work has not been submitted elsewhere for a degree, diploma or a course.

Signature of HOD

(Dr. N.Kishorjit Singh)

Signature of Examiner 1: ____________________

Signature of Examiner 2: ____________________

Signature of Examiner 3: ____________________

Signature of Examiner 4: ____________________

Acknowledgement

I extend my sincere thanks to the institute who has provided me a chance to do this project. I would like to express my thanks to Dr N. Kishorjit Singh, project supervisor, for his vital suggestion, meticulous guidance and constant motivation which went a long way in successful completion of the project. I cannot move on without thank to my group mates who has always been an inspiration to me and provide constant guidance throughout this project. I would like to thank Dr Navanath Saharia, Dr Kabita T., Dr Prerna Mohit, and Kaushal Bhardwaj who has been organising the bimonthly presentation time to time which helped me to finish the project on time. On a moral personal note my deepest appreciation and gratitude to my beloved friends, who has been an inspiration and have provided me a unrelenting encouragement and support.

- Rahul Ranjan

Contents

Abstract **ii**

Declaration **iv**

Certificate **v**

Certificate **vi**

Acknowledgement **vii**

Table of contents **viii**

List of tables **xi**

List of figures **xii**

List of abbreviations **xv**

1 Introduction **0**

2 Existing System & Analysis **4**

2.1 Introduction 5

2.2 Related work 5

2.3 System Analysis 7

2.3.1 Software development cycle 7

3 System Design 9

3.1 Introduction 10

3.2 System Architecture 11

3.3 Preparation of Input-pipeline 12

3.3.1 Input-pipeline for code-mixed dataset 16

3.3.2 Convolutional Neural Network (CNN) Architecture 20

3.3.3 Bidirectional-LSTM 22

3.3.4 CUDA Deep Neural Network (CuDNN) library 25

3.3.4.1 Benefits of CuDNN library 25

3.3.5 Bidirectional - Gated Recurrent Unit (Bi- GRU) 26

3.3.6 Deep learning models for code-mixed data 27

3.3.7 System architecture for code-mixed dataset 27

3.3.8 Collation with Pre-trained models on Image dataset 30

3.4 Implementation 30

3.4.1 Convolutional Neural Network (CNN) Architecture 30

3.4.2 Bidirectional-LSTM 35

3.4.3 CUDA Deep Neural Network (CuDNN) library 37

3.4.3.1 Benefits of CuDNN library 37

3.4.4 Collation using Pre-trained models on Image dataset 38

3.4.5 CUDA Deep Neural Network (CuDNN) library 38

3.4.5.1 Benefits of CuDNN library 38

3.4.6 Gated Recurrent Unit (GRU) 38

3.4.7 VGG-19 40

3.4.8 A hybrid CNN-Bi-LSTM architecture 42

4 Results **44**

4.1 Introduction 45

4.2 Exploratory Analysis 45

5 Conclusion & Future Work **57**

Bibliography **59**

List of Tables

4.1 Performance on Logistic Regression and ANN models 51

4.2 Performance on LSTM and GRU models 51

4.3 Performance of CUDA deep neural network LSTM(CuDNNLSTM) and GRU (CuDNNGRU) on Test data 52

4.4 Performance of CNN and VGG-19 on satellite images test data 52

4.5 Performance of Bi-LSTM and a hyubrid CNN + Bi-LSTM architecture on satellite images test data 54

4.6 Performance of a hybrid CNN + Bi-LSTM architecture on **code-mixed input data** 54

List of Figures

2.1 Software development cycle . . . 8

3.1 Architecture of the System . . . 11

3.2 Correlation Matrix for traffic dataset . . . 13

3.3 Distribution of target class . . . 13

3.4 2011 census areas are mostly arranged in hierarchies and sub-divided into LSOAs . . . 14

3.5 2011 Census Geography hierarchy going from Country to Local Authority to Middle Layer Super Output Area (MSOA) to Lower Layer Super Output Area (LSOA) to Output Area . . . 15

3.6 Safe or danger (serious or fatal) zone's satellite images based on **Rural areas** (as per mentioned in input data) . . . 16

3.7 Safe or danger (serious or fatal) zone's satellite images based on **Urban areas** (as per mentioned in input data) . . . 17

3.8 LSOA with longitude and latitude . . . 17

3.9 LSOA and population region-wise . . . 18

3.10 Geopandas x and y points with LSOA and latitude and longitude . . . 18

3.11 Nearest neighbours of LSOA (E01000374) . . . 19

3.12 Array of RGB Matrix . . . 20

3.13 Matrix multiplies Kernel matrix . . . 21

3.14 CNN architecture . . . 21

3.15 CNN architectural / layers design . . . 22

3.16 An image's channel tensor in each layer 23

3.17 LSTM Architecture 23

3.18 The repeating module in an LSTM contains four interacting layers 24

3.19 Forget Layer 24

3.20 Architecture of Bi-LSTM Model with 2 convolutional layers 25

3.21 Gated Recurrent Unit 26

3.22 System architecture for code-mixed dataset 28

3.23 Implemented deep learning model's summary for code-mixed or mixed-input dataset 29

3.24 Array of RGB Matrix 31

3.25 Matrix multiplies Filter Matrix 32

3.26 Matrix multiplies Kernel matrix 32

3.27 CNN Architecture 33

3.28 An image's channel tensor in each layer 34

3.29 LSTM Architecture 35

3.30 The repeating module in an LSTM contains four interacting layers 35

3.31 Forget Layer 36

3.32 Actual internal structure of Bi-LSTM Model with 2 convolutional layers . 37

3.33 VGG-19 Architecture 41

4.1 Road traffic accident density in UK during 2013-2018 46

4.2 The reported accidents location in an LSOA 46

4.3 The reported accidents location in a LSOA is shown using Open Street Map with specific features of the accident 47

4.4 Shows the severity of road accidents of the reported accidents in an LSOA 47

4.5 Represents the relation between speed limit and number of casualties . . 47

4.6 Fatal accidents during 2013-2018 48

4.7 Accident severity by road classification 48

4.8 Accidents severity by road types during 2013-2018 49

4.9 Accidents severity by types of Junction during 2013-2018 49

4.10 Speed limit versus Number of accidents 50

4.11 Speed limit and Number of accidents with reference to severity level . . 50

4.12 Lighting conditions of accidents at different severity 50

4.13 Weather conditions during accidents with different severity level 51

4.14 Training accuracy vs Validation accuracy 52

4.15 Training loss vs Validation loss 53

4.16 Training and validation accuracy for code-mixed input dataset till 11 epochs 55

4.17 Training and validation loss for code-mixed input dataset till 11 epochs 55

4.18 Actual vs predicted value based on location (safe or danger) 56

List of abbreviations

	B
Bi	Bidirectional
	C
CNN	Convolution Neural Network
CuDNN	CUDA deep neural network library
	G
GRU	Gated Recurrent Unit
	L
LSTM	Long Short Term Memory

Chapter 1

Introduction

"An optimist sees an opportunity in every calamity; a pessimist sees a calamity in every opportunity" - Anonymous

Road accidents have extreme effect on society which cause lots of fatalities and wounds. The road accidents are significant well spring of passings, property harm and losses all through the globe consistently. According to National Crime Records Bureau(NCRB) 2016 report communicates that there were 464,674 impacts which caused 148,707 passings in India[1]. It was accounted for a car accident pace of about 0.8 per 1000 vehicles in 2015 contrasted with 0.9 per 1000 vehicles in 2012, and a 11.35 casualty rate for every 100,000 individuals in 2015[2]. Deep learning has extraordinary potential in learning patterns and has incredible potential in managing large dataset identified with road traffic accidents. To address this significant issue we took a look at assessing the severity of road traffic accidents deep into consideration. Through this approach, we additionally attempted to implement an appropriate perception to assess the underlying driver of road accident. To execute this idea, we gathered dataset from road transport authority of UK which discharges the publicly released information for research and examination purposes. Afterward, we saw there were around 37 factors on which the accident was reported and we pictured the dataset to discover the underlying driver of road traffic accidents in the UK and afterward took a stab at anticipating the fatality of the road traffic accidents dependent on different factors. As the dataset, we gathered were exceptionally Imbalanced because of this, the deep learning model was giving inclinations with towards the major class of the model. To tackle this issue we took a stab at inspecting the information before encouraging it to preparing model. Here we defining a road accident as which happened or began on a way or road open to open traffic; brought about at least one people being executed or harmed, and in any event one moving vehicle was included. Single vehicle accident in which one vehicle alone (and no other road client) was included will be incorporated. Multi-vehicle crashes are considered just a single mishap gave that the progressive crashes occurred at very short intervals.

In this experiment, we have made a first attempt to estimate the severity of road accidents into three classes as serious, slight and fatal in case of text-formatted dataset. he fatal road accidents can be characterized as the road accidents which lead until the very end or casualty of individuals and causes a substantial harm and damage and loss of life. The serious accidents are termed as the accidents which cause injury to the people during accidents and leads to hospitalisation of person involved in the accidents and these accidents lead to breaks, blackouts, inward injuries, pounding, extreme cuts and slash, serious general stun requiring therapeutic treatment and some other genuine sores involving detainment in hospital, whereas the slight severity of an accident can be defined as

[1]https://en.wikipedia.org/wiki/Traffic_accidents_in_Indiacite_notencrb20163

[2]https://en.wikipedia.org/wiki/Traffic_accidents_in_India#cite_notencrb20163

the road accidents which cause auxiliary wounds, for example, sprains or wounds. People whining of stun, yet who have not supported different wounds, ought not be considered in the measurements as having been harmed except if they show very clear side effects of stun and have gotten restorative treatment or seemed to require therapeutic consideration. The text-format dataset was highly imbalanced due to presence of less number of fatal and severe accidents in comparison to slight accidents, there were in total 691,641 datapoints were present in the text-format dataset and out of which 578,791 datapoints belonged to slight, 104,597 severe and 8,253 fatal accidents class. To overcome this problem, we converted multi class-classification problem into binary classification by doing this we converted the three classes into minor(1) and major(0) classes in which severe and fatal classes were included in minor(1) class whereas slight was included in major(0) class.

Previously the neural networks have been used for different purposes to save life of mankind such as Human activity recognition using neural networks[1] and made a significant improvement in the data prediction and analysis domain. Indeed, some outstanding standard machine learning algorithms, for example, Support vector machines, Random forests, Hidden markov models, Bayesian networks, Gaussian networks have been applied in well being, catastrophic event, prediction in emergency, and various different spaces. There has been lots of work done in the field of text classification/mining to obtain the state of the art results by training complex neural network architectures. In this experiment, we have used various deep learning approaches which consists of complex-hybrid neural networks architectures such as Artificial neural network, Long short term memory (LSTM), Gated Recurrent Unit (GRU), hybrid CNN with Bi-LSTM/Bi-GRU architectures on image as well as code-mixed dataset and CUDA deep neural network with LSTM/GRU to estimate the severity of the road traffic accidents on text, image and code-mixed dataset. The textual dataset basically consisted of traffic accidents dataset, population dataset and casualities dataset related to road traffic accidents. These datasets provided a point by point information about the conditions of individual damage road accidents in UK from 2013 onwards, the kinds of vehicles included and the considerable setbacks. The measurements relate just to individual damage mishaps on open roads that are accounted to the police, and along these lines recorded, utilizing the STATS19 accidents revealing structure. Data on only damage accidents, with no human losses or accidents on private roads or vehicle leaves are excluded from this information.

The satellite images dataset was scraped using the longitude and latitude column mentioned in the text dataset and the region was constrained with in the range of (52.12, 83.33), i.e., central London zone. To scrap the satellite images, haversine formula was used with grid-search approach. There were in total 5000 each safe (1) and danger (0) class satellite images were downloaded to train the model. Moving a step further, we prepared a repository of code-mixed dataset which is basically a combination of text and image dataset. The text dataset is the structured data on accidents, traffic and population density. By doing this, we are combining structured data and satellite imagery in a mixed-input neural network which is an attempt to improve the model's predictive capability. The multiple inputs and mixed-input data technique was used to prepare the input pipeline for the deep learning models. In this experiment, we have implemented multi-layer perceptron (MLP) to prepare the input pipeline with categorical (structured) and imagery dataset for the models. The MLP layer is implemented on structured data and convolutional layer is implemented on satellite images. The both layers were concatenated to form the deep learning architecture for code-mixed dataset. The bidirectional LSTM was used to classify the safe and danger location using the code-mixed dataset.

Chapter 2

Existing System & Analysis

Outline: This chapter presents the following:

1. Literature survey
2. Summary

2.1 Introduction

There are very few works in road traffic accidents using the techniques of Neural network and its variants and may be even none using deep learning. This project is aimed at building various deep learning models to estimate the severity of road accidents and tried to build models which are very fast, efficient and robust on such a large dataset.

2.2 Related work

In this experiment, an attempt has been made to address the problem of estimating the severity of road traffic accident by deploying code-mixed data using deep learning where the source of dataset is Government of United Kingdom which maintained a repository with 0.7 millions data-points/raw vectors reported on more than 40 factors/features. There are few works done to predict the severity of road traffic accidents using neural network and deep learning but either they are using less data-points or their model is performing poorly. There are few based on Machine learning and some of these are discussed below :

Sharaf Alkheder *et al*[2] based on Severity Prediction of Traffic Accident Using an Artificial Neural Network aimed at eliminating the serious issue of predicting the severity of road traffic accidents using artificial neural network. In their work, they have used 5973 road traffic accidents data recorded during 2008 to 2013 in Abu Dhabi and there were 48 features were given for each accidents and finally they have used 16 features after preprocessing the input data to train the artificial neural network model. Before training their model they splitted training and validating data as 90% and 10% respectively. The model obtained a test accuracy of 74%. The model's performance improved by applying K-means clustering over the data before feeding it to prediction model.

The system proposed by Pantelis Kopelias *et al* [3] which is based on the analysis of Geometric, Operational, and Weather Effects on Crash Number and Severity. They have presented an analysis work which highlights various factors which can be evaluated using both aggregate and non-aggregate data. In their work, Severity was evaluated with constants for the distinctive effect of fatalities, wounds, and material harms. The observed a major outcome from this in depth analysis work that there is an absence of any major correlation and most of the accidents events were largely attributed to driver behaviour and other situations that cannot be accounted for by geometric, operational and other temporary values. Findings of this exploration recommend that fixed and worldly roadway qualities what's more, the nearness of rain or wet asphalts may clarify (and likely add to) about 5% to 10% of the accidents and seriousness of accidents seen in 2004 and 2005 on the Attica Tollway.

There are some major works done in this field to improve the earlier research in this area using multilayer perceptron (MLP) and fuzzy adaptive resonance theory (ART) neural network which was proposed by Miao Chong *et al* [4] was another further step towards the development in this field, their work was one of the important step towards improvement in this area and brought results that helped mankind in making the value of life more precious. Their work was to bring the relationship between driver injury severity and driver, roadway and environment characteristics was examined. They considered the accident data for 1997 for the Central Florida area which covers major areas with respect to traffic data Orange, Osceola, and Seminole Counties. This work was specific to two-vehicles accidents that occurred at signalized intersections. The Multilayer prerceptron (MLP) obtained a training accuracy and a testing accuracy of 65.6% and 60.4% respectively, whereas the ordered logit model was able to perform correctly in case of classifying 58.9% and 57.1% for the training and testing phases, respectively. Results also show that rural intersections are more dangerous than urban intersections in terms of driver injury severity and the speed ratio increases the likelihood of injury severity. The drivers at fault experiences less severe injury than those who are not at fault and it was also found that wearing seat belts reduces the chances of severe injury on a large scale. It was also observed that drivers of passenger cars are likely are more likely to experience more severe injury than drivers of vans or pickup truck so this statement concludes the importance of vehicle type. Finally, impact at the driver side causes severe injury than elsewhere.

The work done by Garrido Rui *et al* [5] was based on using the ordered probit model to evaluate the contribution of features or factors to the injury severity looked by motor vehicle tenants engaged with road accidents. In this worked, they observed that the motor vehicle tenants travelling in light vehicles at two-way roads on a dry surfaces tend to suffer more injuries than those who travels in heavy vehicles at one-way on wet road surfaces. It was also observed that driver seat is the safest seating position, urban areas have less severe accidents than rural areas and finally, women tend to more likely to experience the severe accidents.

The experiment performed by Miao Chong *et al* [6] majorly focused to model the severity of injury resulting from road traffic accidents using decision trees and artificial neural network (ANN). The data was obtained from National Automotive Sampling System (NASS) General Estimates System (GES). As per the model implemented by them that in all the cases the decision tree outperforms the artificial neural network. They also observed that there most important factor that result in fatal injury are : driver's seat belt usage, lighting condition of the roadway and driver's alcohol consumption level. This research work is an step towards accident data-mining. In this research work the decision tree obtained an accuracy of 67.54% for Non injury class, 64% for Possible injury class, 60.37% for Non-incapacitating Injury class 71.38% for Incapacitating Injury and 89.46% for Fatal Injury class, whereas artificial neural network obtained an accuracy of 60.45%, 57.58%, 56.8%, 61.32% and 75.51% respectively for each classes.

2.3 System Analysis

The scope of this experiment is very wide and this can be used for predicting severity of road traffic accidents using code-mixed input data throughout the globe. This experiment is implemented using the dataset obtained from road traffic accidents repository from UK traffic authority and preprocessed properly to obtain the desired result. Later, We have tried to estimate and visualise the various factors behind the road accidents in UK and was able to estimate the severity of road traffic accidents. The project was implemented using various Machine learning and deep learning architectures.

2.3.1 Software development cycle

The iterative model is used during the development of this project, since most of the machine learning projects are iterative in nature; as one progresses in the project, one find oneself iterating on a section until reaching the satisfactory best performing model and thereafter, we go next task to implement and test. Since, we go for feedback and improve the project further more. In above figure 3.1 we have used an iterative model which consists of various software developments cycles such as :

- Initialisation : This section mainly deals with defining of the problem statement and finding some way to solutions. This is one of the most important part of development cycle of any projects.

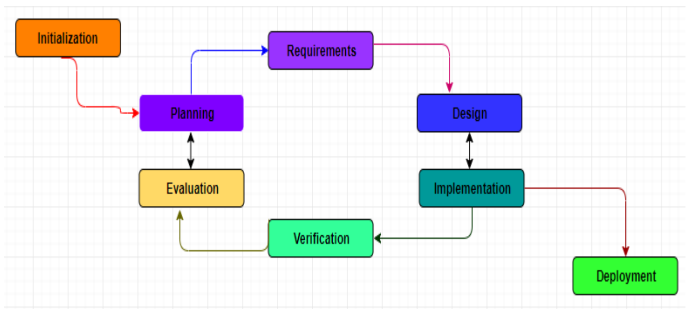


Figure 2.1: Software development cycle

- Planning and Requirements : These two steps are quite similar and are very much related to each other, in this phase we have to map out the specification required, documents needed, lists all software and hardware requirements, etc.
- Analysis and design : The analysis and design phase is achieved in order to list down all business logic, models required, and gathering the dataset to obtain the desired target. In design phase, we list down all the required technical details needed to implement the machine learning and deep learning models.
- Testing : It is the most important phase this is achieved after the design and implementation and it analyses the designed system to find any faults and drawbacks with in the models. If anything found wrong this step helps in reverting all the processes to obtain the desired best results.
- Evaluation : After design, implementation and we go for evaluating our result of the models to analysis the predictions or observations made are correct or not.

The below section consists of constraints regarding software requirements, hardware requirements and dataset used for training the various deep learning models :

- Notebooks - Google colaboratory, Kaggle notebooks and jupyter notebooks were used as the free and open sourced research tool for running the machine learning and deep learning architectures which requires high computational power to use the GPU extensively. To train CUDA deep neural network it is mandatory to have GPU (Nvidia).
- Modules / Libraries : Python2.* or Python3.*, keras, tensorflow, numpy, scikit-learn, Branca, Folium, Seaborn, etc.
- Hardware requirements : Laptop (I3), RAM :4gb, GPU (over cloud) - 1xTesla K80, 12GB VRAM, etc.

Chapter 3

System Design

Outline: This chapter presents the following:

1. Introduction
2. Preparation of Input-pipeline
3. Methodology
4. System Model
5. Summary

3.1 Introduction

The most important part of this project will be designing phase. The designing phase started with collecting the dataset from road accidents dataset repository UK[1]. Later, We scraped the latitude and longitude of the reported acccidents from the dataset using Google Maps Static API and implemented the Haversine formula which is used to get the distance in metres between two points described by latitudes and longitudes. In order to get a set of 'dangerous' and 'safe' areas which do not overlap, a grid system was created and used by splitting Central UK (particularly London) into squares of 0.0005 latitude x 0.0005 longitude (which also allows for easy rounding of locations in the accident dataset). This corresponds to grid squares (technically rectangles) of 56m high (latitude) and 35m wide (longitude), which is similar to the 30m x 30m stated by this paper[7] to be a common area size for traffic accident analysis. The image of grid squares overlaid over a map of traffic accidents in an area of central London - the pattern of these along roads can clearly be seen.

The dataset we collected was having 578,791 slight accidents, 104,597 serious accidents and 8,253 fatal accidents which later merged into as a collection of reported accidents out of which we scraped 5000 dangerous and safe areas satellite images each. The dataset is ready to be visualized and trained for prediction purpose. Before implementing various complex deep learning models we started by implementing Convolution to evaluate the dataset and the ability of a Neural network algorithm to classify safe and dangerous zones on this dataset. Visualization and analysis of various factors which lead to road accidents have already been achieved so this is not converted in this section. Later, various deep learning models has to be implemented. During the design process we will be implementing the following steps:

- Preparation of Input-pipeline
- Scraped the satellite images of reported locations into safe and danger zones using Google Maps Static API.
- Improving the image quality for better estimation.
- Implemented Convolutional neural network architecture (CNN).
- Implemented hybrid architecture of CNN and Bidirectional Long Short Term Memory (Bi-LSTM) similar to [8] in order to implement a better predictive model.

[1]https://data.gov.uk/search?filters%5Btopic%5D=Transport

- Fine-tuned the deep learning architectures based on methods mentioned in [9] to obtain the best performing model on accident image dataset.
- To produce the state-of-the-art result, We are aimed at comparing the obtained test results of this experiment with state-of-the-art pre-trained models such as ImageNet, VGG-16, and VGG-19.

3.2 System Architecture

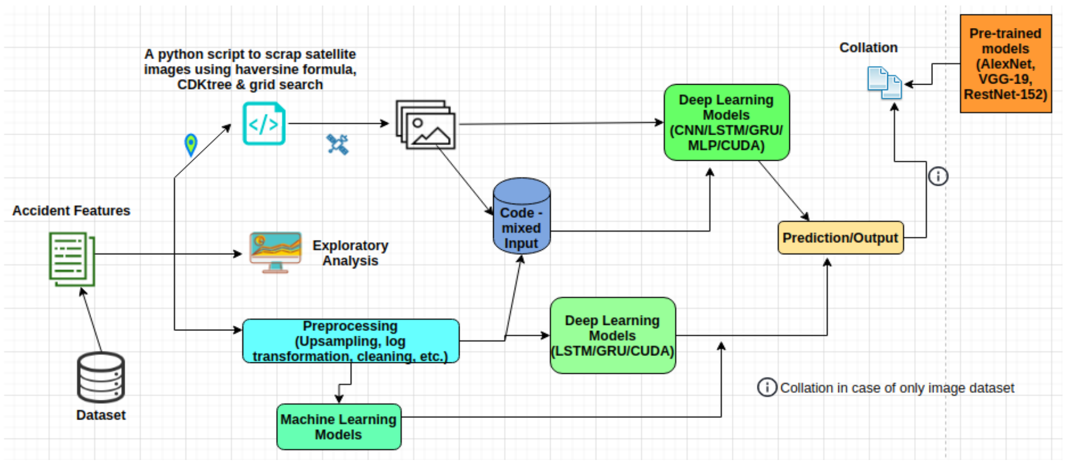

A python script to scrap satellite images using haversine formula, CDKtree & grid search
Collation
Pre-trained models (AlexNet, VGG-19, RestNet-152)
Deep Learning Models (CNN/LSTM/GRU/ MLP/CUDA)
Accident Features
Code - mixed Input
Prediction/Output
Exploratory Analysis
Preprocessing (Upsampling, log transformation, cleaning, etc.)
Deep Learning Models (LSTM/GRU/CUDA)
Collation in case of only image dataset
Dataset
Machine Learning Models

Figure 3.1: Architecture of the System

The figure 3.1 represents the proposed architecture, this is the architecture which is being implemented to classify the safe and danger zones satellite images based on reported road traffic accidents in Central UK. The first step was to collect the dataset from UK road transport authority repository. The second step describes a python script used to scrap the road traffic satellite images under safe and danger areas in the Central UK area particularly London. Preprocessing step is implemented basically to enhance the quality of images so that it can be used for training the deep learning model to produce the best predictive model. The deep learning models we will be implementing such CNN, LSTM/Bi-LSTM/GRU/Bi-GRU with CUDA deep neural network along with a hybrid architecture of CNN and CUDA Bi-LSTM/GRU. Thereafter, the best predictive model will be compared with some of the state-of-the-art pre-trained models on images dataset to evaluate the performance of this proposed system. At the end of the project, We are trying to implement the Web application using TensorflowJS and FLASK so that this system can deployed on mobile devices. Below we have discussed all steps in details.

3.3 Preparation of Input-pipeline

The dataset was collected from road traffic accident UK repository[2]. The dataset had 691,641 data points in total and more than 30 factors using which it was prepared and the dataset contained road accidents information from 2013 to 2017 and there were in total approx 0.65 million road traffic accident cases were reported in that dataset from UK. The dataset was based on three severity level of accidents such as Severe, Slight and Fatal. The dataset was based on three severity level of accidents such as Severe, Slight and Fatal. The dataset was preprocessed based on factors such as Accident_Index, Location_Easting_OSGR, Location_Northing_OSGR, Longitude, Latitude, Accident_Severity, Number_of_Vehicles, Number_of_Casualties, Day_of_Week, Date, Road_Class, Road_Type, Speed_limit, Junction_Type, Light_Conditions, Weather_Conditions, and soon. Our first objective was to find the root cause for road traffic accidents in the UK, to obtain this we did a proper visualization of data points based on each factors. We extracted more important features and dropped irrelevant features such as Police_Force, Local_Auth_(Highway), Road_Number, Pedestrian_Crossing-Human_Control, Pedestrian_Crossing-Physical_Facilities, Did_Police_Officer_Attend _Scene_of_Accident and soon. To extract the relevant features correlation matrix (shown in fig 3.2) was plotted. For estimating the road traffic accident we considered accident severity as the main criteria or label for classification or prediction task.

Initially the dataset consisted of numerical representation and for visualising the dataset we needed to convert it into relevant textual representation based on documentation released by Data regulatory authority UK. The Population dataset based on LSOA to understand the population distribution in the accident prone region to analyse the severity of road accidents happens in both populated or less-populated zone. In population dataset we dropped the irrelevant factors such as date, geography, Rural Urban, Gender, Lives in a household or communal establishment, Schoolchild or full-time student aged 4 and over at their non term-time address.

In order to balance the input data even-upsampling technique was applied based on target class distribution. The figure 3.3 represents the distribution of target class in which it can be easily seen that slight accidents were higher in number than fatal and serious accidents. This imbalanced distribution lead to overfit the machine and deep learning models which basically affected the predictive capacity of the model. This problem was solved by using even-upsampling technique by re-sampling the less frequent class in same amount as the frequent class and by converting the multi-class classification problem

[2]https://data.gov.uk/search?filters%5Btopic%5D=Transport

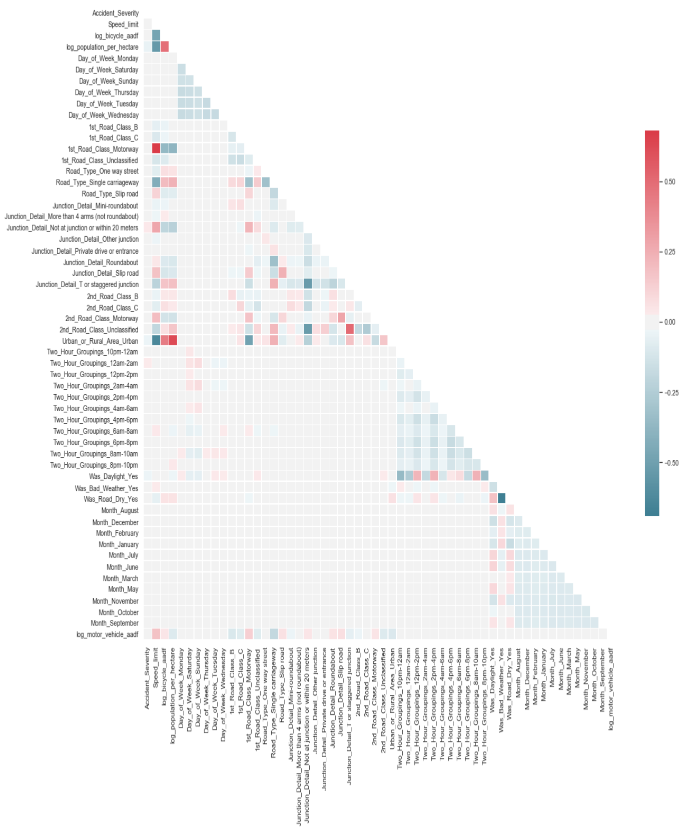


Figure 3.2: Correlation Matrix for traffic dataset

into binary-class classification problem which means converting slight(1), severe(2) and fatal(3) classes into major(0) and minor(1) target classes.

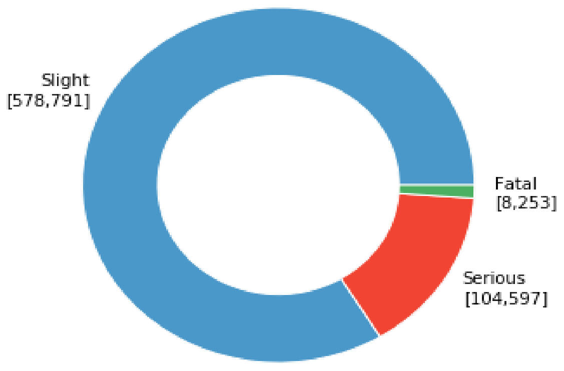


Figure 3.3: Distribution of target class

The reported location's latitude and longitude obtained was used to scrape the satellite images using a python script which uses Google Static API, haversine formula [10], grid square approach and CDKTree (to estimate nearest neighbours). The obtained satellite images were obtained in danger and safe classes based on the fact that the input location co-ordinates were reported any accidents or no accidents were reported. A step ahead, we are focussed on further improving the model by implementing deep learning models on code-mixed input data dataset which consists of text & image dataset. The code-mixed input data was prepared using the A Lower Layer Super Output Area (LSOA) which is a geographic hierarchy to represent the statistics of small areas particulary used in UK. Every reported accidents have specific LSOA based on the region where accident happend due to which same LSOA (in our case a LSOA region coded as E01000001) can be alloted to more than one accidents. The figure 3.4 and 3.5 represents the LSOA area for UK region. The LSOAs were used to map location with other factors in the dataset. The LSOAs were used to align pollution dataset with traffic accidents dataset. Based on number of accidents occured the class labels were assigned to the dataset as danger or safe. The satellite images obtained were aligned with text dataset and code-mixed input dataset was prepared, to train the deep learning models. The comparison of various pre-trained model and our implemented model is done only in case of dataset with satellite images. If a region contains atleast one accident then it was considered as danger zone

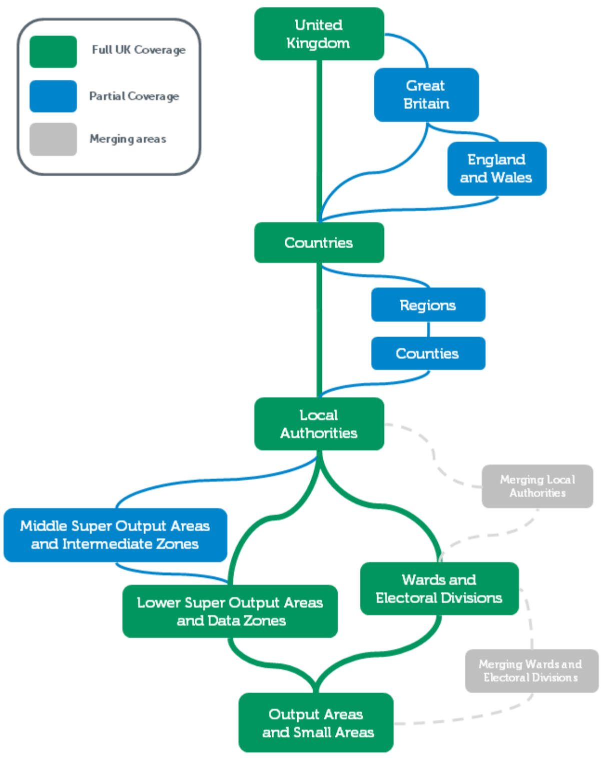

Full UK Coverage
Partial Coverage
Merging areas
United Kingdom
Great Britain
England and Wales
Countries
Regions
Counties
Local Authorities
Merging Local Authorities
Middle Super Output Areas and Intermediate Zones
Lower Super Output Areas and Data Zones
Wards and Electoral Divisions
Merging Wards and Electoral Divisions
Output Areas and Small Areas

Figure 3.4: 2011 census areas are mostly arranged in hierarchies and sub-divided into LSOAs[4]

whereas safe areas can be described as areas containing green areas with no roads or areas were no accidents were reported and also contains sub-urban areas with roads which have similarities with those, where accidents did occur. Safe areas may be difficult to classify by deep learning models but these are important to learn by deep learning models, since

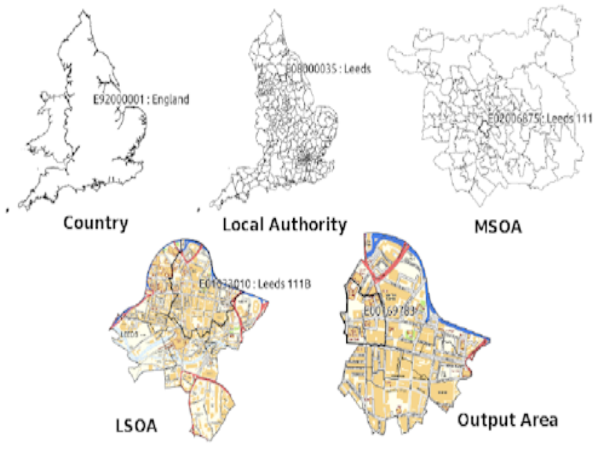
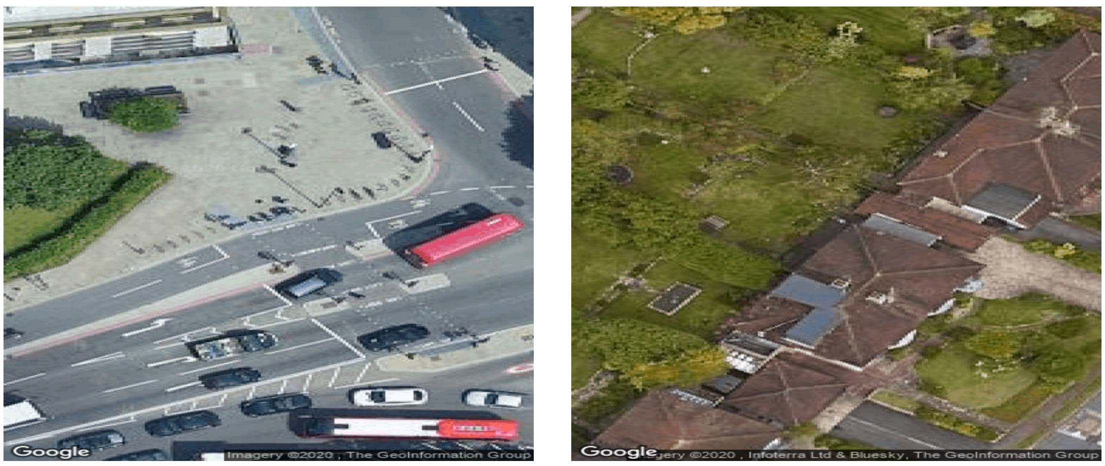


Figure 3.5: 2011 Census Geography hierarchy going from Country to Local Authority to Middle Layer Super Output Area (MSOA) to Lower Layer Super Output Area (LSOA) to Output Area[6]

these images will help the model in learning difference between roads and fields as well as difference types of roads.

In order to get a set of ’dangerous’ and ’safe’ areas which do not overlap, a grid system was created and used by splitting Central UK into squares of 0.0005 latitude x 0.0005 longitude (which also allows for easy rounding of locations in the accident dataset). This corresponds to grid squares (technically rectangles) of 56m high (latitude) and 35m wide (longitude), which is similar to the 30m x 30m to be a common area size for traffic accident analysis. The grid squares was used to download the satellite images. After, scraping the satellite images we then preprocessed it so that proper images are to be provided to the deep learning models in order to implement a highly predictive model. Below figure shows the dangerous and safe zones as:


Google
Imagery ©2020 , The GeoInformation Group
Google
Imagery ©2020 , Infoterra Ltd & Bluesky, The GeoInformation Group

Input Satellite Image of dangerous and safe areas

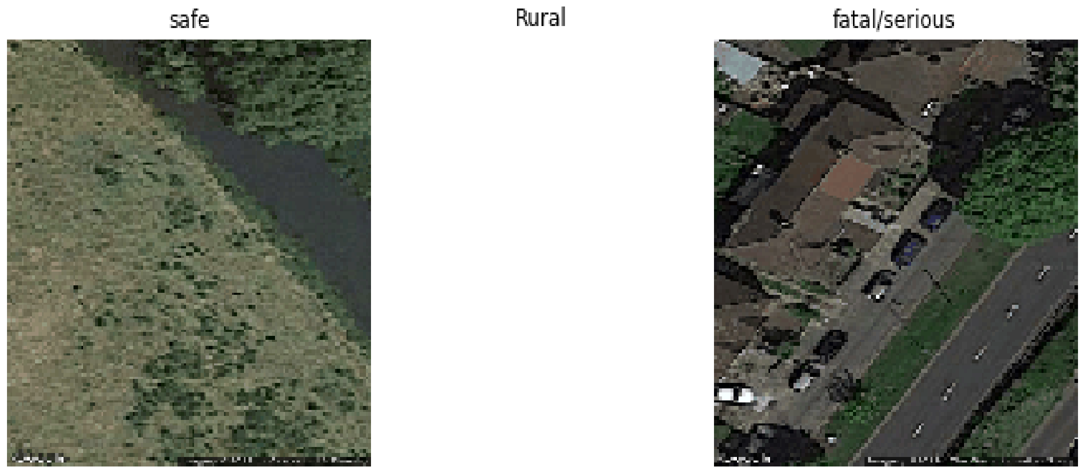


Figure 3.6: Safe or danger (serious or fatal) zone's satellite images based on **Rural areas** (as per mentioned in input data)

Below figure3.6 and 3.7 shows the satellite images of safe and danger (serious / fatal accident zone) location and the zone is classified into two areas Urban and Rural (as per location mentioned in the input dataset):

The preprocessing of input pipeline involves converting images into array of pixel values to fed into the deep learning models such as Convolutional neural network (CNN), Bidirectional long short term memory (Bi-LSTM), Bidirectional gated recurrent unit (Bi-GRU), a hybrid architecture of CNN with Bi-LSTM/Bi-GRU and CUDA deep neural network. For image dataset, the best predictive model is collated [11] with some of the state-of-the-art pre-trained deep neural network models on ImageNet to evaluate this experiment. This one one of the same problem stated by this paper [12] which is basically a problem of satellite images classification.

3.3.1 Input-pipeline for code-mixed dataset

In the step of preparing the input pipeline, we collected the accident structured dataset from UK road transport authority repository as mentioned earlier. The structured dataset consisted of more than 30 features such as Accident_Index, Longitude, Latitude, Accident_Severity, Number_of_Vehicles, Number_of_Casualties, Date, Day_of_Week, 1st_Road_Class, etc. To extract the relevant feature vectors from the input structured dataset we implemented the correlation matrix and scrapped the latitude and longitude

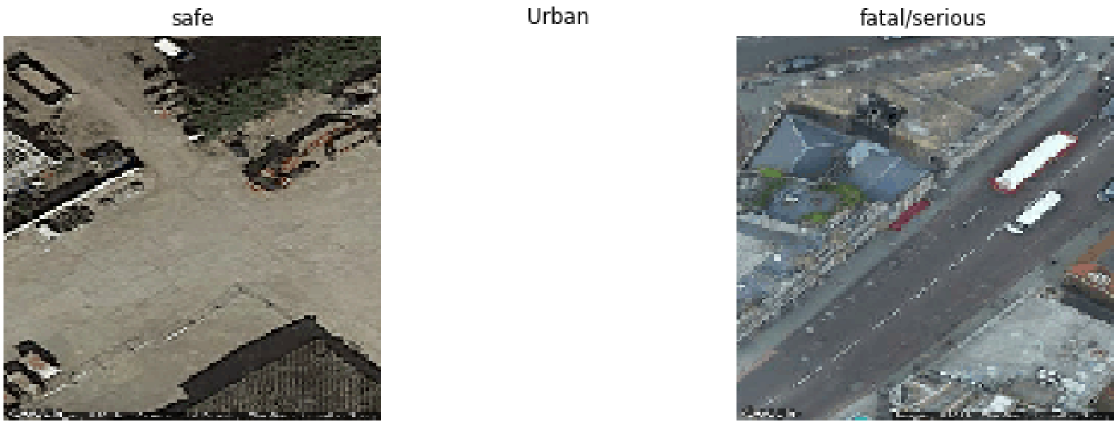
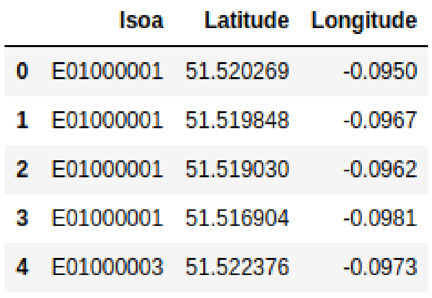


Figure 3.7: Safe or danger (serious or fatal) zone's satellite images based on **Urban areas** (as per mentioned in input data)

from the dataset with their respective LSOA regions as shown in following figure 3.8: On

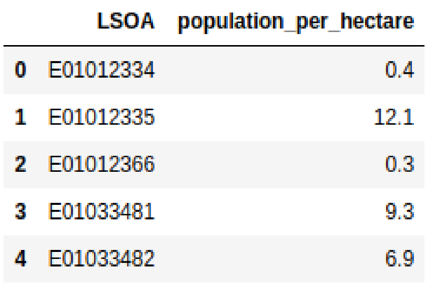


	lsoa	Latitude	Longitude
0	E01000001	51.520269	-0.0950
1	E01000001	51.519848	-0.0967
2	E01000001	51.519030	-0.0962
3	E01000001	51.516904	-0.0981
4	E01000003	51.522376	-0.0973

Figure 3.8: LSOA with longitude and latitude

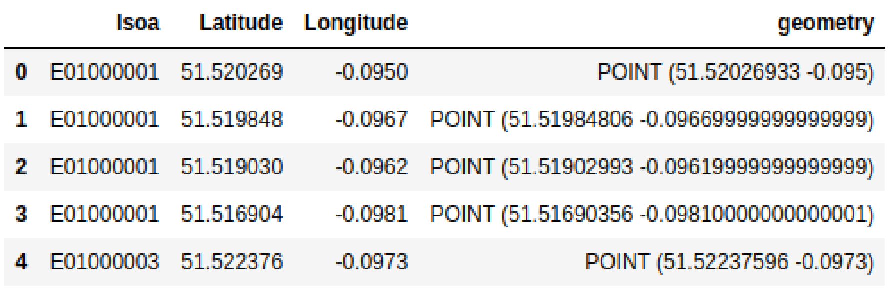
the basis of LSOA, we collected the information of the population density within a particular region as shown in figure 3.9: Later, we took accident traffic dataset into consideration and dropped irrelevant feature vectors such bicycle_aadf, motorbike_aadf, car_aadf, bus_aadf and merged feature vectors such as CP, S_Ref_Latitude, S_Ref_Longitude, bicycle_aadf, motor_vehicle_aadf this with the population dataset based on LSOA of each region.

Added rounded latitiude and longitude columns and a grid square column to the road accident dataset and considered in total 20000 samples from the entire dataset by dropping off duplicate grid_square values. Based on the accident severity or frequence of accidents, we splitted the dataset into safe and danger labels as discussed earlier. To obtain the satellite images based on latitude and longitude, we bounded the geographical range as



	LSOA	population_per_hectare
0	E01012334	0.4
1	E01012335	12.1
2	E01012366	0.3
3	E01033481	9.3
4	E01033482	6.9

Figure 3.9: LSOA and population region-wise

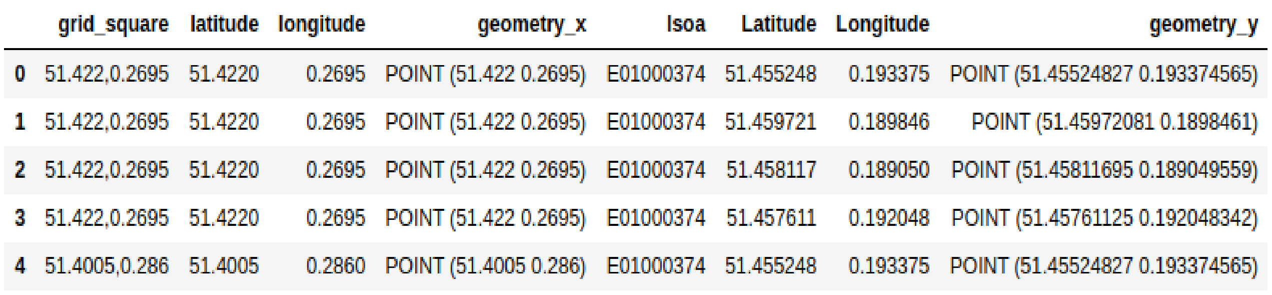


	lsoa	Latitude	Longitude	geometry
0	E01000001	51.520269	-0.0950	POINT (51.52026933 -0.095)
1	E01000001	51.519848	-0.0967	POINT (51.51984806 -0.09669999999999999)
2	E01000001	51.519030	-0.0962	POINT (51.51902993 -0.09619999999999999)
3	E01000001	51.516904	-0.0981	POINT (51.51690356 -0.09810000000000001)
4	E01000003	51.522376	-0.0973	POINT (51.52237596 -0.0973)

Figure 3.10: Geopandas x and y points with LSOA and latitude and longitude

lat_min, lat_max = 51.257, 51.719 ; lon_min, lon_max = -0.542, 0.291 and grid_size = 0.0005. This helped in scarping the satellite images from the central zone with better feature representation and less noise in the training dataset. The location was splitted into danger_squares and safe_squares zone. To find the correct population density for each row in the safe_squares zone the nearest LSOA was necessary to match on the basis of latitude and longitude. This was achieved by finding nearest neighbour LSOA center point to each safe_square location. We used the Geopandas GeoDataFrames to fasten the process of matching each nearest neighbours of the LSOA. The Geopandas GeoDataFrames converts latitude and longitude to Geopandas x and y points. The cKDTree function uses these x and y points to further map the different regions as shown in figure 3.10:

Based on the LSOA regions the nearest neighbours were merged together using the geopandas x and y points with latitude and longitude and cKDTree function by adding the nearest neighbours for each safe_square point as shown in below figures **??** : We further added the population density to this dataset based on the LSOA e.g. -E01000374 and obtained the similar dataset for danger_square locations and appended the safe_square locations to it in order to make a complete single dataset for training the deep learning models. The safe locations were labelled as 1 whereas danger locations were labelled as 0. The list of grid_squares from the safe and danger sets were used to



	grid_square	latitude	longitude	geometry_x	lsoa	Latitude	Longitude	geometry_y
0	51.422,0.2695	51.4220	0.2695	POINT (51.422 0.2695)	E01000374	51.455248	0.193375	POINT (51.45524827 0.193374565)
1	51.422,0.2695	51.4220	0.2695	POINT (51.422 0.2695)	E01000374	51.459721	0.189846	POINT (51.45972081 0.1898461)
2	51.422,0.2695	51.4220	0.2695	POINT (51.422 0.2695)	E01000374	51.458117	0.189050	POINT (51.45811695 0.189049559)
3	51.422,0.2695	51.4220	0.2695	POINT (51.422 0.2695)	E01000374	51.457611	0.192048	POINT (51.45761125 0.192048342)
4	51.4005,0.286	51.4005	0.2860	POINT (51.4005 0.286)	E01000374	51.455248	0.193375	POINT (51.45524827 0.193374565)

Figure 3.11: Nearest neighbours of LSOA (E01000374)

download images from the Google Static Maps API as did earlier.

In previous version of this experiment we already implemented machine learning algorithm and various deep learning algorithms on structured or text dataset, such as :

- Logistic Regression
- Random Forest
- Artificial Neural Network (ANN)
- LSTM (Long Short Term Memory) and Bidirectional (Bi) LSTM
- GRU (Gated Recurrent Unit) & Bi-GRU
- Bi-LSTM/GRU with CUDA deep neural network

Now, We have implemented various deep learning architectures on Image and Code-mixed dataset in this section of the project entitled Estimating the Severity of Road Traffic Accidents byDeploying Satellite Images and Deep Learning for example implemented:

- Convolutional Neural Network (CNN)
- Pre-trained architectures VGG-16 and VGG-19
- A hybrid CNN - Bidirectional Long short term memory (CNN-BiLSTM) & CNN - Bidirectional Gated Recurrent Unit (CNN-Bi-GRU) with CUDA deep neural network.
- Multi-layers Perceptron (MLP)

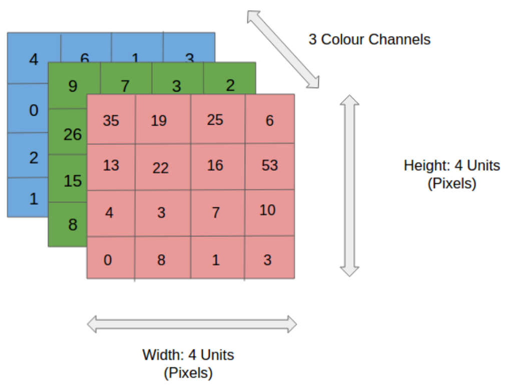


Figure 3.12: Array of RGB Matrix[7]

- A CNN-Bi-LSTM model trained by concatenated output from MLP and CNN model.

3.3.2 Convolutional Neural Network (CNN) Architecture

In neural networks, CNN is one of main architecture to do image classifications,object detection and other classifications task.
CNN classifications takes vectors as input, process it and does the classification task under certain categories Eg., An image in figure 4.17 of $4 \times 4 \times 3$ array of matrix of RGB (3 refers to RGB values) : Training deep learning models like CNN, they need some parameters to be passed such as filters, kernels, pooling layers and then the fully connected layers, after that the flatten layer is used to convert the matrix vector into stack of vectors. Then the output from flatten layer is feeded to the fully connected dense layers where the softmax layer is applied to predict the desired output.

We did the zero padding before fed the input to the convolution as it is done not to lose any useful information from the data when pooling layer is to be applied as it reduces the dimensionality of the data but extracts the useful information. Then we fed the output to the first convolutional layer which has the ReLU activation function. The output from this layer is fed to the next pooling layer which further reduced the dimensions of the input data and kept the information still in the network and finally the output from the convolutional layer is fed to the flatten layer. Below figure represents how convolution applies on the input data. Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below figure 3.13.

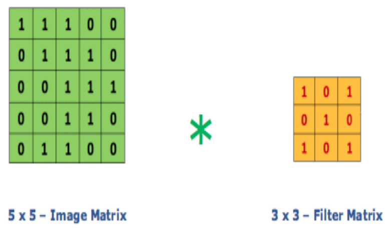


Figure 3.13: Matrix multiplies Kernel matrix[8]

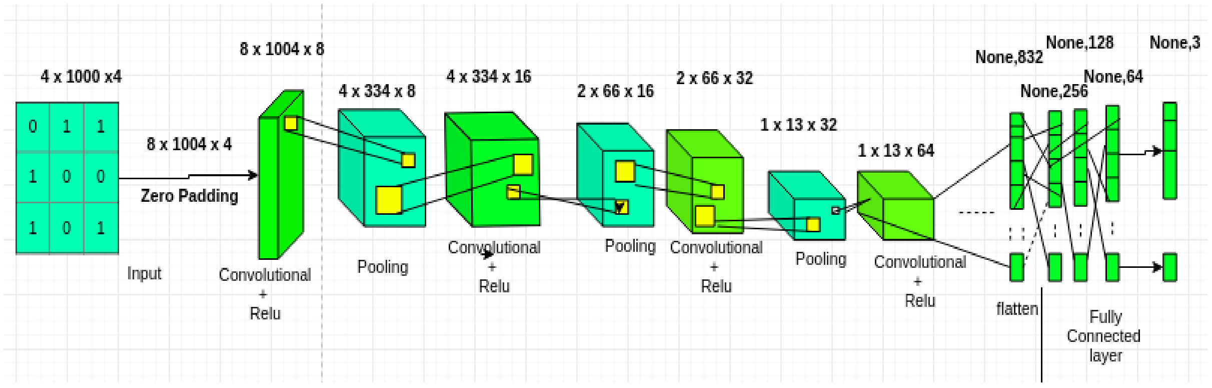


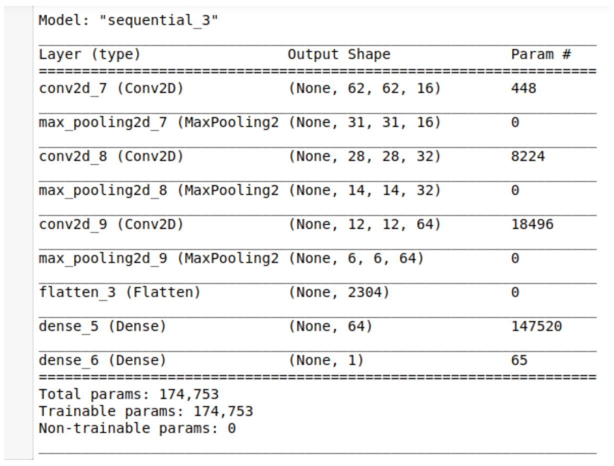
Figure 3.14: CNN architecture

Convolutions of an image with different filters are useful and helps in performing operations such as edge detection, blur and sharpen by applying filters, strides and all these parameters. Strides 1 means the filter moves by window size of 1 and when 2 then filter moves by window size of 2.
Pooling is basically used to reduce the number of parameters or dimensionality and in our system, it is max pooling which we used for down sampling of the input data and helpful when data is too large.

- Max Pooling

Max pooling takes the largest element from the rectified feature map. Convolutional neural network (CNN) with 4 convolutional layers & as shown in figure 3.14 was initially implemented. The CNN architecture used in this experiment has the following layer or architectural design has shown in figure 3.15 : The CNN model implemented in this experiment consists of following architectural parameters :

- The input to the model was $64 \times 64 \times 3$ channel tensor
- 3 Convolutional layers



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 62, 62, 16)        448
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 31, 31, 16)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 32)        8224
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                147520
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
Total params: 174,753
Trainable params: 174,753
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.15: CNN architectural / layers design

- 3 Max-Pooling layers
- Additionally, a flatten layer and 2 dense layers (including output layer)
- Optimizer : Adam
- Loss : binary_crossentropy
- Batch_size : 32
- Epoch : 11
- Train_sample : 6000 samples ; Val_sample : 2000 samples ; Test_samples : 2000 samples
- There were in total 174,753 params out of all of them were trainable params
- EarlyStopping, ModelCheckpoint, ReduceLROnPlateau were used to prevent the CNN model from overfiting and remembering the data-points.

The channel tensor of a particular test image for each channel in each layer is shown in figure 3.16:

3.3.3 Bidirectional-LSTM

LSTMs commonly called as Long Short Term Memory, LSTM is an improvement over RNN and it has capability to retain informations which it already learned and LSTM also supports long term dependencies. In this project we are aimed at implementing the

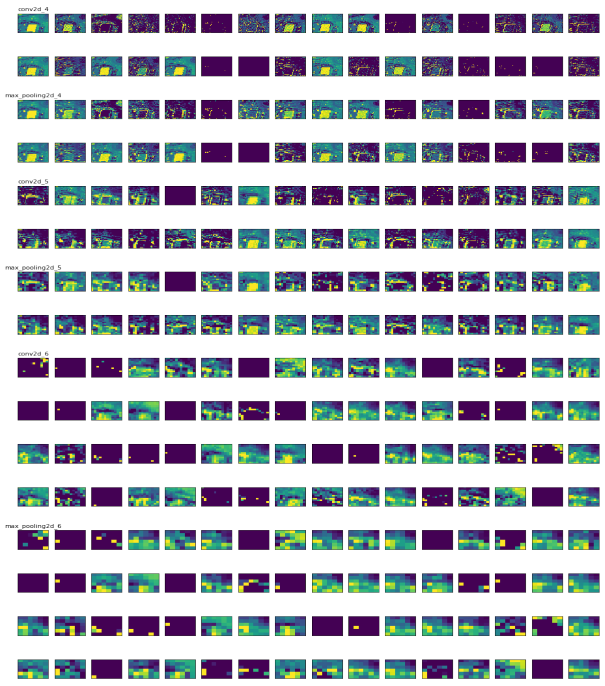


Figure 3.16: An image's channel tensor in each layer

Bidirectional LSTM with CUDA deep neural network library which remembers information in both direction forward and backward to predict the output. Below figure 3.17 and 3.19 shows the LSTM architecture : Let us discuss the LSTM architecture in brief : The figure 3.19 represents the LSTM network which is responsible for forgetting the irrelevant information and decided by sigmoid layer. To forget information gate multiplies input by 0 and output returned by forget layer is 0. If the layer finds some useful information then it multiplies the input by 1 and keeps the complete information. This below formula is used for the calculating the forget layer:

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{3.1}$$

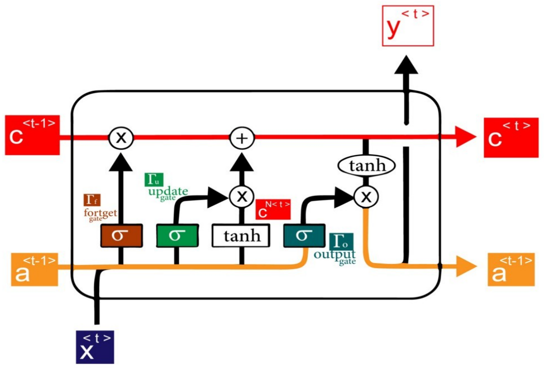


Figure 3.17: LSTM Architecture[9]

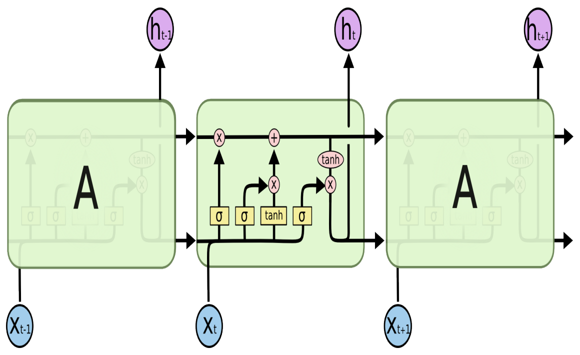


Figure 3.18: The repeating module in an LSTM contains four interacting layers[10]

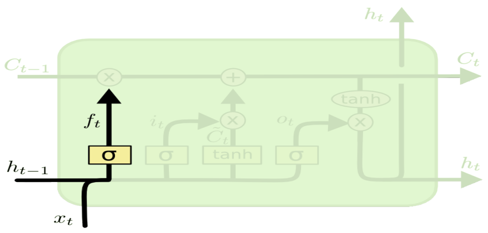


Figure 3.19: Forget Layer[11]

The following formula is used to store information in the cell state :

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{3.2}$$

and

$$candidatevalue, (C_t) = \sigma(W_C \times [h_{t-1}, x_t] + b_C) \tag{3.3}$$

The cell state is updated from C_{t-1}, into the new cell state C_t with below formula

$$C_t = f_t \times C_{t-1} + i_t \times C_t \tag{3.4}$$

We have to decide what we want as the output or how the output is to be designed but the output is the filtered output C_t. Following formula will be used to filter the output :

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.5}$$

and,

$$h_t = o_t \times tanh(c_t) \tag{3.6}$$

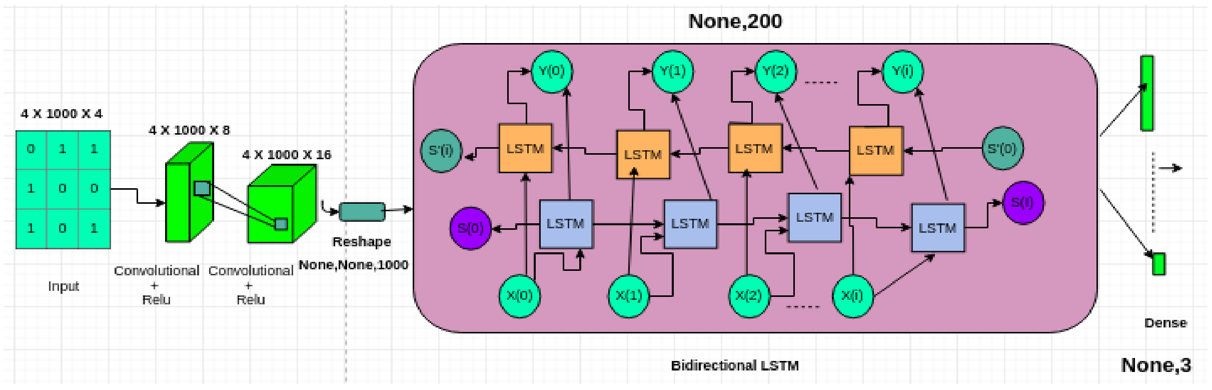


Figure 3.20: Architecture of Bi-LSTM Model with 2 convolutional layers

The hybrid architecture of the CNN and Bi-LSTM was implemented in the similar manner. Below, We have shown the actual architecture of the Bi-LSTM model with 2 convolutional layers as shown in the figure 3.20.

3.3.4 CUDA Deep Neural Network (CuDNN) library

The NVIDIA's CUDA Deep Neural Network library (CuDNN) is a GPU-accelerated and highly robust library that improves the performance of the model and it can substantially speed up training of the model. It allows neural network researchers to focus on training neural networks and allow them enough time and resources to work and develop something innovative rather than spending time on low-level GPU performance tuning.

3.3.4.1 Benefits of CuDNN library

There are following benefits of using CUDA deep learning library apart from improving the GPU Performance

- It applies convolution in forward and backward direction.
- It also applies Pooling, Softmax and activation functions in forward and backward direction such as Sigmoid and other layers.

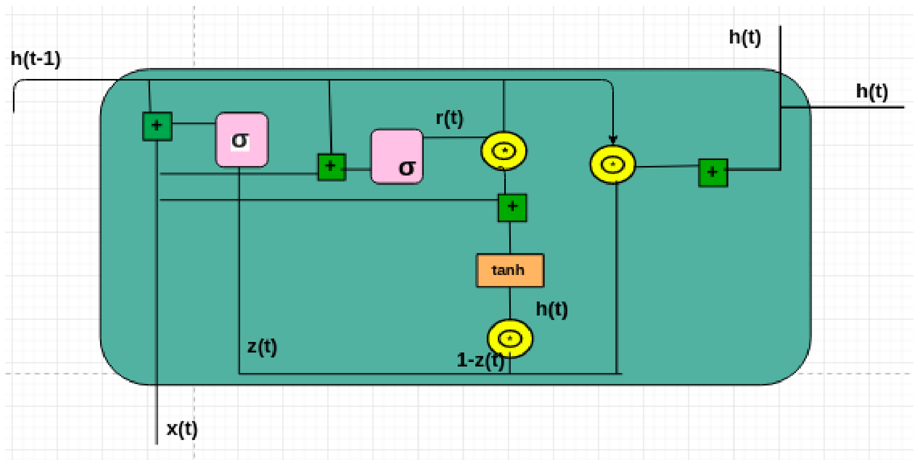


Figure 3.21: Gated Recurrent Unit

3.3.5 Bidirectional - Gated Recurrent Unit (Bi- GRU)

Before going into Bi-GRU, let us discuss GRU which stands for Gated Recurrent Unit and it is an improved version of RNN and it also solves the issue of Vanishing gradient problem. It has 2 gates, one is update gate and other is reset gate. These two gates are responsible for which information to keep and which one to discard respectively. Below is the GRU architecture as shown in figure 3.21 :

where,
$\sigma - Sigmoidfunction$
$+ - plusoperation$
⊙ - Hadamard product operation
tanh - tanh function

Update gate (z_t) is calculated by using following formula:

$$z_t = \sigma(W^(z)x_t + U(z)h_{t-1}) \tag{3.7}$$

When x_t is provided as current input, it is multiplied by its own weight W(z). The same goes for h_{t-1} which holds the information for the previous t-1 units and is multiplied by its own weight U(z). Then the sigmoid layer is applied over the output of this and stored as update gate output.

Reset gate (r_t) decides the information to be discarded and can be calculated using this below mentioned formula -

$$r_t = \sigma(W^(r)x_t + U(r)h_{t-1}) \tag{3.8}$$

Now it is time to design a network in GRU to keep the information to be feeded in next layer and it is calculated as follows:

$$h_t = tanh(Wx_t + r_t \bigodot Uh_{t-1}) \tag{3.9}$$

$$h_t = z_t \bigodot h_{t-1} + (1 - z_t) \bigodot h_t \tag{3.10}$$

Deep learning models were fine-tuned to obtain a better predictive model.

3.3.6 Deep learning models for code-mixed data

The code-mixed dataset consisted of text as well as image dataset. The LSOA with latitudes and longitudes were used to align the image dataset with text data of the road traffic accidents using cDKTree. Earlier, we tried to improve the predictive capability of the model using structured data on accidents, traffic, population density and with satellite imagery to train the model to predict whether a location would be a traffic accident hot-spot or not.
In this part of the experiment, we made an attempt to combine satellite imagery with structured data to form a mixed-input neural network in order to improve the model's predictive capability.

3.3.7 System architecture for code-mixed dataset

In this section, we implemented various deep learning algorithms such as Multi-layer perceptron (MLP), Convolutional neural network (CNN), and Bi-LSTM to improve the predictive capability of the deep learning models on code-mixed dataset. The following system architecture has been used for implementing deep learning models on code-mixed dataset of structured and images data as shown in below figure : The structured data was sorted in the same order as images based on the labels of the images which was latitude and longitude of the image's or accident location. We implemented a function to preprocess the structured dataset and convert the categorical data into vector rep-

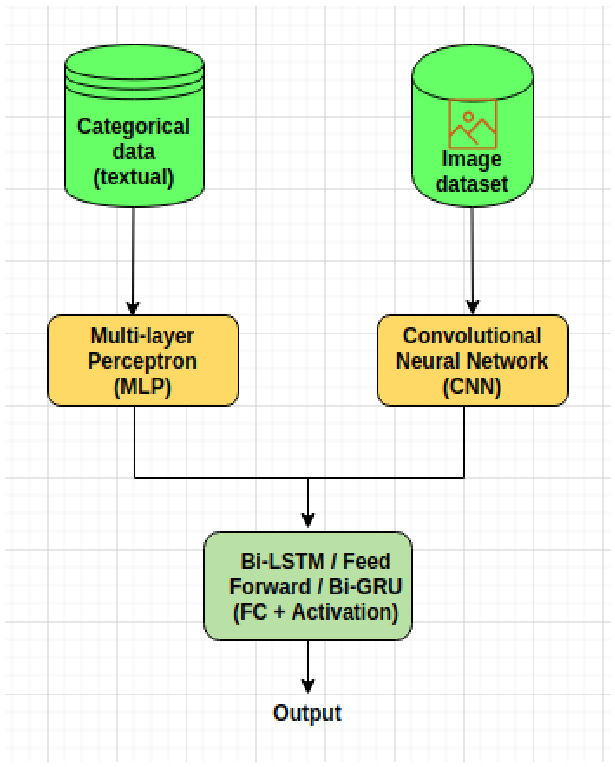
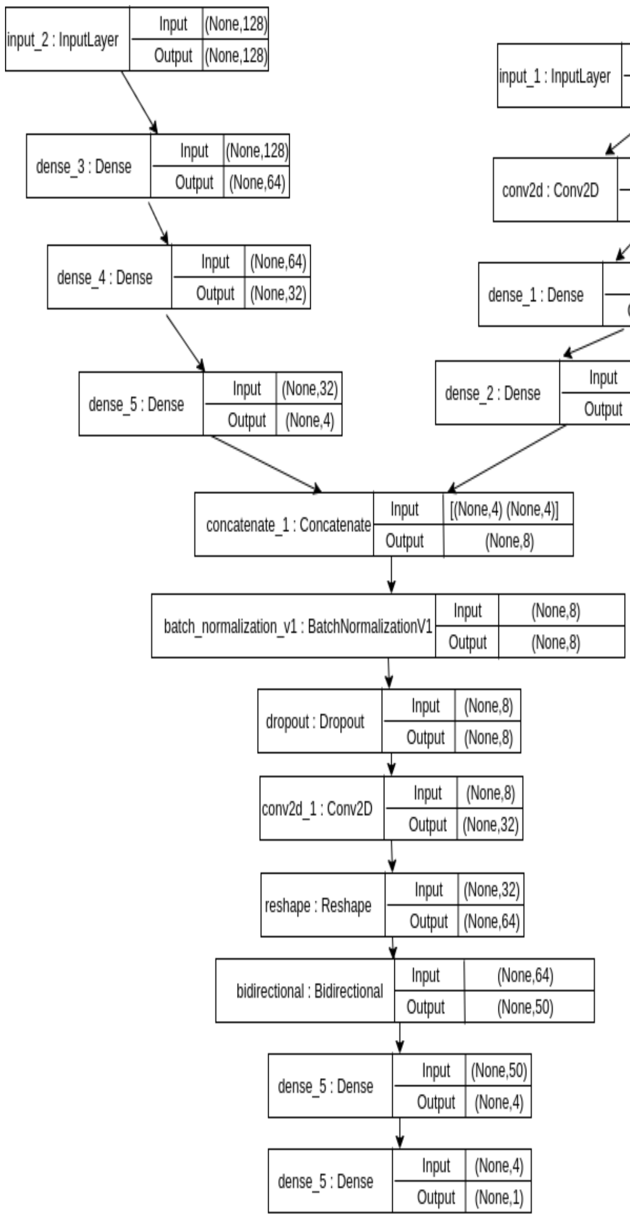

Categorical data (textual)
Image dataset
Multi-layer Perceptron (MLP)
Convolutional Neural Network (CNN)
Bi-LSTM / Feed Forward / Bi-GRU (FC + Activation)
Output

Figure 3.22: System architecture for code-mixed dataset

resentation before fed it to the MLP layers and splitted the structured data into test and train using train_test_split function. After preprocessing the structured and image dataset, we implemented a 3 dense layers multi-layer perceptron (MLP) model with relu-activation function. For the image dataset, we created a CNN model followed by fully connected layer (batch normalisation, dropout) with relu activation function and 2 dense layers. Initialised the input shape and channel dimension where the channel dimension was number of channels in last dimension. Further, we combined the output of MLP and CNN models by concatenating it and used the concatenated output from both models as input to the Bi-LSTM model. The MLP model operates on the textual or structured dataset and CNN model operates on the image dataset and the combination of output of both the layers is fed into the third model which is Bi-LSTM in out case. The output of MLP and CNN are of 4-dim vectors similarly on concatenated output is the 8-dim vector. The below figure 3.23 shows the architectural design of the proposed deep neural network architecture for code-mixed input data as :



input_2 : InputLayer | Input (None,128) | Output (None,128)
dense_3 : Dense | Input (None,128) | Output (None,64)
dense_4 : Dense | Input (None,64) | Output (None,32)
dense_5 : Dense | Input (None,32) | Output (None,4)
input_1 : InputLayer | Input (None,32) | Output (None,32)
conv2d : Conv2D | Input (None,32) | Output (None,64)
dense_1 : Dense | Input (None,64) | Output (None,8)
dense_2 : Dense | Input (None,8) | Output (None,4)
concatenate_1 : Concatenate | Input [(None,4) (None,4)] | Output (None,8)
batch_normalization_v1 : BatchNormalizationV1 | Input (None,8) | Output (None,8)
dropout : Dropout | Input (None,8) | Output (None,8)
conv2d_1 : Conv2D | Input (None,8) | Output (None,32)
reshape : Reshape | Input (None,32) | Output (None,64)
bidirectional : Bidirectional | Input (None,64) | Output (None,50)
dense_5 : Dense | Input (None,50) | Output (None,4)
dense_5 : Dense | Input (None,4) | Output (None,1)

Figure 3.23: Implemented deep learning model's summary for code-mixed or mixed-input dataset

3.3.8 Collation with Pre-trained models on Image dataset

The output of the best proposed deep learning model is to be collated with the state-of-the-art CNN models pre-trained models on Image dataset such VGG-16, VGG-19 and ImageNet. These models is to be imported directly using the library available to import these models to load the pre-trained weights and errors. The same test data which is to be used to find the performance in this experiment is to be used to find the performance on pre-trained model and in this way the our implemented model's performance is to be evaluated.

3.4 Implementation

The Implementation section will consist of various deep learning architectures. We will basically start with CNN and fine-tunned it properly to obtained a most accurate model but We will also try to implement other models to evaluate the performance of this rea-life problem to find the most accurate and efficient model. To achieve this target We are aimed at implementing Bi-LSTM/LSTM,Bi-GRU/GRU with CUDA deep neural network architecture (an nvidia's library) with LSTM or GRU. Let's see implementation of each architectures separately:

3.4.1 Convolutional Neural Network (CNN) Architecture

In neural networks, CNN is one of main architecture to do image classifications,object detection and other classifications task.

CNN classifications takes vectors as input, process it and does the classification task under certain categories Eg., An image in figure 3.24 of $4 \times 4 \times 3$ array of matrix of RGB (3 refers to RGB values) as shown in figure 3.24

Training deep learning models like CNN, they need some parameters to be passed such as filters, kernels, pooling layers and then the fully connected layers, after that the flatten layer is used to convert the matrix vector into stack of vectors. Then the output from flatten layer is feeded to the fully connected dense layers where the softmax layer is applied to predict the desired output. We did the zero padding before fed the input to the convolution as it is done not to lose any useful information from the data when pooling

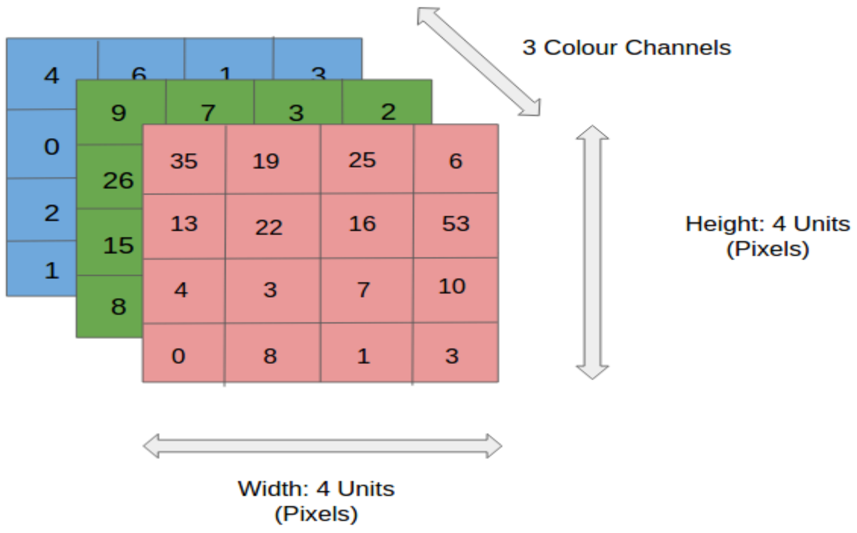
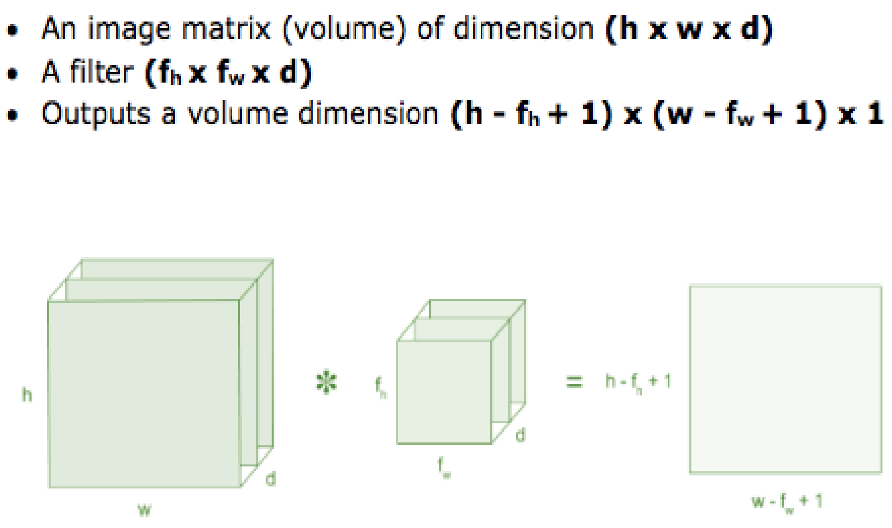


Figure 3.24: Array of RGB Matrix[12]

layer is to be applied as it reduces the dimensionality of the data but extracts the useful information. Thereafter we fed the output to the first convolutional layer which has the ReLU activation function. The output from this layer is fed to the next pooling layer which further reduced the dimensions of the input data and kept the information still in the network and finally the output from the convolutional layer is fed to the flatten layer. Below figure represents how convolution applies on the input data as shown in figure 3.25 . Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below figure 3.26: Convolutions of an image with different filters are useful and helps in performing operations such as edge detection, blur and sharpen by applying filters, strides and all these parameters. Strides 1 means the filter moves by window size of 1 and when 2 then filter moves by window size of 2. Pooling is basically used to reduce the number of parameters or dimensionality and in our system, it is max pooling which we used for down sampling of the input data and helpful when data is too large.

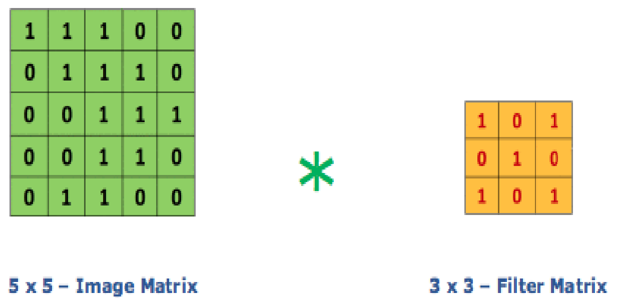
- Max Pooling

Max pooling takes the largest element from the rectified feature map. In this step, We have implemented the various deep learning architectures on the code-mixed dataset (text as well as image) obtained from UK road traffic accident repository. As discussed in earlier section, We extracted the latitude and longitude from the textual dataset and used it to obtain the satellite images into two classes ,i.e., danger and safe from Google Static API. The satellite images was consisted of locations of Central UK.



- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **($h - f_h + 1$) x ($w - f_w + 1$) x 1**

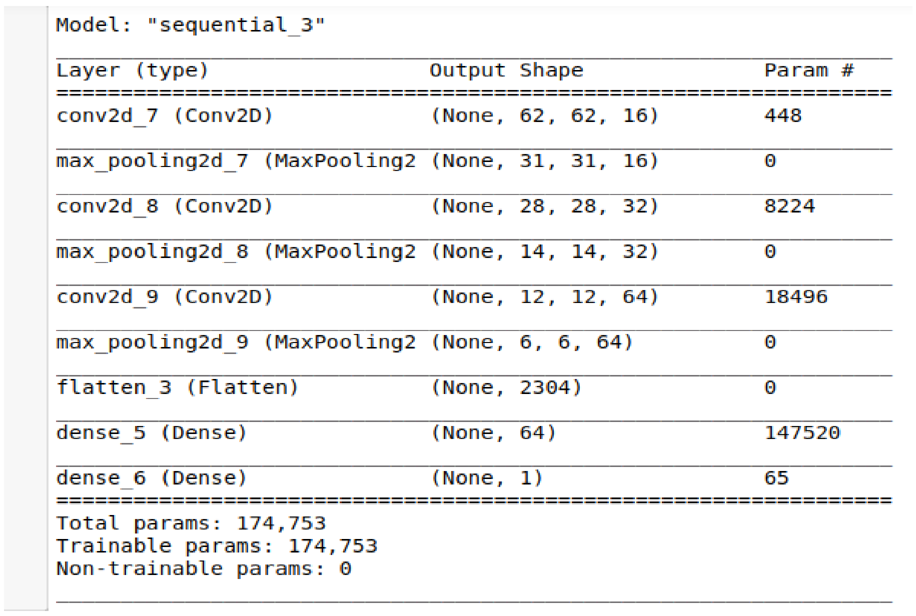
Figure 3.25: Matrix multiplies Filter Matrix[13]



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Figure 3.26: Matrix multiplies Kernel matrix[14]



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_7 (Conv2D)            (None, 62, 62, 16)        448       
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 31, 31, 16)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 32)        8224      
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 64)        18496     
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_5 (Dense)              (None, 64)                147520    
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65        
=================================================================
Total params: 174,753
Trainable params: 174,753
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.27: CNN Architecture

The Convolutional neural network (CNN) is one of the important deep learning architecture in neural network used for image classification and other classification objectives. CNN architecture has been discussed earlier in this report in section 2.5.1 and below diagram 3.27 shows the CNN architecture used in this project : The CNN model implemented in this project consisted of following architectural design :

- The input to the model was $64 \times 64 \times 3$ channel tensor
- 3 Convolutional layers
- 3 Max-Pooling layers
- Additionally, a flatten layer and 2 dense layers (including output layer)
- Optimizer : Adam
- Loss : binary_crossentropy
- Batch_size : 32
- Epoch : 11
- Train_sample : 6000 samples ; Val_sample : 2000 samples ; Test_samples : 2000 samples
- There were in total 174,753 params out of all of them were trainable params

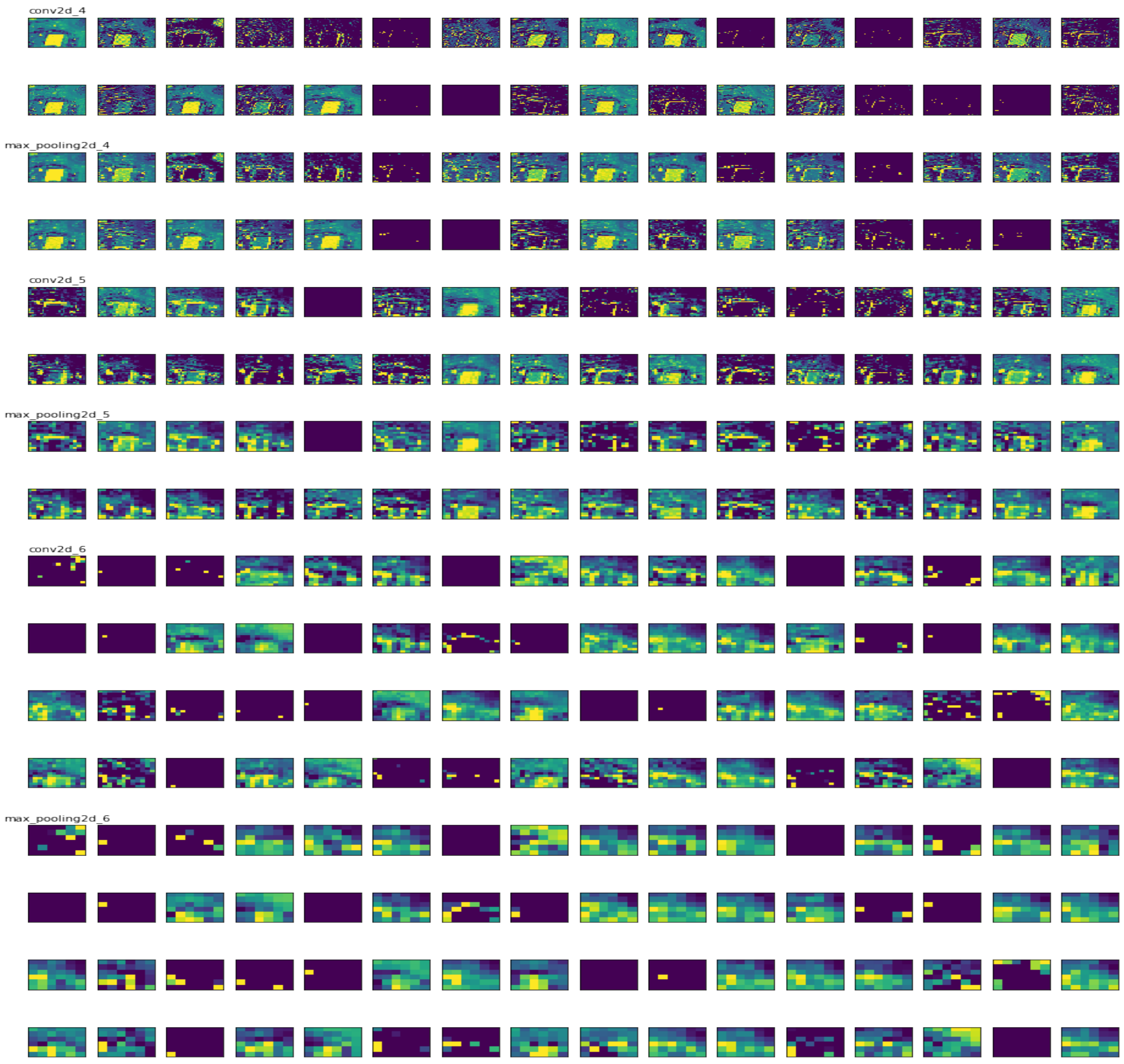


Figure 3.28: An image's channel tensor in each layer

- EarlyStopping, ModelCheckpoint, ReduceLROnPlateau were used to prevent the CNN model from overfitting and remenbering the data-points.

The channel tensor a particular test image for each channel in each layer as shown in figure 3.28:

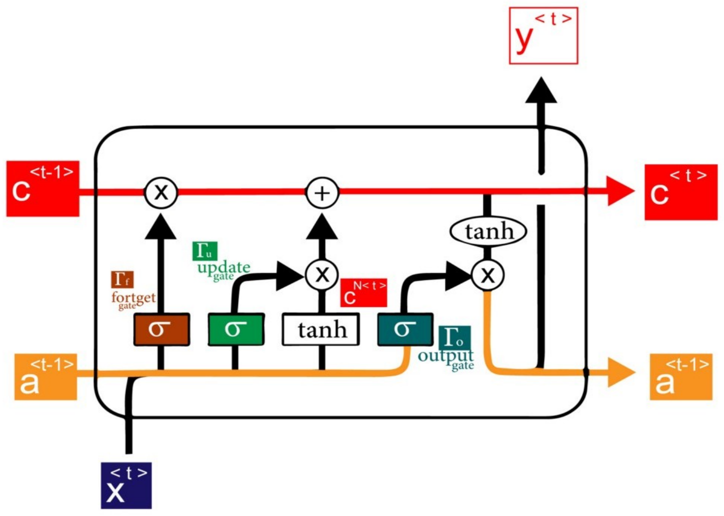


Figure 3.29: LSTM Architecture[15]

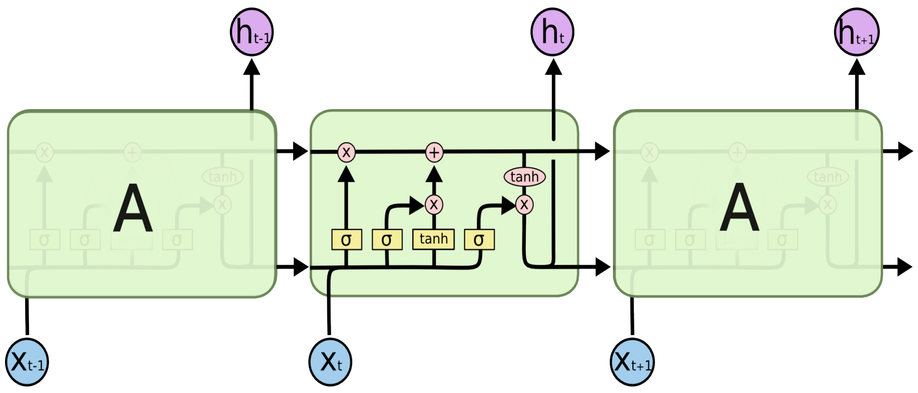


Figure 3.30: The repeating module in an LSTM contains four interacting layers[16]

3.4.2 Bidirectional-LSTM

LSTMs commonly called as Long Short Term Memory, LSTM is an improvement over RNN and it has capability to retain informations which it already learned and LSTM also supports long term dependencies. In this project we are aimed at implementing the Bidirectional LSTM with CUDA deep neural network library which remembers information in both direction forward and backward to predict the output. Below figure 3.29 shows the LSTM architecture : Let's discuss the LSTM architecture in brief : This is the first step in the LSTM network which deals with information which is to be thrown out or forget by the LSTM network which is decided by sigmoid layer. If the LSTM layer want to forget the information then the forget gate multiplies input in forget layer with 0 and the forget layer output 0. If the layer finds some useful information then it multiplies the same with 1 and keeps the complete information. The formula mentioned below is used

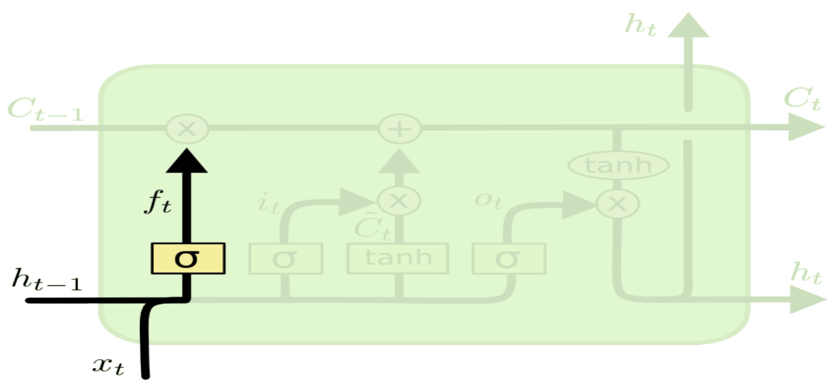


Figure 3.31: Forget Layer[17]

for the calculating the forget layer:

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{3.11}$$

The further steps in LSTM involves the information need to be kept by the model which essential and it needed to be stored. The following formula is used to do so :

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{3.12}$$

and

$$candidatevalue, (C_t) = \sigma(W_C \times [h_{t-1}, x_t] + b_C) \tag{3.13}$$

The cell state is updated from C_{t-1}, into the new cell state C_t with below formula

$$C_t = f_t \times C_{t-1} + i_t \times C_t \tag{3.14}$$

Atlast, we have to decide what we want as the output or how the output is to be designed that but the output will certainly now will be the filtered output as C_t has been updated.For this following formula will be used as

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.15}$$

and,

$$h_t = o_t \times tanh(c_t) \tag{3.16}$$

The hybrid architecture of the CNN and Bi-LSTM is to be implemented in the similar manner. Below, We have shown the actual architecture of the Bi-LSTM model with 2

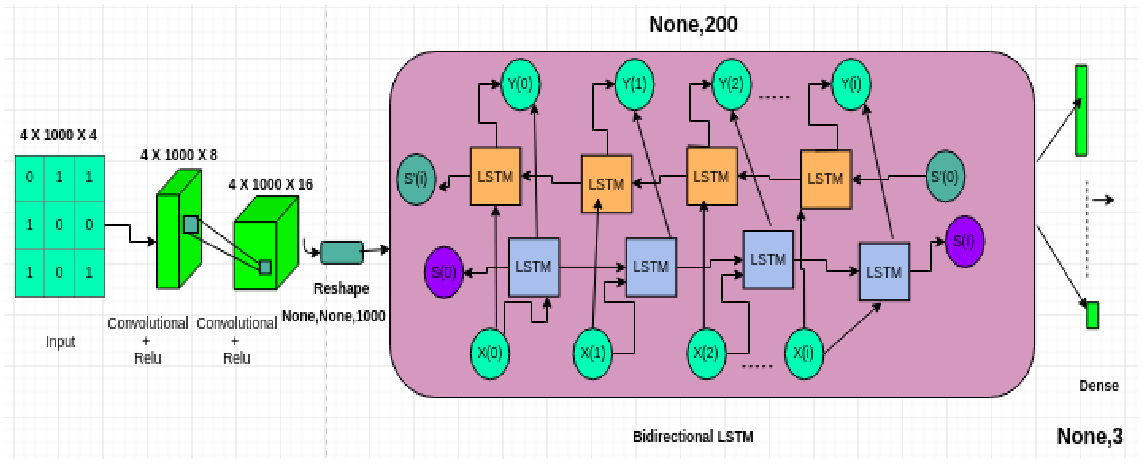


Figure 3.32: Actual internal structure of Bi-LSTM Model with 2 convolutional layers

convolutional layers as shown in the figure 3.32:

3.4.3 CUDA Deep Neural Network (CuDNN) library[18]

The NVIDIA's CUDA Deep Neural Network library (CuDNN) is a GPU-accelerated and highly robust library that improves the performance of the model and it can substantially speed up training of the model. It allows neural network researchers to focus on training neural networks and allow them enough time and resources to work and develop something innovative rather than spending time on low-level GPU performance tuning.

3.4.3.1 Benefits of CuDNN library

There are following benefits of using CUDA deep learning library apart from improving the GPU Performance

- It applies convolution in forward and backward direction.
- It also applies Pooling, Softmax and activation functions in forward and backward direction such as Sigmoid and other layers.

[18]https://developer.nvidia.com/cuda

3.4.4 Collation using Pre-trained models on Image dataset

The output of the best proposed deep learning model is to be collated with the state-of-the-art CNN models[19] pre-trained models on Image dataset such ImageNet, VGG-16 and VGG-19. These models is to be imported directly using the libray available to import these models separately and the same test data which is to be used to find the performance of proposed model will be similarly used to find the performance on pre-trained modwl and in this way the proposed model will be evaluated. The performance of these pre-trained model on practical applications

3.4.5 CUDA Deep Neural Network (CuDNN) library[20]

The NVIDIA's CUDA Deep Neural Network library (CuDNN) is a GPU-accelerated and highly robust library that improves the performance of the model and it can substantially speed up training of the model. It allows neural network researchers to focus on training neural networks and allow them enough time and resources to work and develop something innovative rather than spending time on low-level GPU performance tuning.

3.4.5.1 Benefits of CuDNN library

There are following benefits of using CUDA deep learning library apart from improving the GPU Performance

- It applies convolution in forward and backward direction.
- It also applies Pooling, Softmax and activation functions in forward and backward direction such as Sigmoid and other layers.

3.4.6 Gated Recurrent Unit (GRU)[21]

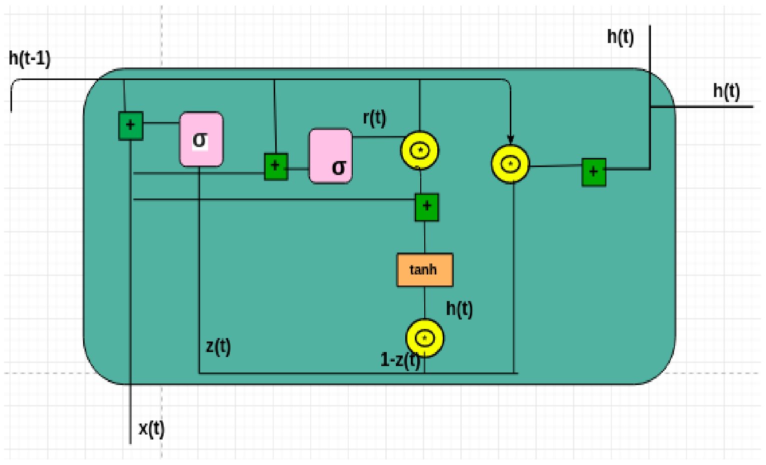
GRU stands for Gated Recurrent Unit and it is an improved version of RNN and it also solves the issue of Vanishing gradient problem. It has 2 gates, one is update gate and

[19]https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[20]https://developer.nvidia.com/cuda

[21]https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

other is reset gate. These two gates are responsible for which information to keep and which not to or which one to discard respectively. Below is the GRU Architecture as shown in fig 3.3.16



3.3.16 Gated Recurrent Unit

where,
$\sigma - Sigmoidfunction$
$+-plusoperation$
$\odot$ - Hadamard product operation
tanh - tanh function

Update gate (z_t) is calculated by using following formula:

$$z_t = \sigma(W^(z)x_t + U(z)h_{t-1}) \tag{3.17}$$

When x_t is provided as current input, it is multiplied by its own weight W(z). The same goes for h_{t-1} which holds the information for the previous t-1 units and is multiplied by its own weight U(z). Then the sigmoid layer is applied over the output of this and stored as update gate output.

Reset gate (r_t) decides the information to be discarded and can be calculated using this below mentioned formula -

$$r_t = \sigma(W^(r)x_t + U(r)h_{t-1}) \tag{3.18}$$

Now it is time to design a network in GRU to keep the information to be feeded in next layer and it is calculated as follows:

$$h_t = tanh(Wx_t + r_t \bigodot Uh_{t-1}) \tag{3.19}$$

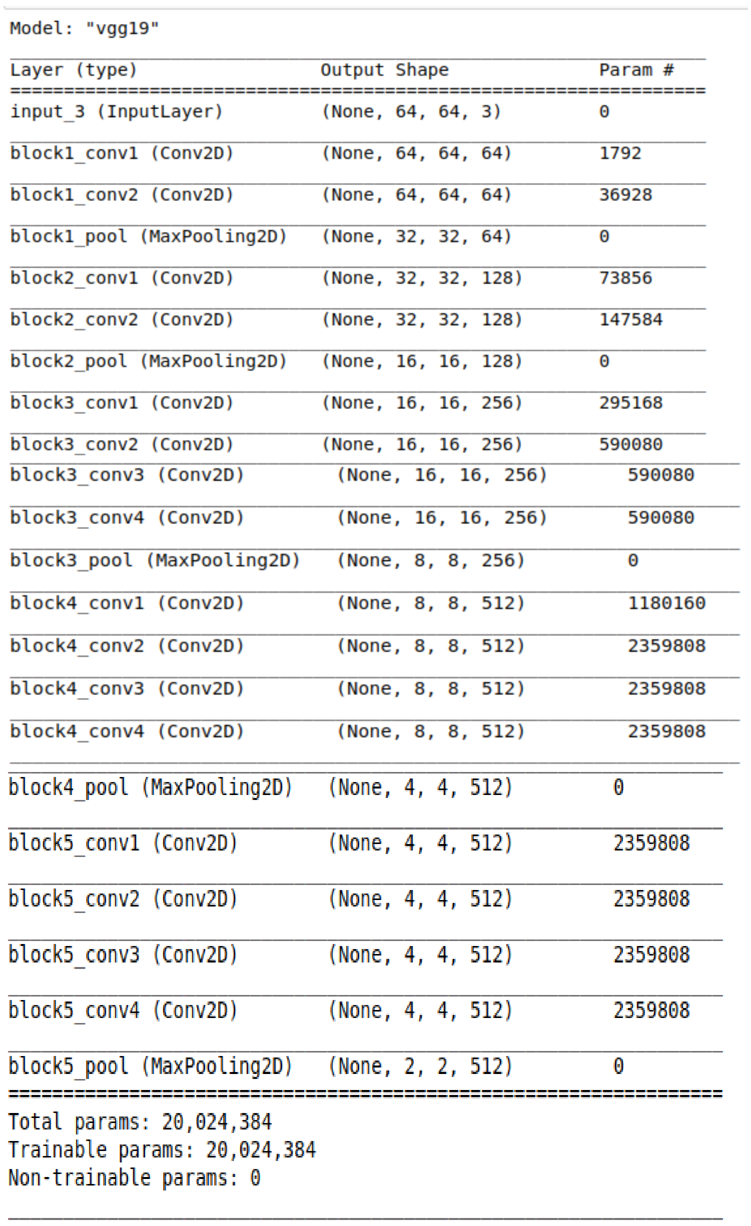
$$h_t = z_t \bigodot h_{t-1} + (1 - z_t) \bigodot h_t \tag{3.20}$$

3.4.7 VGG-19

The VGG-19 is a large scale image recognition pre-trained architecture based on ImageNet. VGG-19 is a Convolutional Neural Network with 19 deep layers and this is a pre-trained model model on more than a million images from ImageNet dataset[22]. The pre-trained VGG-19 model can classify images into 1000 object classes and it has learned rich feature dataset for a wide range of images. The model has input size of 224 × 224 and it has also an alternative model with 16 deep convolutional layers named VGG-16. Here, We are loading the VGG-19 model and loading the ImageNet weights with input 64 × 64 × 3 channel. The architecture of VGG-19 is shown below in figure 3.33: The VGG-19 architecture shown above was used for testing the result obtain on the used road traffic accident dataset. The VGG-19 architecture was further added with dense layers and the outputs from pre-trained model was used to train new densely connected classifier layers. The VGG-19 consisted of following params :

- The input to the VGG-19 was 64 × 64 × 3 which was used to extract pre-trained features with 2 × 2 × 512 tensor channel before training on new data-points.
- Pre-trained VGG-19 consisted of 16 Convolutional layers and 5 Maxpooling layers.
- Total params : 20,024,384 ; Trainable params: 20,024,384 ; Non-trainable params: 0
- Training samples : 6000 ; Test samples : 2000 ; Validation samples : 2000
- 2 dense layers with *relu* and *sigmoid* activation function respectively was implemented.
- Batch_size : 32 ; Epoch : 19

[22]http://www.image-net.org/



```
Model: "vgg19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.33: VGG-19 Architecture

To improve the performance of existing VGG-19 model, We fine-tuned the existing model by reshaping the input data-points as target size of (200,200) and batch_size = 6000 before using input in ImageDataGenerator function for train, test and validation dataset. The input to VGG-19 model while loading the ImageNet features was $200 \times 200 \times 3$ channel tensor. Pre-trained feature and label vectors were extracted from existing VGG-19 model. Before training the new input data by VGG-19 model, the input tensors were reshaped into $6 \times 6 \times 512$ channel and the model was further implemented with 3 dense layers. The earlier implemented VGG-19 model depicted that it was overfitting on the input data, in order to eliminate the overfitting, We have used regularisation in this implementation with following params :

- The input tensor is $6 \times 6 \times 512$ with pre-trained convolutional and maxpooling layers
- Added 3 dense classifier with 2 relu and a sigmoid activation function respectively
- Kernel_regularisation = 0.005 (in order to prevent from overfitting)
- Optimizer : RMS with lr=1e-4
- Batch_size : 32 and Epoch : 19
- Training samples : 6000 ; Test samples : 2000 ; Validation samples : 2000

This improved model of VGG-19 is not depicting any kind of overfitting.

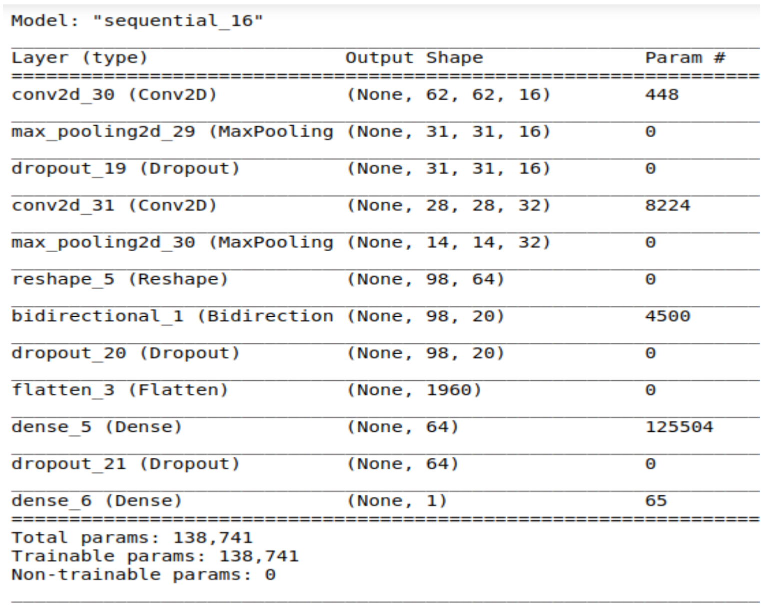
3.4.8 A hybrid CNN-Bi-LSTM architecture

In case of a hybrid architecture of CNN and Bidirectional LSTM (Bi-LSTM) with the following params :

- The input tensor is $64 \times 64 \times 3$ with a hybrid architecture of CNN - Bi-LSTM architecture.
- Implemented 2 Convolutional layers, 2 Maxpooling layers and Bi-LSTM.
- Dropout value of 0.25 and 0.50
- 2 dense layers with relu and sigmoid activation function respectively.

- Kernel_regularisation = 0.005 (in order to prevent from overfitting)
- Optimizer : RMS with lr=1e-4
- Batch_size : 128 and Epoch : 19
- Training samples : 6000 ; Test samples : 2000 ; Validation samples : 2000

Following image shows the architecture of a hybrid CNN & Bi-LSTM model :



```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 62, 62, 16)        448
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 31, 31, 16)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 31, 31, 16)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 28, 28, 32)        8224
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
reshape_5 (Reshape)          (None, 98, 64)            0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 98, 20)            4500
_________________________________________________________________
dropout_20 (Dropout)         (None, 98, 20)            0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1960)              0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                125504
_________________________________________________________________
dropout_21 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
Total params: 138,741
Trainable params: 138,741
Non-trainable params: 0
_________________________________________________________________
```

a hybrid CNN & Bi-LSTM model architecture

Chapter 4

Results

Outline: This chapter presents the following:

1. Introduction
2. Exploratory data analysis
3. Evaluations Results

4.1 Introduction

In this section, We have discussed the exploratory analysis of our project in depth and meaning of each and every features and how it is impacting our results. The results associated with our project are discussed in details, i.e, to evaluate how better and efficient each Machine learning and deep learning model performed on the road traffic accidents data-set and how it is performing as compared to other state of the art model.

4.2 Exploratory Analysis

In this experiment, our main aim was to estimate the severity of road traffic accidents using various machine learning and deep learning architectures on UK road traffic accidents dataset. The road accidents data-set had approx 0.7 million datapoints during 2013 to 2017. There were more than 40 features or factors on the based of that the dataset was prepared. Our main task was to find the relevant features to acquire in-depth from the dataset. We also performed visualization on the input dataset and found interesting results.

Based on longitude and latitude, a scattered map was visualised to identify the specific areas at high risk and low risk. Figure 4.1 represents areas with more dense points having more reported accidents than areas having less dense points. Figure 4.1 represents that region at the coordinate (0,51.8) is highly dense which means this region is highly prone to road traffic accidents, whereas regions near coordinates (-4,54.5) to (-2, 54) are having less density which signifies a safe zone with less number of accidents. We plotted the latitude and longitude on the actual map of the LSOA using Folium, Branca and Open Street Maps API[1]. Figure 4.2 shows the accidents reported in LSOA [E01004736] in the year 2017 and there were 167 accidents reported in this LSOA. We have visualized these 166 locations in fig4.2.

We made an attempt to find features of the accidents at a particular location such as level of severity, weather condition, light conditions, number of casualties, time, road type, junction details and speed limit. Below figure 4.3 shows the specification of the accidents reported for better analysis of the accident in a particular LSOA and at the particular location. The figure 4.3 shows the other features at the time of accidents in a

[1]https://www.openstreetmap.org/

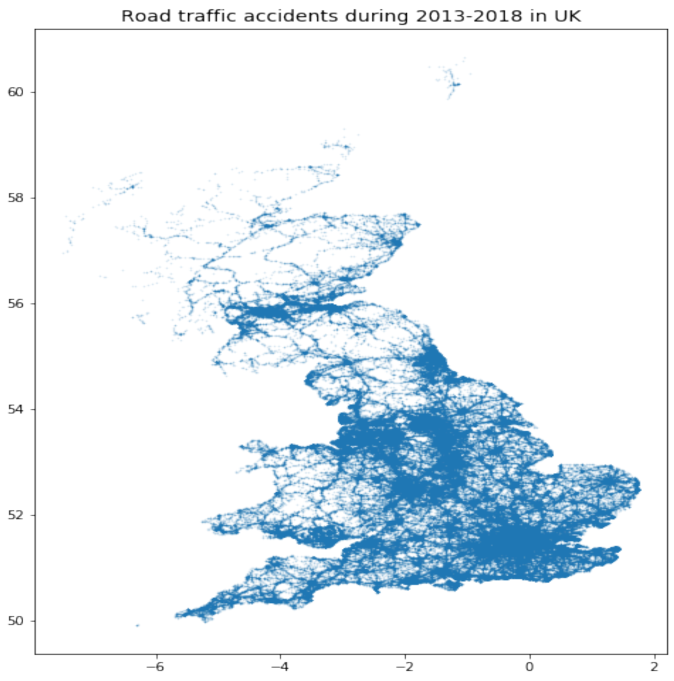


Figure 4.1: Road traffic accident density in UK during 2013-2018

particular LSOA at a particular location which is one of the major visual finding of this project this help in better analysis of a a particular location in a LSOA. In figure 4.4, We have tried to visualize the accident severity level location wise in an LSOA using Open Street Map API. The map has been displayed as a heat map. We have made an attempt to visualise the number of casualties with respect to the speed limit in order to find the most severe accident region as shown in figure 4.5.

It is clear from above figure4.5 that when the speed limit is at 70 mph or higher, chances of fatal accidents casualties is higher than slight accident casualties, whereas for 30 mph the chances of slight accidents casualties are higher than fatal accidents casualties. In figure 4.5, only slight and fatal classes are compared for analysing the relationship between

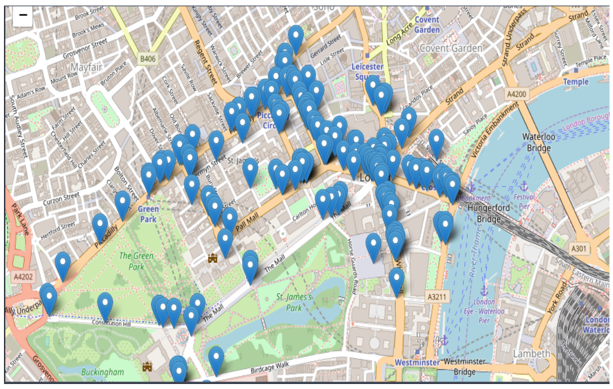
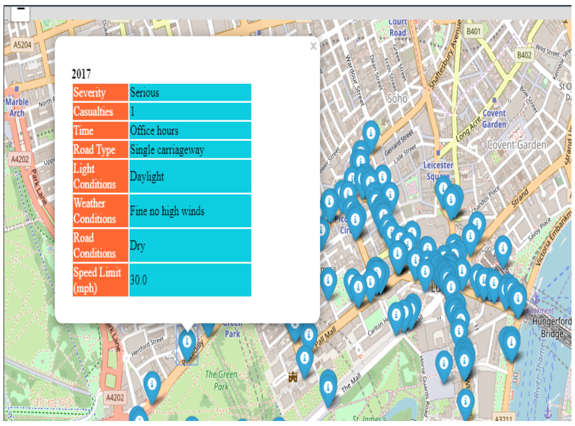


Figure 4.2: The reported accidents location in an LSOA



2017

Severity	Serious
Casualties	1
Time	Office hours
Road Type	Single carriageway
Light Conditions	Daylight
Weather Conditions	Fine no high winds
Road Conditions	Dry
Speed Limit (mph)	30.0

Figure 4.3: The reported accidents location in a LSOA is shown using Open Street Map with specific features of the accident

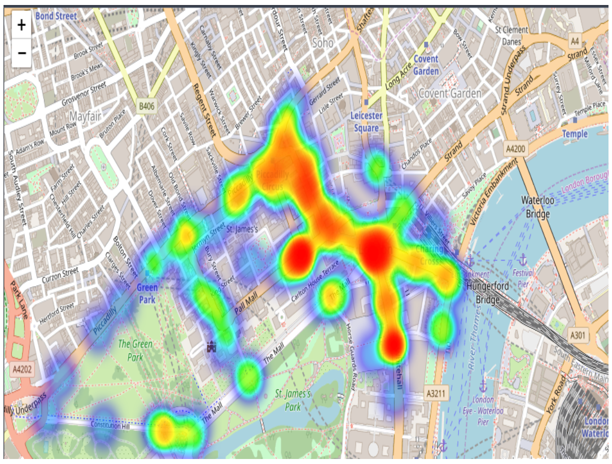


Figure 4.4: Shows the severity of road accidents of the reported accidents in an LSOA

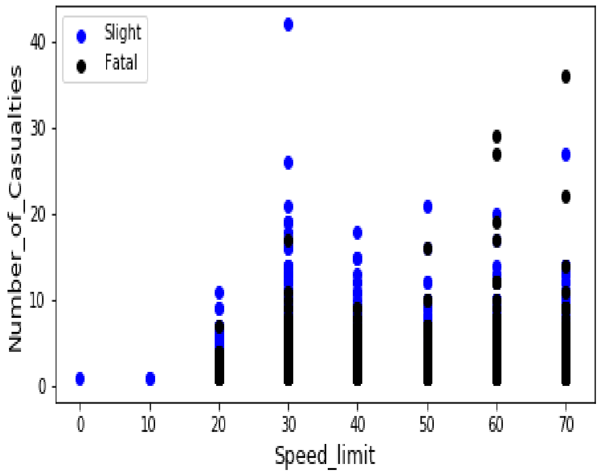


Figure 4.5: Represents the relation between speed limit and number of casualties

speed limit and number of casualties. We tried to find the locations where only fatal accidents occurred during 2013-2017 in UK as shown in figure 4.6. It can be observed

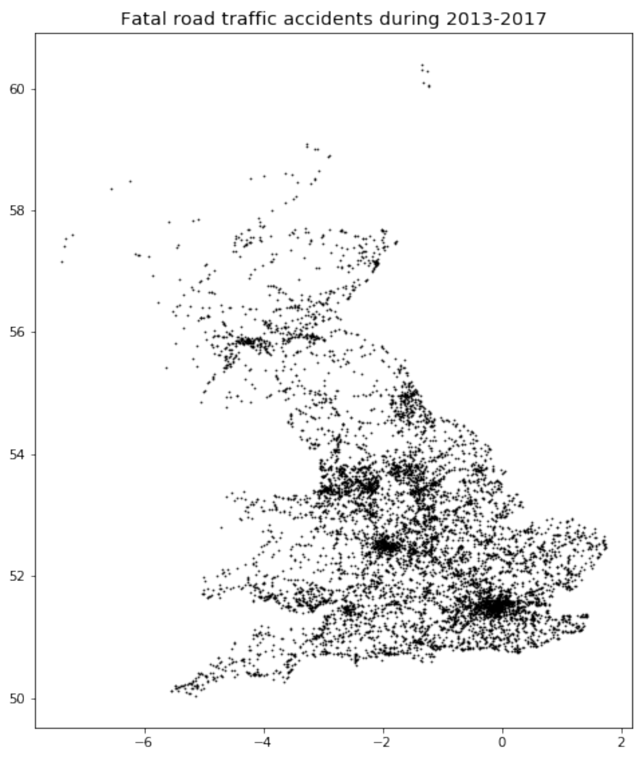


Figure 4.6: Fatal accidents during 2013-2018

from figure 4.6 that at the right most bottom the amount of fatal accidents is dense which means it is a danger region. The plotting between accidents severity and road type is shown in figure 4.7: The figure 4.7 shows the road class (A) has the highest amount

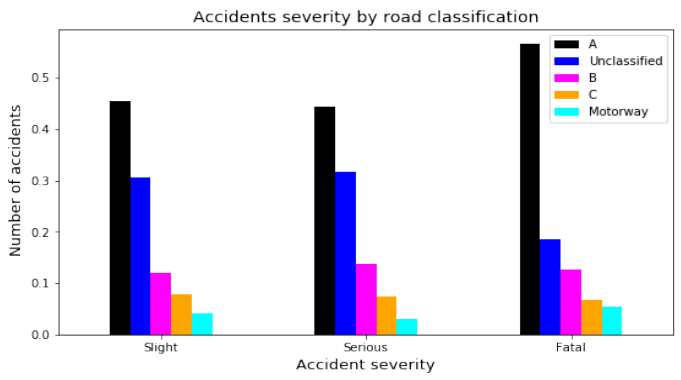


Figure 4.7: Accident severity by road classification

slight accidents reported during 2013-2018 and the road class (A) is really dangerous and has the highest number of accidents among all the severity classes. Figure 4.8 shows the severity of accidents based on the type of roads and number of accidents happened

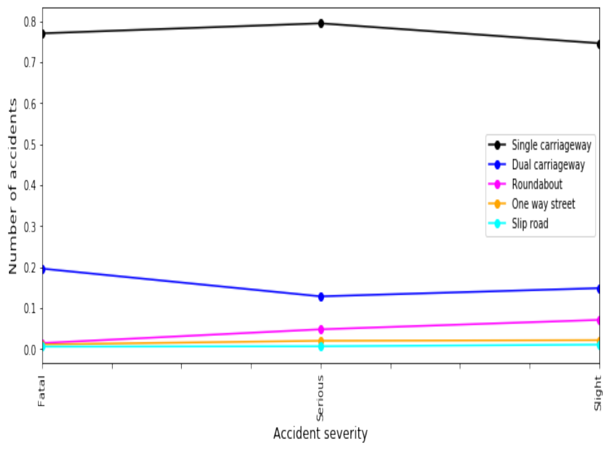


Figure 4.8: Accidents severity by road types during 2013-2018

during 2013-2018. Figure 4.9 represents the type of junction and number of accidents

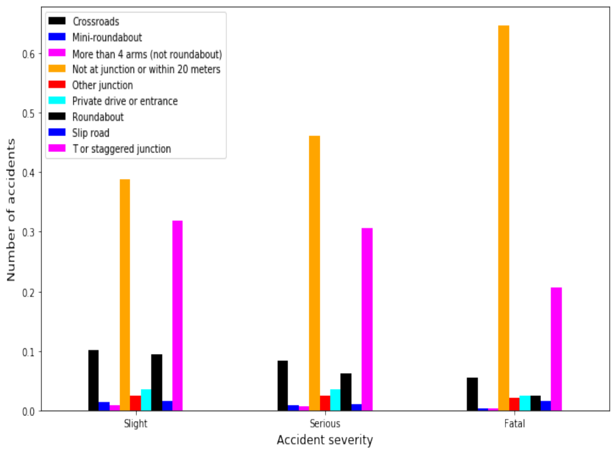


Figure 4.9: Accidents severity by types of Junction during 2013-2018

happened as slight, serious or fatal. We can observe from this figure that classes of road accidents severity tend to occur about 20 metres near a junction and followed by T or staggered junctions. The figure 4.10 can be interpreted as the speed limit of 30 mph is responsible for large amount of casualties whereas 70 mph causes less number of road traffic accidents, but figure 4.5 suggests that 70 mph causes fatal accidents whereas 30 mph causes slight accidents (by severity). Figure 4.11 shows relationship between number of accidents and speed limit but in reference to level of severity.

It can be clearly observed that the speed limit of 60 mph leads to more fatal accidents which is followed by serious accidents and slight level of accidents are the highest in number of accidents happened during 2013-2018 in UK. The lighting condition is one of the important factor in context to the road accidents and in figure 4.12, it can observed

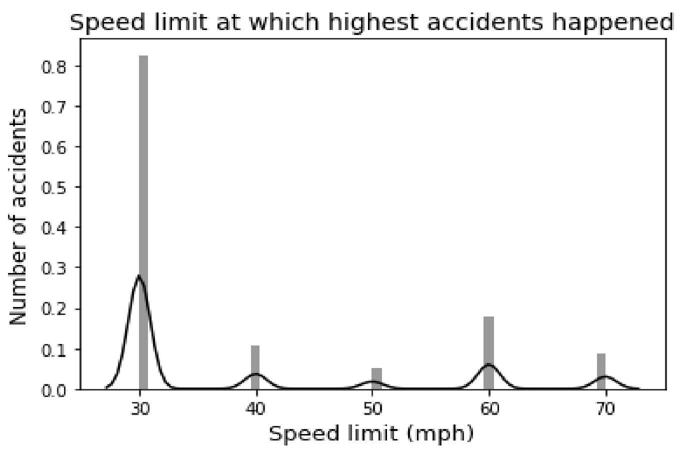


Figure 4.10: Speed limit versus Number of accidents

Figure 4.11: Speed limit and Number of accidents with reference to severity level

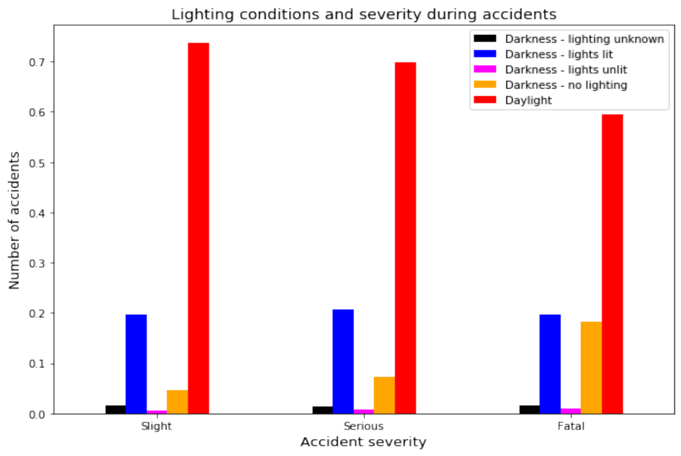


Figure 4.12: Lighting conditions of accidents at different severity

that most of the accidents occurred during darkness/at nights which is 0.9% and during daylight which is approx 0.4% of accidents. Figure 4.12 is the normalised value representation of number of accidents. Weather plays a key role in context of road accidents and it can be observed from figure 4.13 which shows the normalised value of accidents and it can be observed that snowing no high winds class has highest accidents rate for all three severity classes. We made an attempt to estimate the severity of road traffic accidents by deploying code-mixed dataset using deep learning. In our experiment, prepraring the input pipeline with exploratory analysis was a challenging task since the input data were highly imbalanced and we tried to make it a balanced dataset. Below following tables have been shown as a comparison of performance achieved by different model on training and testing the models. Let us discuss the results of each models:

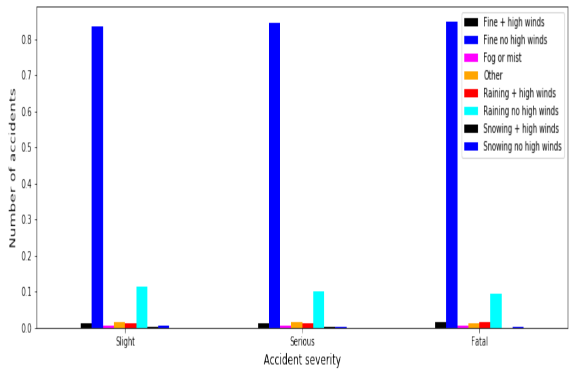


Figure 4.13: Weather conditions during accidents with different severity level

Table 4.1: Performance on Logistic Regression and ANN models

File	**Logistic Regression**			**Artificial Neural Network**		
	Precision	Recall	F-score	Precision	Recall	F-score
major(0)	0.20	0.43	0.27	0.19	0.55	0.28
minor(1)	0.86	0.66	0.75	0.91	0.54	0.68
	Training Acc.	Testing Acc.		Training Acc.	Testing Acc.	
score	0.56	0.63		0.55	0.64	

Following figure 4.14 and 4.15 shows the graphical representation of the performance of VGG-19 model : Below figure shows the loss metrics : To improve the performance of existing VGG-19 model, We fine-tuned the existing model by reshaping the input data-points as target size of (200,200) and batch_size = 6000 before using input in ImageDataGenerator function for train, test and validation dataset. The input to VGG-19 model while loading the ImageNet features was $200 \times 200 \times 3$ channel tensor. Pre-trained feature and label vectors were extracted from existing VGG-19 model. Before training the new input data by VGG-19 model, the input tensors were reshaped into $6 \times 6 \times 512$ channel and the model was further implemented with 3 dense layers. The earlier implemented VGG-19 model depicted that it was overfitting on the input data,

Table 4.2: Performance on LSTM and GRU models

File	**LSTM**			**GRU**		
	Precision	Recall	F-score	Precision	Recall	F-score
major(0)	0.23	0.52	0.32	0.24	0.53	0.33
minor(1)	0.89	0.65	0.76	0.86	0.67	0.77
	Training Acc.	Testing Acc.		Training Acc.	Testing Acc.	
score	0.65	0.68		0.65	0.68	

Table 4.3: Performance of CUDA deep neural network LSTM(CuDNNLSTM) and GRU (CuDNNGRU) on Test data

File	**CuDNNLSTM**			**CuDNNGRU**		
	Precision	Recall	F-score	Precision	Recall	F-score
$major(0)$	0.24	0.48	0.36	0.27	0.52	0.36
$minor(1)$	0.88	0.72	0.80	0.90	0.75	0.82
	Training Acc.	Testing Acc.		Training Acc.	Testing Acc.	
$score$	0.70	0.79		0.74	0.83	

Table 4.4: Performance of CNN and VGG-19 on satellite images test data

File	**Convolutional Neural Network (CNN)**			**VGG-19 (Improved)**		
	Precision	Recall	F-score	Precision	Recall	F-score
$major(0)$	0.78	0.85	0.81	0.87	0.64	0.81
$minor(1)$	0.83	0.76	0.80	0.81	0.88	0.84
	Training Acc.	Testing Acc.		Training Acc.	Testing Acc.	
$score$	0.84	0.79		0.75	0.72	

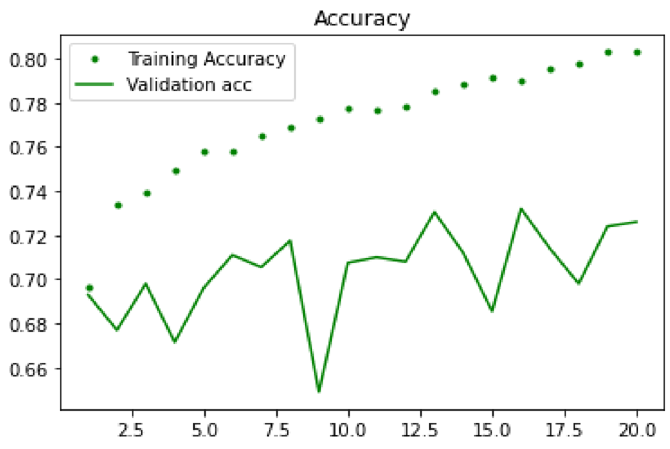


Figure 4.14: Training accuracy vs Validation accuracy

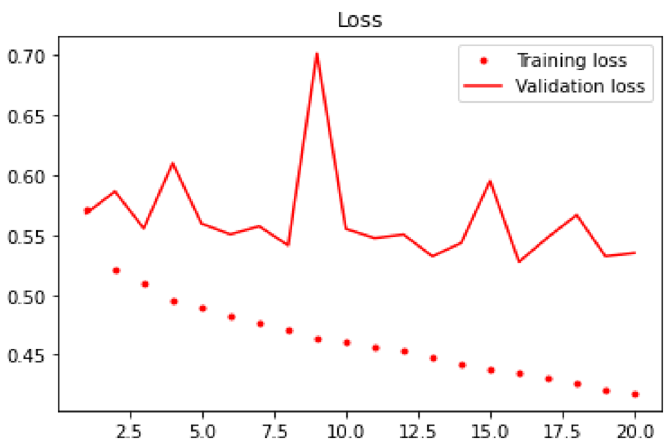


Figure 4.15: Training loss vs Validation loss

in order to eliminate the overfitting, We have used regularisation in this implementation with following params :

- The input tensor is 6 × 6 × 512 with pre-trained convolutional and maxpooling layers
- Added 3 dense classifier with 2 relu and a sigmoid activation function respectively
- Kernel_regularisation = 0.005 (in order to prevent from overfitting)
- Optimizer : RMS with lr=1e-4
- Batch_size : 32 and Epoch : 19
- Training samples : 6000 ; Test samples : 2000 ; Validation samples : 2000

Table 4.5: Performance of Bi-LSTM and a hyubrid CNN + Bi-LSTM architecture on satellite images test data

File	**Bidirectional LSTM (Bi-LSTM)**			**CNN + Bi-LSTM (hybrid)**		
	Precision	Recall	F-score	Precision	Recall	F-score
$safe(0)$	0.84	0.81	0.82	0.86	0.81	0.84
$danger(1)$	0.84	0.86	0.85	0.82	0.86	0.83
	Training Acc.	Testing Acc.		Training Acc.	Testing Acc.	
$score$	0.93	0.82		0.94	0.85	

Table 4.6: Performance of a hybrid CNN + Bi-LSTM architecture on **code-mixed input data**

File	**CNN + Bi-LSTM**		
	Precision	Recall	F-score
$safe(0)$	0.92	0.80	0.86
$danger(1)$	0.89	0.82	0.85
	Training Acc.	Testing Acc.	
$score$	0.93	0.91	

The accuracy metric and loss metric was recorded and shown below (accuracy and loss metric was obtained on training and validating the model for 11 epochs) : To prevent the over-fitting of the deep neural networks, we the l1 regularization, early-stopping and dropout.

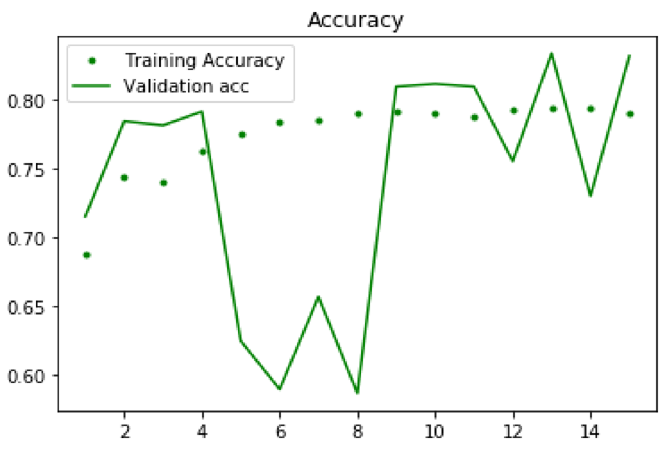


Figure 4.16: Training and validation accuracy for code-mixed input dataset till 11 epochs

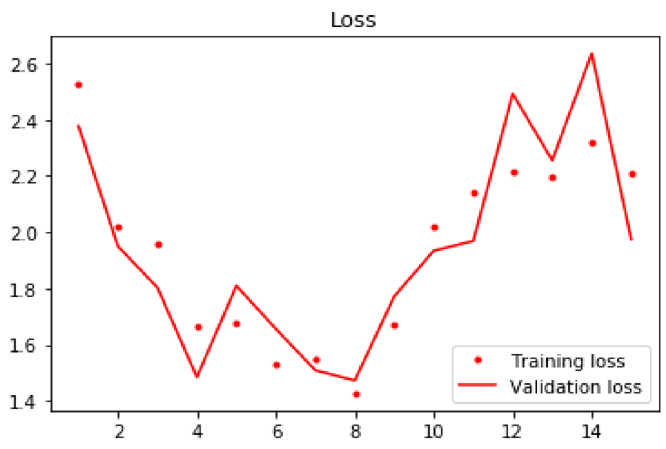
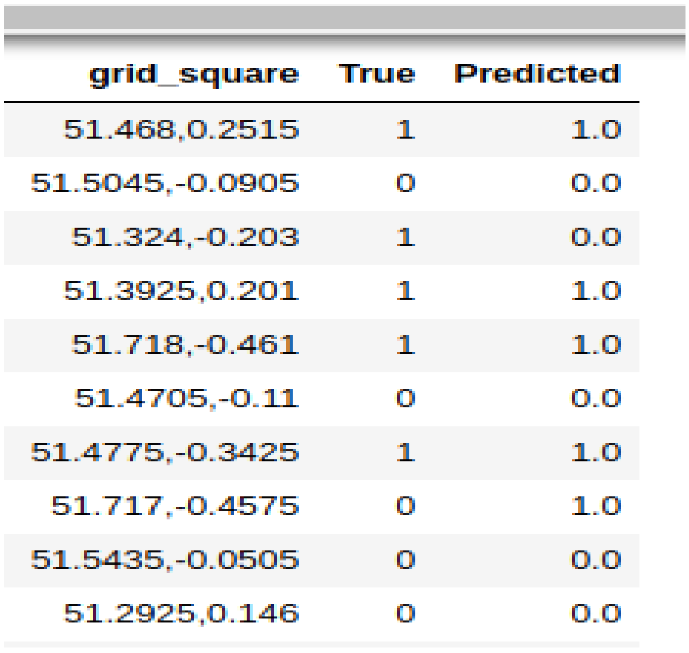


Figure 4.17: Training and validation loss for code-mixed input dataset till 11 epochs



grid_square	True	Predicted
51.468,0.2515	1	1.0
51.5045,-0.0905	0	0.0
51.324,-0.203	1	0.0
51.3925,0.201	1	1.0
51.718,-0.461	1	1.0
51.4705,-0.11	0	0.0
51.4775,-0.3425	1	1.0
51.717,-0.4575	0	1.0
51.5435,-0.0505	0	0.0
51.2925,0.146	0	0.0

Figure 4.18: Actual vs predicted value based on location (safe or danger)

The following table shows the actual label of the place ,i.e., whether the location is safe or danger and the other column represents the predicted value as whether the input image is safe or dangerous as shown in below figure 4.18:

Chapter 5

Conclusion & Future Work

This experiment was aimed at estimating the severity of road traffic accidents and finding the relationship among the various factors responsible for road traffic accidents. In this experiment, We also did a comparative study among state-of-the-art deep learning models in order to improve the predictive capacity of the models. We studied the open source road traffic accidents data and this study and experiment will certainly help towards the safety of mankind and measure to be taken to reduce the impact of a road accident. The scope of the work in this area has no limit and it needs a lot of improvement every single day. We represented results in tabular format for analysing this experiment in depth. On text dataset, the CUDA deep neural network with CuDNNGRU architecture outperformed the other architectures with a highest test accuracy of 83%. This model was followed by CUDA deep neural network with CuDNNLSTM which gave an accuracy of 79%. On image dataset, the CNN model with 84% training accuracy and 79% testing accuracy, whereas when the same model was compared with pre-trained VGG-19 architecture, the VGG-19 obtained a training accuracy of 75% and a testing accuracy of 72%. On code-mixed input dataset, a hybrid architecture of CNN and Bi-LSTM with input-pipeline implemented using multi-layer perceptron (MLP) with 2 dense layers concatenated with CNN with 2 dense layers, 1 convolutional layer obtained a training accuracy of 94% and a testing accuracy of 85%. On same dataset, the CNN with 3 convolutional layers and 2 max-pooling layers obtained a training accuracy of 93% with tesing accuracy of 91% which outperformed other models. Further, we are working on to improve the deep learning models by applying BERT or XLnet or Transformer model on this dataset as a pattern summarizer.

Bibliography

[1] S. Oniga and J. Suto, *Human activity recognition using neural networks.* 2014, Proceedings of the 2014 15th International Carpathian Control Conference, ICCC, May 28-30, pp. 403-406.

[2] A. Sharaf, T. Madhar, and T. Salah, *Severity Prediction of Traffic Accident Using an Artificial Neural Network.* Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/for.2425, 2016.

[3] K. Pantelis, K. P. Fanis Papadimitriou, and P. Panos, *Urban Freeway Crash Analysis, Geometric, Operational, and Weather Effects on Crash Number and Severity.* Dec 2008, Transportation Research Record Journal of the Transportation Research Board.

[4] A. Hassan T. and M. A.Abdel-Aty, *Devel-opment of Artificial Neural Network Models to Predict Driver Injury Severity in TrafficAccidents at Signalized Intersections.* Transportation Research Record 1746 Paper No. 01-2234.

[5] G. Rui, B. Ana, A. de Almeida, and E. JoséPaulo, *Prediction of road accident severity using the ordered probit model.* 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain, Transportation Research Procedia 3 (2014) 214 – 223.

[6] M. Chong, A. Abraham, and M. Paprzycki, *Traffic Accident Analysis Using Decision Trees and Neural Networks*, 06 2004, vol. 2.

[7] K. S. M. Y. Najjar, A., *(2017). Combining Satellite Imagery and Open Data to Map Road Safety. In: AAAI.*

[8] P. Z. Y. Niu, J. Wen and Y. Xue, *2019, "A Hybrid R-BILSTM-C Neural Network Based Text Steganalysis," in IEEE Signal Processing Letters, December 2019, vol. 26, no. 12, pp. 1907-1911.*

[9] G. G. . T. Y. . H. S. . S. O. . M. K. (2016)., *Fine-tuning deep convolutional neural networks for distinguishing illustrations from photographs. Expert Systems with Applications. 66. 10.1016/j.eswa.2016.08.057.*

[10] G. Anisya, Swara, *(2017). Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route. IOP Conference Series: Earth and Environmental Science. 97. 012004. 10.1088/1755-1315/97/1/012004.*

[11] D. V. . P. D. del Rio F., Messina P., *(2018). Do Better ImageNet Models Transfer Better... for Image Recommendation ? ArXiv, abs/1807.09870.*

[12] . M. Pritt G. Chern, "Satellite Image Classification with Deep Learning, *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, 2017, pp. 1-7, doi: 10.1109/AIPR.2017.8457969.*