Traffic and Accident Prediction for Images and Video Using Deep Learning Techniques

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Sahith D

17BCE0422

Chitirala Naveen Aaditya

17BCE0566

Under the guidance of

Prof. Sathiya Kumar C

SCOPE

VIT, Vellore

June, 2021

DECLARATION

I hereby declare that the thesis entitled "Traffic and Accident Prediction for Images and Video Using Deep Learning Techniques" submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Professor. Sathiya Kumar C.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Sahith D (17BCE0422)

Chitirala Naveen Aaditya (17BCE0566)

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Traffic and Accident Prediction for Images and Video Using Deep Learning Techniques" submitted by Sahith D (17BCE0422), Chitirala Naveen Aaditya (17BCE0566), School of Computer Science and Engineering, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him under my supervision during the period, 01.02.2021 to 09.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 8/6/2021

Signature of the Guide

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

It is consistently a delight to remind the fine individuals in the Engineering program for their earnest direction I got to maintain my aptitudes in building all through this Capstone Project. I want to thank my Project Guide Professor. Sathiya Kumar C. for the inspirational mentality he portrayed for my work and persistently upheld me every way under his guidance. He helped me throughout, from beginning counsel to support to finish my undertaking. I express massive delight and profound feeling of appreciation towards my guide Professor. Sathiya Kumar C. for giving me this chance and the direction all through this undertaking as well as my entire undergrad venture.

Sahith D (17BCE0422)

Chitirala Naveen Aaditya (17BCE0566)

Student Name

Executive Summary

This project is mainly based on CNN algorithm. There are four classes of images. They are dense traffic, sparse traffic, fire accident and normal accident. We consider the high and low traffic detection under traffic section and the fire and normal accident are considered under the accident section. Our main objective is to classify the given input image or video into high traffic, low traffic, fire accident and normal accident by obtaining the trained model from input traffic and accident dataset. First, the video is divided into frames using openCV and then series of operations such as dimension reduction, noise reduction, are done on the each frame(image) and passed through the trained CNN model to identify the image. This live process identification form the surveillance cameras can predict traffic and suggest best traffic routes for the commuters and also save lives by immediately responding to the accidents and fire accidents on road. Traffic congestion can be easily recognized by placing the cameras in junctions and areas

Where we expect usual traffic but accidents can happen anywhere on the road. So we can take an advantage of the infrastructure used for autonomous cars which need 5G network pole for every 500mtrs of distance along side of the road. Implementing this system has many advantages like does not need special infrastructure, easy for police surveillance, accident alert, safer for people(women,children) travelling on road , traffic moderation and reduce congestion.

Keywords: CNN, OpenCV, Keras , Tensorflow , On road accidents, fire accidents, dense traffic and sparse traffic datasets.

List of Figures

List of Tables

List of Abbreviations

ITS	Intelligent transport system
CNN	Convolutional neural networks
DBN	Deep belief network
AI	Artificial intelligence
DNN	Deep neural network

1.INTRODUCTION

1.1. Theoretical Background:

As rapid growth of deep-learning approaches, numerous different responsibilities in clinical diagnostics, robotics, etc. have been found to apply. However, it is unanticipated that there are hardly any successful applications in the system, particularly in view of the large amount of video and image traffic used to monitor the urban road network and the highway, have been reported. Depending on the current situation, most cameras act as passive monitors and cannot detect the congestion automatically on time. When it comes to the alarm system, the detection of traffic jams relies mainly on lots of staff to manually report congestion.

The monitoring of the current surveillance system and the identification of congestions in the traffic control hall are highly tedious and time-consuming. Moreover, all cameras can't be watched with human eyes because many cameras are on the highway that cover a large region. However, it is important to detect traffic congestion quickly in a large region.

1.2. Motivation:

The monitoring of the current traffic control systems using the traffic monitoring hall is extremely difficult for a person. Moreover, all cameras can not be viewed on individual eyes given the several cameras that cover a large-scale area. In large regions it is nevertheless important to detect traffic congestion quickly. Early detection can prevent extended traffic congestion, which is a significant application in intelligent transport system with devastating development from the initial controlled traffic congestion (ITS).

In many fields of the transport structure, from traffic flow prediction to traffic congestion recognition, deep learning algorithms have the possibility for implementation. Traffic classification is one of the important elements of an ITS, which can be used widely in the analysis of traffic management, traffic flow strategies and so on. An intelligent transport system therefore needs to be proposed.

1.3. Aim of the proposed Work:

Our main aim is to identify the type of image or video uploaded. In this project we are identifying the type of traffic or the type of accident. In type of traffic we predict whether the image is of high traffic or low traffic. In type of accident we predict whether the images are of normal accident or fire accident.

1.4. Objective(s) of the proposed work:

The objective of our study is to implement traffic detection system using Convolutional Neural Networks (CNN). Our main objective is to classify the given input image or video into high traffic, low traffic, fire accident and normal accident by obtaining the trained model from input traffic and accident dataset. First, the video is divided into frames using openCV and then series of operations such as dimension reduction, noise reduction, are done on the image and passed through the trained CNN model to identify the image.

2. Literature Survey

2.1. Survey of the Existing Models/Work

[1] "H. Lei et al" proposed that accurate cell image classification of the human epithel-2 a key role in diagnosing and following autoimmune diseases. One among the more important challenges is huge variations in intra-class lighting. We are proposing a framework for classification of HEp-2 cell images based on the very DSRN. In particular, we use a residual 50-layer network is significantly more to remove discriminatory characteristics. It instead uses time consistency at the level of the feature. It enhances the per-frame characteristics by adding the nearby characteristics along the motion paths. Two available datasets are used to evaluate the proposed method. A strategy related to transfer learning of cross model is developed in contrast to the previous deep learning models learned from scratch.

[2] "P. Wang, L. Li, Y. Jin, and G. Wang" proposed that It is important to detect traffic congestion. Although the surveillance system has been in use on the road for years, it is difficult for various illuminations, weather and other disturbances to automatically identify and report the congestion of traffic in complicated transport. The human eye-based detection process is time consuming and tedious, because it does not satisfy the demands of practical applications with the machine detection accuracy. This paper proposes a new classifier to generate four TrafficNets by means of convolutionary neuronal networks. This new structure trains and tests congestion and non-congestion images. Over 30,000 image database from existing traffic monitoring videos is extracted. Trained and tested with the database, traffic accuracy is compared and training times are compared. The research shows that the precision of the method proposed is far greater than the conventional method based on functional extraction without profound knowledge, up to 90 percent.

[3] " X. Zhu, Y.Wang, J. Dai, L. Yuan, and Y.Wei" proposed that Detection accuracy is affected by degenerated objects in videos. Existing work tries to use temporal information, but such methods are not fully trained. We present flow-oriented add-on, a precise and end-to-end video object detection learning framework. It instead uses time consistency at the level of the feature. It enhances the per-frame characteristics by adding the nearby characteristics along the motion paths. Our method improves significantly on strong baselines on single frames, particularly with fast-moving objects in the ImageNet. Without any additional bells and whistles, this framework is designed and consistent with the other best systems to win the 2016 ImageNet VID Challenges. It'd release the code.

[4] "Z. Zhao, W. Chen, X.Wu, P. C. Chen, and J. Liu" proposed that One of the essential problems in intelligent transport systems is the short-term traffic forecast. Exact forecasts allow travellers to make suitable modes of travel, travel routes and departure time that are significant in traffic control. A more desired approach for the analysis of traffic data is a workable way to promote forecast accuracy. The provision of ample data of traffic and computational power in recent years has led us through in-depth learning approaches to increase the accuracy of the traffic projections. It instead uses time consistency at the level of the feature. It enhances the per-frame characteristics by adding the nearby characteristics along the motion paths. A new LSTM model for traffic forecasting is proposed. The LSTM network will achieve a much best performance in comparison with other representative prediction models.

[5] "P. Li, D. Wang, _ L. Wang, _and H. Lu" proposed that deep learning in visual tracking has recently been very successful. This paper aims at examining the most up-to-date methods of tracking_ based on profound learning. Firstly, we introduce deep visual_ tracking backgrounds including basic _concepts for visual tracking and related algorithms for deep learning. Secondly, The systemic traffic volume evaluation enables suitable change in control measures to direct

traffic streams in order to achieve certain performance objectives. A suitable and practical source of traffic data is video traffic monitoring. The document presents a classification method based on video surveillance data for road traffic conditions. The deep trackers are much more effective than the low-level handmade trackers. Features from various convolutionary layers have many various characteristics and usually lead to more robust trackers due to their effective combination.

[6] "P. Wang and J. Di" proposed that the rapid development of deep education has dramatically improved its performance in image classification and object recognition. These promising results can also be used in scattering media imaging to improve understanding of spike patterns. In this article, the scattering media is a multimode fibre, and the generating patterns are transmitted by 4000 images from face to face. A SpeckleNet is proposed and developed based on a convolutionary network using these 3600 speckle patterns and its output layer for the SVM classification is activated. In comparison with the accuracy of pure SVM method the binary classification accurate of the proposed CNN SpeckleNet architecture is around 96 percent. The promising results confirmed the possibility to reduce optical and computational cost of optical sensing through combination with deep learning.

[7] **"** J. Zhao, Z. Zhang, W. Yu, and T.-K. Truong" proposed that In ships detection from SAR images, convolutionary neural networks found applications. Some challenges are, however, hindering their progress. First, the boundary boxes detected are not too compact. Secondly, a lot of small and densely clustered ships lack detections. Thirdly, analogical scattered objects on the ground are erroneously detected as ships. This paper proposes a cascade-coupling visual care method for the detection of SAR vessels by CNN. The newly submitted 3C2N model is considered as a qualified generator of ship proposals as the spatial information of the images is used more adequately. A sequence of cascade detectors is It may also detect the specific object displayed on the screen at any given time based on user input. For this assignment, a sequential frame extraction method of films is employed, as well as a deep learning methodology of CNN and Fully Connected CNN. The approach has an accuracy of around 75 percent. The proposals

could be further refined, reducing the missing detections and false alarms significantly. Moreover, data are used to remove ship-like goals on land from a digital elevation model.

[8] “T. Pamula ”proposed that in order to determine traffic control strategies in ITS, classification is a vital task. The systemic traffic volume evaluation enables suitable change in control measures to direct traffic streams in order to achieve certain performance objectives. A suitable and practical source of traffic data is video traffic monitoring. The document presents a classification method based on video surveillance data for road traffic conditions. Classify your content related to video and establish congestion measures related to traffic observed using the confusing neural network. Four traffic conditions that match the LOS categories are distinguished. The network is validated by video from a range of traffic monitoring sites. The trained CNN is able to process video data systemically through traffic management subsystems of ITS. The method proposed is better than the quality of video data and less sensitive.

[9] “M. Barth and K. Boriboonsomsin” proposed that over the years, intelligent systems for transport (ITS) have been developed to solve a range of transport problems. The main goals of ITS were mainly to improve safety and efficiency in transportation. However, ITS can have important environmental advantages as well. The vehicles' pollution and associated air quality have become long-term concerns. More recently, the consumption of energy and GHG emissions have increased their concerns. Many current ITS programmes are capable of cutting consumption of energy and GHG emissions; however, ITS programmes that minimise the transport's environmental effects can also be specifically designed. In this paper, we outline briefly various types of ITS programmes and their possible benefits of environment. They reviewed many transport system projects in North eastern university especially aimed at reduced consumption of energy and emissions of GHGs.

[10] “A. Krizhevsky, I. Sutskever, and G. E. Hinton” proposed that they trained the images more than 1 million in 900 classes through a deep convolutionary neural network. By testing the data, they achieved the error rate for the dataset is around 25 percent, which is having less error rate compared to the previous existing systems. There are five convolutionary layers in the nerve network, more than 50 million parameters with around 600,000 neurons, with a 1000 way softmax. . It instead uses time consistency at the level of the feature. It enhances the per-frame characteristics by adding the nearby characteristics along the motion paths. Our method improves significantly on strong baselines on single frames, particularly with fast-moving objects in the ImageNet. Without any additional bells and whistles, this framework is designed and consistent with the other best systems to win the 2016 ImageNet VID Challenges.

[11] **“**D. D. Pukale, S.G. Bhirud, V.D. Katkar” proposed that in recent years, the use of computationally intensive Neural Networks for processing large amounts of data has expanded. Researchers employ Deep Convolution Neural Networks to evaluate photos for a range of applications. This research provides a Deep Convolution Neural Network-based Content-based Image Retrieval method. Gray pictures, RGB colour space, and YCbCr colour space images arranged into different clusters are used in the experiments. As a performance criterion, the precision-recall crossover point is used.

[12] “Bongjin Oh, Junhyeok Lee” proposed that based on combining two CNN’s, this study presents an architecture for recognising scene photos. A CNN is used to train huge scene images, while another is utilised to extract items from the scene images. For this assignment, a sequential frame extraction method of films is employed, as well as a deep learning methodology of CNN and Fully Connected CNN. The approach has an accuracy of around 80 percent. During the scene image recognition stage, the object lists are organised by scene classes and utilised as a guide to determine all the top five classes.

[13]"Ulagamuthalvi, J.B. Janet Felicita, D Abinaya" proposed that when a specific scene is presented, the object detection method finds a specific object. Classifiers such as Neural Networks and SVM are used to train classifiers so that they can detect an object when a fresh image is presented. In this study, we offer a model for extracting text from images. The texts are detected and localised using bounding boxes. The model is trained using a neural network, with the training set consisting of a large number of photos with text. The model's performance is evaluated, and also it is discovered that it detects the texts from the image presented. Object identification is a key problem in computer vision, as it seeks to identify common things in images.

[14] "Meghajit Mazumdar, V Sarasvathi, Akshay Kumar" proposed that a strategy for developing an application to recognize objects from various videos is proposed in this paper. The application can categorise the video into one of several genres. It may also detect the specific object displayed on the screen at any given time based on user input. For this assignment, a sequential frame extraction method of films is employed, as well as a deep learning methodology of CNN and Fully Connected CNN. The approach has an accuracy of around 80 percent.

[15] "Ke Chen; Yanying Cheng; Hui Bai; Chunjie Mou; Yuchun Zhang" proposed that this research compares and contrasts five modern detection of fire based on vison. Color detection of flame is used in conjunction with various parameters in these detection of fire systems. Support Vector Machine-Based Image Fire Detection Research Yanying Cheng; Hui Bai; Chunjie Mou; Yuchun Zhang; Ke Chen; Yanying Cheng; Hui Bai; Chunjie Mou; Yuchun Zhang Surveillance for fire detection in comparison to classic detection of fire system based on sensors, detection of fire system based on vison got favor towards them. The extensive usage of detection of fire based on vison has been aided by the popularity and demand for video monitoring in various different venues. The primary way for identifying fire in an image is to look at the color of it. Fire, on the other hand, comes in a variety of colors. Color detection is used with numerous additional approaches to increase the accuracy of detection systems based on fire. Edge

detection, motion detection, area covered by flames, fire expansion are some of the techniques that diverse researchers have integrated and classified the fire and non-fire images correctly. There are also a variety of thresholds that can be utilized to distinguish between fire and non-fire in any frame. Depending on the type of location and its light intensity, these thresholds must be changed. Also, if the difference in successive frames and the area covered by the flames is more than the threshold, it supports the existence of fire.

2.2 Summary/Gaps identified in the Survey

It this study, the most common classification of images was KNN. The SVM was also used with satisfactory results to classify hyperspectral pictures. During the separation of traffic images and accident images, all these traditional image processing methods were difficult to use with regard to different scenarios and troubles. Hinton suggested a way to perform the DBN learning in an unsupervised manner.

Classification method changed its course of research into the deep learning method and moved to the level of artificial intelligence (AI). The unsupervised learning approaches used in medical image analysis included a RBM, DBN and autoencoder. At the ILSVRC 2015, ResNet Architecture first achieved a top5 validation error of 3.57 per cent in combination with six models in different depths. Yu et al. successfully used and improved performance in very deep residual networks to recognise melanoma. For a high accuracy cell staining pattern classification. Pham et al. exploited and effectively applied residual learning to the recognition of human action. Residual learning for traffic applications should be further explored. Most existing systems are regarded as problems of binary class.

3. OVERVIEW OF PROPOSED SYSTEM

3.1 Introduction and Related Concepts:

Deep learning (DL):

Deep learning (DL) is simulator of mechanisms of the human brain for data processing and make decisions by generating patterns. Deep-learning (D L) comes under ML(Machine-Learning) in A.I(ARTIFICIAL INTELLIGENCE) functions which can be able to learn unstructured and unrecognized data without any supervising and is named as D-N-N(DEEPNEURAL-NETWORK) and it mentions transferring the data through many layers from raw input to extract features.

The most well-famed AI techniques are Machine learning which is used for huge data processing and the self-compatible algorithm used to have a better analysis of patterns with old and new data that is added to the algorithm.

Deep learning uses a classified level of AI that is built by an example a brain(human), which has junctions and neurons connected such as lattice which enables systems to process in a non-linear method of data.

Every layer and the levels in deep learning are used to transform its input data into a complex representation and abstraction. Were Image recognition which is used in our project takes a raw input from the user it may be pixel matrices in which the image process through several layers in which the first layer used to extract and encode edges of the picture and later encodes the arrangement of the picture edges and extract features respectively and this process is learning by itself by repeating or while going through each level of algorithm.

To complete this, DL accompanies by algorithms that are multi-layered in a structure called neural networks. And it can perform some tasks to complete and understand data. Every layer can also be taught as a filter how to work specifically to obtain results. reference the forecast of cardiovascular sicknesses.

CNN

Convolutional Neural networks (CNN) are a form of neural network commonly used in computer viewing. Its name originates from the secret layers of its type. Instead of the usual activation function, it uses convolution functions and pooling Before understanding convolution functions and pooling functions, we have to understand how to use computer vision techniques in order to conceptualize the hidden mechanism of the CNN.

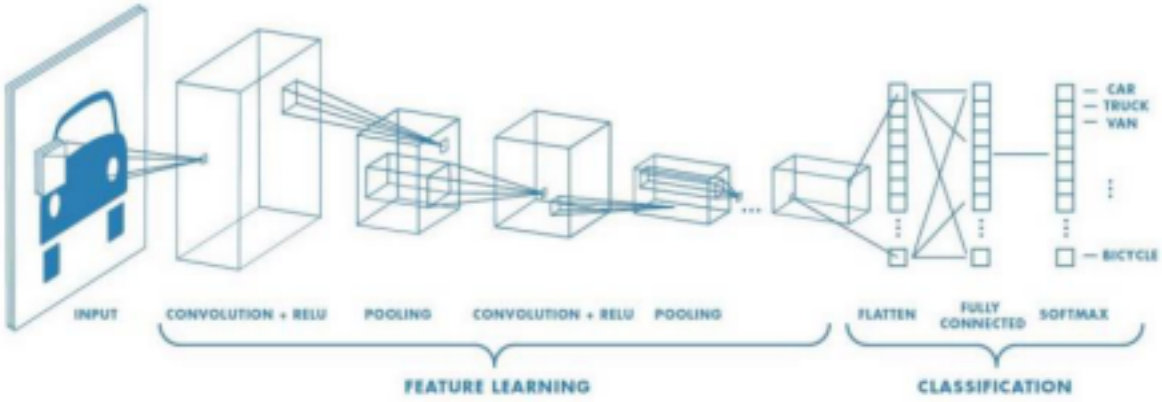
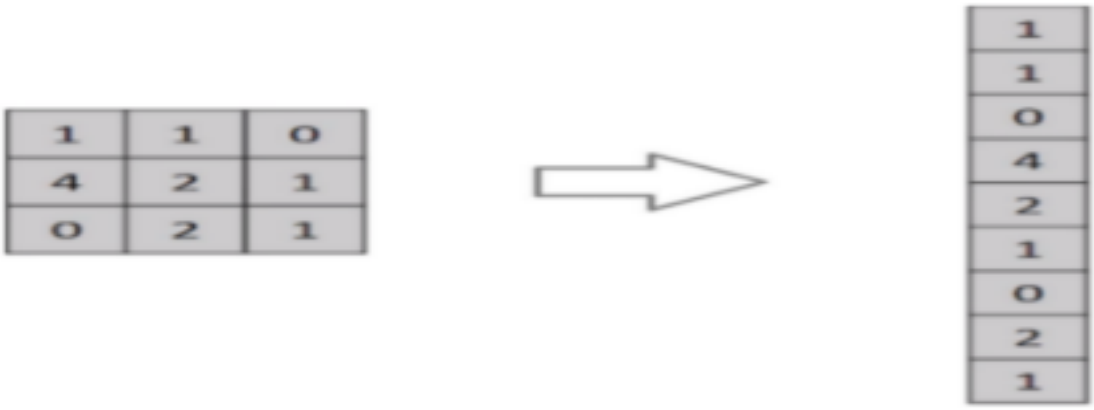


Fig:3.1 Flow chart of CNN Algorithm

The gap between the machines and humans were bridged by AI and the focus of researchers and developers turned to the main field of Computer vision and the advancements of DL has increased day by day on Image and video recognition, NLP, etc. were based on one algorithm which is Convolution Neural Network. Deep learning algorithms which enemrate weights of given resemblance.

ConvNet is used while classification is used for learning the filters. ConvNet is used to three-dimensional and progressive dependencies the image resolution of the image part without any feature default and for prediction accuracy. Feed forward is used to flatten the dimension of the image.



Flattening of 3*3 image into 9*1 vector

Fig3.2 feed-forward neural network

Convolution layer:

This layer is used to extract the details and the features. Before the start of the algorithm, a kernel is used to compare it to the image dimensions and shifts through the image, and matrix multiplications occur over the image. Kernels usually are 3x3 matrix and ConvNets can also be used to all Convolutional layers and for low-level feature extraction Layer 1 of convolution and by adding layers can familiarize the High-level features.

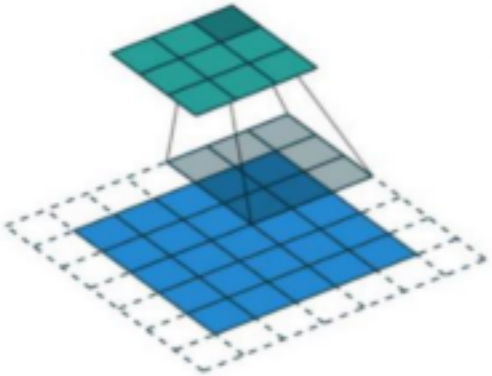


Fig 3.3 convolution operation with Stride length=2

Padding:

Two types of results can be expected during the image pass through these layers were the Same padding where we resize it and apply kernel over for convolved matrix of giving input dimensions and Valid Padding were, we the same operation repeats but without padding.

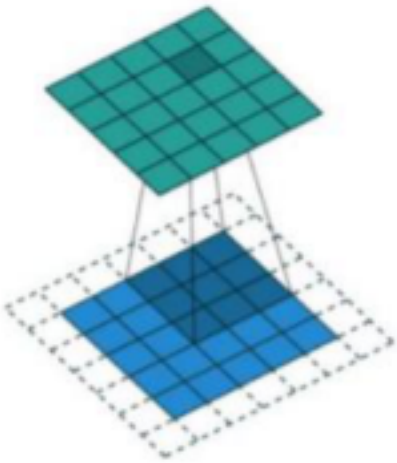


Fig 3.4 padding

Non Linearity (ReLU):

The usage of ReLU is for non-linearity in a ConvNet. ConvNet can learn from real world would and be positive linear values. Additional non-linear functions, which are sigmoid or tan h , and used in place of ReLU. ReLU is used by the bulk of data scientists because it outperforms the other two.

$$A_i^k = f(X_i^{(k)}) = f\left(\sum_{c=1}^{C} W_i^{(c,k)} \otimes X_{(i-1)}^c + B_i^k\right)$$

Pooling Layer:

Convolution layer is interchangeable like p.layer so that reduction reduction occurs of the given image and for processing the data it reduces the computational power throughout the dimensional reduction and feature extraction while training the model. Two types of pooling Average pooling which is for after convolved matrix on averaging the values of the kernel for Noise reduction and Max pooling yields the maximum values from the image kernel and this also used to and de-noising the reduced dimensions of the image. In our project, we use max pooling.

$$P_i^k = \max_{\substack{(j-1)S+1<t_x\leq jS \\ (j-1)H+t_y\leq jH}} \{A_i^k(t_{x,y})\}$$

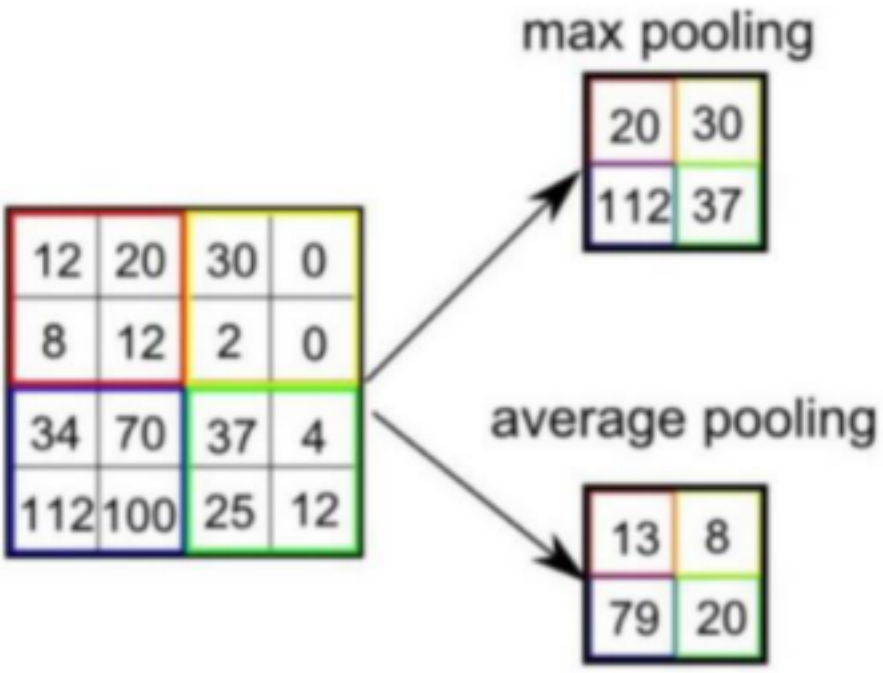


Fig 3.5 Pooling layer after kernel added to dimentions

Fully Connected Layer:

For a non-linear combination representing high-level features for learning to convolutional layer. And convert the input image to a column vector and the flattened output while training the model. And the number of times the algorithm through training data set is an epoch, series of epochs the model classifies the low-level features for images using Soft max technique in classification.

$$P(y=j|\boldsymbol{x}) = \frac{\exp[\sum_{m=1}^{M} \widehat{P_m}\ (y=j|\boldsymbol{x})]}{\exp[\sum_{k=1}^{K}\sum_{m=1}^{M} \widehat{P_m}\ (y=k|\boldsymbol{x})]}$$

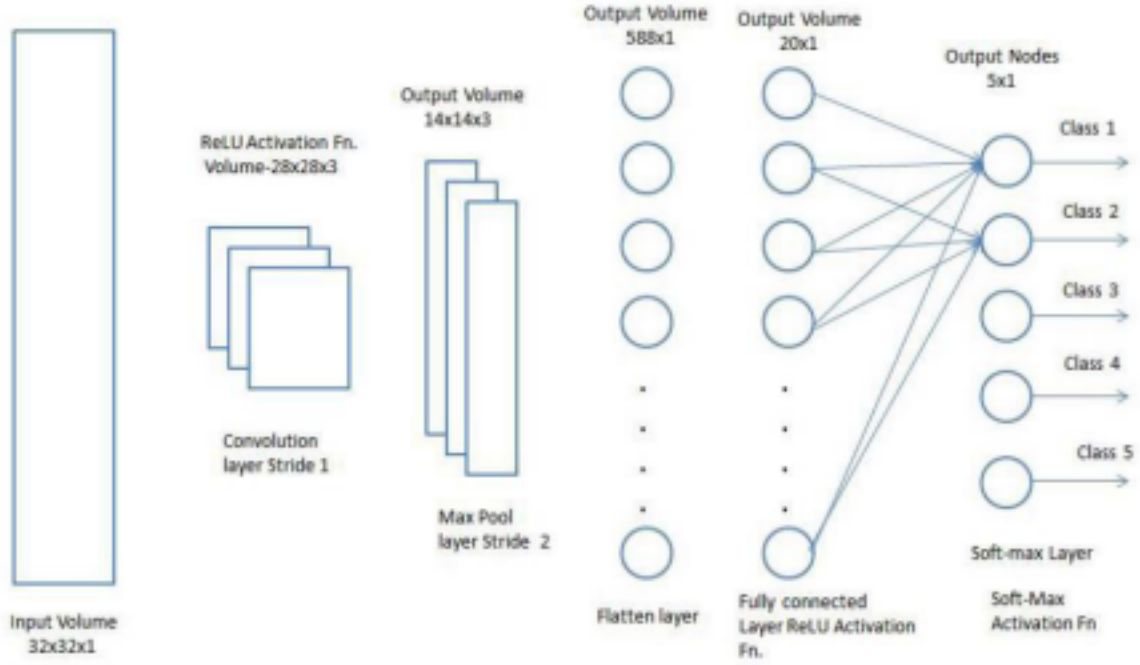


Fig 3.6 Fully connected layers

3.2 Framework, Architecture or Module for the Proposed System:

There are four modules in our proposed system. They are:

- Collecting the Dataset.
- Image preprocessing.
- Applying CNN
- Recognition

3.2.1 DATASET COLLECTION

Web images are collected from different classes of the input traffic scene. The output value of class of scenes is provided together with the collection of images. We created four folders. They are intense traffic, Low traffic, normal accident, Fire accident. Each directory includes 900 images for training purpose and validation purposes. The name of the folder is same as the class value for the output classification.

3.2.2 IMAGE PRE-PROCESSING

In this implementation, there is no need for much pre-processing. The data set for training and testing is classified as input via the "Flow from directory" keras function. This provides the pre-process necessary, for example dimensional reductions. In the same way, the test input image is a dimensional reduction and a numpy array conversion.

3.2.3 TRAINING USING CONVOLUTIONAL 2D NEURAL NETWORK

We used convolutional 2D neural network available in keras for training and testing our model. The overall architecture of Conv2D is shown below.

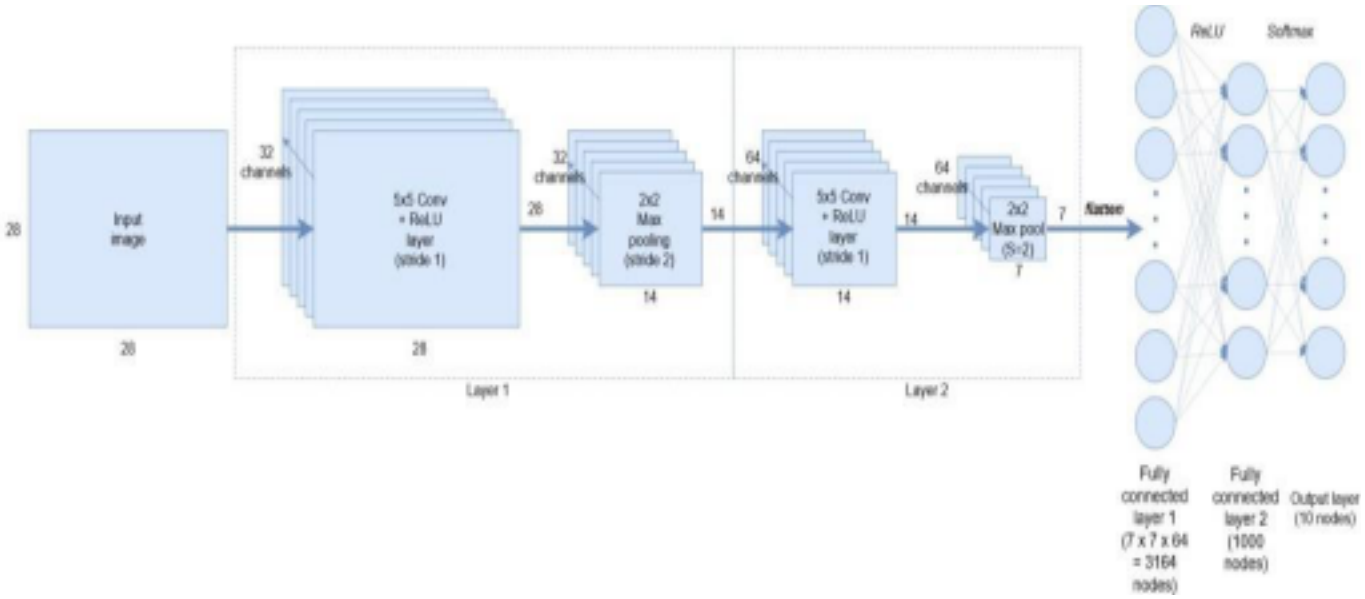


Fig .3.2.1 Sequential Model

Sequential Model:

Keras models are in two forms one is sequential and other one is functional. The sequential model is likely for deep learning networks. It allows the sequence of layers of the network to be easily stacked from input to output.

The first line declares sequential model type ().

Adding 2D Convolutional layer:

In processing 2D images we should add a 2D convolution layer. The count of channels for output is the first parameter given to Conv2D(). This is 32 output channels. Kernel size, which is a 5-to-5 moving window in this case, followed by steps in x and y(1, 1). Rectified linear unit, also we must lastly provide the model with input size for the layer. The input shape only needs to be declared by the first layer, Keras is sufficiently good to calculate the size of tensors from there.

Adding 2D max pooling layer:

The next step is to add the above mentioned layer. We just indicate the size of the bundle in x and y directions and strides in the above case.

Adding another convolutional + max pooling layer:

Next, with 64 output channels, we add another convolution + max layer of pooling. In the function Conv2D(), the default strides argument is (1, 1), so we can leave it out. Keras's default strides argument is to match the pool size. For this layer the input tensor is (batch size, 28, 28, 32) - the image size is 28 x 28 and the output number is 32. The output of the previous layer is 32.

Flatten and adding dense layer:

We need to enter into fully connected layers. In order do that we need to flatten the output. In line with our own architecture we specify 1000 nodes in the next two lines – we use the Dense() layer in Keras, each of them activated with a ReLU function. The second is the classification of our soft-max or output layer of the number of our classes.

Training neural network:

We have to specify, in the training model, the loss function or what optimizer the framework can use (i.e.Adamoptimiser, Gradient descent etc.).Crossentropy standard function (keras.losses.categoricalcrossentropy) for categorical classification. The Adam optimizer is used (keras.optimizers.Adam). Finally, when we run evaluate() on our model, we can state a metric that is calculated.We pass on our training data for the first time in this case. The batch size is the next argument. We use a batch size of 32 in this case. Then we go through the no. of training periods.

3.2.4 RECOGNITION

Finally, we transfer validation/test data to the appropriate function, convert the input image into numpy array to a classified output that consists of dense traffic, sparse traffic, fire accident or normal accident.

SYSTEM ARCHITECTURE:

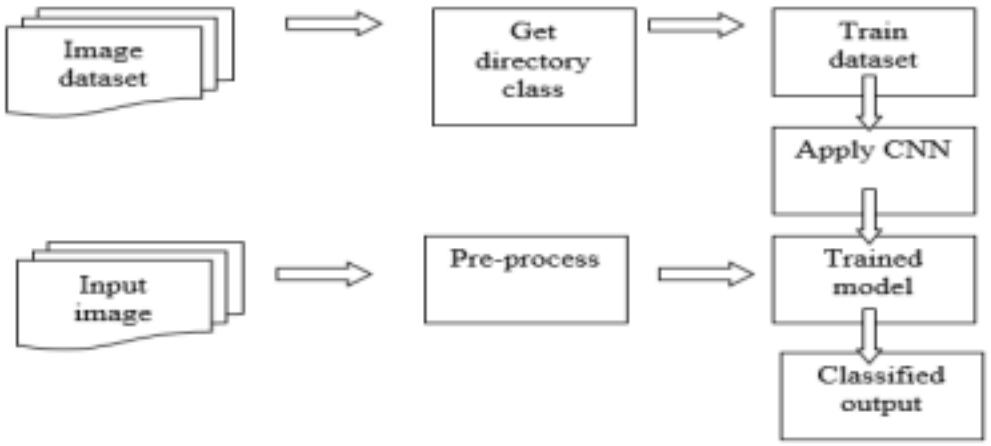


Fig 3.2.4 SYSTEM ARCHITECTURE:

3.3. Proposed System Models

USE CASE DIAGRAM

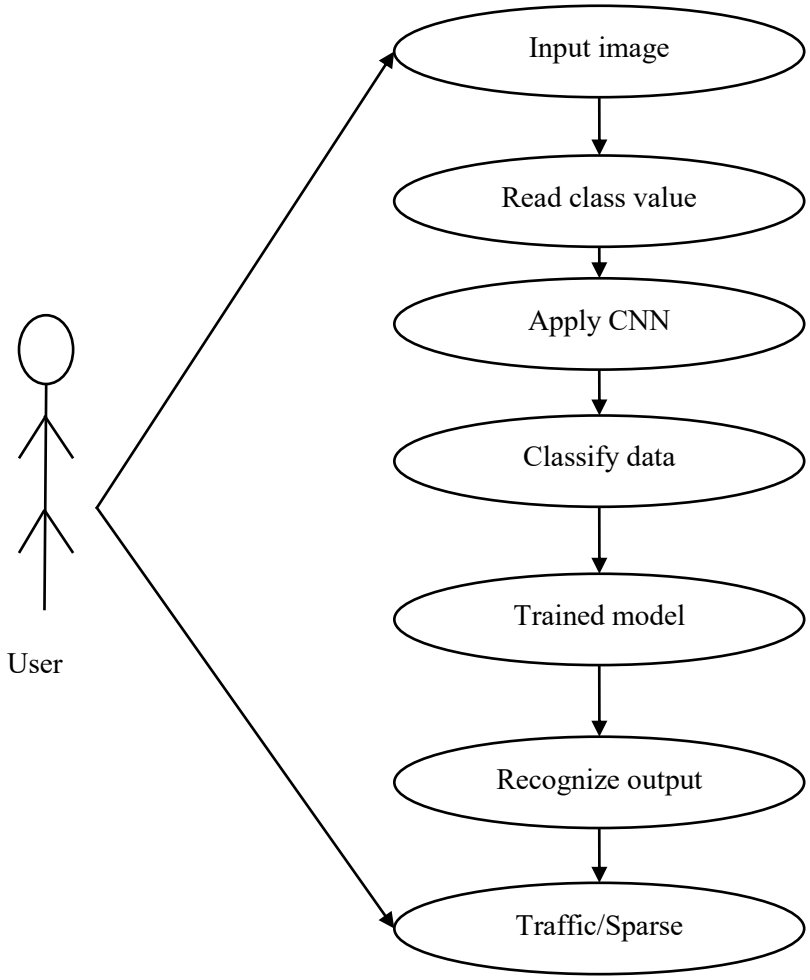


Fig3.3.1: Use case Diagram

The figure above is the Traffic system's use case diagram, where user uploads image of traffic scene from folder, the algorithm work to generate the identified output as dense traffic or sparse.

SEQUENCE DIAGRAM

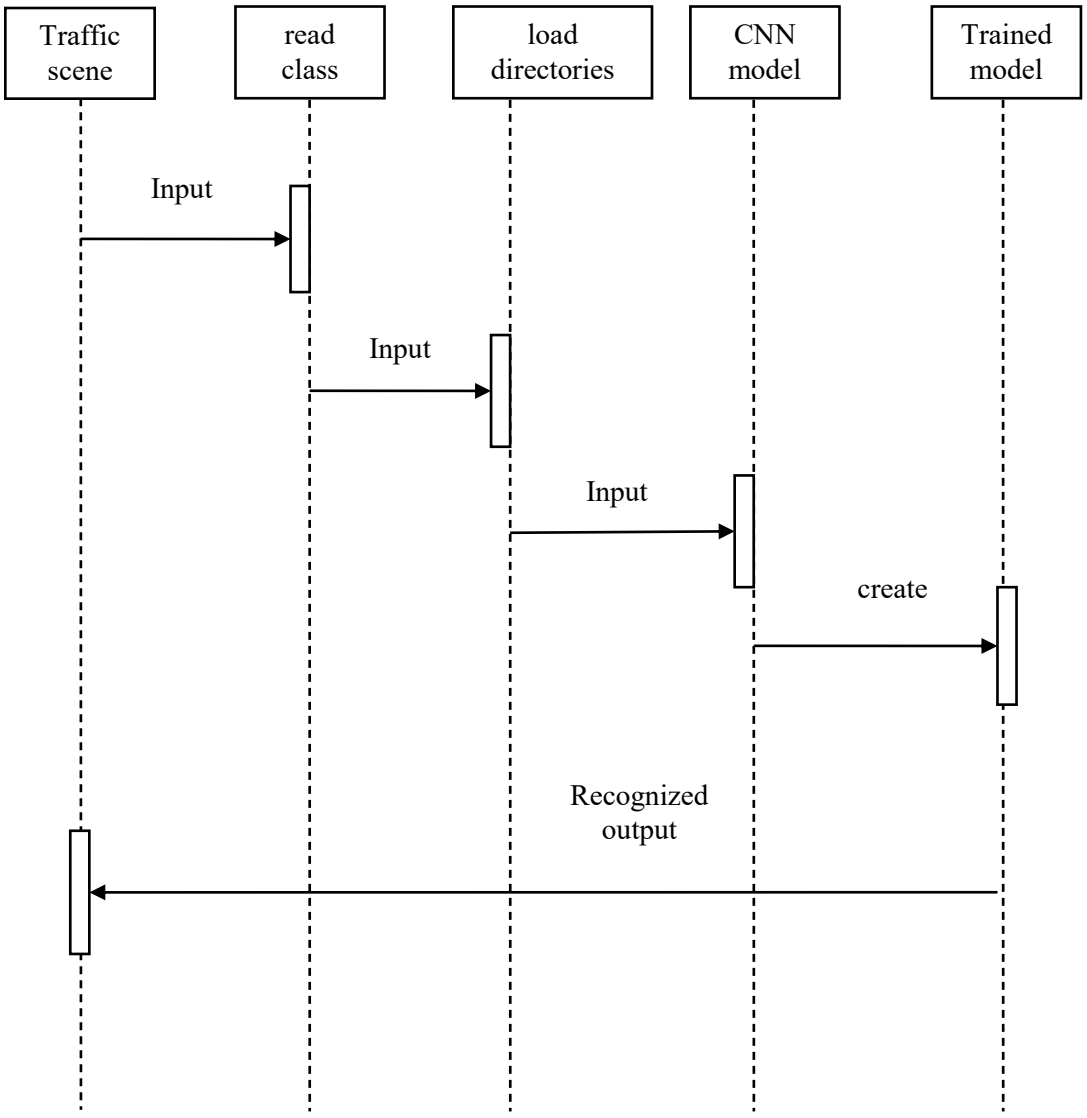


Fig3.3.2: Sequence Diagram

The figure above determines the sequence of how the traffic dataset is trained. The traffic dataset is trained by applying CNN model.

COLLABORATION DIAGRAM

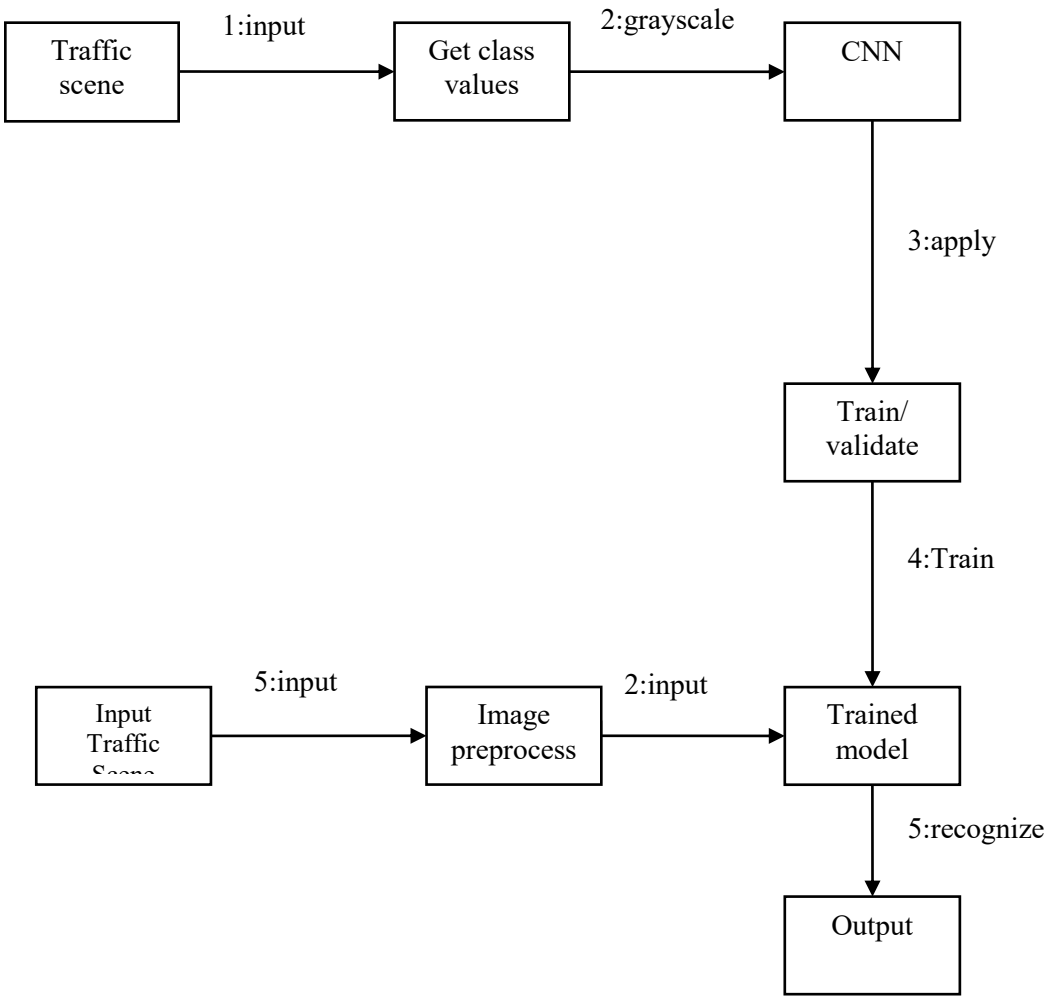


Fig3.3.3: Collaboration Diagram

The figure above shows the collaboration graph of the system proposed. It contains order of how the traffic dataset is trained by applying CNN model and the order of image upload by the user, preprocess it and type of image is identified.

ACTIVITY DIAGRAM

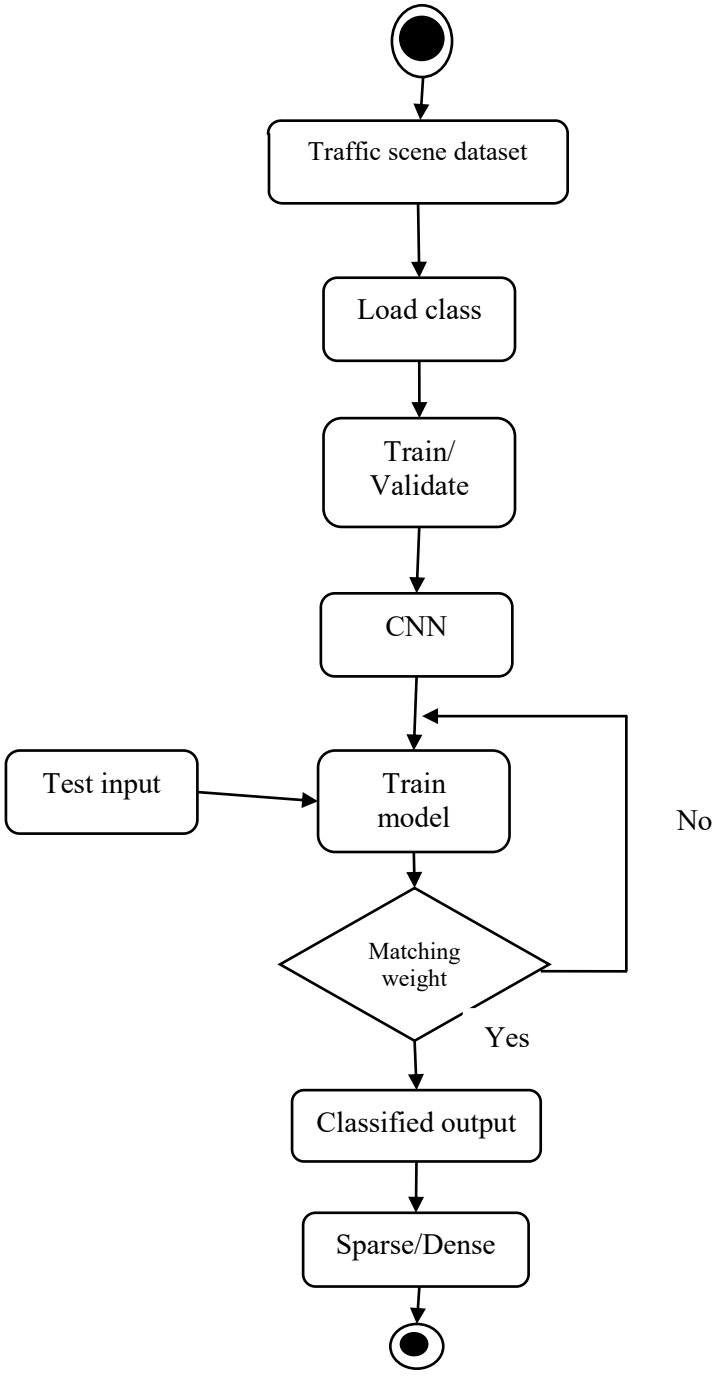


Fig3.3.4: Activity Diagram

The above activity diagram depicts the various activities done to get the type of traffic or type of accident predicted.

DEPLOYMENT DIAGRAM

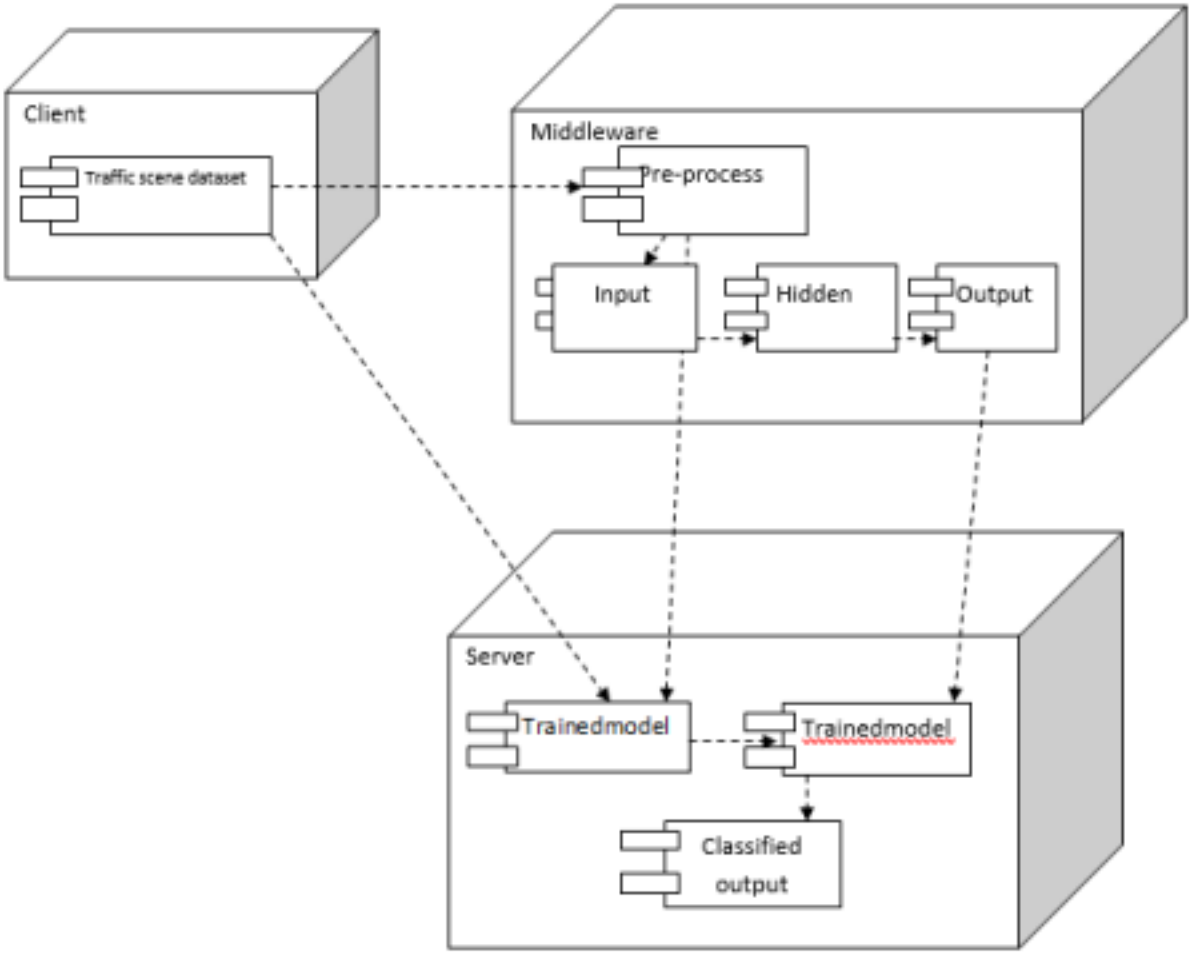


Fig3.3.5: Deployment Diagram

The figure above is the deployment diagram which connected the client, server and the middleware.

4. Proposed System Analysis and Design

4.1 Introduction

DETAILED DESIGN

The Neural Network is generally made up of several hidden layers. In most Conv2D, the hidden layers are multiplied by different random weights between 0 and 1 with three layers with 32 and 64 neurons respectively, But the Conv2D Network had a design in which two hidden layers are identical, and each hidden layer consists of a broad set of neurons, i.e. 256 dense neurons are used. We have promising results by using this deep network.

Table 1: Sample data representation of labeling.

Layer operation	No of feature images	Feature map size	Validation steps	Total parameters
Conv2D	900	150X150	300	900

We have used five layered CNN models with Relu Optimisation in this test phase. In them we used five levels of Convolutionary Neural Networks (CNN) model: a layer for convolution or a layer for max pooling, or a layer for sub sampling.

4.2 Requirement Analysis

4.2.1 Functional Requirements

4.2.1.1 Product Perspective

This project is a web-based portal that is used to report the type of traffic or type of accident in a particular area. The web application should be compatible with all browser platforms. User uploads the image or a video of the traffic present in his locality. The location of the traffic found by the user Data has to be stored in a database following all the ACID properties.

4.2.1.2 Product features

Cross-platform support: supports most known and commercial operating systems. Operating support.

Detailed categorizations of products: Exact class categorization for the user shall be shown by the system.

Detailed user report: The system shall provide the user with immediate traffic.

4.2.1.3 User characteristics

This project is aimed to be used by the traffic management system where they can get to know the type of traffic or type of accident through online portal and takes required action. It will be designed in a way that even an illiterate can also make use of the product. But this application which we are developing is mainly aims to the traffic system as they need to keep up to the technology of current generation.

4.2.1.4 Assumption & Dependencies

To access this web application internet is required.

4.2.1.5 User Requirements

To create a portal lets user to know the type of accident or type of traffic. To create a web-based transport system. To get the image of video prediction with the accuracy of more than 80%.

4.2.2 Non Functional Requirements

4.2.2.1 Product Requirements

4.2.2.1.1 Efficiency

This application must store an image of garbage even bigger than 5MB. This application can store a total of 300 images depending on the free space available. This application must predict the type of image or accident within 2 seconds. This database can be scaled to add more images of garbage.

4.2.2.1.2 Reliability

For any number of users input the system must produce accurate results. Also, the application needs to be available all the time.

4.2.2.1.3 Portability

The application is a full stack web application. Hence, it is compatible to all the browsers that support and render web pages. It is easily portable as it is implied on a regular computer. The user can access the computer from the place where the system was installed.

4.2.2.1.4 Usability

Our portal is intuitive and self-explanatory. It has an easy workflow of tasks that can be done by the user. The software system is simple to comprehend, discover and manage. The portrait as signal can be provided by the reader.

4.2.2.2 Organizational Requirements

4.2.2.2.1 Implementation Requirements

The system needs a python supporting environment along with tensorflow, openCV and pandas installed in it for performing some operations using image values.

4.2.2.2.2 Engineering Standard Requirements

One must have basic knowledge and understanding of the concepts of Machine Learning, Deep Learning, Python and image processing.

4.2.2.3 Operational Requirements

- **Economic**

  This project will determine the economic impact of the system on the company. The costs incurred during the project development phase must be justified. Consequently, the system built was made budgetary by the fact that most of the technologies used were readily available. The only items to be purchased were personalized items.

- **Environmental**

  This project needs an explicit computer to process the data that is being recorded in traffic cameras.

- **Social**

  The main social requirement of the project is the acceptance rate of the user. The user using the application shouldn’t rebel with the application but instead should accept the product and promote the product if necessary.

- **Political**

  The government can also support implementation of this by allocating dedicated ambulance services to this system so that response would be quick.

- **Ethical**

  The application is to adhere and oblige all the ethical standards set by the ethical board committee. This product's use does not include any unethical practices and adheres to all legal societal prosperities; as long as it is used in a good manner, there will be no ethical issues.

- **Health and Safety**

  Implementation of this in traffic video monitoring could actually save many lives by responding immediately to accidents, as humans cannot always check everything.

- **Sustainability**

  Since there is no confidentiality in data collected and obtained there would be no problems like theft and hacking.

- **Legality**

  It is published as an open-source software which can be implemented in a traffic monitoring system.

- **Inspectability**

  The application must be inspectable which will be ensured by the admin/head of the command chain.

4.2.3 System Requirements

4.2.3.1 H/W Requirements

- Intel Pentium Processor or higher.
- Recommended Intel core i3 gen
- Recommended min 1GB RAM or higher.
- Recommended min 10 GB disk storage space or higher

4.2.3.2 S/W Requirements

- Windows 7 or higher
- Python 3.6 and related libraries
- OpenCV
- TensorFlow

5. Results and Discussion

EXPERIMENTAL EVALUATIONS

The experiment is carried out with the input parameters in the convolution neural network. Our model is tested on the input of 200 images on each class once the training is completed. The experimental results show that 80 percent of our model is accurate. We used an epoch of 20, with a number of epochs, the precision increased.

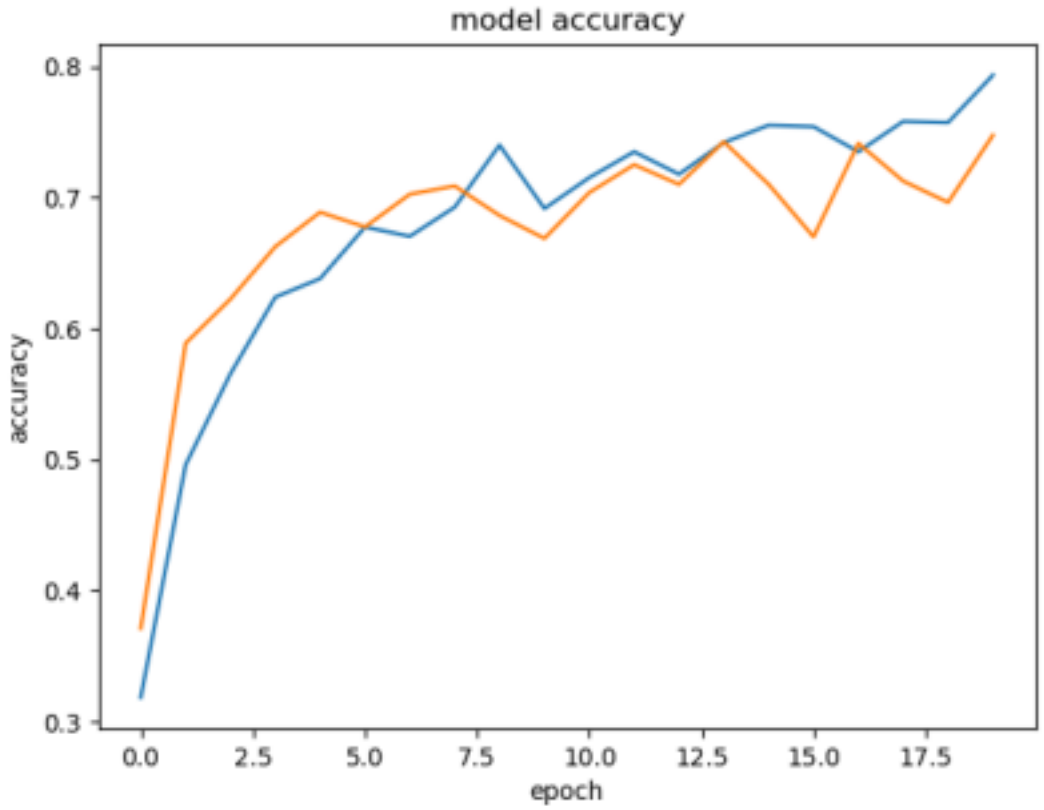


Fig5.1 : Model Accuracy

The following figure shows the model loss in our experimental method.

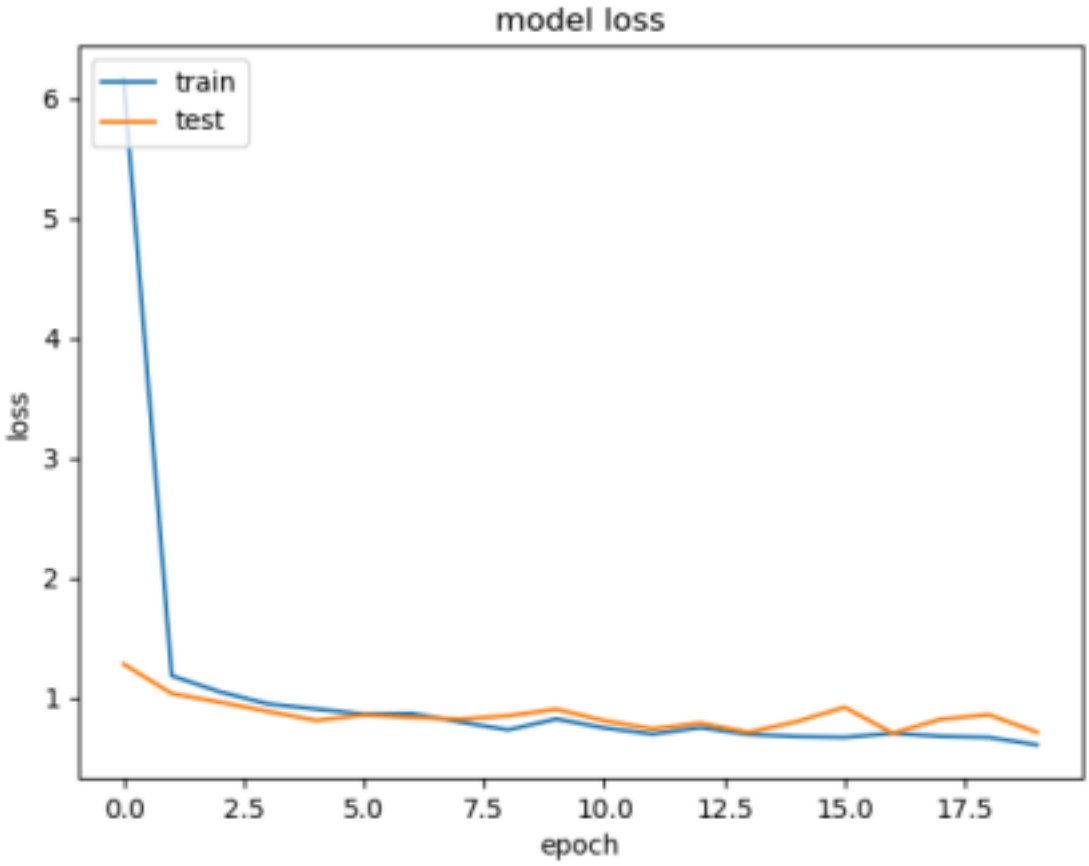


Fig 5.2: Model Loss

This project is mainly based on CNN algorithm. There are four classes of images. They are dense traffic, sparse traffic, fire accident and normal accident. We consider the high and low traffic detection under traffic section and the fire and accident are considered under the accident section. Our main objective is to classify the given input image or video into high traffic, low traffic, fire accident and normal accident by obtaining the trained model from input traffic and accident dataset. First, the video is divided into frames using openCV and then series of operations such as dimension reduction, noise reduction, are done on the image and passed through the trained CNN model to identify the image. This live process identification form the

surveillance cameras can predict traffic and suggest best traffic routes for the commuters and also save lives by immediately responding to the accidents and fire accidents on road.

5.3 Screenshots of Implementation

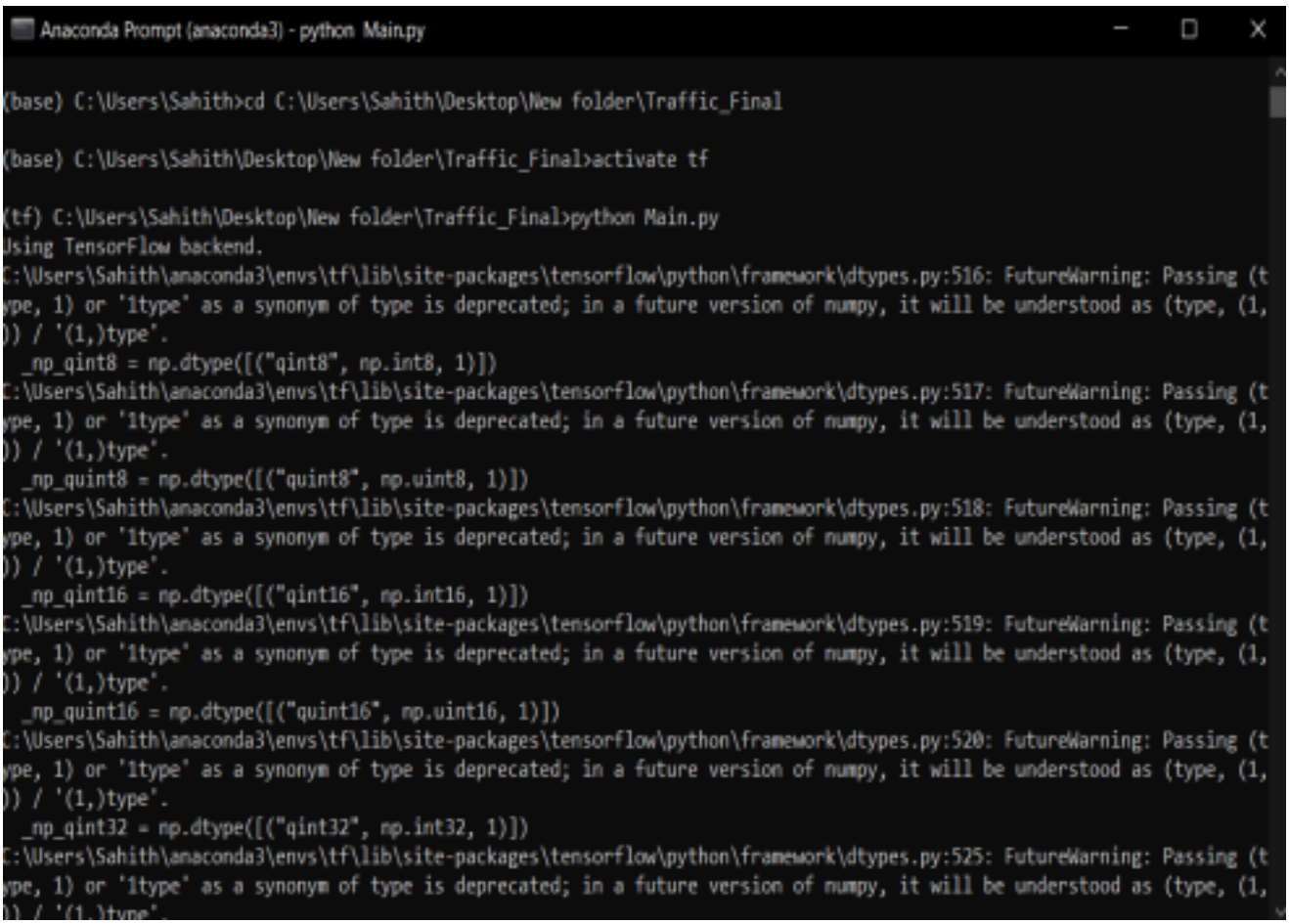


Fig 5.3.1 Activating tensor flow

The following screen shows the main page of application

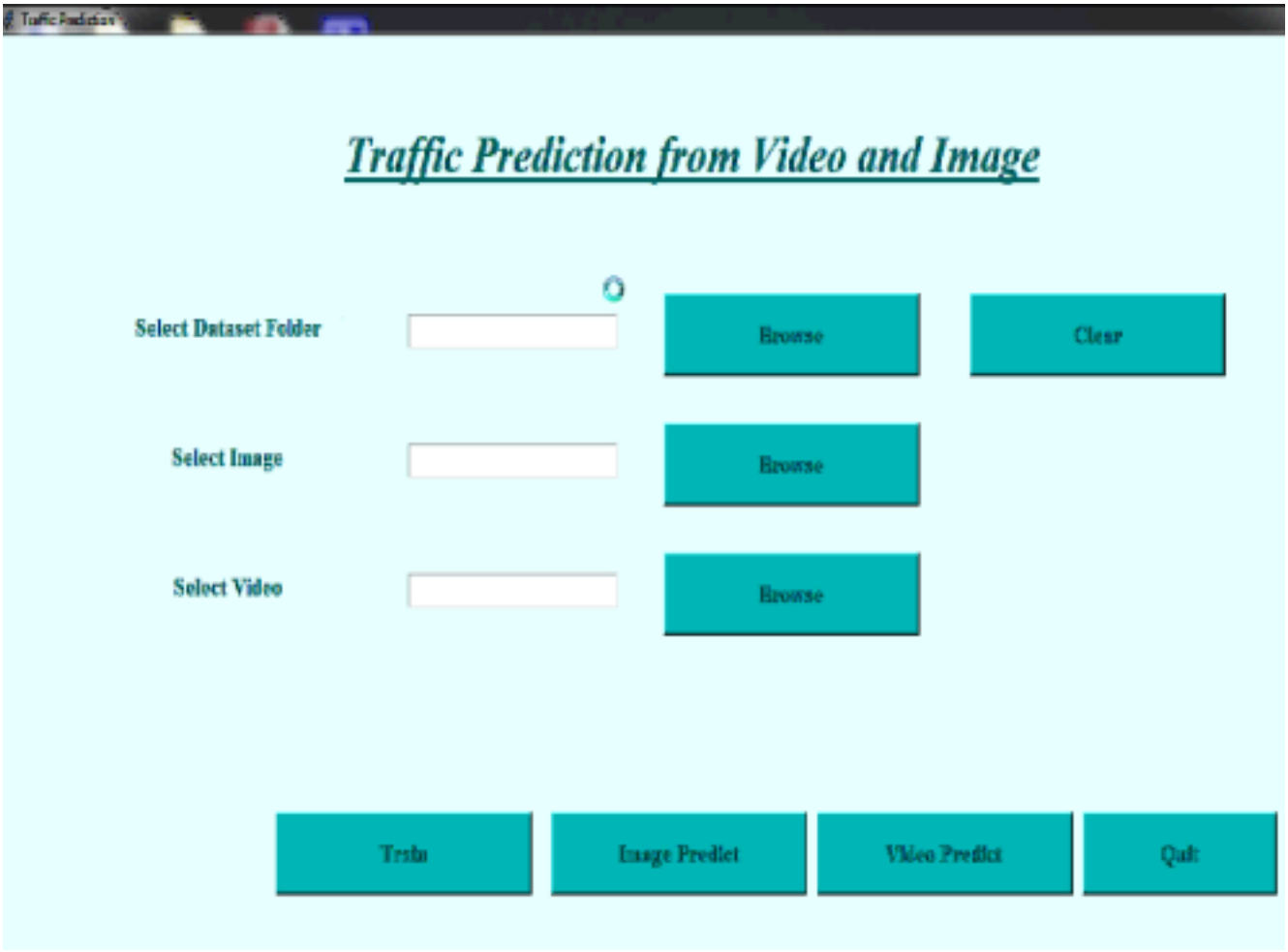


Fig 5.3.2

The following screen shows that the dataset is loaded in the application for train

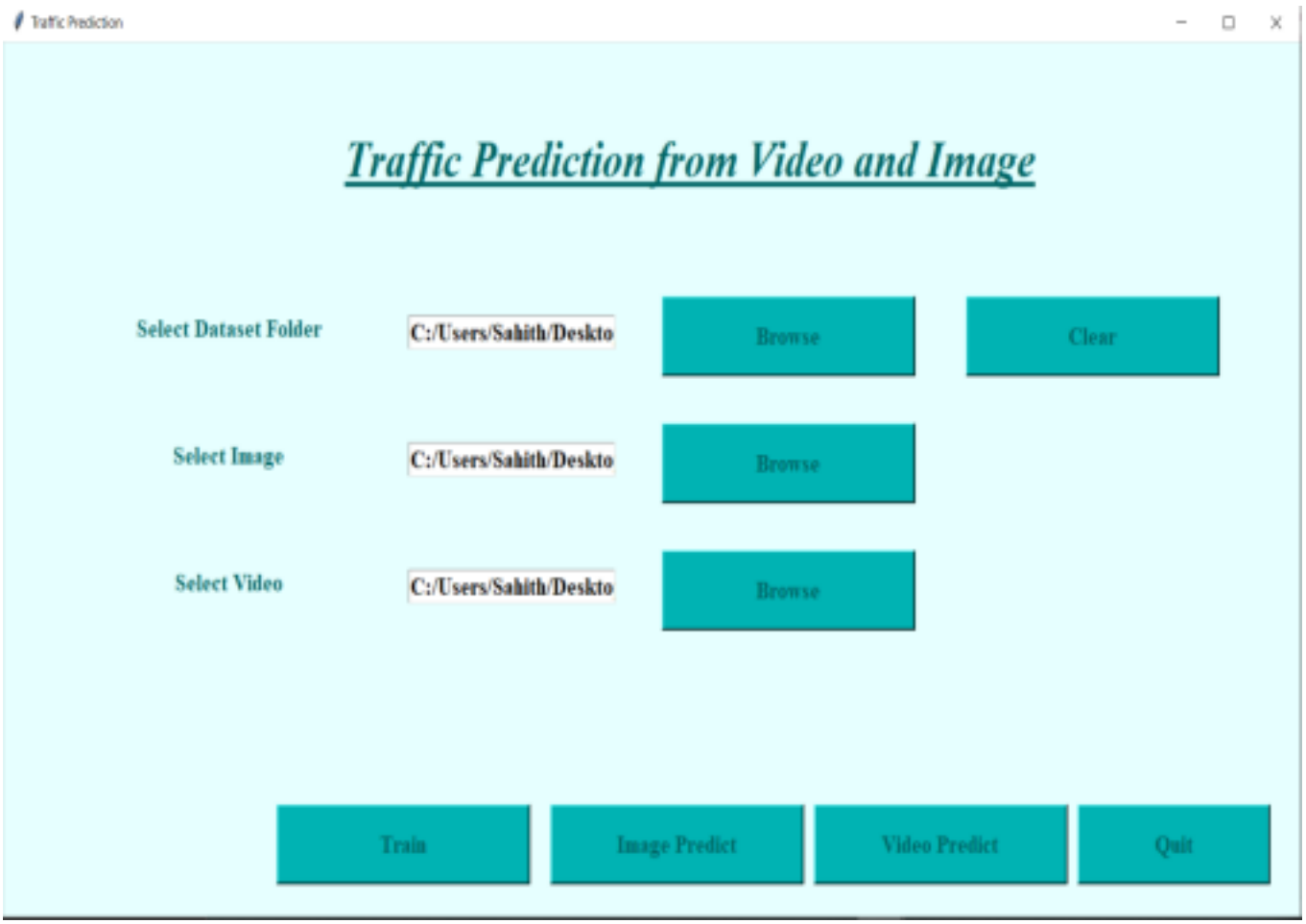


Fig 5.3.3

Training of dataset for 20 epoch is done

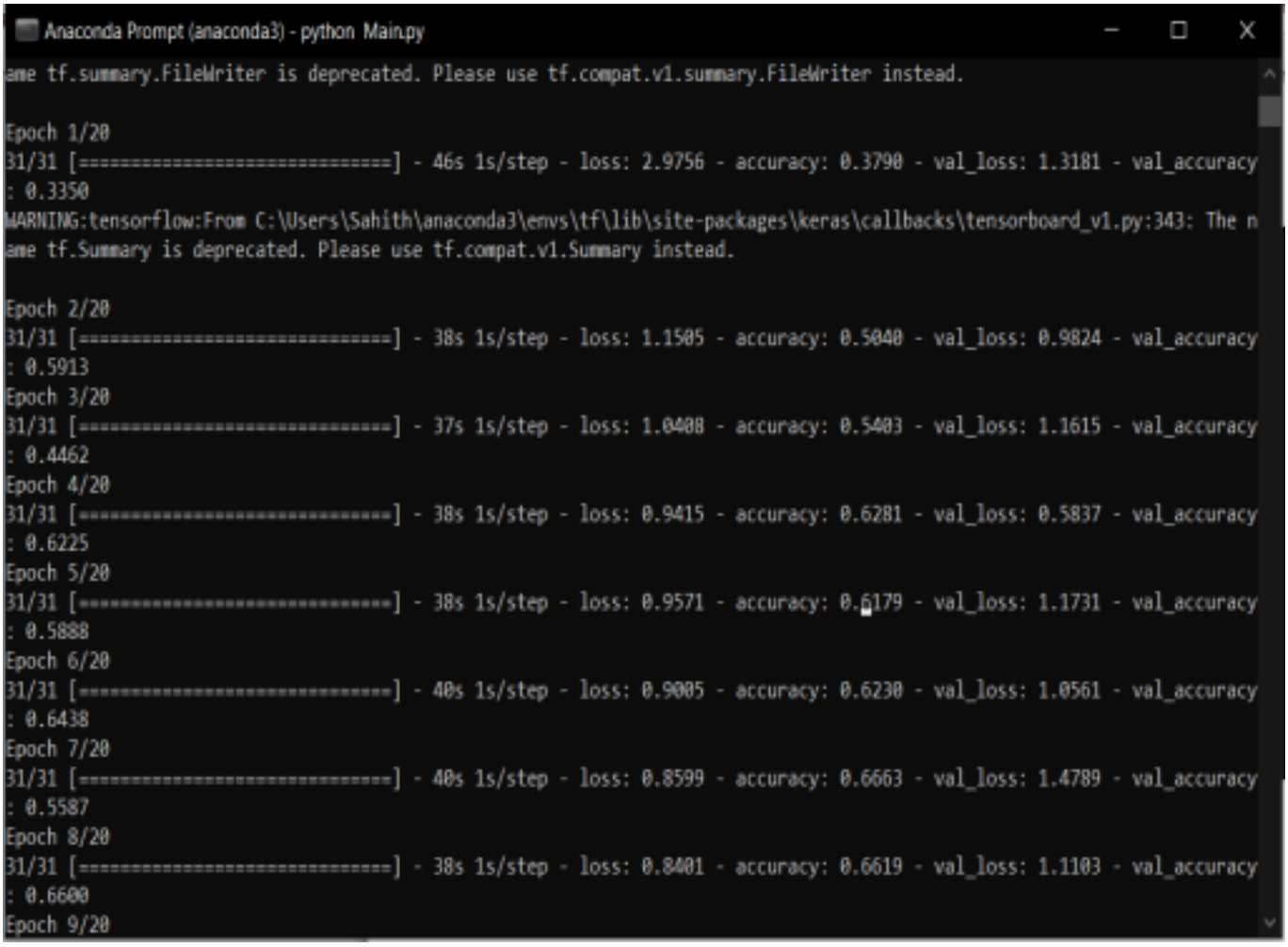


Fig 5.3.4.1

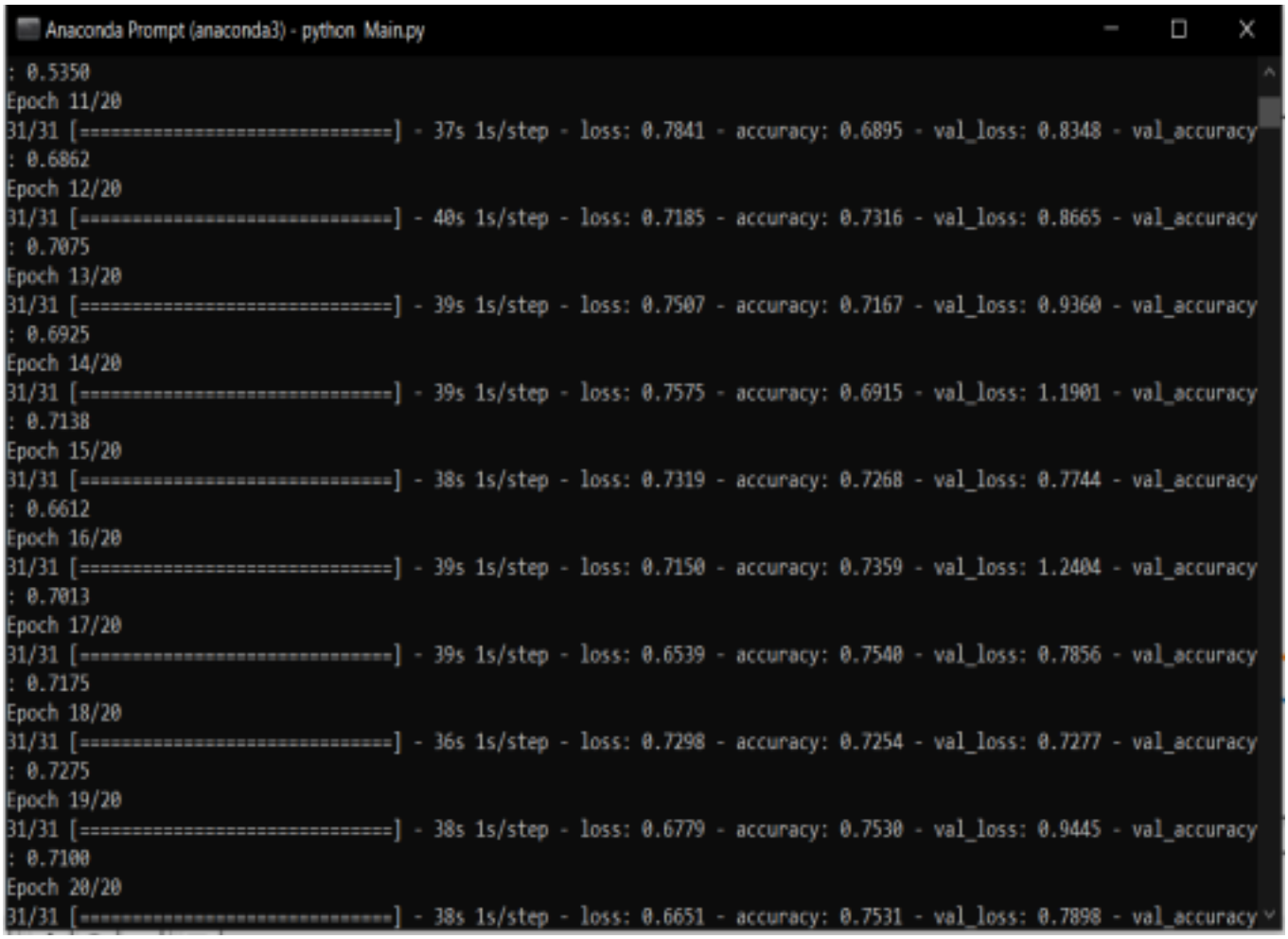

Anaconda Prompt (anaconda3) - python Main.py
: 0.5350
Epoch 11/20
31/31 [==============================] - 37s 1s/step - loss: 0.7841 - accuracy: 0.6895 - val_loss: 0.8348 - val_accuracy
: 0.6862
Epoch 12/20
31/31 [==============================] - 40s 1s/step - loss: 0.7185 - accuracy: 0.7316 - val_loss: 0.8665 - val_accuracy
: 0.7075
Epoch 13/20
31/31 [==============================] - 39s 1s/step - loss: 0.7507 - accuracy: 0.7167 - val_loss: 0.9360 - val_accuracy
: 0.6925
Epoch 14/20
31/31 [==============================] - 39s 1s/step - loss: 0.7575 - accuracy: 0.6915 - val_loss: 1.1901 - val_accuracy
: 0.7138
Epoch 15/20
31/31 [==============================] - 38s 1s/step - loss: 0.7319 - accuracy: 0.7268 - val_loss: 0.7744 - val_accuracy
: 0.6612
Epoch 16/20
31/31 [==============================] - 39s 1s/step - loss: 0.7150 - accuracy: 0.7359 - val_loss: 1.2404 - val_accuracy
: 0.7013
Epoch 17/20
31/31 [==============================] - 39s 1s/step - loss: 0.6539 - accuracy: 0.7540 - val_loss: 0.7856 - val_accuracy
: 0.7175
Epoch 18/20
31/31 [==============================] - 36s 1s/step - loss: 0.7298 - accuracy: 0.7254 - val_loss: 0.7277 - val_accuracy
: 0.7275
Epoch 19/20
31/31 [==============================] - 38s 1s/step - loss: 0.6779 - accuracy: 0.7530 - val_loss: 0.9445 - val_accuracy
: 0.7100
Epoch 20/20
31/31 [==============================] - 38s 1s/step - loss: 0.6651 - accuracy: 0.7531 - val_loss: 0.7898 - val_accuracy

Fig 5.3.4.2

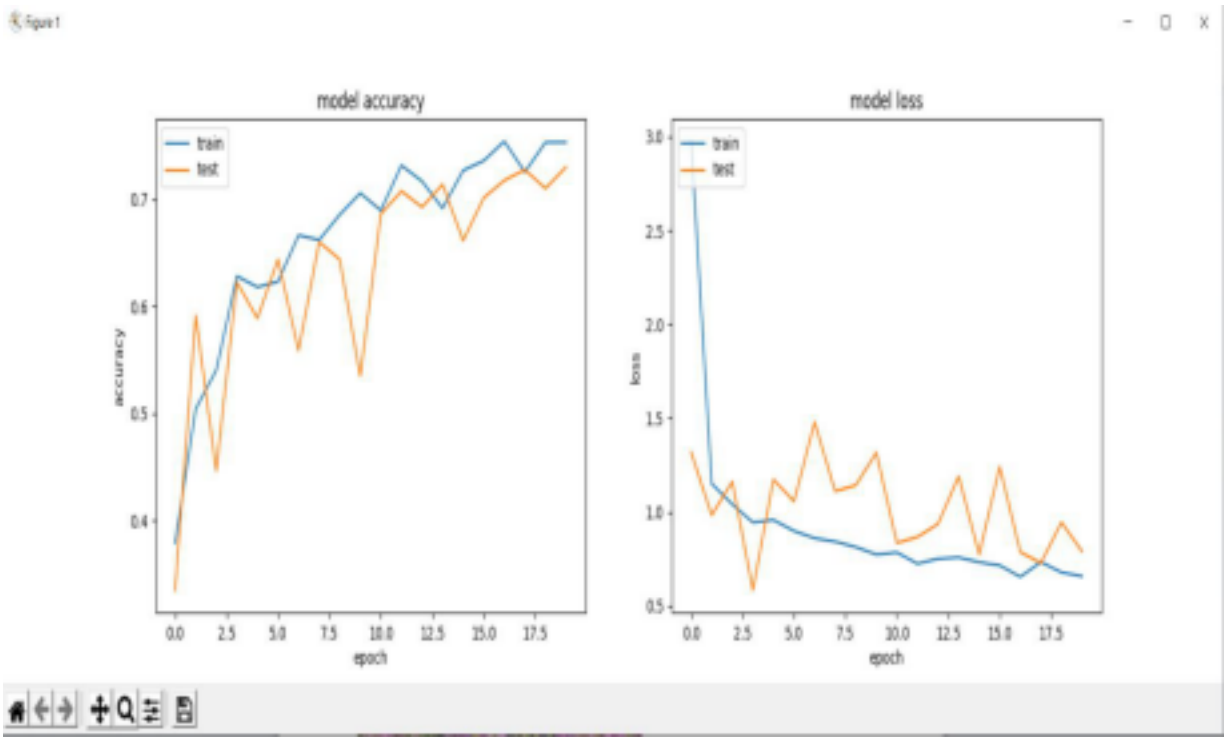


Fig 5.3.5 model accuracy and model loss

Predicted output embedded on the image

Fig 5.3.6.1 predicted heavy traffic

Fig 5.3.6.2 predicted heavy traffic

Fig 5.3.6.3 predicted accident

Fig 5.3.6.4 predicted accident

Fig 5.3.6.5 predicted low traffic

Fig 5.3.6.6 predicted low traffic

Fig 5.3.6.7 predicted fire accident

Fig 5.3.6.8 predicted fire accident

Output predicted for a vedio by dividing into frames

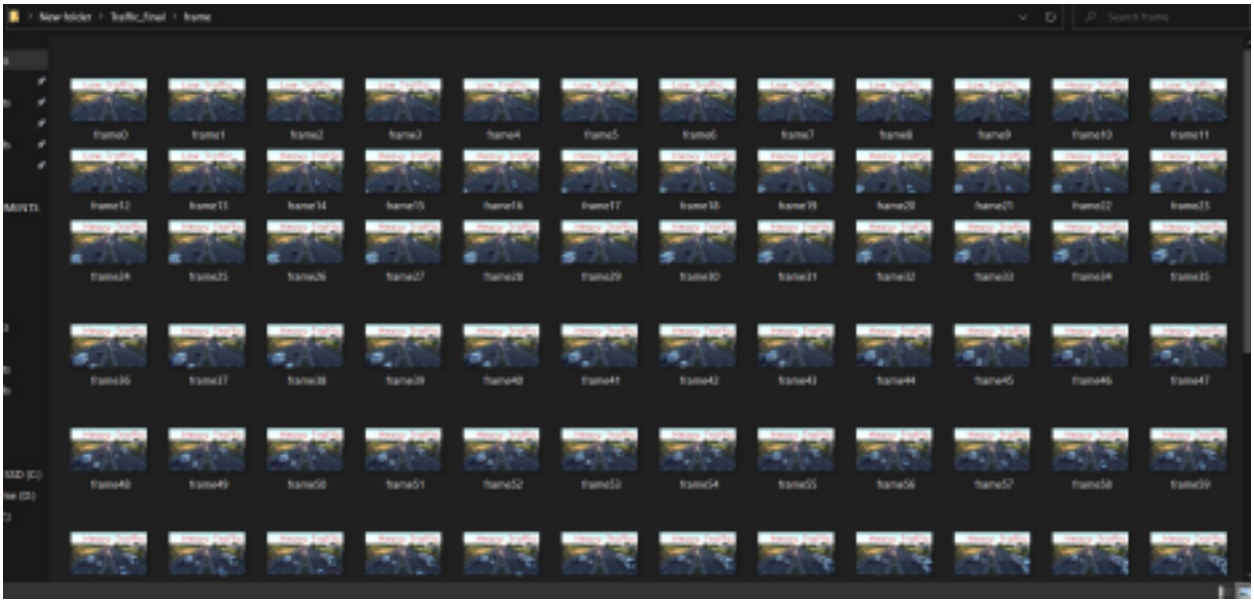


Fig 5.3.10 predicted video frames

Fig 5.3.11 console output

Test Case Analysis

SNO	ID	Test Description	Test Procedure	Name of the program executed.	Expected Result	Actual Result
1	01	Dataset input	Train button is clicked before loading dataset	Main.py	Load Dataset details	Alert to "Select Dataset"
2	02	Training	Train button is clicked after loading dataset	Main.py	Images input from dataset folder and start training	Train Started
3	03	Image input	Click on Image predict button before loading input	Main.py	Load Input details	Alert to "Select Input Image"
4	04	Image Output	Image predict button is clicked after loading input	Main.py	Input details to be loaded	Output of classification embedded on the Image

6. REFERENCES

[1] H. Lei et al., ``A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning," Pattern Recognit., vol. 79, pp. 290302, Jul. 2018.

[2] P. Wang, L. Li, Y. Jin, and G. Wang, ``Detection of unwanted traffic congestion based on existing surveillance system using in freeway via a CNN-architecture trafficNet," in Proc. 13th IEEE Conf. Ind. Electron. Appl., May/Jun. 2018, pp. 11341139.

[3] X. Zhu, Y.Wang, J. Dai, L. Yuan, and Y.Wei, ``Flow-guided feature aggregation for video object detection," in Proc. ICCV, Mar. 2017, pp. 408417.

[4] Z. Zhao,W. Chen, X.Wu, P. C. Chen, and J. Liu, ``LSTM network: A deep learning approach for short-term traffic forecast," IET Intell. Transp. Syst., vol. 11, no. 2, pp. 6875, Mar. 2017.

[5] P. Li, D. Wang, L. Wang, and H. Lu, ``Deep visual tracking: Review and experimental comparison," Pattern Recognit., vol. 76, pp. 323338, Apr. 2018.

[6] P. Wang and J. Di, ``Deep learning-based object classification through multimode fiber via a CNN-architecture SpeckleNet," Appl. Opt., vol. 57, no. 28, pp. 82588263, 2018.

[7] J. Zhao, Z. Zhang, W. Yu, and T.-K. Truong, ``A cascade coupled convolutional neural network guided visual attention method for ship detection from SAR images," IEEE Access, vol. 6, pp. 50693-50708, 2018.

[8] T. Pamula, ``Road traffic conditions classification based on multilevel filtering of image content using convolutional neural networks," IEEE Intell. Transp. Syst. Mag., vol. 10, no. 3, pp. 1121, Jun. 2018.

[9] M. Barth and K. Boriboonsomsin, ``Environmentally beneficial intelligent transportation systems," IFAC Proc. Volumes, vol. 42, no. 15, pp. 342345, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ``Imagenet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Pro-cess. Syst., 2012, pp. 10971105.

[11] D. D. Pukale, S.G. Bhirud, V.D. Katkar "Content-based Image Retrieval using Deep Convolution Neural Network"13 September 2018.

[12] Bongjin Oh, Junhyeok Lee "A case study on scene recognition using an ensemble convolution neural network" 26 March 2018.

[13] Ulagamuthalvi; J.B. Janet Felicita; D Abinaya "An Efficient Object Detection Model Using Convolution Neural Networks" 10 October 2019.

[14] Meghajit Mazumdar; V Sarasvathi; Akshay Kumar "Object recognition in videos by sequential frame extraction using convolutional neural networks and fully connected neural networks"21 June 2018.

[15] Ke Chen; Yanying Cheng; Hui Bai; Chunjie Mou; Yuchun Zhang " Research on Image Fire Detection Based on Support Vector Machine " 21 December 2017.

Appendix:

Main.py code

```
import tkinter as tk
from tkinter import Message ,Text
from PIL import Image, ImageTk
import pandas as pd

import tkinter.ttk as ttk
import tkinter.font as font
import tkinter.messagebox as tm
import matplotlib.pyplot as plt
import csv
import numpy as np
from PIL import Image, ImageTk
from tkinter import filedialog
import tkinter.messagebox as tm
import train as TR
import predict as PR
import video as vid
import cv2
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.preprocessing import image

bgcolor="#e6ffff"
bgcolor1="#00b3b3"
fgcolor="#006666"


def Home():
        global window
        def clear():
            print("Clear1")
            txt.delete(0, 'end')
            txt1.delete(0, 'end')
            txt2.delete(0, 'end')



        window = tk.Tk()
        window.title("Traffic Prediction")


        window.geometry('1280x720')
        window.configure(background=bgcolor)
        #window.attributes('-fullscreen', True)

        window.grid_rowconfigure(0, weight=1)
        window.grid_columnconfigure(0, weight=1)
```

```
messagel = tk.Label(window, text="Traffic Prediction from Video and Image" ,
                    bg=bgcolor  ,fg=fgcolor  ,width=50  ,height=3,font=('times', 30, 'italic bold underline'))
messagel.place(x=100, y=20)

lbl = tk.Label(window, text="Select Dataset Folder",width=20  ,height=2  ,fg=fgcolor  ,bg=bgcolor ,font=('times', 15, ' bold ') )
lbl.place(x=100, y=200)

txt = tk.Entry(window,width=20,bg="white" ,fg="black",font=('times', 15, ' bold '))
txt.place(x=400, y=215)

lbl1 = tk.Label(window, text="Select Image",width=20  ,height=2  ,fg=fgcolor  ,bg=bgcolor ,font=('times', 15, ' bold ') )
lbl1.place(x=100, y=300)

txt1 = tk.Entry(window,width=20,bg="white" ,fg="black",font=('times', 15, ' bold '))
txt1.place(x=400, y=315)

lbl2 = tk.Label(window, text="Select Video",width=20  ,height=2  ,fg=fgcolor  ,bg=bgcolor ,font=('times', 15, ' bold ') )
lbl2.place(x=100, y=400)

txt2 = tk.Entry(window,width=20,bg="white" ,fg="black",font=('times', 15, ' bold '))
txt2.place(x=400, y=415)

def browse():
        path=filedialog.askdirectory()
        print(path)
        txt.delete(0, 'end')
        txt.insert('end',path)
        if path !="":
                print(path)
        else:
                tm.showinfo("Input error", "Select Dataset Folder")

def browse1():
        path=filedialog.askopenfilename()
        print(path)
        txt1.delete(0, 'end')
        txt1.insert('end',path)
        if path !="":
                print(path)
        else:
                tm.showinfo("Input error", "Select Image")

def browse2():
        path=filedialog.askopenfilename()
        print(path)
        txt2.delete(0, 'end')
```

```
def trainprocess():
        sym=txt.get()
        if sym != "" :
                TR.process(sym,4)
                tm.showinfo("Input", "Training Successfully Finished")
        else:
                tm.showinfo("Input error", "Select Dataset")

def predictimgprocess():
        sym=txt1.get()
        if sym != "" :
                result=PR.process(sym)
                #tm.showinfo("Input", res)
                img = cv2.imread(sym)
                font = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(img, str(result) ,(19,51), font, 1,(0,0,255),5,cv2.LINE_AA)
                cv2.imwrite("output.jpg", img)
                plt.imshow(image.load_img("output.jpg"))
                plt.show()
                #plt.imshow(image.load_img(np.random.choice(image_files)))
                #plt.show()

        else:
                tm.showinfo("Input error", "Select image File")

def predictvideoprocess():
        sym=txt2.get()
        if sym != "" :
                vid.process(sym)
                #tm.showinfo("Input", res)
                cap = cv2.VideoCapture('video.avi')

                # Read until video is completed
                while(cap.isOpened()):
                  # Capture frame-by-frame
                  ret, frame = cap.read()
                  if ret == True:

                    # Display the resulting frame
                    cv2.imshow('Frame',frame)

                    # Press Q on keyboard to  exit
                    if cv2.waitKey(25) & 0xFF == ord('q'):
                      break

                  # Break the loop
```

```
                    if cv2.waitKey(25) & 0xFF == ord('q'):
                        break

                # Break the loop
                else:
                    break

            # When everything done, release the video capture object
            cap.release()

            # Closes all the frames
            cv2.destroyAllWindows()
        else:
            tm.showinfo("Input error", "Select Video")

    browse = tk.Button(window, text="Browse", command=browse  ,fg=fgcolor  ,bg=bgcolor1  ,width=10  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    browse.place(x=65, y=200)

    browse1 = tk.Button(window, text="Browse", command=browse1  ,fg=fgcolor  ,bg=bgcolor1  ,width=10  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    browse1.place(x=650, y=300)

    browse2 = tk.Button(window, text="Browse", command=browse2  ,fg=fgcolor  ,bg=bgcolor1  ,width=10  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    browse2.place(x=650, y=400)

    clearButton = tk.Button(window, text="Clear", command=clear  ,fg=fgcolor  ,bg=bgcolor1  ,width=20  ,height=2 ,activebackground = "Red" ,font=('times', 15, ' bold '))
    clearButton.place(x=650, y=200)

    trainbutton = tk.Button(window, text="Train", command=trainprocess  ,fg=fgcolor  ,bg=bgcolor1  ,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    trainbutton.place(x=270, y=600)

    prbutton = tk.Button(window, text="Image Predict", command=predictingprocess  ,fg=fgcolor  ,bg=bgcolor1  ,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold
    prbutton.place(x=540, y=600)

    vpbutton = tk.Button(window, text="Video Predict", command=predictvideoprocess  ,fg=fgcolor  ,bg=bgcolor1  ,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, ' bo
    vpbutton.place(x=890, y=600)

    ANNbutton = tk.Button(window, text="NEURAL NETWORKS", command=ANNprocess  ,fg=fgcolor  ,bg=bgcolor1 ,width=15  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    ANNbutton.place(x=420, y=600)


    quitWindow = tk.Button(window, text="Quit", command=window.destroy  ,fg=fgcolor  ,bg=bgcolor1  ,width=15  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
    quitWindow.place(x=1060, y=600)

    window.mainloop()
Home()
```

Convert.py

```
import cv2
import os
from os.path import isfile, join

def convert_frames_to_video(pathIn,pathOut,fps):
    frame_array = []
    files = [f for f in os.listdir(pathIn) if isfile(join(pathIn, f))]

    #for sorting the file names properly
    files.sort(key = lambda x: int(x[5:-4]))

    for i in range(len(files)):
        filename=pathIn + files[i]
        #reading each files
        img = cv2.imread(filename)
        height, width, layers = img.shape
        size = (width,height)
        print(filename)
        #inserting the frames into an image array
        frame_array.append(img)

    out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'DIVX'), fps, size)

    for i in range(len(frame_array)):
        # writing to a image array
        out.write(frame_array[i])
    out.release()


def process():
        pathIn= 'frame/'
        pathOut = 'video.avi'
        fps = 10.0
        convert_frames_to_video(pathIn, pathOut, fps)
```

Predict.py

```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.models import Sequential, load_model
from keras.preprocessing import image

import datetime
import time
import requests

img_width, img_height = 150, 150
model_path = './models/model.h5'
model_weights_path = './models/weights.h5'
model = load_model(model_path)
model.load_weights(model_weights_path)

def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  x = np.expand_dims(x, axis=0)
  array = model.predict(x)
  result = array[0]
  answer = np.argmax(result)
  res=""
  print(answer)
  if answer == 0:
    res="Accident"
    print("Label: Accident")
  elif answer == 1:
    res="Heavy Traffic"
    print("Labels: Heavy Traffic")
  elif answer == 2:
    res="Fire Accident"
    print("Label: Fire Accident")
  elif answer == 3:
    res="Low Traffic"
    print("Label: Low Traffic")

  return res

def process(path):
        result = predict(path)
        url = "https://www.fast2sms.com/dev/bulk"
        mobile="MOBILENUMBER"
        from_date = datetime.datetime.today()
        print(from_date)
        currentDate = time.strftime("%d:%m:%Y:%H:%M:%S")
```

```
def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  x = np.expand_dims(x, axis=0)
  array = model.predict(x)
  result = array[0]
  answer = np.argmax(result)
  res=""
  print(answer)
  if answer == 0:
    res="Accident"
    print("Label: Accident")
  elif answer == 1:
    res="Heavy Traffic"
    print("Labels: Heavy Traffic")
  elif answer == 2:
    res="Fire Accident"
    print("Label: Fire Accident")
  elif answer == 3:
    res="Low Traffic"
    print("Label: Low Traffic")

  return res

def process(path):
        result = predict(path)
        url = "https://www.fast2sms.com/dev/bulk"
        mobile="MOBILENUMBER"
        from_date = datetime.datetime.today()
        print(from_date)
        currentDate = time.strftime("%d:%m:%Y:%H:%M:%S")
        msg="Hi " + result + " is present " + currentDate
        payload = "sender_id=FSTSMS&message="+msg+"&language=english&route=p&numbers="+mobile
        headers = {'authorization':"Key",'Content-Type': "application/x-www-form-urlencoded",'Cache-Control': "no-cache",}
        response = requests.request("POST", url, data=payload, headers=headers)
        print(response.text)

        return result
```

Train.py

```
import sys
import os
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras import callbacks
import matplotlib.pyplot as plt


def process(path,classes_num):
        epochs = 20
        train_data_path = path+'/train'
        validation_data_path = path+'/test'
        img_width, img_height = 150, 150
        batch_size = 32
        samples_per_epoch = 1000
        validation_steps = 300
        nb_filters1 = 32
        nb_filters2 = 64
        conv1_size = 3
        conv2_size = 2
        pool_size = 2
        #classes_num = 4
        lr = 0.0004

        model = Sequential()
        model.add(Convolution2D(nb_filters1, conv1_size, conv1_size, border_mode
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

        model.add(Convolution2D(nb_filters2, conv2_size, conv2_size, border_mode
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(pool_size, pool_size), dim_ordering='t

        model.add(Flatten())
        model.add(Dense(256))
        model.add(Activation("relu"))
        model.add(Dropout(0.5))
```

```
        History=model.fit_generator(
            train_generator,
            samples_per_epoch=samples_per_epoch,
            epochs=epochs,
            validation_data=validation_generator,
            callbacks=cbks,
            validation_steps=validation_steps)

        target_dir = './models/'
        if not os.path.exists(target_dir):
          os.mkdir(target_dir)
        model.save('./models/model.h5')
        model.save_weights('./models/weights.h5')

        #Plot Loss and Accuracy

        plt.figure(figsize = (15,5))
        plt.subplot(1,2,1)
        plt.plot(History.history['accuracy'])
        plt.plot(History.history['val_accuracy'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.savefig('Accuracy.png')
        plt.legend(['train', 'test'], loc='upper left')

        plt.subplot(1,2,2)
        plt.plot(History.history['loss'])
        plt.plot(History.history['val_loss'])
        plt.title('model loss')
        plt.ylabel('loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper left')
        plt.savefig('Loss.png')
        plt.show()
        plt.show()

#process("D:\Traffic\data",4)
```

Vedio.py

```
import cv2
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.models import Sequential, load_model
from keras.preprocessing import image
import matplotlib.pyplot as plt
from PIL import ImageFont, ImageDraw, Image
import numpy as np
import os
from os.path import isfile, join
import convert as cvt


img_width, img_height = 150, 150
model_path = './models/model.h5'
model_weights_path = './models/weights.h5'
model = load_model(model_path)
model.load_weights(model_weights_path)

def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  x = np.expand_dims(x, axis=0)
  array = model.predict(x)
  result = array[0]
  answer = np.argmax(result)
  res=""
  print(answer)
  if answer == 0:
    res="Accident"
    print("Label: Accident")
  elif answer == 1:
    res="Heavy Traffic"
    print("Labels: Heavy Traffic")
  elif answer == 2:
    res="Fire Accident"
    print("Label: Fire Accident")
  elif answer == 3:
    res="Low Traffic"
    print("Label: Low Traffic")
```

```
    res="Heavy Traffic"
    print("Labels: Heavy Traffic")
  elif answer == 2:
    res="Fire Accident"
    print("Label: Fire Accident")
  elif answer == 3:
    res="Low Traffic"
    print("Label: Low Traffic")

  return res

# Function to extract frames
def FrameCapture(path):

        # Path to video file
        vidObj = cv2.VideoCapture(path)

        # Used as counter variable
        count = 0

        # checks whether frames were extracted
        success = 1
        while success:
                # vidObj object calls read
                # function extract frames
                success, image = vidObj.read()
                # Saves the frames with frame-count
                cv2.imwrite("frame/frame%d.jpg" % count, image)
                result = predict("frame/frame"+str(count)+".jpg")
                image = cv2.imread("frame/frame"+str(count)+".jpg")
                font = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(image, str(result) ,(219,151), font, 5,(0,0,255),5,c
                cv2.imwrite("frame/frame"+str(count)+".jpg", image)
                print(count,result)
                count += 1
                if count==100:
                        break


def process(path):
        # Calling the function
        FrameCapture(path)
        cvt.process()
```