

COMBINED OBJECT DETECTORS METHOD FOR PEDESTRIAN DETECTION

A Project Report submitted in partial fulfillment of the requirements for the award

of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Chilakamarri Sai Srinivas (221710302017)

Sathunuri Dinesh (221710302059)

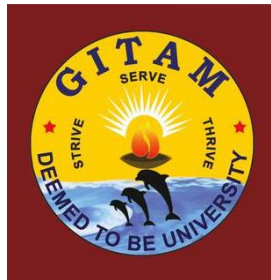
Saimpu Sai Mohit (221710302056)

Potlapally Shantan (221710302049)

Under the esteemed guidance of

Dr.G.Yugandhar

(Assistant Professor)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

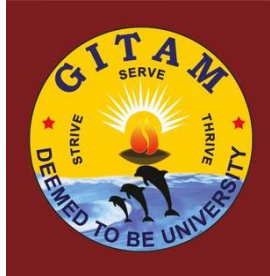
(Deemed to be University)

HYDERABAD

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM

(Deemed to be University)



DECLARATION

I/We, hereby declare that the project report entitled “**Combined Object Detectors Method for Pedestrian Detection**” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s)

221710302017

221710302059

221710302056

221710302049

Name

Chilakamarri Sai Srinivas

Sathunuri Dinesh

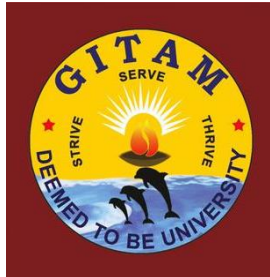
Saimpu Sai Mohit

Potlapally Shantan

Signature(s)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM

(Deemed to be University)



CERTIFICATE

This is to certify that the project report entitled “**Combined Object Detectors Method for Pedestrian Detection**” is a bonafide record of work carried out by **Chilakamarri Sai Srinivas(221710302017)**, **Sathunuri Dinesh(221710302059)**, **Saimpu Sai Mohit(221710302056)**, **Potlapally Shantan (221710302049)** submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide

Dr.G.Yugandhar
(Assistant Professor)

Head of the Department

Dr.S.Phani Kumar
(Professor)

ACKNOWLEDGEMENT

Our Major Project would not have been successful without the help of several people. we would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honourable Pro-Vice Chancellor, Prof. N. Siva Prasad for providing necessary infrastructure and resources for the accomplishment of our seminar.

We are highly indebted to Prof. N. Seetharamaiah, Principal, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved Prof. S. Phani Kumar, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this seminar and encouragement in completion of this seminar.

We hereby wish to express our deep sense of gratitude to Dr. S Aparna, Assistant Professor, Department of Computer Science and Engineering, School of Technology and to Mrs. G. Sri Sowmya Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the summer Internship.

We are also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our seminar work.

Sincerely,

Chilakamarri Sai Srinivas(221710302017)

Sathunuri Dinesh(221710302059)

Saimpu Sai Mohit(221710302056)

Potlapally Shantan (221710302049)

TABLE OF CONTENTS

1	ABSTRACT	1
2	INTRODUCTION	2
	2.1 PROBLEM DESCRIPTION	5
	2.2 MOTIVATION FOR THE PROJECT	5
	2.3 OBJECTIVE	5
	2.4 APPLICATIONS	5
	2.5 LIMITATIONS	5
3	LITERATURE SURVEY	7
	3.1 INTRODUCTION	7
	3.2 RECENT WORK/RESEARCH	7
4	PROBLEM IDENTIFICATION AND OBJECTIVES	9
	4.1 PROBLEM DEFINITION	9
	4.2 REQUIREMENT ANALYSIS	9
	4.3 METHODS TO IMPLEMENT THE PROJECT	10
	4.4 FINANCIAL FEASIBILITY	10
	4.5 OPERATIONAL FEASIBILITY	10
	4.6 RESOURCE FEASIBILITY	10
	4.7 TIME FEASIBILITY	11
	4.8 BOUNDARIES	11
5	SYSTEM METHODOLOGY	12
	5.1 DATASET	13
	5.2 MODELS USED	14
	5.2.1 HOG+SVM	14
	5.2.2 CNN	14
	5.3 ACCURACY METRICS	16
6	OVERVIEW OF TECHNOLOGIES	18
7	IMPLEMENTATION	21
	7.1 CODING	21
	7.2 TESTING	24
8	RESULTS AND DISCUSSION	29
9	CONCLUSION	32
10	REFERENCES	33

LIST OF FIGURES

Fig 2.1	Classification of Machine Learning	4
Fig 5.1	Project workflow	13
Fig 5.2	HOG+SVM Architecture	14
Fig 5.3	CNN Architecture	14
Fig 5.4	Combined model flowchart	16
Fig 7.1	Output-1.1	25
Fig 7.2	Output-1.2	26
Fig 7.3	Output-1.3	26
Fig 7.4	Output-2.1	27
Fig 7.5	Output-2.2	28
Fig 7.6	Output-2.3	28
Fig 8.1	Architecture	29
Fig 8.2	Result-1	30
Fig 8.3	Result-2	30
Fig 8.4	Result-3	31

ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine
CNN	Convolutional Neural Networks

1. ABSTRACT

The detection of pedestrians from the environment as using it is very important for working of automatic driving. There have been researches on this for detecting pedestrians, vehicles, traffic signs, etc. The main objective of the pedestrian detection system is to deliver the system with accurate information in real-time about the surrounding environment so that it can reduce the number of traffic accidents that can occur and save lives.

In this project, the models used for pedestrian detection are HOG+SVM which is a classical machine learning model and CNN, a deep learning model. These two models are combined and used for pedestrian detection to improve the accuracy of detecting pedestrian and keep the miss rate as low as possible. The dataset used in this project is INRIA Person Dataset which contains static pedestrian images in various angles of standing positions and also has positive and negative samples.

2. INTRODUCTION

Improvement in technology has resulted in revolution of our world and has eased our lives over the years and keeps improving every day. Creation of unique tools and resources, and having helpful information. In addition, modern technology given us different useful devices such as smartphones, smartwatches and other comfort gadgets.

As the time passes, the improvement in technology is happening exponentially increasing the speed and functionalities of computers. With all of these new inventions our lives are easier, faster, better, and more fun. These days many tasks performed by humans are being carried out by machines and robots. As a result, artificial intelligence and machine learning, deep learning has become buzzwords recently in the technology sector.

Many companies and organizations are trying to create excellent applications based on Artificial intelligence and Machine learning, Deep learning—organizations, namely Google, Microsoft, and other IT companies working on these projects.

We also have a fair number of resources to work on these AI and ML projects, and there is a considerable number of libraries developed by many organizations. For example, python is the most used language these days, and its usage has been growing for the last five years. Python programming language is known for its simplicity, and the number of libraries it offers to the user is simply unique.

Some of the libraries are named below:

- **TensorFlow library:** This library is helpful to work with Ai and ML models. It offers a bunch of tools and utilities. Google develops it.
- **Pytorch library:** This library is also the most popular one and offers many methods to work on AI and ML projects. Facebook's AI Research Lab develops it

Artificial Intelligence: Artificial Intelligence is the recreation of human insights in machines and they are modified to think like people and also imitate their activities. Moreover, this term can be imposed on any machine that can show the characteristics related to a human mind, such as learning, rationalising and problem-solving.

Machine Learning: Machine Learning (ML) is the ponder of computer calculations that improve automatically through encounter. It is a subset of counterfeit intelligence. Machine learning

calculations construct a demonstration based on test information, known as "training data," to create expectations or choices without being unequivocally programmed to do such operations. Machine learning calculations are used in a wide assortment of applications, such as mail sifting and computer vision, where it is troublesome or unfeasible to develop conventional calculations to perform the required tasks. A subset of machine learning is closely associated with computational insights, focusing on making forecasts utilizing computers. The think about numerical optimization conveys strategies, theory, and application spaces to the field of machine learning.

Deep Learning: Deep Learning is a family of machine learning strategies based on artificial neural systems with representation learning. Learning can be directed, semi-supervised or unsupervised. Deep-learning models such as neural systems, conviction systems, redundant neural systems, and convolutional neural systems have been connected to areas counting computer vision, machine vision, machine interpretation, bioinformatics, medicate plan, where they have delivered comes about comparable to and in a few cases outperforming human master execution.

There are particular learning algorithms that we can implement based on the problem we are dealing with, so let us see those learning algorithms.

Supervised Learning: Supervised Learning, also known as Administered Learning, it is the most famous worldview of AI. It is the least demanding to the most straightforward to carry out. It is the same as showing a kid with the utilization of glimmer cards. The information is given as models with names, and it can take it for a learning calculation. These models mark combines individually, and permit the calculation to anticipate the name for model, and giving it input and if it has anticipated the correct answer. However on the long run, the calculation will figure out how to find the specific idea of the relationship among model and their names. When wholly prepared, the regulated learning calculation will want to notice another, at no other time seen the model and anticipate a decent mark for it.

Supervised learning is regularly portrayed as an assignment situated along these lines. It is profoundly centered around a solitary undertaking, taking care of an ever-increasing number of guides to the calculation until it can precisely perform on that task. It is the learning type that we will most likely experience, as displayed in many primary applications, such as Ad Popularity and Spam Classification.

Unsupervised Learning: Unsupervised Learning Solo Learning is especially something contrary to Directed Learning or Supervised Learning. It includes no marks. All things being equal, our

calculation would be taken care of a ton of information and given the devices to comprehend the properties of the information. From that point, it can figure out how to gather, bunch, and put together the information in order with the result that a human (or other clever calculation) can come in and figure out the recently coordinated information.

For example, let's imagine a scenario where we had an large data set of each exploration paper ever distributed. We had a solo learning calculation that realized how to bunch these in such a manner, so we were consistently mindful of the movement inside a specific examination area. Start to begin an examination project by guiding the work into the network that the calculation can see. Review the work and take notes, then calculate and make ideas about related work that may wish to refer to, and work that may assist us in using that area of examination forward. With such a device, our profitability can be incredibly helped. Since unaided learning depends on the information and its properties, we can say that unaided learning is information-driven. The results from an unaided learning task are constrained by the information and how it is organized. We might see solo learning crop up a few regions are Recommender frameworks, purchasing propensities, and gathering client log.

Reinforcement Learning: Reinforcement Learning is the preparation of AI models to let them make a grouping of decision. The model understands how to achieve an objective in a dubious, complex climate. In support of learning, computerized reasoning appearances a game-like circumstance. The system experiments to find an answer. The fake insight gets either results or punishments for the activities it can perform. Its objective is to expand the all-out remuneration. The user sets the strategy, that is, the game's guidelines then he doesn't give the model clues or ideas for tackling the game. It is up to the model to sort out some way to play out the errand.

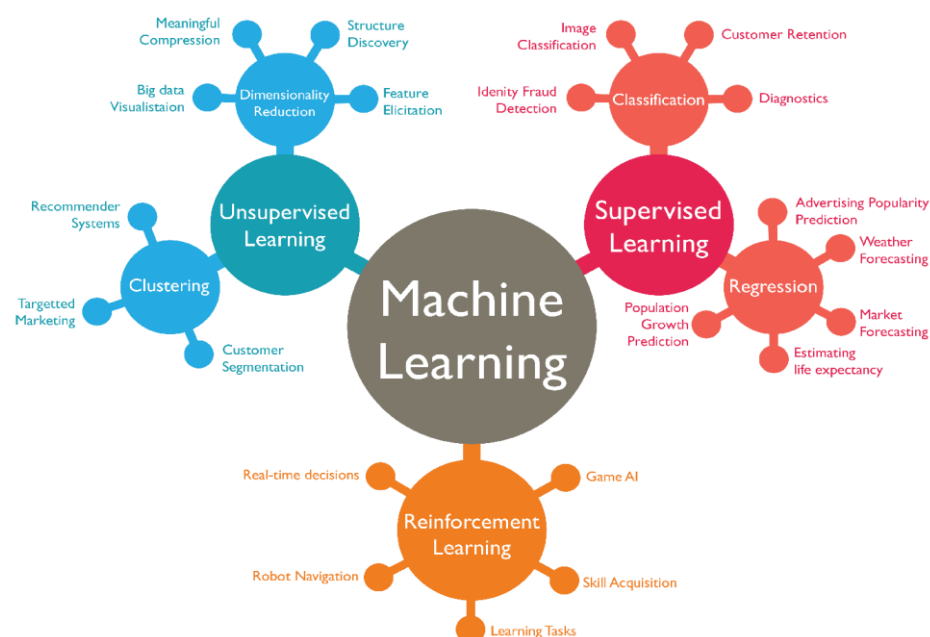


Fig 2.1 Classification of Machine Learning

2.1 PROBLEM DESCRIPTION

There have been significant researches on the working of automated driving systems and the systems should be developed in such a way that it can process accurate information of its surrounding using the input from environment and making decisions to reduce traffic accidents and save lives.

2.2 MOTIVATION FOR THE PROJECT

The detection of pedestrians from the environment as using it is very important for working of automatic driving. There have been researches on this for detecting pedestrians, vehicles, traffic signs, etc. The main objective of the pedestrian detection system is to deliver the system with accurate information in real-time about the surrounding environment so that it can reduce the number of traffic accidents that can occur and save lives.

Although there are many Deep Learning and classical machine learning models out there used for pedestrian detection, we will be trying to achieve good accuracy by developing a simple prediction model that will be trained on images and then used to make predictions.

2.3 OBJECTIVE

The main objective is to develop a model with HOG+SVM and CNN to improve the detection accuracy and identify the pedestrians in the image.

2.4 APPLICATIONS

Surveillance applications, traffic safety, blurring pedestrians for privacy issues, and human-robot interaction. In the last decade, pedestrian detection received much attention as a research topic, which resulted in a broad number of available techniques.

2.5 LIMITATIONS

- **Image/sensor noise:** Low-priced cameras will interpolate the pixels of raw sensors so that it can be used to produce accurate colors. Sometimes this can cause sensor noise that will prove difficult for detection purposes.

- **Blurring:** In some environments can cause blurred image, mainly if the user uses a common smartphone that do not have good form of stabilization.

- **Lighting conditions:** The brightness of the image can be a great factor in improving the detection process. If the image is dark and the model cannot predict the required output due to low clarity image.

- **Resolution:** Higher resolutions mean that there more pixels per inch resulting in more pixel information and creating a high-quality, crisp image. With more information on the image with good resolution the prediction will be more accurate.

3. LITERATURE SURVEY

3.1 INTRODUCTION

The importance of literature survey is that it is the most basic advance in the product improvement measure. Prior to development of the device, it is important to pitch in the economy, time factor, and support strength. When those things are fulfilled, the accompanying advances choose which working framework and language can build up the application. When the software engineers begin assembling the system, the developers need a ton of outside help. This outside help can be from senior developers, books, papers or sites. Prior to continuing with the undertaking, the above focuses for building up the proposed framework is to be followed.

3.2 RECENT WORK/RESEARCH

The detection of humans has been an important research task and challenging work as their appearance can change and they can be seen in different poses. To identify the features of humans in an image and identify them in varying background and illuminations we need a robust method. The locally normalized Histogram of Oriented Gradient (HOG) descriptors can provide excellent performance. These proposed descriptors are edge orientation of histograms, and different descriptors such as SIFT descriptors and also shape contexts, but these are calculated on a dense grid of uniformly spaced cell. The gradients for the image are taken along the horizontal and vertical axes and magnitude and directions are found then the bins are used to store the information of magnitude and direction of cells. Trained SVM is given these descriptors to identify the pedestrians in image.

For an automated driving system recognition of traffic signals is an important task and there are other factors in real world scenarios such as motion blur, illuminations, different viewpoints. There have been few models developed previously to understand the traffic signs. The recognition of these signs of different colors has proven difficult for the models. The deep convolutional neural networks can be trained on large datasets and used for the extracting features from images and can be used for different applications.

The automatic driving cars should be able to identify the environment on road and this can be called as Traffic scene perception. This can be divided into three types which are tracking motion of objects, detecting objects and recognising objects. The main ability of TSP is to identify the objects effectively. So using a simple learning detector can help in extracting the required features from the image train the model and use it for identification of the required classes. The main advantage of using a learning based detector is it is much faster, efficient.

You look only once(YOLO) has also been a famous model that can be used for the purpose of object detection. The deep learning based object detections work better due to good accuracy values and they can help in generating proposal networks that help in classifying the object easily rather than traditional object detectors. The object detection algorithms such as YOLO and SSD can generate region proposal networks to identify the object and classify them. R-CNN, Fast R-CNN, and Faster R-CNN are also few CNN models which have been used for detection of object on a large scale.

The automated driving cars should be able to distinguish between the traffic sign to be able to operate efficiently for this purpose there have been proposals such as use of a hybrid detector of HOG descriptor with SVM combined with simple convolutional neural network to improve the color detection and extract the candidate regions to identify the require shape and color of the traffic light. Similar method has been used in this project.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

Problem analysis is the essential step while working on projects like this. It gives us a complete idea of the problem and the requirements for proceeding with the project. Problem analysis deals with the problem definition, hardware requirements, and software requirements, existing methods, and what methods can be implemented to achieve the final result, scope, and limitations. This detailed analysis provides us with valuable insights, which helps us understand better. In this chapter, we deal with the following:

- Problem definition
- Requirement analysis
- Methods to implement the project
- Feasibility study such as Financial, Operational, Resource, and Time feasibility.

4.1 PROBLEM DEFINITION

The detection of pedestrians from the environment as using it is very important for working of automatic driving. There have been researches on this for detecting pedestrians, vehicles, traffic signs, etc. The main objective of the pedestrian detection system is to deliver the system with accurate information in real-time about the surrounding environment so that it can reduce the number of traffic accidents that can occur and save lives.

4.2 REQUIREMENTS ANALYSIS

Now coming on to the requirements for this project, it depends on which method you will choose while implementation as there are many ways in which this project can be implemented. But the generalized requirements of both hardware and software are mentioned below:

Hardware requirements:

- Processor: Intel-i5/Intel-i7 or any AMD equivalent
- RAM: 8GB minimum or higher
- GPU: A decent graphic card with CUDA rating 7.5 - 4GB memory (This is only needed if we choose to develop a model from scratch and train it on large datasets)

Software requirements:

- Operating system: Windows 10/Linux
- Python installation: Anaconda
- IDE: Visual studio code or PyCharm

4.3 METHODS TO IMPLEMENT THE PROJECT

- i. **Using the pre-trained model:** Using the pre-trained model gives us a head-start as we do not need to develop the model from scratch. We can load the model and use it in their programs we like. This method saves time and reduces the burden on us. But in this method, we might not be able to change the structure of the model.
- ii. **Developing the model from scratch:** Developing the model from scratch gives us access to change the layers of the Deep Learning model according to their use. We can also make any changes to the model in between. Now we can pick a data set and train the model on it until good accuracy is achieved. Now, this model can be saved and used whenever needed.

4.4 FINANCIAL FEASIBILITY

The cost indulged in this project is zero as this is done for academic purposes. However, if this project were to be released into the real-world market, it would not incur any costs. Therefore, we can call it a low-budget project. The cost incurred on the first two methods mentioned above is almost negligible.

4.5 OPERATIONAL FEASIBILITY

The operational feasibility deals with how well the proposed system solves the problem. Any one of the above methods satisfies the problem partially but needs to be investigated further as the machine learning model's accuracy may improve over time or maybe in the future due to technological advancements. Method one and method two mentioned above in technical feasibility use the systems resources mentioned in the requirements section.

This project also fits into real-world business as machine learning is one of the recent advancements in the technology sector. Various organizations are working on this type of project and have demand for this type of project and could be an excellent real-world application if it could be enhanced more. It is also affordable in terms of expenditure, and it should be continuously refined wherever there is scope for improvement if deployed as a real-world application.

4.5 RESOURCE FEASIBILITY

The number of resources used by the program depends on what method is being to train and develop the machine learning model. When we develop the model from scratch, then most of the system resources are consumed while training. When the model trains on such massive datasets, it will require lots of memory to process that image data and store the results. The CPU and GPU are

also continuously working to provide speedy access to the data for the machine learning model to train.

4.6 TIME FEASIBILITY

This is used to understand the time period which the project will take to complete. The project is said to be considered a failure if it takes a lot of time to complete before it is functional. Therefore, it means estimating how much time the system will take to develop. The time taken for the project completion will be on the technical expertise: the programmer's knowledge about the project and his or her capacity to do programs.

However, one specific thing is that it will take much time to train the model on image samples because sometimes the datasets may be massive.

4.7 BOUNDARIES

This particular project is only specific to the detection of pedestrians in images; other objects and features of the image will not be detected here. The detected pedestrian is shown in the bounding box on the image on the output showing the detected image.

Even though it is only limited to pedestrian detection, this project acts as a base for identifying the people from different angles, which is the next level of this project.

The combined model of hog feature with SVM and simple convolutional neural network can also be used for the detection of traffic lights. The hybrid model can be used to identify the color information and extract the required candidate region and identify the shape and color of the traffic light to better the working of automatic driving cars. This method has also been used in this project for the detection of pedestrians.

5. SYSTEM METHODOLOGY

Here In this chapter, we deal with the workflow in which this project was implemented.

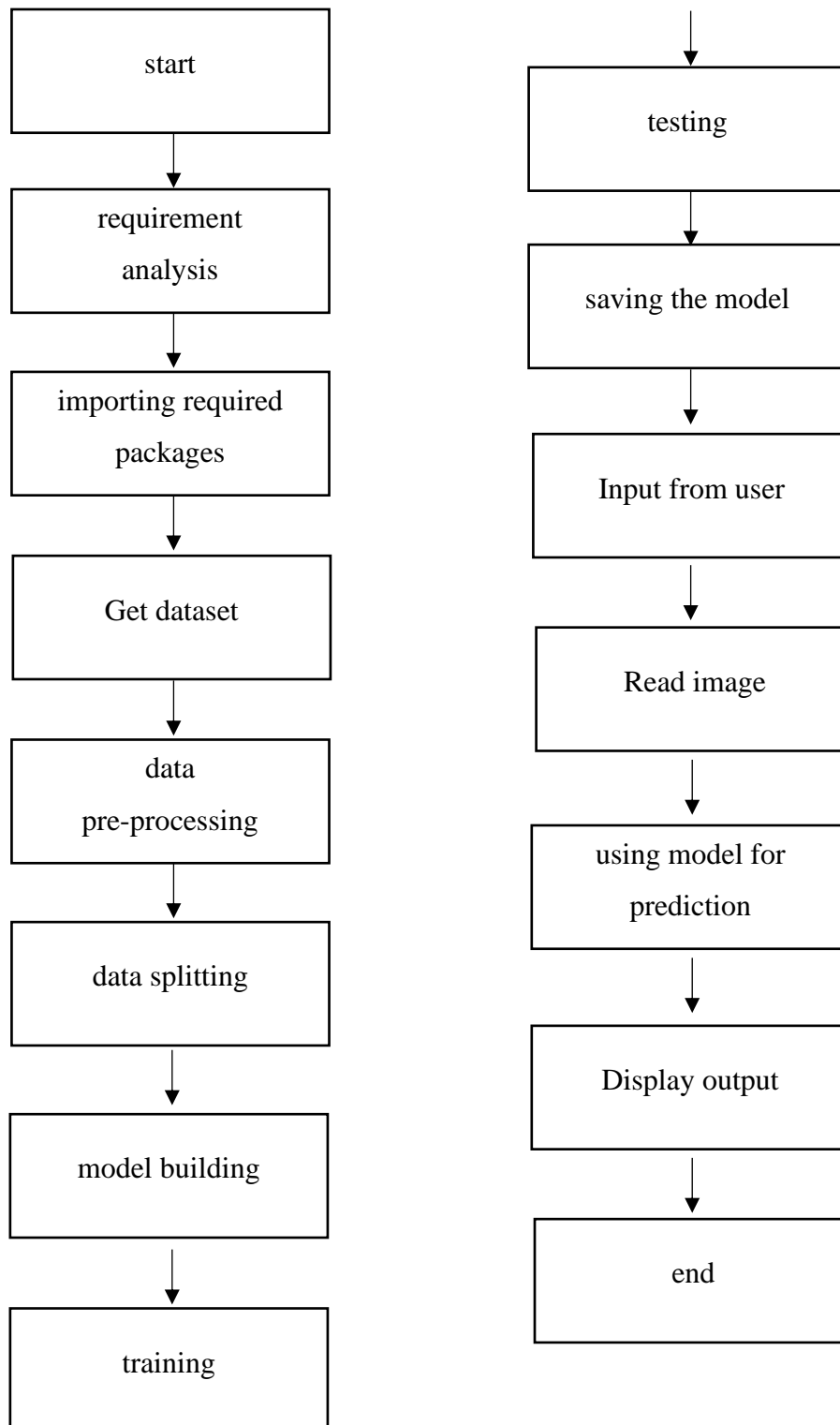


Fig 5.1 Project workflow

REQUIREMENTS ANALYSIS

Now coming on to the requirements for this project, it depends on which method you will choose while implementation as there are many ways in which this project can be implemented. But the generalized requirements of both hardware and software are mentioned below:

Hardware requirements:

- Processor: Intel-i5/Intel-i7 or any AMD equivalent
- RAM: 8GB minimum or higher
- GPU: A decent graphic card with CUDA rating 7.5 - 4GB memory (This is only needed if we choose to develop a model from scratch and train it on large datasets)

Software requirements:

- Operating system: Windows 10/Linux
- Python installation: Anaconda
- IDE: Visual studio code or PyCharm

IMPORTING REQUIRED PACKAGES:

For every project there are certain libraries or modules which provide certain functionality for the program. These are very essential for the program in order to run the program properly. So, at the start of the program required packages are imported. Some of the library/module names are mentioned below.

- TensorFlow
- NumPy
- Keras
- PIL
- OS
- Pickle
- Tkinter

5.1 DATASET

The dataset that is used in this project is INRIA Dataset. It has pictures gathered from different sources such as personal photos, Google, etc., and the pictures are of high definition. The dataset contains positive and negative samples of images for testing and training. The images in the dataset are static images containing pedestrians in different standing postures.

5.2 MODELS USED

5.2.1 HOG+SVM

Histogram of oriented gradients (HOG) is used for feature extraction from the image in the human detection process and the linear support vector machines (SVM) are used for human classification based on the feature extraction data from HOG.

The HOG features are usually used for object detection and HOG decomposes an image into small squared cells, then computes the histogram of oriented gradients in each cell, then normalizes the result using a block-wise pattern, and return a descriptor for each cell.

Stacking the cells into a squared image region can be used as an image window descriptor for object detection, for example by means of an SVM.

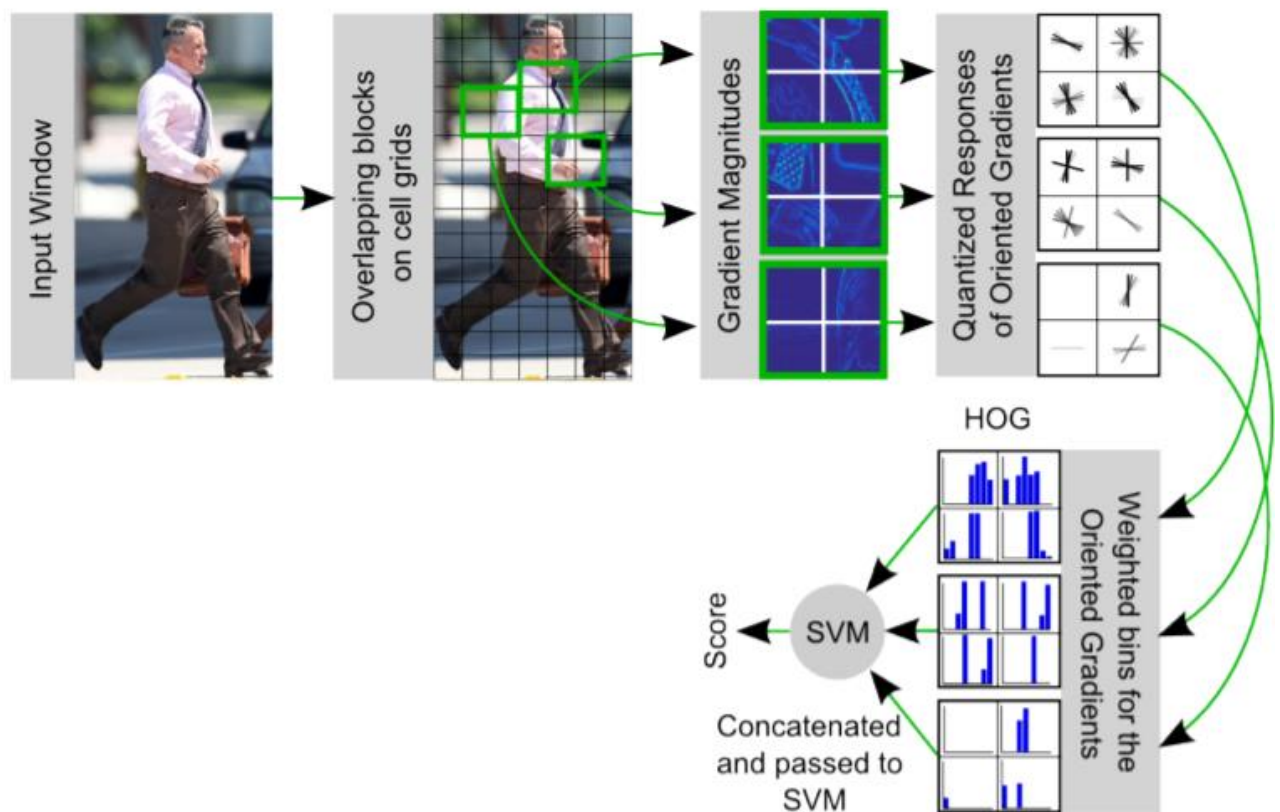


Fig 5.2 HOG+SVM Architecture

5.2.2 CNN

A Convolutional Neural Network (CNN) is a Deep Learning algorithm. It can take an image as an input and extract the image features using the method of convolutions, then pass it to the neural network with random weights to train and fix the right weights and produce required detection results when used.

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

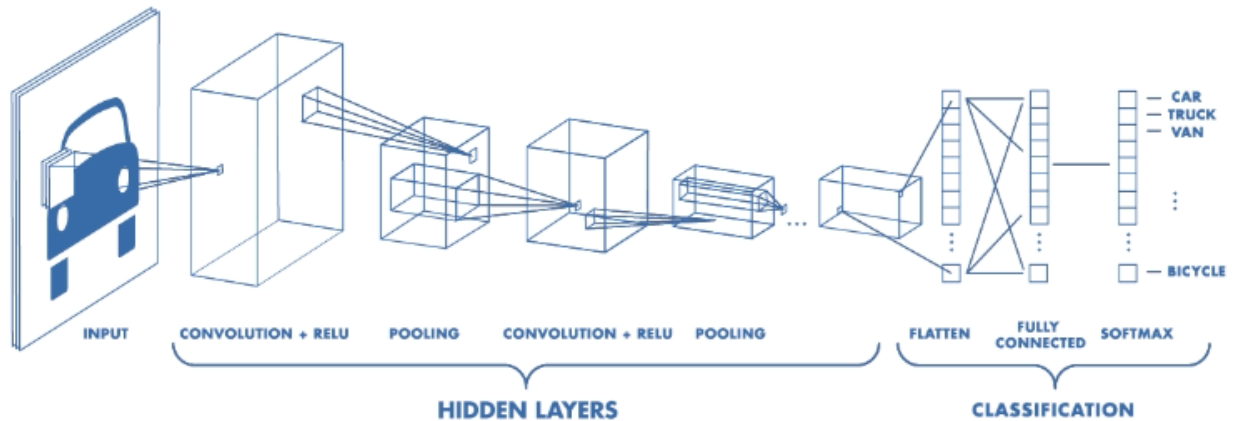


Fig 5.3 CNN Architecture

CONVOLUTION LAYER:

In this layer a dot product between matrices is performed and a matrix with learnable parameters is kernel and other matrix is restricted part of receptive field. The kernel is smaller spatially compared to the image but is more in depth. The image is composed of three channels which are RGB.

POOLING LAYER:

The main purpose of pooling layer is to reduce the spatial size of the image decreasing the amount of computation and weights required for the representation. The pooling layer replaces the output at certain locations by deriving the statistics of nearby outputs.

FULLY CONNECTED LAYER:

This layer is containing neurons with full connectivity with preceding and succeeding layers. This layer helps in identifying the features of the image resulting in the right prediction of output and during the training process the weights between the neurons is calculated.

NON-LINEARITY LAYERS:

Convolutions are linear operations whereas images are not linear which is the reason we use non-linearity layers to induce non-linearity in the image having the activation maps.

The two models are combined and then used for training of the model. For training of the model 70% of the dataset is taken as the training set while rest is used for testing. The batch size or number of images used for training is 128 per epoch.

Initially the uploaded image is resized to require input size then it is passed to the HOG+SVM model and confirm if there are any pedestrians in the image. If they are present then the image is passed to CNN model for improving the accuracy of the detection. The dataset containing the images are used to trained this model and save the weights for future use.

The model's robust structure, all because of the two stage model gave it a nice accuracy of 89%. This is more than sufficient to be used in real-time scenarios. It should be noted that the developed project did use trimmed videos instead of untrimmed ones like usually seen in real-time.

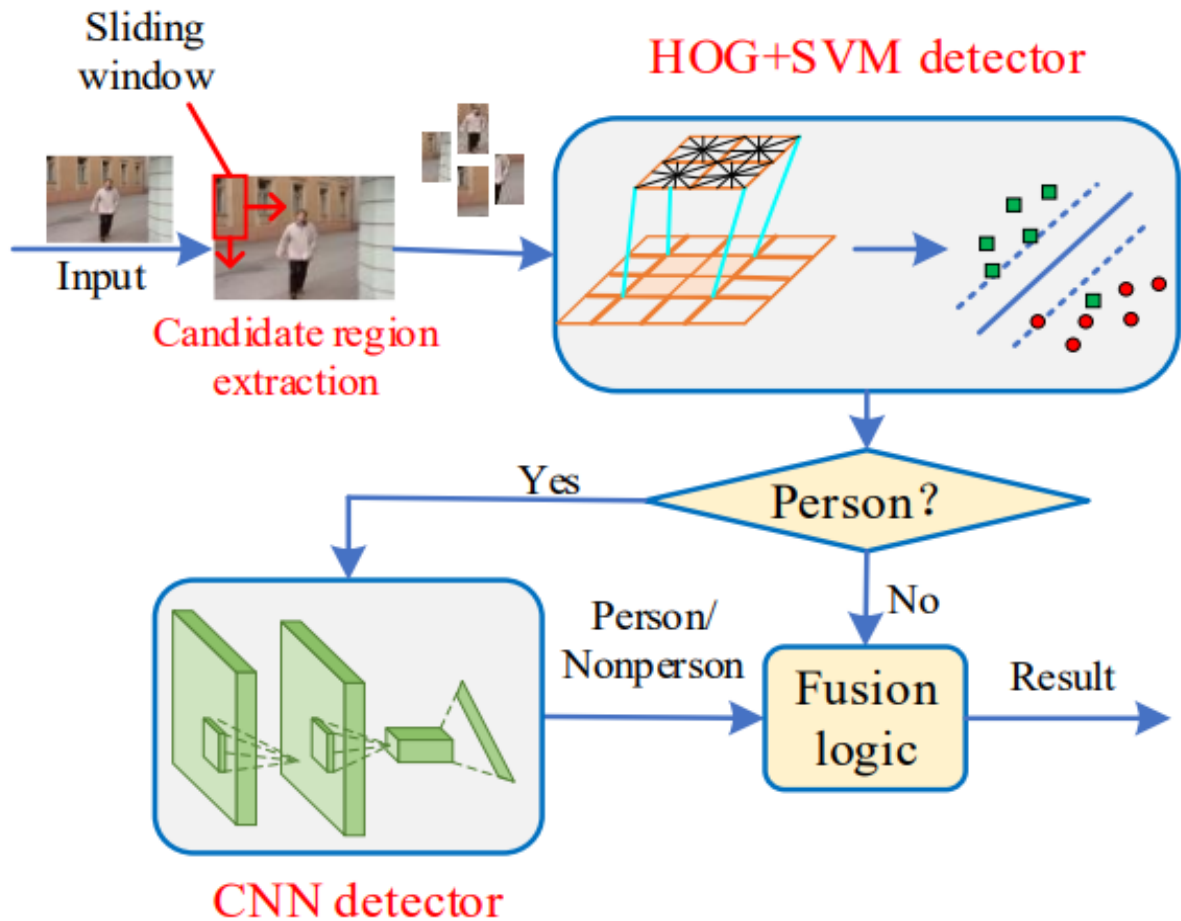


Fig 5.4 Combined model flowchart

5.3 ACCURACY METRICS:

Accuracy is one of the key challenges to be met when developing an effective model. For checking of the accuracy, the following techniques are used.

a. **Accuracy** : Accuracy is the usually considered as the standard evaluation metric like the name suggests, which can be defined as

$$Accuracy = \frac{\text{True Positives} + \text{True Negative}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negative}}$$

b. **Precision** : In precision the goal is to find out what fraction of all positive predictions, are true positives. This is another popular way to evaluate a model's accuracy.

$$Precision(P) = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

c. **Recall** : In recall, the goal is to calculate of all actual positives, what fraction is actually predicted as true positives. This is

$$Recall (R) = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

d. **F1-Score**: It is also known F score which is a weighted harmonic mean of precision and recall. When the formulae of precision and recall are observed, it can be understood as the F1- score is 1.0 is best and F1-score is 0.0 is the worst.

6. OVERVIEW OF TECHNOLOGIES

A deep learning project uses many libraries to complete its execution. Since, its realistically very difficult to develop a deep learning model from scratch, most developers often use pre-defined libraries which are present in various libraries such as TensorFlow which contains many deep learning models and neural networks, scikit learn to split the data into testing and training sets, NumPy to get the vector values and so on. These libraries contain methods and models which have

been developed after several years of research and testing which makes the users' life way easier than developing their own models. It also reduces the resource requirements as developing models from scratch takes way too much time and is not a realistic target especially for beginners.

The various modules used in the project include:

TensorFlow: TensorFlow is a well-known deep learning library developed by Google for their internal purposes primarily. TensorFlow's primary goal is to work on deep learning models and neural networks and their trainings. TensorFlow was created by a team in Google called as Google Brain. TensorFlow is a special library which can make use of multiple CPUs and GPUs to accelerate the training and prediction. For these we need special libraries such as CUDA libraries which is used in the utils folder of the project. TensorFlow is available and usually used in two languages, namely, Python and JavaScript. This is a premier software library used for numerical data computation using data flow graphs. TensorFlow has special nodes which represents the complex or simple mathematical operations and it has edges which represent the multidimensional data arrays called tensors communicated between them.

TensorFlow API's (Application Program Interfaces) are of two types. Those two types are namely Low-Level API and High-Level API.

Low Level API: Complete programming control. TensorFlow core is the low-level API of TensorFlow

High Level API: Built on top of TensorFlow core. This makes repeated tasks easy and more consistent between different users.

NUMPY:

NumPy is a mathematical library for python. It fixes the issues such as python being slow in many numerical and mathematical calculations by making the execution speeds to rival that of traditional and more complex languages such as C and Fortran. Like TensorFlow and Keras, NumPy also allows the user to have an option of running on multiple GPUs at the same time. NumPy gives access to the user to work using powerful mathematical tools traditional python lacks like mathematical functions, algebraic equations, random number generation tools and series transformations like Fourier transforms. It also gives python access to arrays. Python in default only supports lists but NumPy gives it arrays of various dimensions and even vectors. This is due to the fact that NumPy is in fact an extremely well optimized library written in C. Despite being in C, it is made to be a very high-level library which makes it extremely simple to use. Like most other python libraries, it is also open source and simply can extended as Numerical Python. It even has

many functions for trigonometry and statistics. In the project NumPy has been used to get the value of the images as images are three-dimensional matrix which has an RGB value. The extracted features are also matrix and vector values which are converted and stored using NumPy methods.

KERAS:

Keras is an extremely high-level TensorFlow API which is especially created to make our working with TensorFlow and deep learning very simple. Keras offers many features which makes human involvement in the project development quite low. This minimization of human intervention is quite often seen in usual use cases that are encountered while developing the project like layer creation. Keras was built on the widely popular TensorFlow 2.0. Newer versions of Keras are only available for TensorFlow whereas older ones were supported for other Machine Learning tools too like Theano. In other words, Keras can be simply considered as an intuitive, effective and easy to understand interface for the wide and diverse TensorFlow. Most of the models and extraction of features are all handled by keras in this project making it a key development concept in almost every deep learning project. Like TensorFlow Keras also allows the user to have an option of running on multiple GPUs at the same time.

OS: OS is a library that is in-built in the default python. This is primarily used to access the various operations that the operating system provides like file access. This module allows python to access various files in the system. Path functions allows the user to specify a relative path from where a file can be accessed. This module is especially useful since data science projects need continuous access to the various files which hold the data on which the model needs to be trained upon and tested on. Most of the OS module commands match with their UNIX counterparts as these commands are built similarly and even work quite similarly. OS module even has methods to move from the current directory or even remove a directory entirely. There are methods to change the path, create new files, read and write files etc. As python is portable, this module also provides a portable way to use the dependant functionality of the operating system with a few commands. In the project OS module has been the go-to way to access the extremely large datasets. It has also been the module which has been used to save the trained models or the graphs or the architecture and weights files.

CV2: CV2 is a large open-source library for computer vision, machine learning, and image processing, and it now plays a critical role in real-time operations, which are critical in today's systems. It can be used to recognise objects, faces, and even human handwriting in photographs and videos. Python will process the CV2 array structure for analysis when it is combined with various

libraries such as Numpy. We use vector space and perform mathematical operations on these features to identify image patterns and their different features.

The first edition of CV2 was 1.0. It supports Windows, Linux, Mac OS, iOS, and Android and has C++, C, Python, and Java interfaces. The main goal of CV2 when it was created was to make real-time applications as efficient as possible. To take advantage of multi-core computing, everything is written in optimised C/C++

Tkinter: The tkinter package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.) Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Pickle: Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.)

7. IMPLEMENTATION

In this chapter we are going to look at implementation part in this project. In this we have used Visual Studio Code as our IDE. The first thing is to load all modules required for the image handling and some other libraries for other utilities. When the user runs the program and gives any image the dimensions of the image should be captured and resized according to the model input dimensions. Then after we load our model into the program by using function in open cv. Then we take the images given and make a forward pass-through layers of model. After doing this we get the

output as prediction values, so now we decode these predictions and then display the image with a bounding box around the trees.

Now below is a detailed code and how we implemented this project. So first let's start with what are all the modules required to run this program. These are call the pre-requisite modules which are required for running the program as expected.

7.1 SOURCE CODE

```
1  import numpy as np
2  import keras
3  from keras.models import Sequential
4  from keras.layers import Conv2D
5  from keras.layers import MaxPooling2D
6  from keras.layers import Flatten
7  from keras.layers import Dense
8  from keras import optimizers
9  from keras.preprocessing.image import ImageDataGenerator
10 from keras.models import model_from_json
11 from keras.preprocessing import image
12 from keras.callbacks import ModelCheckpoint
13 from keras.models import load_model
14 import pickle
15 import cv2
16 import os
17 import tkinter
18 from tkinter import *
19 from tkinter import filedialog
20 from PIL import Image, ImageDraw, ImageTk
```

```
22 |
23 main = tkinter.Tk()
24 main.title("Pedestrian Detection")
25 main.geometry("1000x800")
26
27 l = ['Pedestrian', 'Not pedestrian']
28
```

```

30 # CNN Model
31 def train_Model():
32     model = Sequential()
33
34     # Step 1 - Convolution
35     model.add(Conv2D(32, (2, 2), input_shape = (160, 80, 3), activation = 'relu'))
36
37     # Step 2 - Pooling
38     model.add(MaxPooling2D(pool_size = (2, 2)))
39
40     # Adding a second convolutional layer
41     model.add(Conv2D(64, (2, 2), activation = 'relu'))
42     model.add(MaxPooling2D(pool_size = (2, 2)))
43
44     # Adding a third convolutional layer
45     model.add(Conv2D(128, (2, 2), activation = 'relu'))
46     model.add(MaxPooling2D(pool_size = (2, 2)))
47
48     # Adding a fourth convolutional layer
49     model.add(Conv2D(128, (2, 2), activation = 'relu'))
50     model.add(MaxPooling2D(pool_size = (2, 2)))
51
52     # Step 3 - Flattening
53     model.add(Flatten())
54
55     # Step 4 - Full connection
56     model.add(Dense(units = 64, activation = 'relu'))
57     model.add(Dense(units = 1, activation = 'sigmoid'))
58
59     # Compiling the CNN
60     model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
61     model.summary()

```

```

59 # Compiling the CNN
60 model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
61 model.summary()
62
63 #loading test and train datasets
64 batch_size = 128
65 train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
66 test_datagen = ImageDataGenerator(rescale = 1./255)
67 training_set = train_datagen.flow_from_directory('INRIAPerson/Train', target_size = (160, 80), batch_size = 32, class_mode = 'binary')
68 test_set = test_datagen.flow_from_directory('INRIAPerson/Test', target_size = (160, 80), batch_size = 32, class_mode = 'binary')
69
70 #training the model
71 hist = model.fit_generator(training_set, steps_per_epoch = 5000//batch_size, epochs = 10, validation_data = test_set,
72 | | | | | validation_steps = 1000//batch_size, verbose=1)
73
74 #saving the model
75 model.save("model/trained.h5")
76
77 f = open('model/history.pckl', 'wb')
78 pickle.dump(hist.history, f)
79 f.close()
80
81 f = open('model/history.pckl', 'rb')
82 data = pickle.load(f)
83 f.close()
84
85 acc = data['accuracy']
86 accuracy = acc[-1] * 100
87
88 print("Accuracy:", accuracy, "\n", end= "\n")
89

```

```

91 def detect():
92     #read input image from user
93     filename = filedialog.askopenfilename(initialdir="testImages")
94
95     imagetest = image.load_img(filename, target_size = (160,80))
96
97     im = Image.open(filename)
98     image_arr = np.asarray(im)
99
100    imagetest = image.img_to_array(imagetest)
101    imagetest = np.expand_dims(imagetest,axis=0)
102
103    # Initialize the HOG descriptor
104    hog = cv2.HOGDescriptor()
105    hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
106
107    #detect
108    detections, weights = hog.detectMultiScale(image_arr)
109
110    if detections==():
111        print("\nNo pedestrian detected")
112
113    else:
114        detections_rectangles = detections.tolist()
115        if os.path.exists('model/trained.h5'):
116
117            detector = keras.models.load_model("model/trained.h5")
118
119            #predict the output
120            detector._make_predict_function()
121
122        else:
123            train_Model()
124
125    predict = detector.predict(imagetest)

```

```

126
127    print(detector.summary(),end='\n\n')
128    print(1[np.argmax(predict)],end='\n\n')
129
130    f = open('model/history.pckl', 'rb')
131    data = pickle.load(f)
132    f.close()
133
134    acc = data['accuracy']
135    accuracy = acc[-1] * 100
136
137    print("Accuracy:",accuracy,"\n",end="\n")
138
139    draw = ImageDraw.Draw(im)
140    for x, y, w, h in detections_rectangles:
141        draw.rectangle(
142            [x, y, x + w, y + h], outline=(255, 0, 0))
143
144    im.show()
145
146 def close():
147     main.destroy()
148
149

```

```

150 #code for UI
151 font = ('Calibri', 16, 'bold')
152 title = Label(main, text='Pedestrian Detection', anchor=W, justify=CENTER)
153 title.config(bg='slateblue', fg='white')
154 title.config(font=font)
155 title.config(height=2, width=70)
156 title.place(x=10, y=10)
157
158 #initialise text field
159 font1 = ('Calibri', 14)
160
161 #testing label
162 label = Label(main, text="Testing the model:")
163 label.config(font=font)
164 label.place(x=10, y=100)
165
166 #button to test a new input
167 testButton = Button(main, text="Upload Image", bg='firebrick1', fg='white', command=detect)
168 testButton.place(x=250, y=200)
169 testButton.config(font=font1)
170
171 #exit the user interface
172 exitButton = Button(main, text="Exit", bg='firebrick1', fg='white', command=close)
173 exitButton.place(x=300, y=500)
174 exitButton.config(font=font1)
175
176 main.config(bg='lightblue')
177 main.mainloop()

```

7.2 TESTING

Testing is an important but it is undoubtedly the most underrated and also important part of the development process. When developing a project, the code must satisfy all the requirements that are put forth by the customer. Thus, without a proper and robust testing conditions will not be achieved. In testing, things such as boundary conditions, how long does it take when large data which can overload the system and also the regular cases are considered.

Primarily, there are two very famous and necessary testing techniques, namely, Black box testing and white box testing.

Black box considers functional aspects of the project. In this testing, all the various user input scenarios are taken into account. It is then tested and checked whether it gives the expected output. In most cases it is used when the actual behaviour of the code is not known. The tester creates the test cases as given and is then tested making it a simple but effective process. In this project, there are two main test cases, that have been taken. One is to randomly select a batch of images and test them to check if they detect the pedestrians correctly or not. The other is to select a image of the user's choice and check whether it gives the right output. Both worked well in the project.

In the white box testing, it is used to test the program flow works, or in other words, the behavioural aspects of the program. So here, how the program flow goes when given different inputs are given and it should follow the flow of the program. In this project each and every step is executed and followed when inputs are given.

Test Case 1:

This is a random test case where we use a random image taken and tested. The below image is the representation of output.

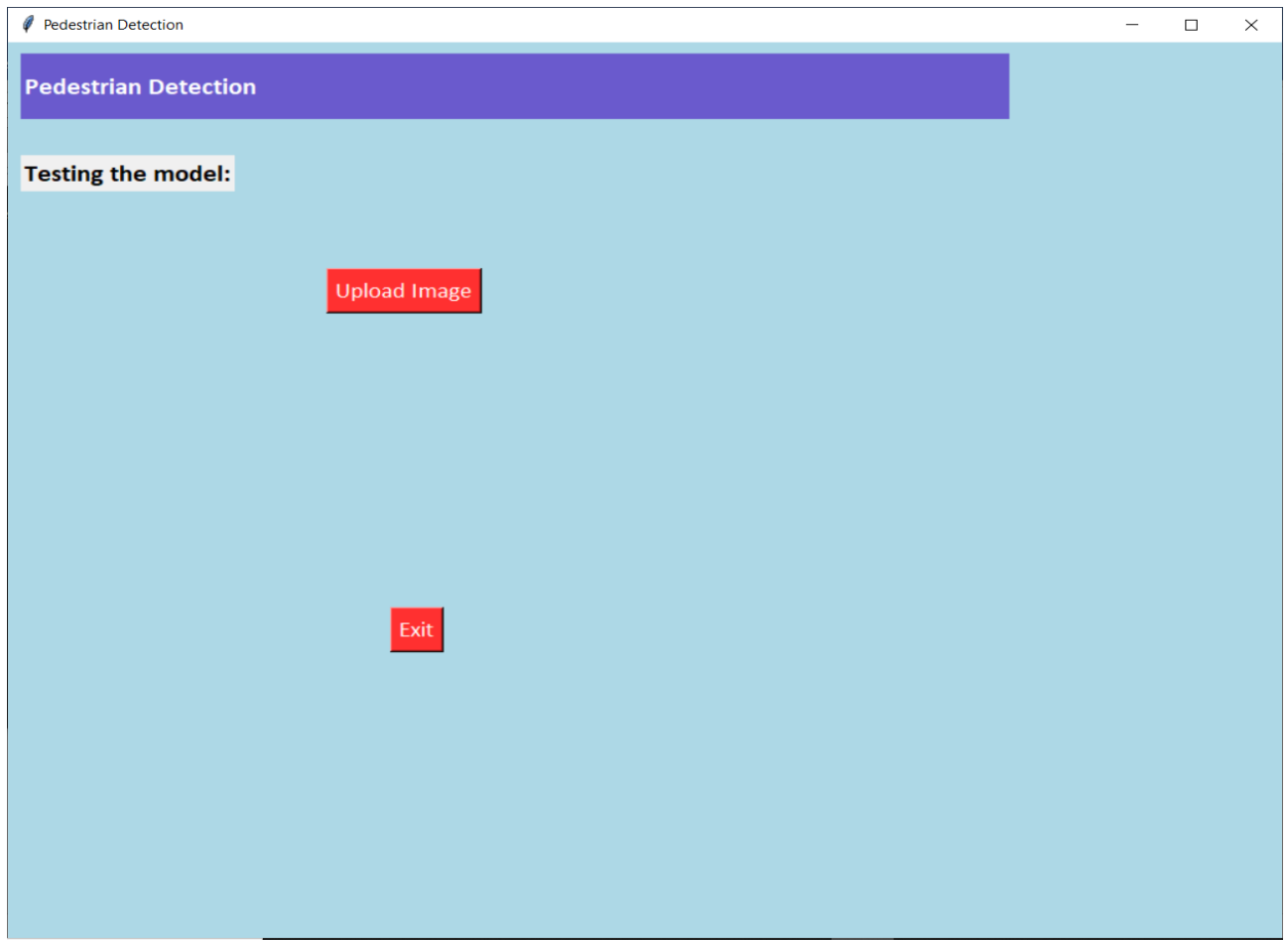


Fig 7.1 Output-1.1

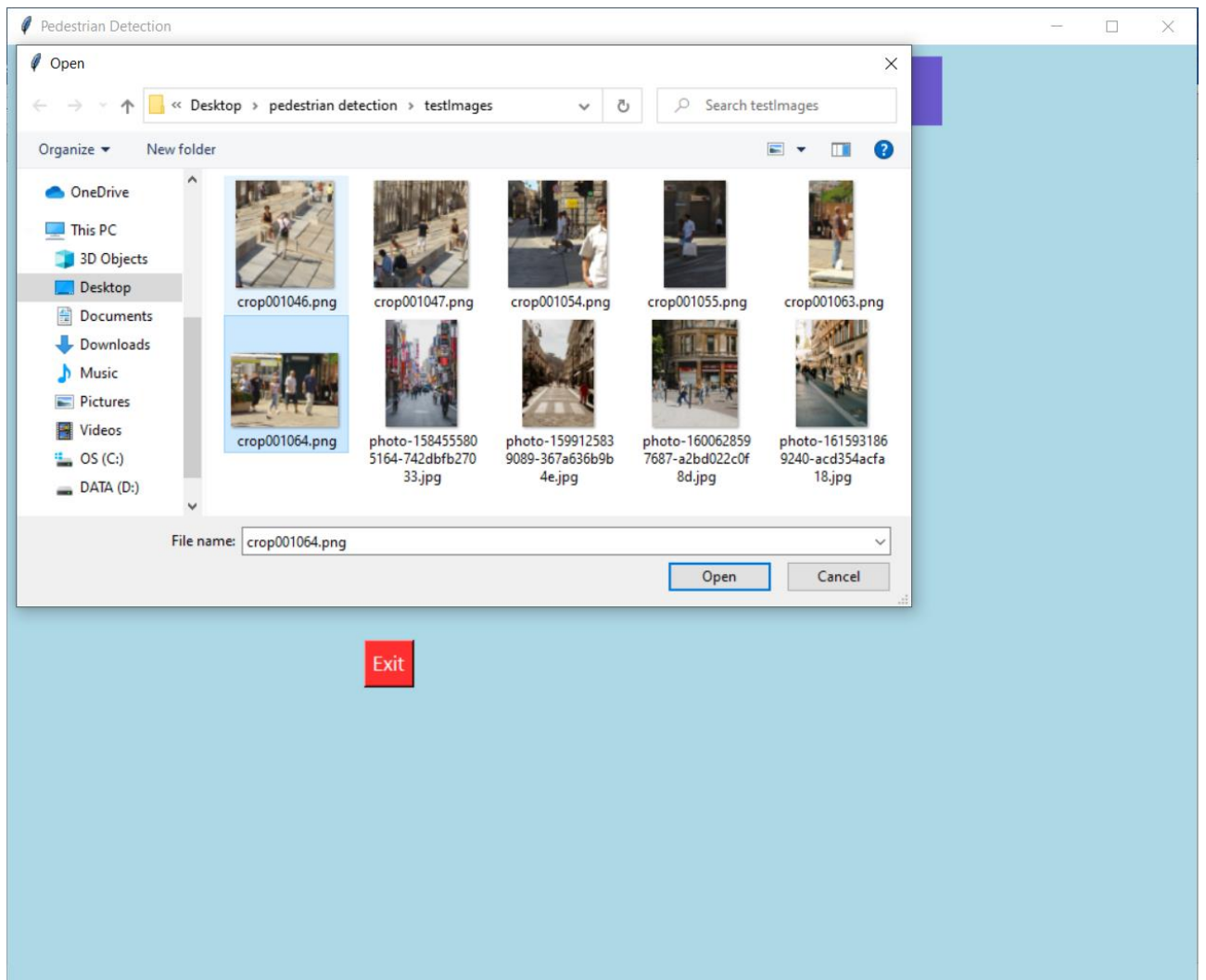


Fig 7.2 Output-1.2

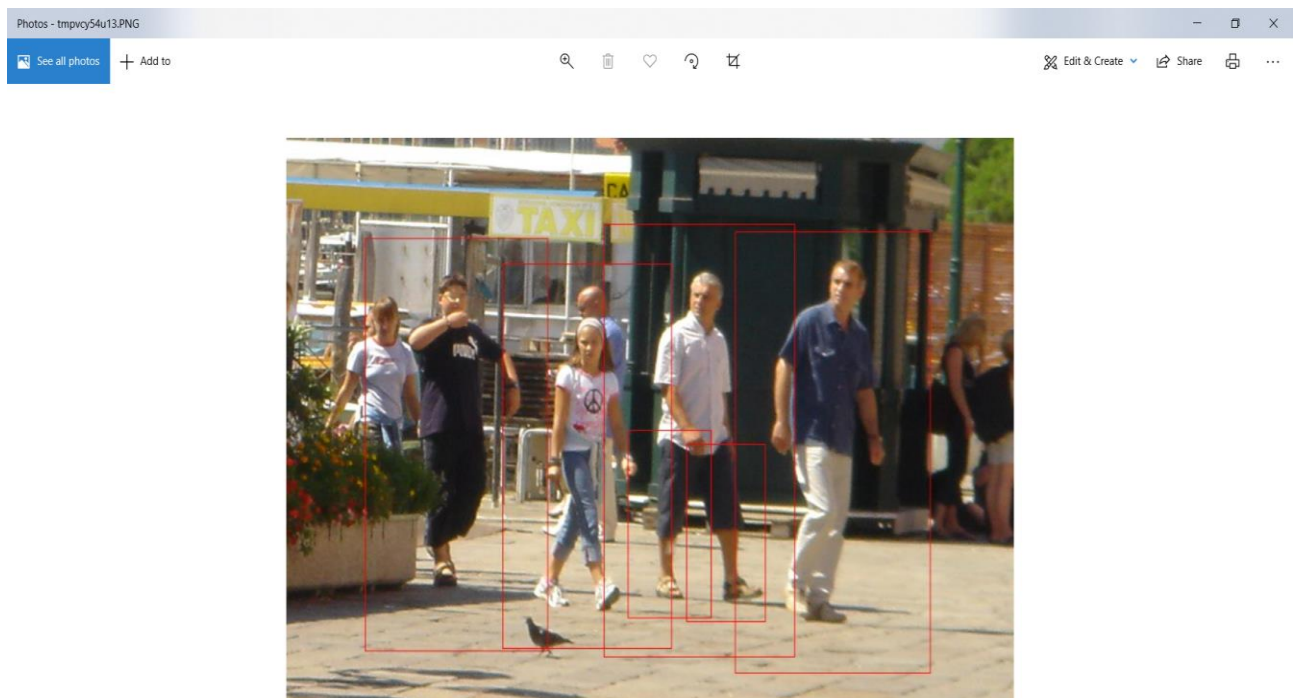


Fig 7.3 Output-1.3

Test Case 2:

This is a testcase where we took an image from the internet containing pedestrian and test it if it can detect them.

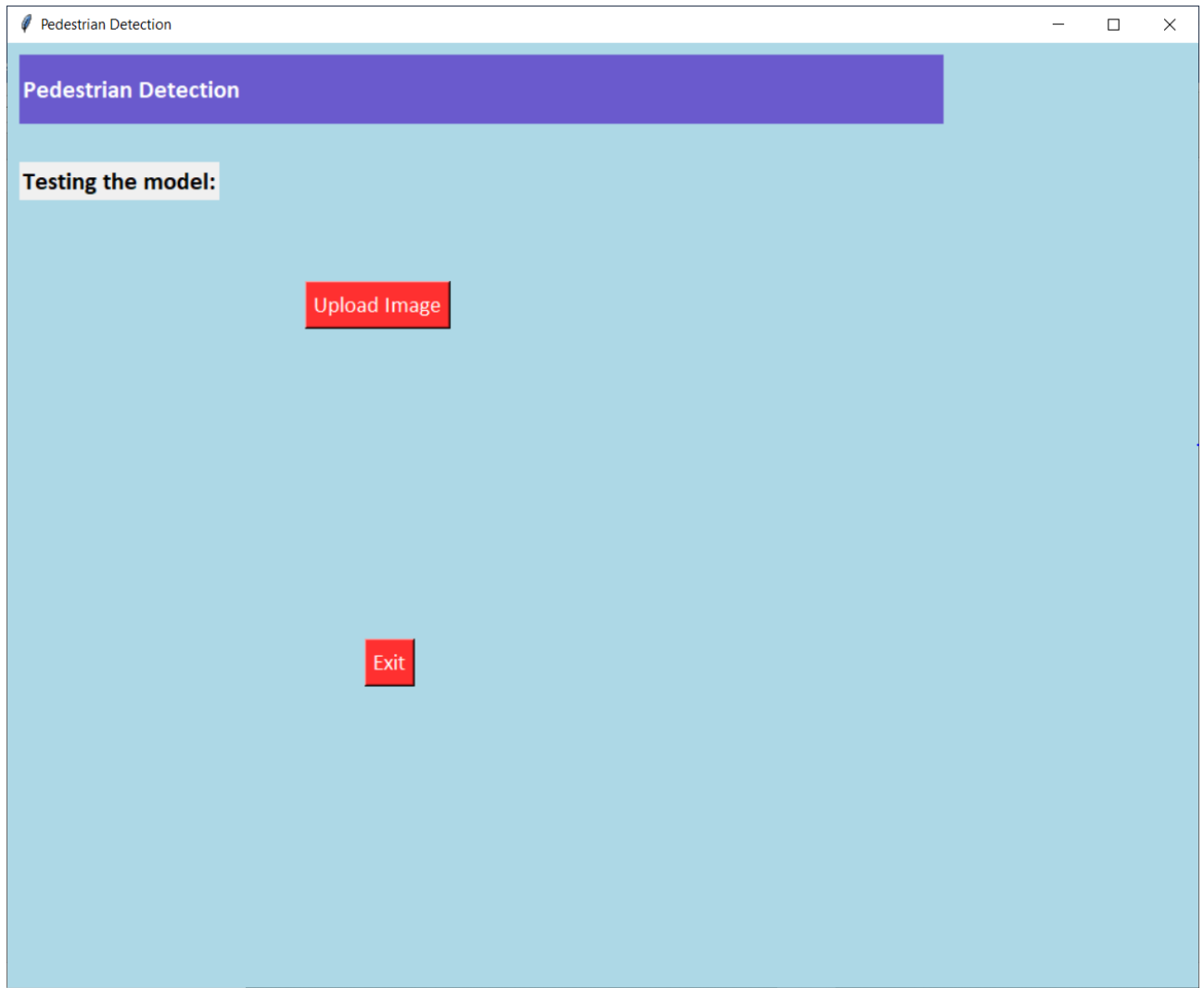


Fig 7.4 Output-2.1

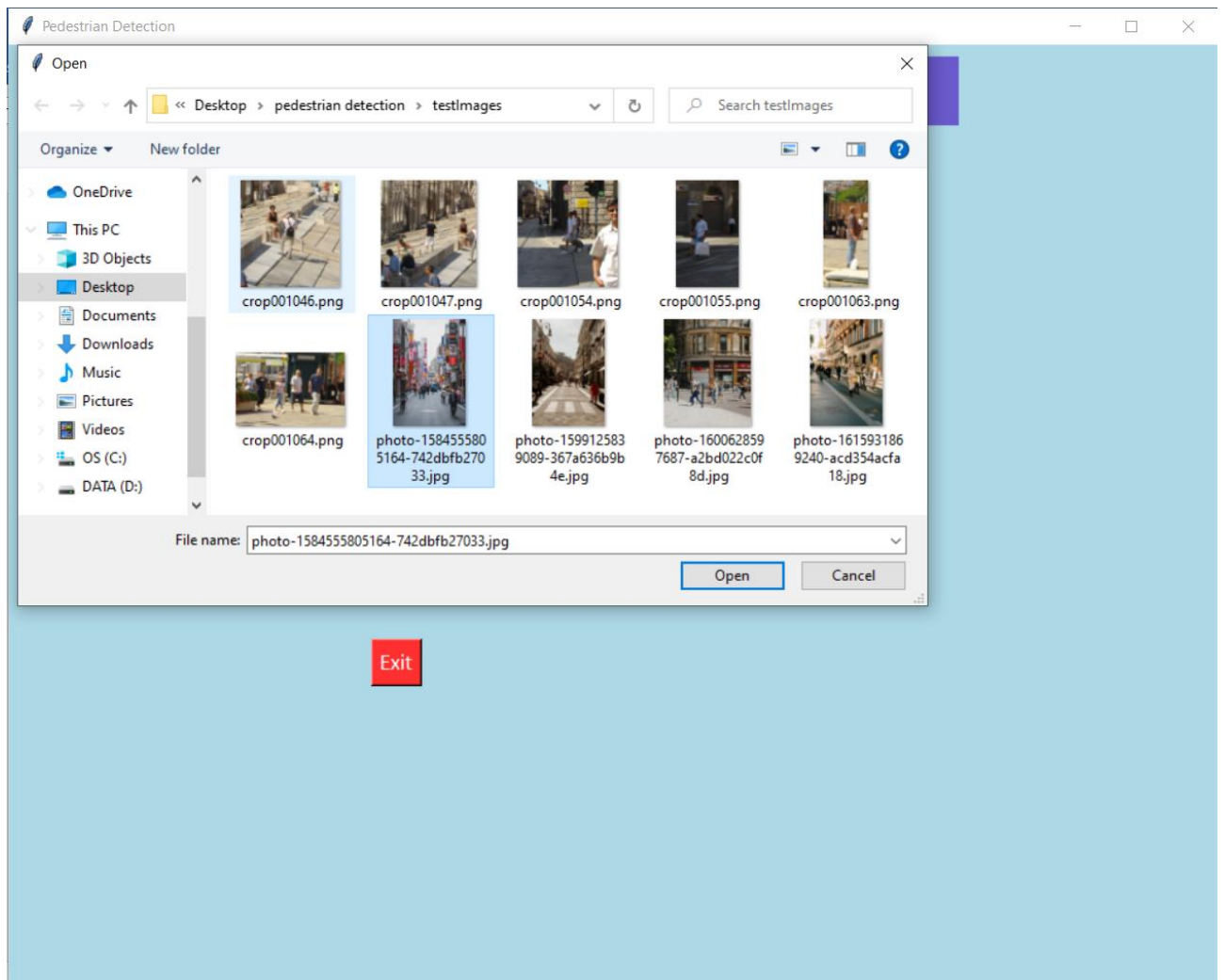


Fig 7.5 Output-2.2

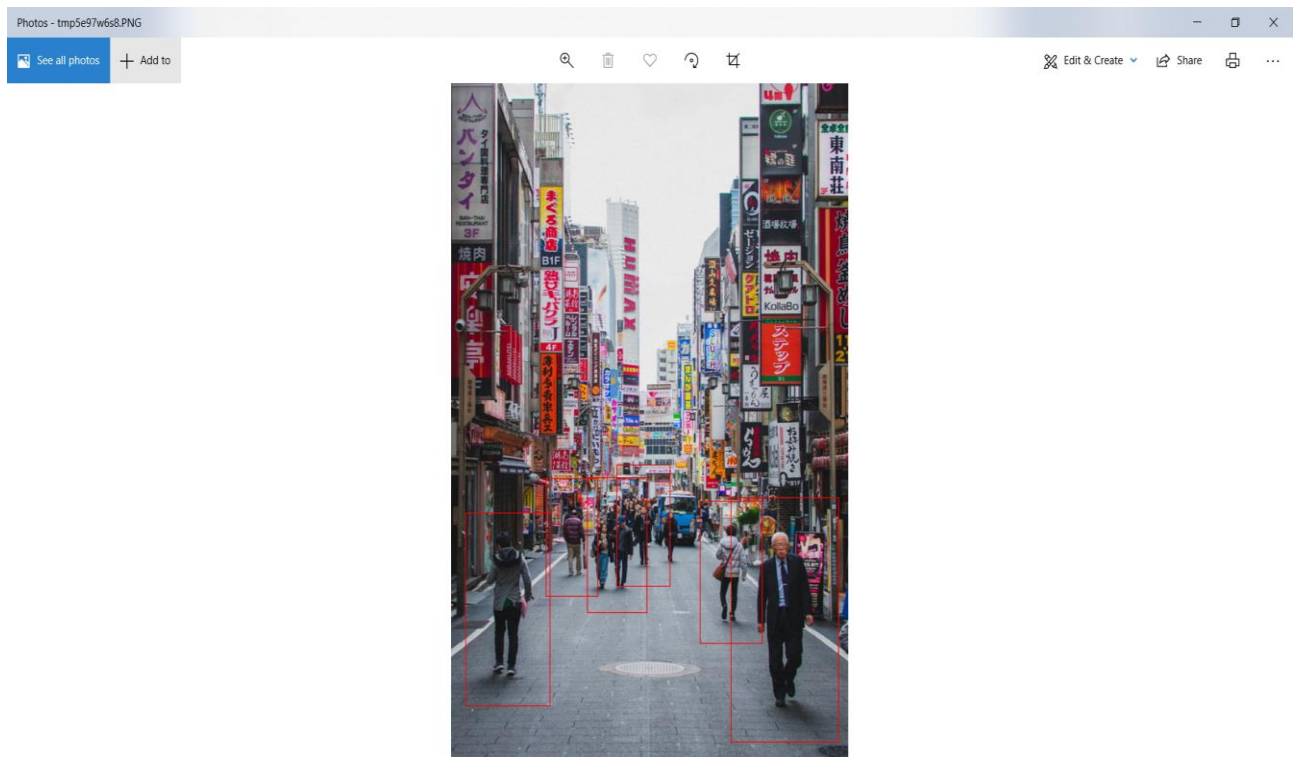
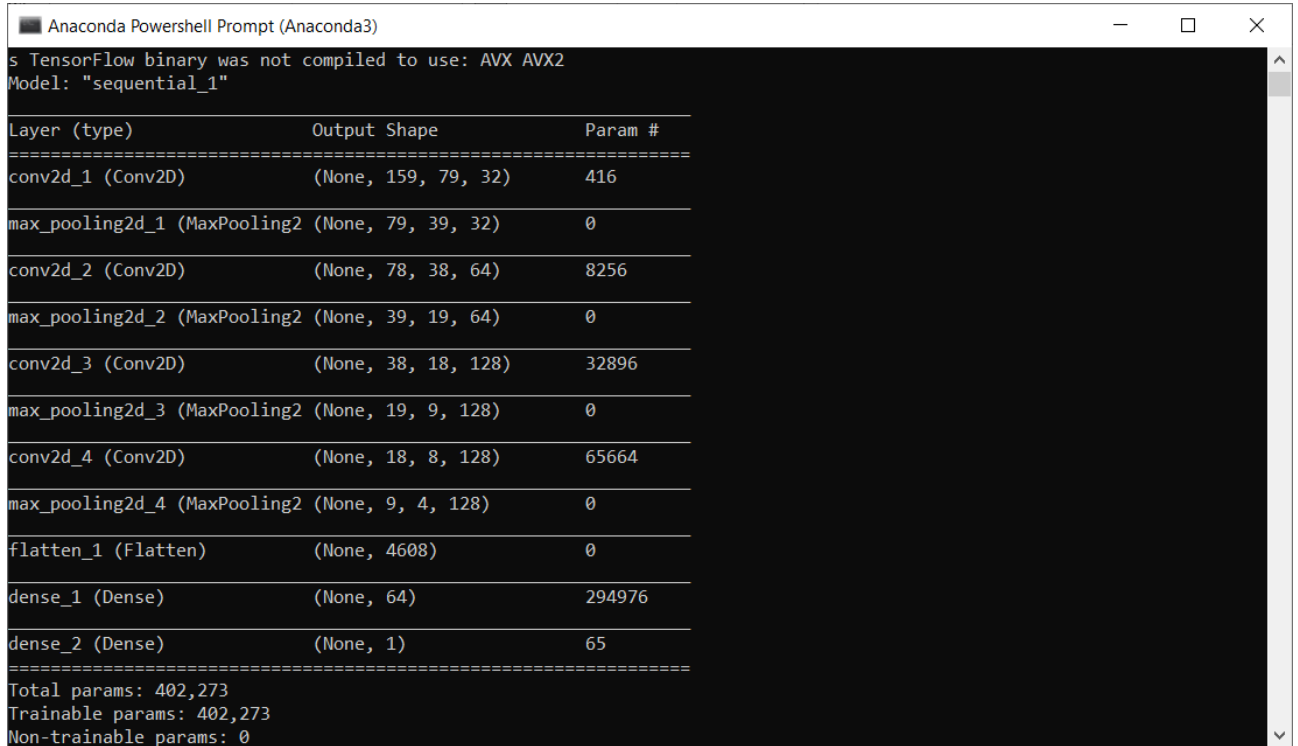


Fig 7.6 Output-2.3

8. RESULTS AND DISCUSSIONS

This chapter deals with the results obtained after training and testing the model. Now after the training is done. The model was tested on different cases as discussed in the above chapter. Now for a user to use the model, it has to be deployed into application using Tkinter.

This below image is architecture of the CNN model used for the pedestrian detection.



```
Anaconda Powershell Prompt (Anaconda3)
s TensorFlow binary was not compiled to use: AVX AVX2
Model: "sequential_1"

Layer (type)                 Output Shape                  Param #
-----
conv2d_1 (Conv2D)            (None, 159, 79, 32)          416
max_pooling2d_1 (MaxPooling2 (None, 79, 39, 32)          0
conv2d_2 (Conv2D)            (None, 78, 38, 64)           8256
max_pooling2d_2 (MaxPooling2 (None, 39, 19, 64)          0
conv2d_3 (Conv2D)            (None, 38, 18, 128)          32896
max_pooling2d_3 (MaxPooling2 (None, 19, 9, 128)           0
conv2d_4 (Conv2D)            (None, 18, 8, 128)           65664
max_pooling2d_4 (MaxPooling2 (None, 9, 4, 128)           0
flatten_1 (Flatten)          (None, 4608)                 0
dense_1 (Dense)              (None, 64)                   294976
dense_2 (Dense)              (None, 1)                    65
-----
Total params: 402,273
Trainable params: 402,273
Non-trainable params: 0
```

Fig 8.1 Architecture

When the code is run from the command prompt the user interface pops up and user needs to click upload button to give an image for detecting pedestrians. The below image is user interface and the result.

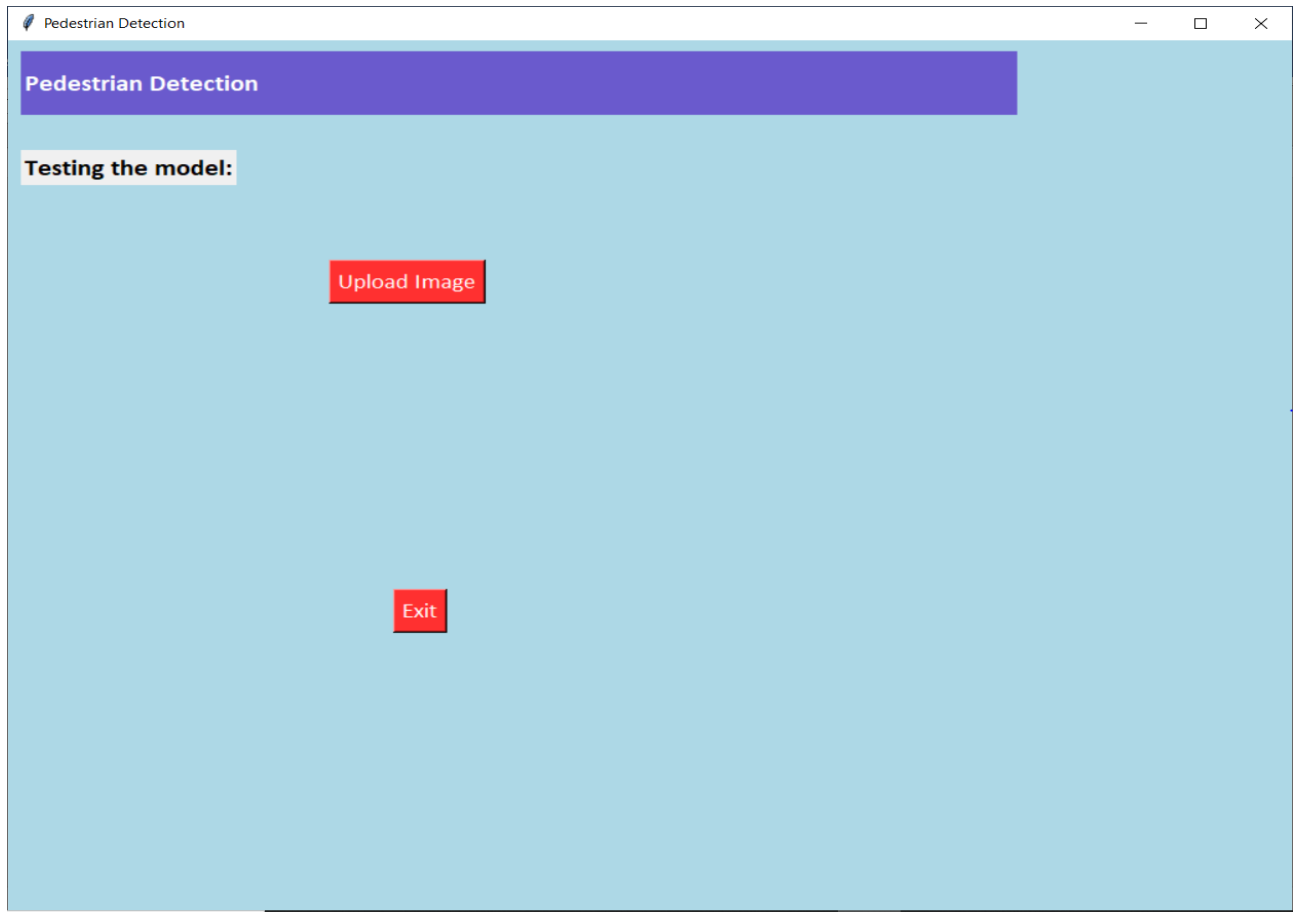


Fig 8.2 Result-1

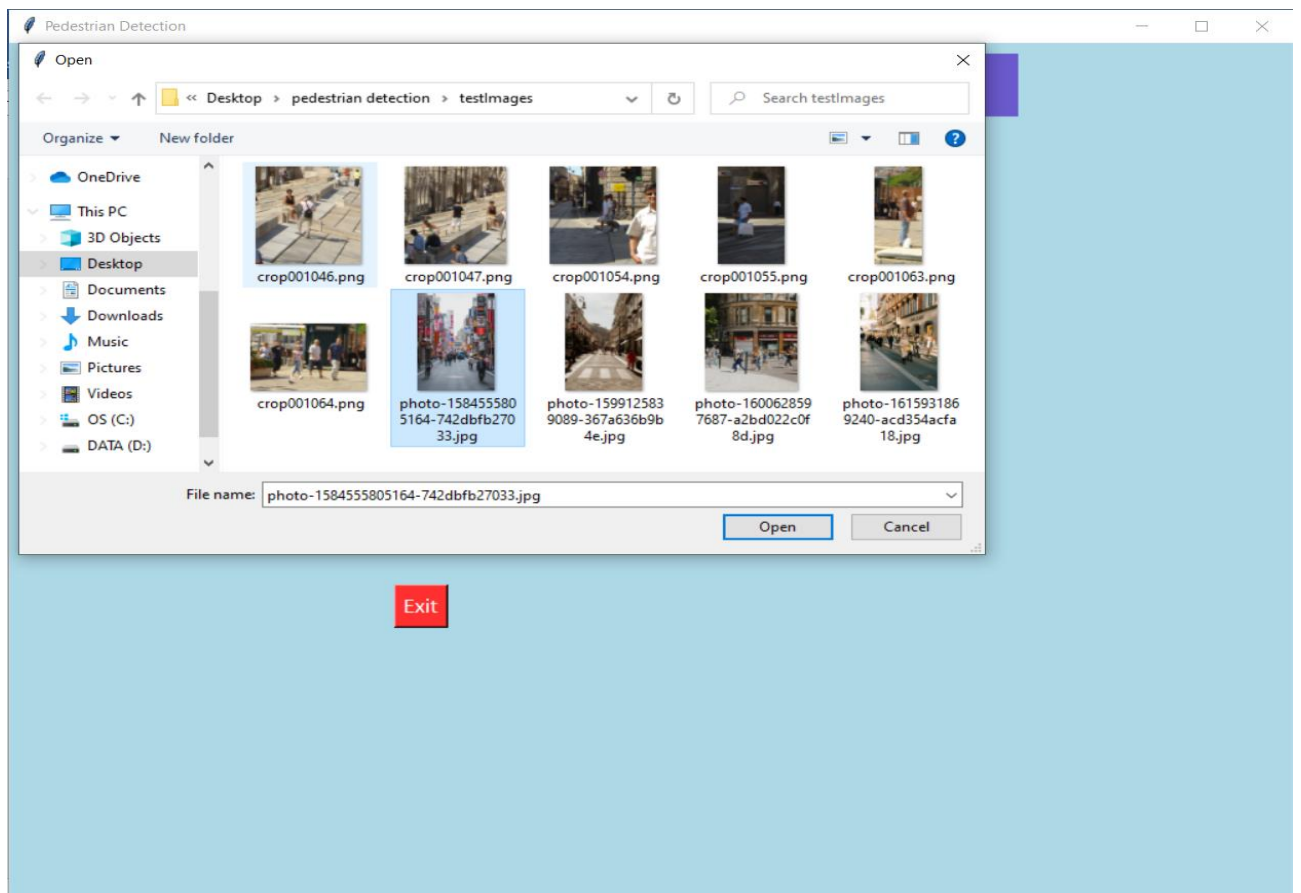


Fig 8.3 Result-2

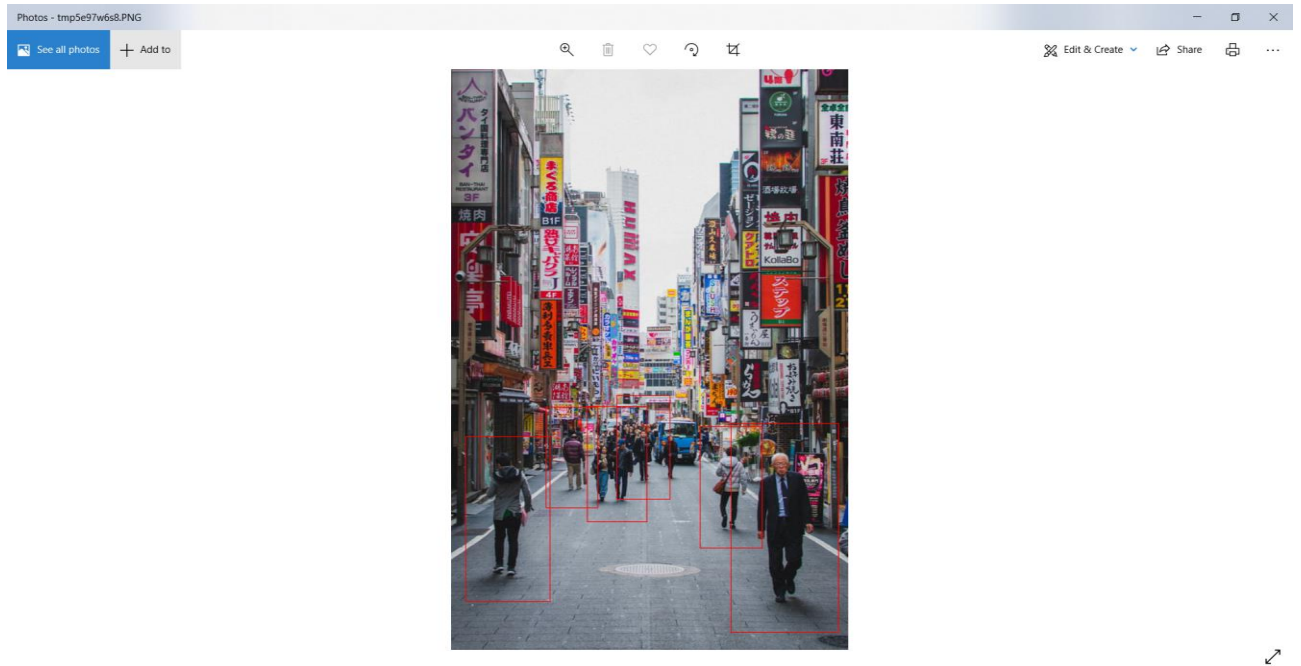


Fig 8.4 Result-3

This image which is uploaded for testing is taken from google and tested for verifying the detection done by the model. By using the combined model of HOG+SVM and CNN the accuracy achieved was 89% .

9. CONCLUSION

This project main goal is to help in improving the detection systems in automated vehicles so that we can avoid cause of accidents and improve security while travelling in vehicles. The detection of pedestrians from the environment as using it is very important for working of automatic driving. There have been researches on this for detecting pedestrians, vehicles, traffic signs, etc. The main objective of the pedestrian detection system is to deliver the system with accurate information in real-time about the surrounding environment so that it can reduce the number of traffic accidents that can occur and save lives.

This project uses a fusion of classical machine learning model and deep learning model to improve the detection accuracy and reduce the miss rate so that it can be useful in real-time efficiently. The project can conclude for now by saying that the model has reached a really acceptable accuracy of 89%.

Coming to the future scope and improvements that can be made to this project include improving the user interface to be easily operatable while using vehicle and making it a webapp or even developing an android application. It can be developed in a way that it takes live video of the while driving and then detect there itself. The ability to add recorded videos using GUI will be a great addition. There are tremendous amounts of research going on and the accuracy of the model has a great chance of improving in the near future.

10. REFERENCES

- [1] S. Ren et al., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks.
- [2] Q. Hu, S. Paisitkriangkrai, C. Shen, A. van den Hengel, F. Porikli, 2016. Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework. IEEE Trans. Intell. Transp. Syst. 17
- [3] Z. Liu, J. Du, F. Tian, J. Wen, 2019. MR-CNN: A Multi-Scale Region Based Convolutional Neural Network for Small Traffic Sign Recognition. IEEE Access
- [4] N. Dalal, B. Triggs, 2005. Histograms of oriented gradients for human detection.

Websites referred

- [1] <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53>
- [2] <https://deeptai.org/machine-learning-glossary-and-terms/stride>
- [3] https://en.wikipedia.org/wiki/Convolutional_neural_network
- [4] <https://cloud.google.com/vision/docs/ocr>
- [5] <https://www.tensorflow.org/>
- [6] <https://stackoverflow.com/questions/24385714/detect-text-region-in-image-using-opencv>