A Project report

on

FINGERPRINT OBJECT ANALYSIS AND IDENTIFICATION

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

by

D. DIVYA	**(184G1A0514)**
D. CHANDRIKA	**(184G1A0512)**
R. MANISH KUMAR	**(184G1A0542)**
B. SHAHID AFRIDI	**(194G5A0507)**

Under the Guidance of

Dr. G. Hemanth Kumar Yadav M.Tech., Ph.D

Associate Professor

Department Of Computer Science & Engineering

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUA & Approved by AICTE)

(Accredited by NAAC with 'A' Grade & Accredited by NBA (EEE, ECE & CSE))

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu-515701.

2021-2022

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTU & Approved by AICTE)

(Accredited by NAAC with 'A' Grade & Accredited by NBA (EEE, ECE & CSE))

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu-515701.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

This is to certify that the project report entitled Fingerprint Object Analysis And Identification is the bonafide work carried out by **D. Divya** bearing Roll Number 184G1A0514**, D. Chandrika** bearing Roll Number 184G1A0512**, R. Manish Kumar** bearing Roll Number 184G1A0542 and **B. Shahid Afridi** bearing Roll Number 194G5A0507 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2021-2022.

Signature of the Guide

Dr. G. Hemanth Kumar Yadav M.Tech., Ph.D

Associate Professor

Head of the Department

Mr. P. Veera Prakash M.Tech., (Ph.D)

Assistant Professor & HOD

Date:

EXTERNAL EXAMINER

Place: Rotarypuram

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we now have the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express my indebted gratitude to our Guide **Dr. G. Hemanth Kumar Yadav, M.Tech., Ph.D, Associate Professor, Computer Science & Engineering**, who has guided us a lot and encouraged us in every step of the project work. We thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep-felt gratitude to **Mr. K. Venkatesh, M.Tech., Assistant Professor,** project coordinator valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We are very much thankful to **Mr. P. Veera Prakash, M.Tech., (Ph.D), Assistant Professor & Head of the Department, Computer Science & Engineering,** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey my special thanks to **Dr. G. Bala Krishna, Ph.D., Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and our friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities**.**

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

Project Associates

DECLARATION

We, Ms. D. Divya with reg no: 184G1A0514, Ms. D. Chandrika with reg no: 184G1A0512, Mr. R. Manish Kumar with reg no: 184G1A0542, Mr. B. Shahid Afridi with reg no: 174G5A0507 students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that the dissertation entitled "FINGERPRINT OBJECT ANALYSIS AND IDENTIFICATION" embodies the report of our project work carried out by us during IV year Bachelor of Technology under the guidance of Dr. G. Hemanth Kumar Yadav, M.Tech., Ph.D, Department of CSE, SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, and this work has been submitted for the partial fulfillment of the requirements for the award of the Bachelor of Technology degree.

The results embodied in this project have not been submitted to any other University of Institute for the award of any Degree.

D. DIVYA	Reg no: 184G1A0514
D. CHANDRIKA	Reg no: 184G1A0512
R. MANISH KUMAR	Reg no: 184G1A0542
B. SHAHID AFRIDI	Reg no: 194G5A0507

CONTENTS **Page No.**

LIST OF FIGURES

LIST OF SCREENS

LIST OF ABBREVIATIONS

CSV	Comma-separated values
CNN	Convolutional Neural Network
CV	Computer Vision
SRS	Software Requirement Specification
UML	Unified modelling language
Numpy	Numerical Python
ML	Machine Learning
SKimage	Scikit-image

ABSTRACT

Fingerprint images in crime scene are important clues to solve serial cases. In this paper we present a complete crime scene fingerprint identification system using Harris Corner detector and OpenCV. Images are acquired from crime scene using methods ranging from precision photography to complex physical and chemical processing techniques and saved as the database.

The images collected from the crime scene are usually incomplete and hence difficult to categorize. Suitable enhancement methods are required for pre-processing the fingerprint images. Minutiae are extracted from the fingerprint images. The features of preprocessed data are fed into the Harris Corner as input to train and test the network. The experimental results demonstrated on database using Open CV-Python shows high accuracy of 80% recognition an partial or full fingerprints in the criminal database.

Keywords:

Machine learning, Harris Corner, Open CV, Minutiae, ReLU, Python

CHAPTER - 1
INTRODUCTION

Fingerprints in the crime scene plays an important role to identify the criminal involved in the crime. Crime scene images (CSI) are images taken from the crime spot. When crime is occurred, the investigator takes both latent and patent sample of fingerprints left behind. Normally in crime scene, the investigators apply a fine dusting powder (aluminium dust or black granular) to the surface in which fingerprints to be extracted. The fingerprint images are subjected to image pre-processing, image feature extraction and identification analysis. In order to solve serial cases, we try to build a model that predicts the Criminal person for a particular crime using Machine Learning.

Earlier there exists many approaches to predict the Crime rate. Lot of work has been carried out to predict crime rate using machine learning dataset Different levels of accuracy have been attained using various machine learning techniques. In the existing system, the prediction of crime rate was done using Machine Learning techniques such as SVM which doesn't give the accurate results. In the existing system, the prediction of crime rate was done using Machine Learning techniques such as SVM which doesn't give the accurate results. To get the accurate outcome, we use Harris Corner detector.

Finger prints in the crime scene plays an important role to identify the criminal involved in the crime. Crime scene images (CSI) are images taken from the crime spot. When crime is occurred, the investigator takes both latent and patent sample of fingerprints left behind. The patent fingerprints are visible by naked eye, so they are simply photographed. But latent fingerprints are invisible and these samples are more difficult to perceptible. These samples can be lifted through different techniques. The use of cyanoacrylate vapors which sticks to prints and make them visible in the present of normal light. This method is much difficult, so normally in crime scene, the investigators apply a fine dusting powder (aluminium dust or black granular) to the surface in which fingerprints to be extracted. The dust actually sticks to the fingerprint then they use clear tape to lift the fingerprint. After the lifting the fingerprints, the prints are scanned and saved in the digital image form. The fingerprints taken from the crime scene is unintentionally made and these images are noisy or partial prints and difficult to identify[1].

1.1 Objective of the Project

The objective is to perform techniques for crime detection. On the basis of a performance evaluation, a best suited predictive model is suggested for the crime detection. The results are summarized in terms of accuracy of openCV techniques taken for detection. The main objective of the system is to analyze the crime rate and to predict whether a fingerprint matches with a crime fingerprint using machine learning algorithm.

1.2 Machine Learning

Machine Learning is the area of study which enables machines to learn without being explicitly programmed. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data". A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.

Machine learning gives a system the ability to learn automatically and improve its recommendations using data alone, with no additional programming needed. Because retailers generate enormous amounts of data, machine learning technology quickly proves its value. When a machine learning system is fed data—the more, the better— it searches for patterns. Going forward, it can use the patterns it identifies within the data to make better decisions. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

1.2.1 Convolutional Neural Network

A convolutional neural network (CNN, or ConvNet) is another class of deep neural networks. CNNs are most commonly employed in computer vision. Given a series of images or videos from the real world, with the utilization of CNN, the AI system learns to automatically extract the features of these inputs to complete a specific task, e.g., image classification, face authentication, and image semantic segmentation.

Different from fully connected layers in MLPs, in CNN models, one or multiple convolution layers extract the simple features from input by executing convolution operations. Each layer is a set of nonlinear functions of weighted sums at

different coordinates of spatially nearby subsets of outputs from the prior layer, which allows the weights to be reused[2].

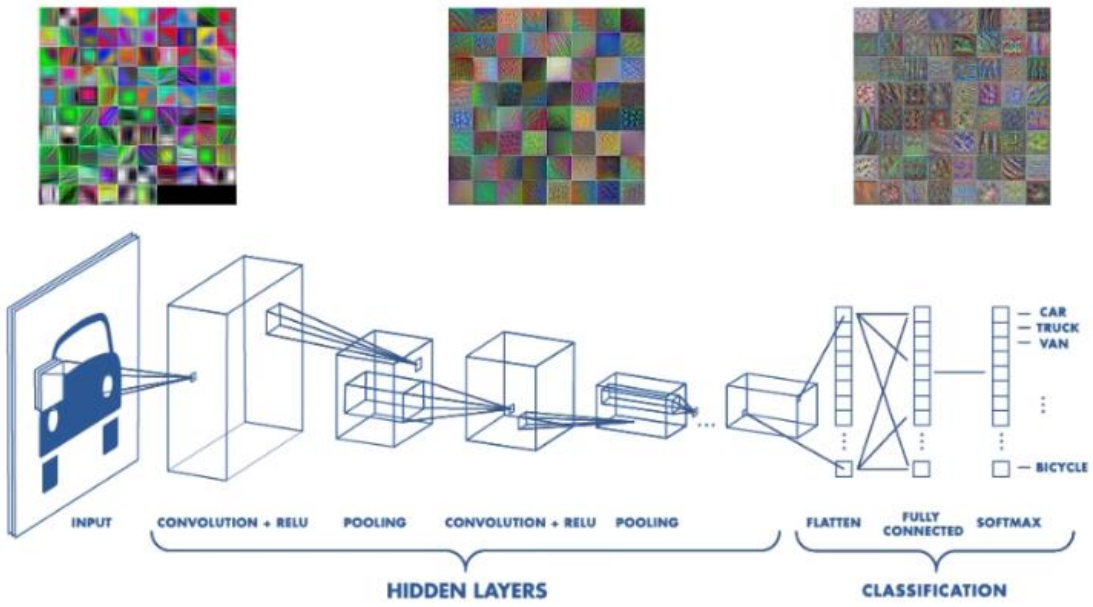


Fig.1.1. Convolutional Neural Network

1.3 Harris Corner Detector

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness.[3] Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation and illumination. Although corners are only a small percentage of the image, they contain the most important features in restoring image information, and they can be used to minimize the amount of processed data for motion tracking, image stitching, building 2D mosaics, stereo vision, image representation and other related computer vision areas.

In order to capture the corners from the image, researchers have proposed many different corner detectors including the Kanade-Lucas-Tomasi (KLT) operator and the Harris operator which are most simple, efficient and reliable for use in corner detection. These two popular methodologies are both closely associated with and based on the local structure matrix. Compared to the Kanade-Lucas-Tomasi corner detector, the Harris corner detector provides good repeatability under changing illumination and rotation, and therefore, it is more often used in stereo matching and

image database retrieval. Although there still exists drawbacks and limitations, the Harris corner detector is still an important and fundamental technique for many computer vision applications.

1.4 OpenCV

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing[3].

1.5 Steps Involved in Model

1) First Determine the type of training dataset
2) Collect/Gather the labelled training data.
3) Split the training dataset into training dataset, test dataset, and validation dataset.
4) Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
5) Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.

6) Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
7) Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

1.6 Convolutional Neural Network

CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. In deep learning, a convolutional neural network **(**CNN/ConvNet**)** is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

1.6.1 Convolution Layer

Convolution is an orderly procedure where two sources of information are intertwined; it's an operation that changes a function into something else. Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g. enhance edges and emboss) CNNs enforce a local connectivity pattern between neurons of adjacent layers.

The first layer of a Convolutional Neural Network is always a Convolutional Layer. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. A convolution converts all the pixels in its receptive field into a single value. For example, if you would apply a convolution to an image, you will be decreasing the image size as well as bringing all the information in the field together into a single pixel. The final output of the convolutional layer is a vector. Based on the type of problem we need to solve and on the kind of features we are looking to learn, we can use different kinds of convolutions.

The most common type of convolution that is used is the 2D convolution layer and is usually abbreviated as conv2D. A filter or a kernel in a conv2D layer "slides" over the 2D input data, performing an elementwise multiplication. As a result, it will be summing up the results into a single output pixel. The kernel will perform the same operation for every location it slides over, transforming a 2D matrix of features into a different 2D matrix of features[4].

There are two main types of separable convolutions: spatial separable convolutions, and depthwise separable convolutions. The spatial separable convolution deals primarily with the spatial dimensions of an image and kernel: the width and the height. Compared to spatial separable convolutions, depthwise separable convolutions work with kernels that cannot be "factored" into two smaller kernels. As a result, it is more frequently used.

1.6.2 ReLU Layer

Convolutional filters start at the upper left corner on top of every pixel in input image and at every position, it's going to dot product and it will produce output which is called activation map and fill it in activation function. ReLU was starting to be used a lot around 2012 when we had AlexNet, the first major convolutional neural network that was able to do well on ImageNet and large-scale data. ReLU stands for **Re**ctified **L**inear **U**nit, and is represented by the function. Sigmoid activation function was quite popular when train neural networks 10 years ago but it has vanishing gradients problem. The general recommendation is that you probably want to stick with ReLU for most cases or default choice because it tends to work well for a lot of different architectures.

1.6.3 Pooling Layer

The pooling operation involves sliding a two-dimensional filter over each channel of feature map and summarising the features lying within the region covered by the filter. Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated

by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

1.6.4 Fully Connected Layer

Fully connected layers are an essential component of Convolutional Neural Networks (CNNs), which have been proven very successful in recognizing and classifying images for computer vision. The CNN process begins with convolution and pooling, breaking down the image into features, and analyzing them independently. The result of this process feeds into a fully connected neural network structure that drives the final classification decision.

Fully connected layers in a CNN are not to be confused with fully connected neural networks – the classic neural network architecture, in which all neurons connect to all neurons in the next layer. Convolutional neural networks enable deep learning for computer vision. The classic neural network architecture was found to be inefficient for computer vision tasks. Images represent a large input for a neural network (they can have hundreds or thousands of pixels and up to 3 color channels). In a classic fully connected network, this requires a huge number of connections and network parameters.

A convolutional neural network leverages the fact that an image is composed of smaller details, or features, and creates a mechanism for analyzing each feature in isolation, which informs a decision about the image as a whole. As part of the convolutional network, there is also a fully connected layer that takes the end result of the convolution/pooling process and reaches a classification decision.

1.7 Dimensionality Reduction

Dimensionality reduction technique can be defined as, "It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information." These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems. It is commonly used in the fields that deal with high dimensional data. It can also be used for data visualization, etc[5].

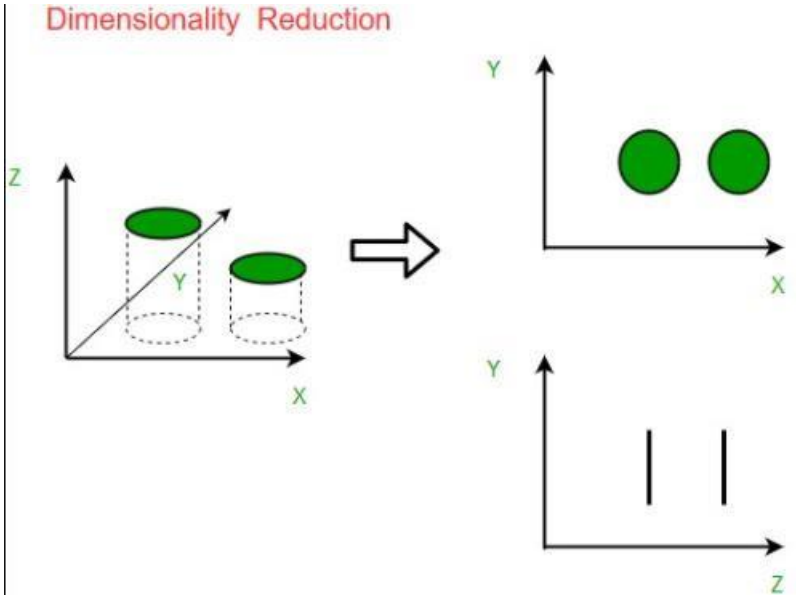


Fig.1.2. Dimensionality Reduction

1.7.1 Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

1.7.2 Feature Extraction

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

1.7.3 Benefits of applying Dimensionality Reduction

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- It removes the redundant features (if present) by taking care of multicollinearity.

CHAPTER - 2
LITERATURE SURVEY

Pavithra [1] proposed an algorithm for crime scene fingerprint image detection by using Convolutional Neural Network (CNN). Images acquire from crime scene is complex in physical appearance. So, image pre-processing and feature extraction is used are fed into the CNN and accuracy of 80% is achieved.

B. Wenxuan [2] proposed an algorithm which is focused on feature extraction by the edges of the fingerprint obtained. For this purpose clustering with neighboring points.

O. I. Abiodun [3] focused using artificial neural network (ANN) for feature extraction from the fingerprint images and identification purpose. The experimental result of the proposed algorithm with some existing algorithms such as GAN, SAE, DBN, RBM, RNN, RBFN, PNN, CNN, SLP, MLP, MLNN.

Serafim [4] proposed an algorithm which is used to segment region of interest in fingerprint image using convolutional neural networks (CNN) without pre-processing steps.

Han [5] proposed fingerprint image enhancement technique termed as adaptive median filter which is used to remove impulse noise. Its performance is measured with existing median filter and it outperforms better with respect to traditional method.

2.1 Proposed system

The fingerprints of the crime scene are taken from the crime location and are converted into digital form. The finger print images of both crime and non-crime person together forms the dataset. Image pre-processing step is included to remove noises present in the images. This enhances the quality of images in the dataset. As a part of enhancement,

Otsu thresholding is used, which converts greyscale image in to binary image. Segmentation is a process of segmenting a precise or trivial detail of the fingerprints which extract the minutiae marking from the dataset. By applying the image thinning operation region of interest (ROI) is extracted. The filtered ROI images are given as input to the Harris Corner Detector.

The dataset is divided into training and testing sets. Then the model will be trained as per our requisition. At the last step, we use Matlab software to identify the suspect fingerprint match with the fingerprint in the dataset or not. If the fingerprint matches, the system identifies the suspect and gives the output as a person is a crime person or not.

CHAPTER - 3

ANALYSIS

3.1 Introduction

The Analysis Phase is where the project life cycle begins. This is the phase where you break down the deliverables in the high-level Project Charter into the more detailed business requirements. Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements. This formal process, which will be developed in more detail, consists of four basic steps.

1. **Elicitation** – I ask questions, you talk, I listen

2. **Validation** – I analyze, I ask follow-up questions

3. **Specification** – I document, I ask follow-up questions

4. **Verification** – We all agree

Most of the work in the Analysis Phase is performed by the role of analyst.

3.2 Software Requirement Specification

SRS is a document created by a system analyst after the requirements are collected. SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The requirements received from clients are written in natural language. It is the responsibility of system analysts to document the requirements in technical

language so that they can be comprehended and useful by the software development team.

3.3 Hardware Requirements

Any Contemporary PC.

3.4 Software Requirements

Operating system: Windows 7 Ultimate or above
Tools: Pycharm
Dataset: CSV file
Languages Used: Python

3.5 PyCharm

3.5.1 INTRODUCTION

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

PyCharm is available in three editions:

- *Community* (free and open-sourced): for smart and intelligent Python development, including code assistance, refactorings, visual debugging, and version control integration.
- *Professional* (paid) : for professional Python, web, and data science development, including code assistance, refactorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.
- *Edu* (free and open-sourced): for learning programming languages and related technologies with integrated educational tools.

Supported languages

To start developing in Python with PyCharm you need to download and install Python from python.org depending on your platform.

PyCharm supports the following versions of Python:

- **Python 2:** version 2.7
- **Python 3:** from the version 3.6 up to the version 3.11

Besides, in the *Professional* edition, one can develop Django , Flask, and Pyramid applications. Also, it fully supports HTML (including HTML5), CSS, JavaScript, and XML: these languages are bundled in the IDE via plugins and are switched on for you by default. Support for the other languages and frameworks can also be added via plugins (go to Settings | Plugins or PyCharm | Preferences | Plugins for macOS users, to find out more or set them up during the first IDE launch).

3.5.2 Installation

- website https://www.jetbrains.com/pycharm/download/ and Click the "DOWNLOAD" link under the Community Section "DOWNLOAD" link under the Community Section the Community Section.
- Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click "Next".

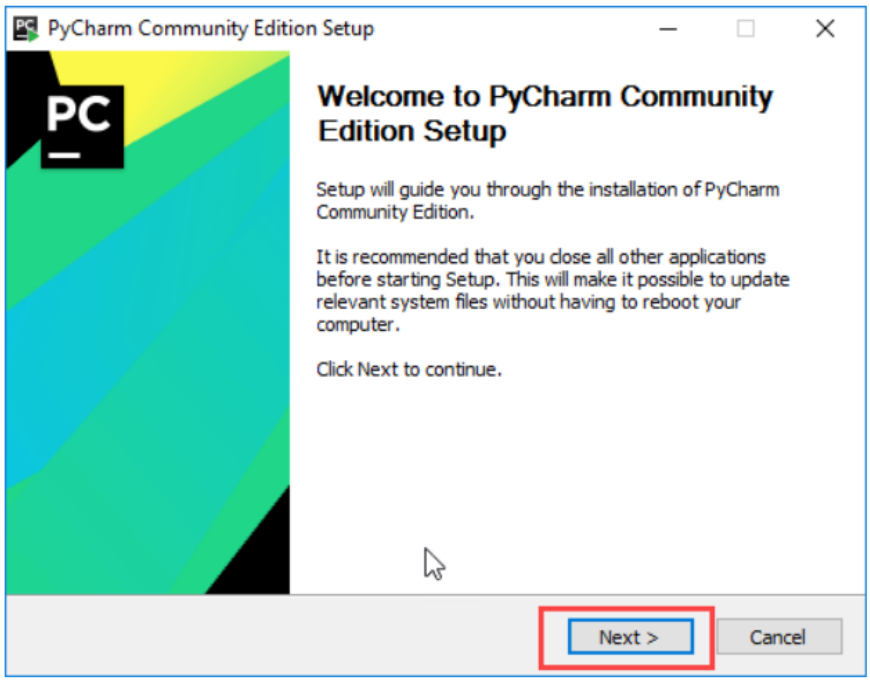


Fig.3.1. PyCharm Community Edition Setup

- On the next screen, Change the installation path if required. Click “Next”.
- On the next screen, you can create a desktop shortcut if you want and click on “Next”.
- Choose the start menu folder. Keep selected JetBrains and click on “Install”.

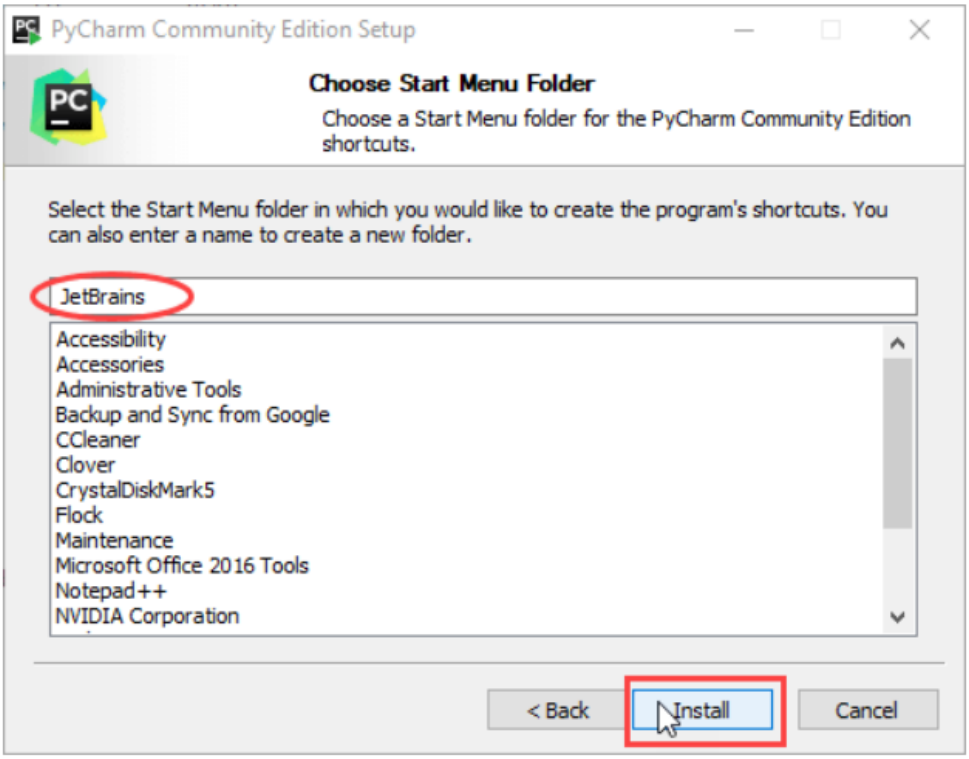


Fig.3.2. Installation of PyCharm

- Wait for the installation to finish.
- Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
- After you click on “Finish,” the Following screen will appear.

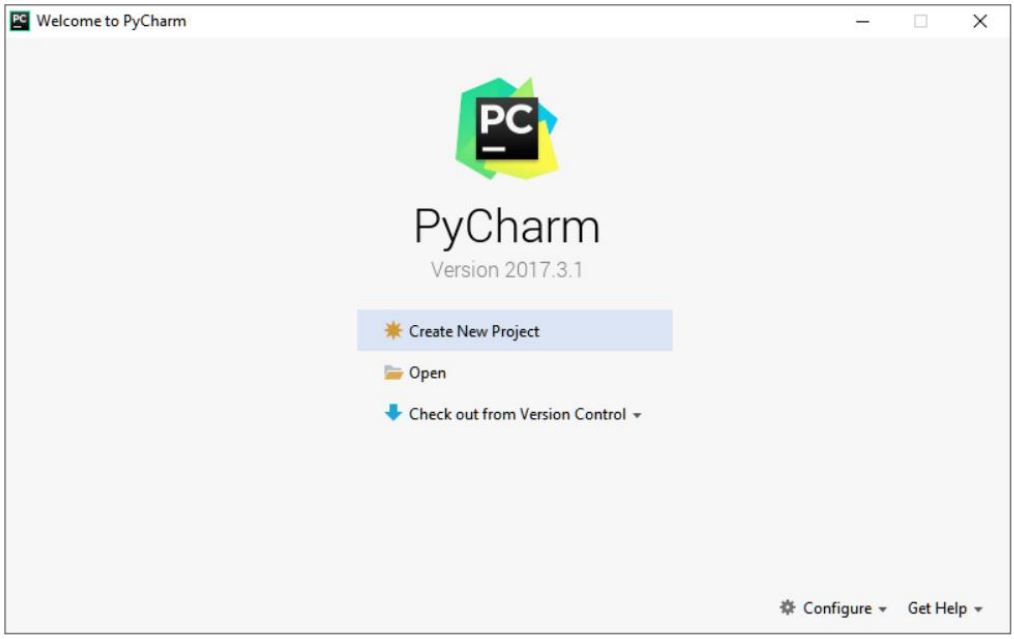
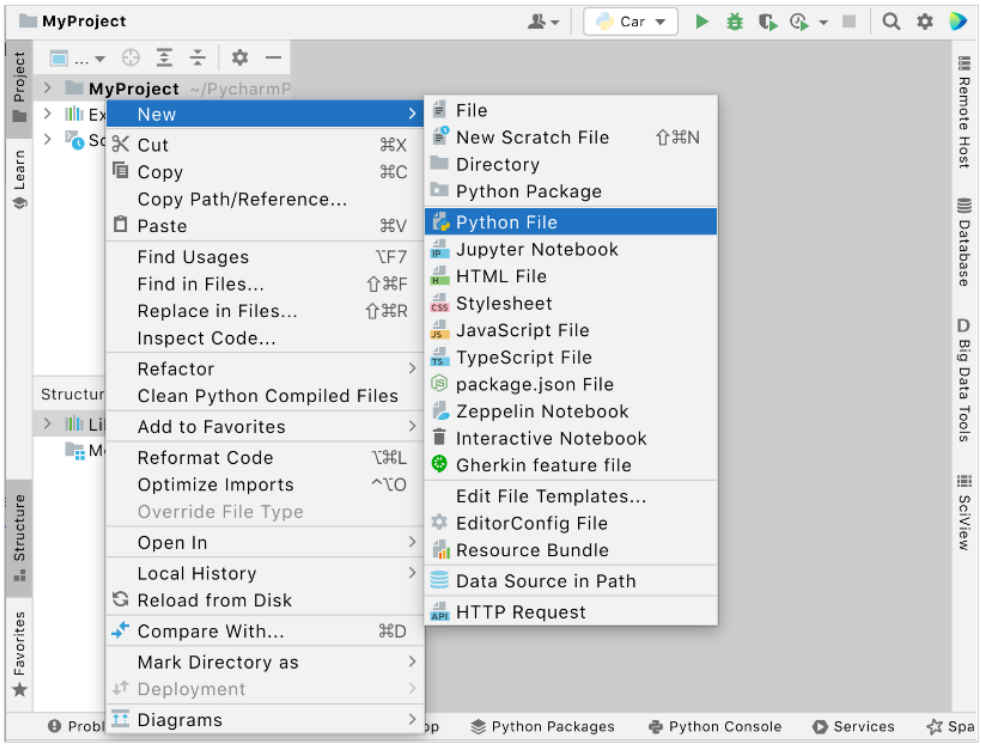


Fig.3.3. Welcome to PyCharm

3.5.3 Create a Python File

- In the Project tool window, select the *project root* (typically, it is the root node in the project tree), right-click it, and select File | New.



- Select the option Python File from the context menu, and then type the new filename.

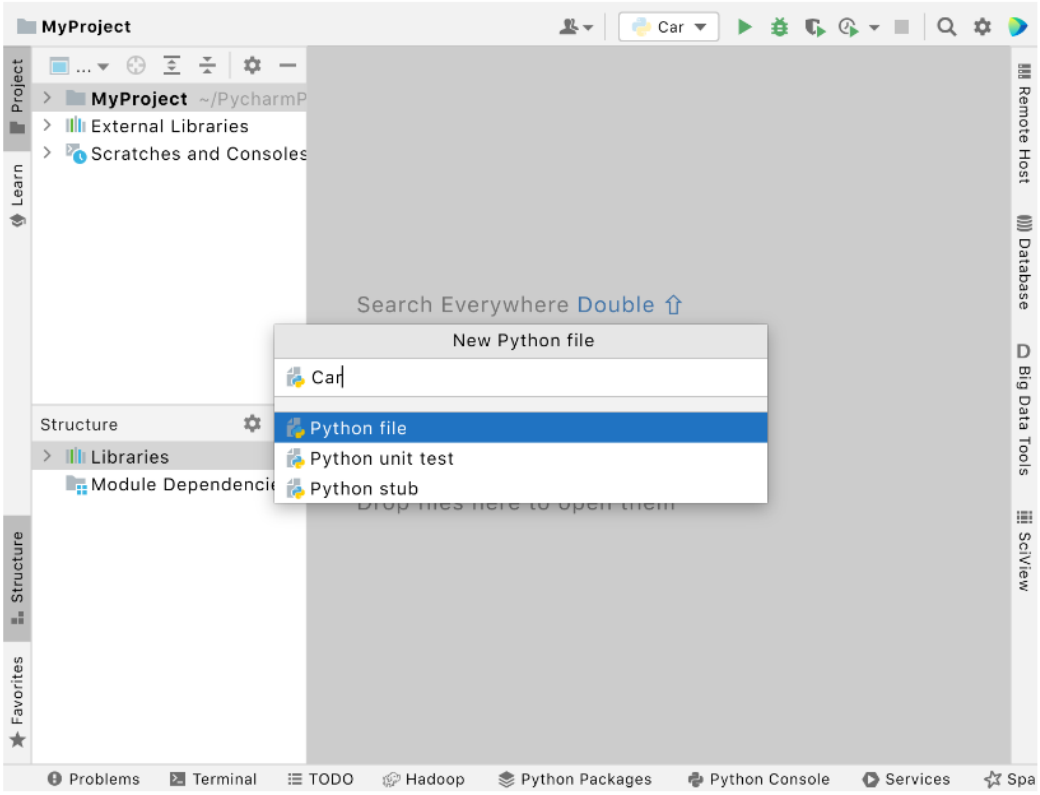


Fig.3.4. Create a Python File

3.5.4 Run, Debug and Test

Run

The easiest way to run an application is to right-click in the editor, and then choose Run <name> from the context menu:

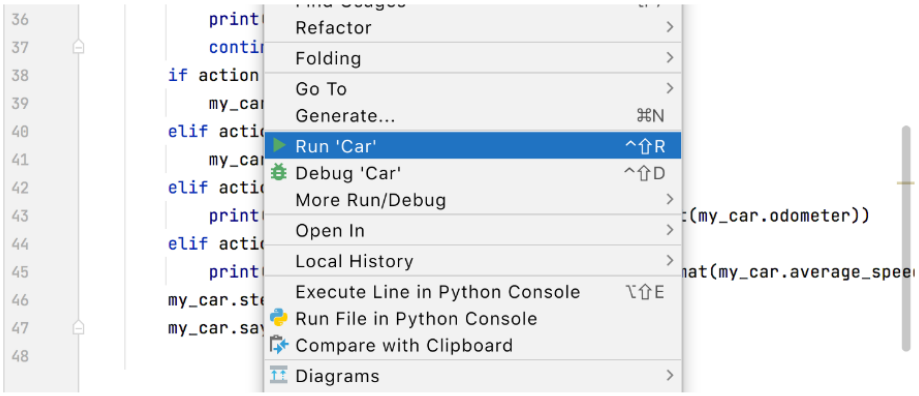


Fig.3.5. Run a Python File

Debug

Debugging starts with placing breakpoints at which program execution will be suspended, so you can explore program data. Just click the gutter of the line where you want the breakpoint to appear.

To start debugging your application, press Shift+F9. Then go through the program execution step by step (see the available options in the Run menu or the Debug tool window), evaluate any arbitrary expression, add watches, and manually set values for the variables.

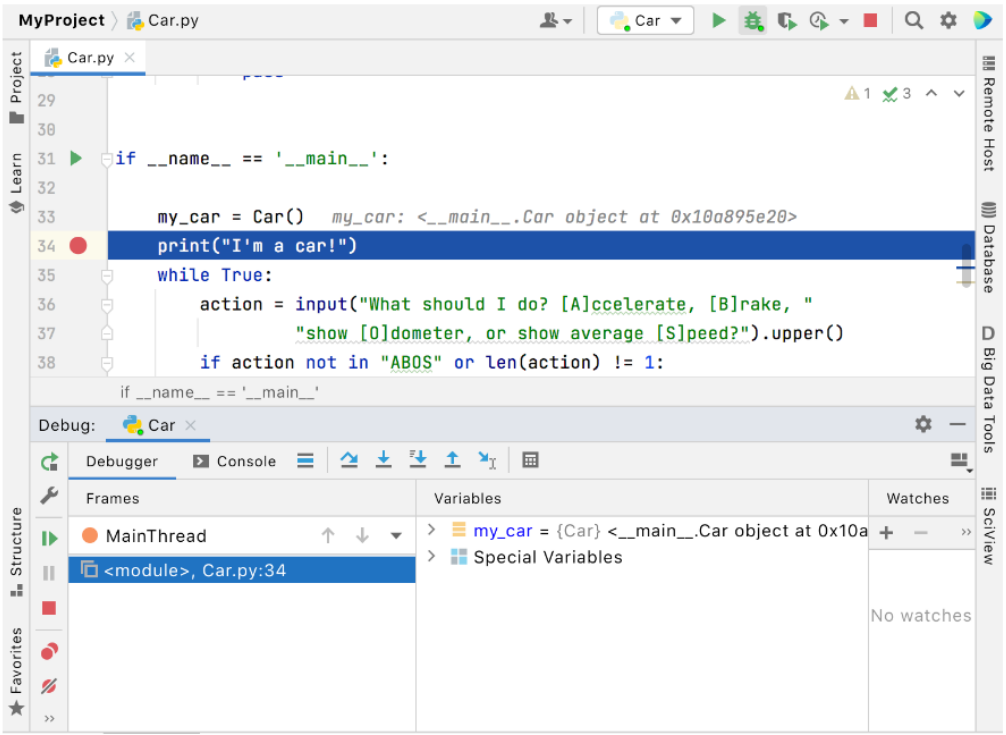


Fig.3.6. Debug a Python File

Test

It is a good idea to test your applications, and PyCharm helps doing it as simple as possible.

With PyCharm, you can:

- Create tests
- Create special testing run/debug configurations.

- Run and debug tests right from the IDE, using the testing run/debug configurations.
- And, finally, the most important thing - you can explore test results in the test runner tab of the Run tool window:

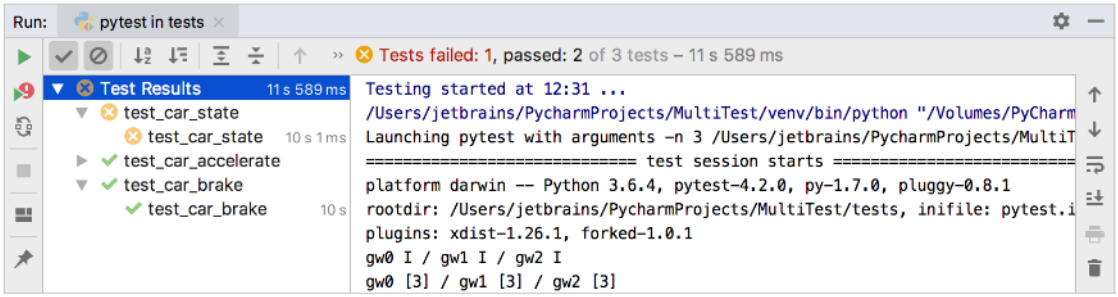


Fig.3.7. Test a Python File

CHAPTER - 4

DESIGN

4.1 UML Introduction

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from a distinctly different perspective.

UML is specifically constructed through two different domains, they are:

- UML Analysis modeling, this focuses on the user model and structural model views of the systems.
- UML Design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

4.1.1 Usage of UML in Project

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time to the market. These techniques include component technology, visual programming, patterns and frameworks. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The UML was designed to respond to these needs. Simply, systems design refers to the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

4.2 Activity Diagram

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

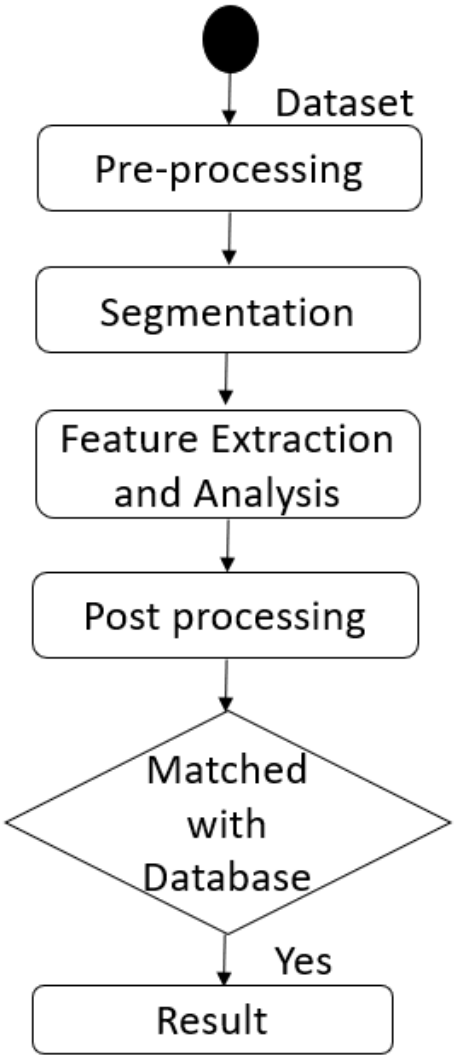


Fig.4.1. Activity Diagram for Crime Detection

4.3 Architecture of Project

An architecture is a way of representing the flow of data of a process or a system (usually an information system). This also provides information about the outputs and inputs of each entity and the process itself. Machine learning architecture defines the various layers involved in the machine learning cycle and involves the major steps being carried out in the transformation of raw data into training data sets capable for enabling the decision making of a system.

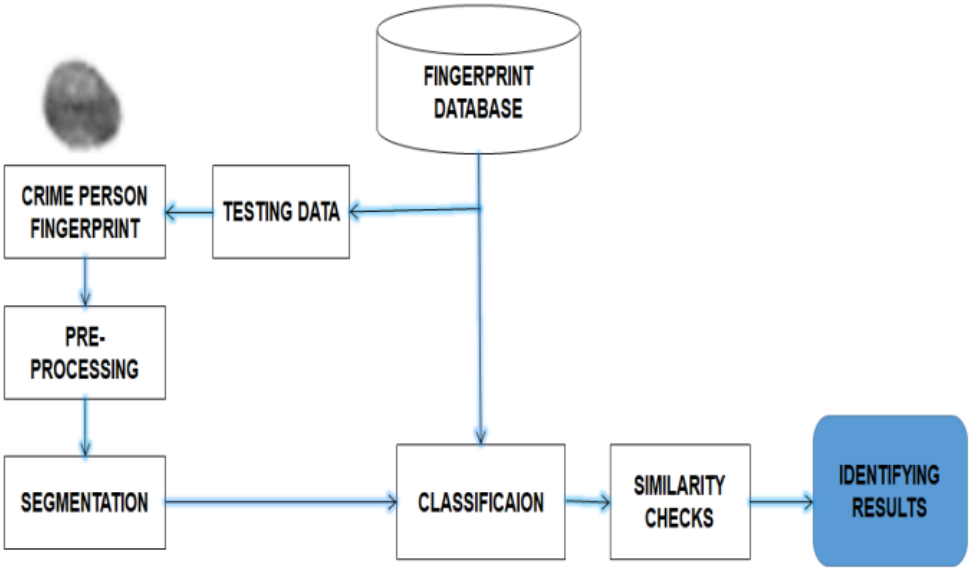


Fig.4.2. Architecture of Crime Detection

The above architecture describes how crime detection can be done. Initially obtain the datasets from kaggle website. After obtaining the datasets, perform data transformation to it in such a way that there shouldn't be any integration problem or any redundancy issue.

Now, apply feature selection techniques to the crime detection dataset which has many features in it. Then a subset of features which are most important in the prediction of future crime. Choose the algorithm that gives the best possible accuracy with the subset of features obtained after feature selection. Applying the algorithms to the dataset actually means that it needs to train the model with the algorithms and test the data so that the model will be fit.

4.4 Steps involved in Design

Data Collection

Data Preprocessing

Model Training

Model Evaluation

4.4.1 Data Collection

Data is an important asset for developing any kind of Machine learning model. Data collection is the process of gathering and measuring information from different kinds of sources.

- ➢ This is an initial step that has to be performed to carry out an Machine learning project. In the present internet world these datasets are available in different websites(Ex: Kaggle, Google public datasets, Data.gov etc)
- ➢ The dataset used in our project is downloaded from the Kaggle website and it contains nearly 12000 records and many different attribute.
- ➢ The dataset consists of some independent attributes and one dependent attribute.
- ➢ So, the aim of the project is to detect the crime based on fingerprints[6].

4.4.2 Data Preprocessing

- ➢ Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.
- ➢ When creating a machine learning project, it is not always the case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So for this, we use data preprocessing tasks.
- ➢ Preprocessing of the data consists of different kinds of steps in which analysis of the data , Data cleaning , Data encoding are part of this[7].

4.4.2.1 Explanatory Data Analysis

- ➢ Exploratory data analysis is an approach of analyzing datasets to summarize their main characteristics, often using statistical graphics and other data visualization methods.
- ➢ The main purpose of EDA is to help look at data before making any assumptions.
- ➢ It can help identify obvious errors, as well as better understanding patterns within the data, detect outliers or anomalous events, and find interesting relations among the variables.
- ➢ Specific statistical functions and techniques you can perform with EDA tools include:

- ➔ Clustering and dimension reduction techniques, which help create graphical display of high dimensional data containing many variables.
- ➔ Univariate visualization of each field in the raw dataset, with summary statistics.
- ➔ Bivariate visualizations and summary statistics that allows you to assess the relationship between each variable in the dataset and the target variable in the dataset and the target variable you're looking at.
- ➔ Multivariate visualizations, for mapping and understanding interactions between different fields in the data.
- ➔ Predictive models, such as linear regression, usw statistics and data to predict outcomes.

➢ This data analysis is of two types:

a . Univariate analysis

b. Bivariate analysis

Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships.

Bivariate data is data that involves two different variables whose values can change. Bivariate data deals with relationships between these two variables.

4.4.2.2 Filling Missing Data & Data Encoding

- ➢ The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.
- ➢ By calculating the mean and Mode: In this way, we will calculate the mean or Mode of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.
- ➢ Data encoding: Since the machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable,

then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

- ➢ Our dataset also consists of different categorical data in which they are encoded in this step[8].

4.4.3 Training The Model

In this step the model is trained using the algorithms that are suitable. Crime detection is a kind of problem in which One variable has to be determined using some independent variables. Regression model is suitable for this kind of scenario.

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output.

Our project implements two algorithms. Harris Corner and Open CV where Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. It was first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector. OpenCV is a great tool for image processing and performing computer vision tasks[9].

4.4.3.1 Convolution Layer

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

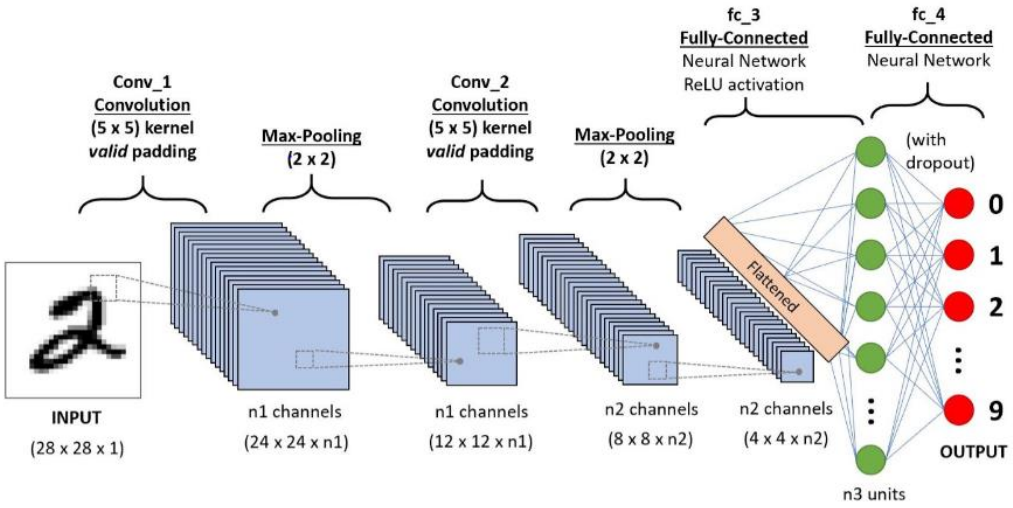


Fig.4.3. Convolution Layer

4.4.3.2 Pooling Layer

Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "downscaling" of the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. Spatial pooling is also called downsampling or subsampling, which reduces the dimensionality of each map but retains the important information.

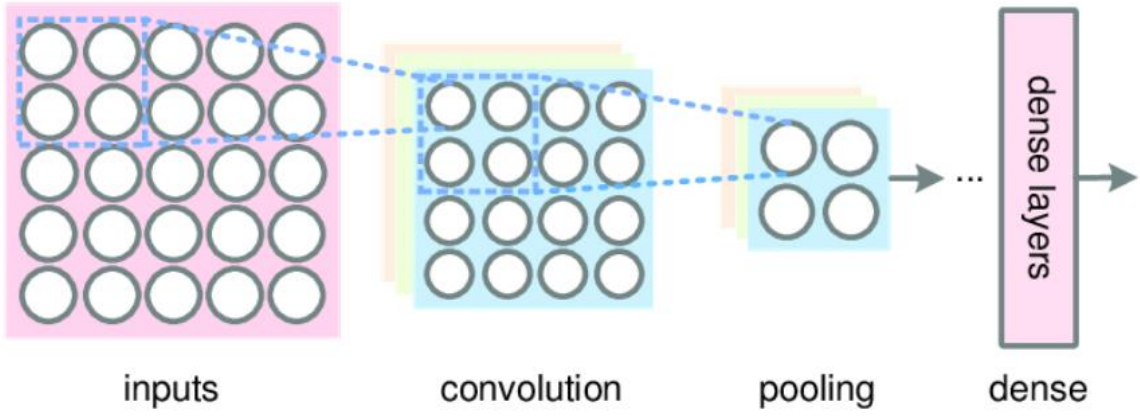


Fig.4.4. Pooling Layer

Max pooling is a **sample-based discretization process**. Its main objective is to downscale an input representation, reducing its dimensionality and allowing for the assumption to be made about features contained in the sub-region binned.

4.4.3.3 Fully Connected Layer

The fully connected layer is as follows.

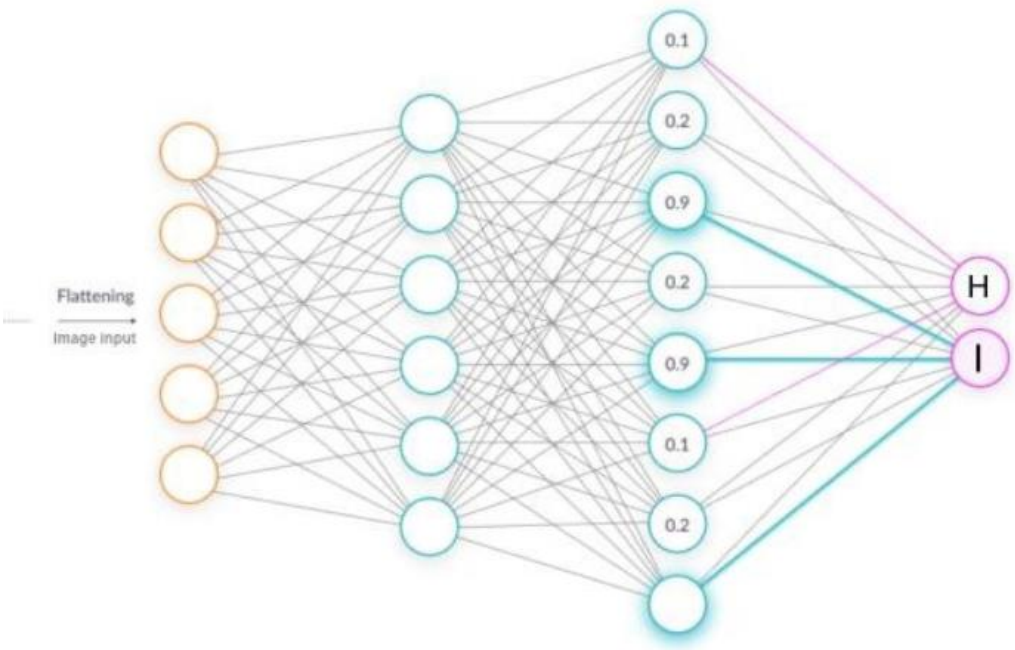


Fig.4.5 Fully Connected Layer

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network[10].

4.4.4 Model Evaluation

In this step the trained model is evaluated by determining the accuracy of the model against the test data.

Various ways to check the performance of our machine learning or deep learning model and why to use one in place of the other. We will discuss terms like:

1. Accuracy
2. Precision
3. Recall
4. Specificity
5. F1 score

6. Precision-Recall or PR curve
7. ROC (Receiver Operating **C**haracteristics) curve
8. PR vs ROC curve.

Out of these we used Accuracy for evaluating our model. Accuracy is the most commonly used metric to judge a model and is actually not a clear indicator of the performance. The worst happens when classes are imbalanced.

CHAPTER - 5

IMPLEMENTATION

Implementation part is made using CSV file containing different attributes with nearly 12000 records. Crime is detected using the data collected with Machine Learning algorithms like Harris Corner, OpenCV. All these algorithms helps to detect the crime. Crime is detected by implementing all the algorithms separately and are compared one with another.

5.1 Libraries Used

Python is increasingly being used as a scientific language. Matrix and vector manipulation are extremely important for scientific computations. Both NumPy and SKimage have emerged to be essential libraries for any scientific computation, including machine learning, in python due to their intuitive syntax and high performance matrix computation capabilities.

Pip

The pip command is a tool for installing and managing Python packages, such as those found in the Python Package Index. It's a replacement for easy installation. The easiest way to install the nfl* python modules and keep them up-to-date is with a Python-based package manager called pip.

pip install (module name)

NumPy

NumPy stands for ‘Numerical Python’ or ‘Numeric Python’. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since arrays and matrices are an essential part of the Machine Learning ecosystem, NumPy along with Machine Learning modules like Scikit-learn, Pandas, Matplotlib, TensorFlow, etc. complete the Python Machine Learning Ecosystem. NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation. NumPy can be imported into the notebook using

import numpy as np.

Pandas

Similar to NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Pandas provides an in-memory 2d table object called Data frame. It is like a spreadsheet with column names and row labels. Hence, with 2d tables, pandas are capable of providing many additional functionalities like creating pivot tables, computing columns based on other columns and plotting graphs. Pandas can be imported into Python using:

import pandas as pd.

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib comes with a wide variety of plots[11]. Plots help to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Matplotlib can be imported into Python using:

import matplotlib.pyplot as plt

SKimage

scikit-image is an image processing Python package that works with NumPy arrays which is a collection of algorithms for image processing. Simple and efficient tools for image processing and computer vision techniques. Accesible to everybody and reusable in various contexts. Built on the top of Numpy, Scipy, and matplotlib. Open source, commercially usable- BSD license.

from skimage.io import imread, imshow

Sklearn

Scikit-learn is a free software machine library for Python programming language.It features various classification , regression and clustering algorithms including Linear regression,Polynomial Regression and XGBoost Regression. In our project we have used different features of sklearn library like:

from sklearn.preprocessing import LabelEncoder

In machine learning, we usually deal with datasets which contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labeled in words.

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important preprocessing step for the structured dataset in supervised learning.

Label encoding converts the data in machine readable form, but it assigns a unique number(starting from 0) to each class of data. This may lead to the generation of priority issues in training of data sets. A label with high value may be considered to have high priority than a label having lower value.

from sklearn.preprocessing import PolynomialFeatures

Polynomial features of sklearn is mainly useful for the implementation of Polynomial regression algorithm of different kind of degrees like 1,2,3,4…....This feature of sklearn help to fit the dataset with Polynomial regression algorithm.So that it helps to Predict the sales.

from sklearn.model_selection import train_test_split

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations

where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

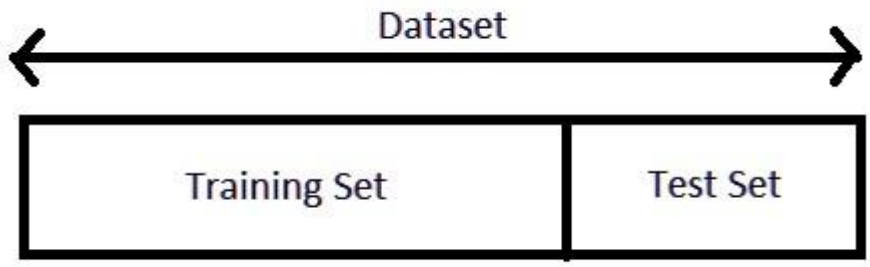


Fig.5.1. Training and Test Dataset

CSV file

The dataset used in this project is a .CSV file.

In computing, a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. CSV is a simple file format used to store tabular data, such as a spreadsheet or database. Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOffice Calc. Its data fields are most often separated, or delimited, by a comma.

A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a .csv extension. CSV files can be used with most any spreadsheet program, such as Microsoft Excel or Google Spreadsheets.

The difference between CSV and XLS file formats is that CSV format is a plain text format in which values are separated by commas (Comma Separated

Values), while XLS file format is an Excel Sheets binary file format which holds information about all the worksheets in a file, including both content and formatting.

5.2 Implementation

5.2.1 Importing all the required Modules and Libraries

All the required libraries like Sklearn and Modules like Numpy , Pandas, Matplotlib , Seaborn are imported into the Jupyter notebook initially into the file created in the notebook.

After importing all the modules and libraries into the notebook , A csv file has to be loaded using Pandas into the notebook. The implementation of these will be as follows:

```
import cv2
import os
import sys
import numpy
import matplotlib.pyplot as plt
from enhance import image_enhance
from skimage.morphology import skeletonize, thin
```

Fig.5.2 Importing of modules and libraries

5.2.2 Data Visualization

As it contains large amounts of data it is not possible to analyse with the human eye normally so the feature of Data Visualization helps to analyse the entire data. The relation between any two features can be only analysed with the Data Visualization technique.

This Visualization can be made in different forms by representing the data in the pictorial forms like graph, bar chart and many other forms.

Some Visualizations are made for the dataset that is collected for the detection of crime.

5.2.2.1 Thinning

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, somewhat like erosion or opening. It can be used for several applications, but is particularly useful for skeletonization. In this mode it is commonly used to tidy up the output of edge detectors by reducing all lines

to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output.

The thinning operation is related to the hit-and-miss transform, and so it is helpful to have an understanding of that operator before reading on.

```
In [ ]: skeleton = skeletonize(img)
        skeleton = numpy.array(skeleton, dtype=numpy.uint8)
        skeleton = removedot(skeleton)
```

Fig.5.3. Thinning

One of the most common uses of thinning is to reduce the thresholded output of an edge detector such as the Sobel operator, to lines of a single pixel thickness, while preserving the full length of those lines (*i.e.* pixels at the extreme ends of lines should not be affected). A simple algorithm for doing this is the following:

Consider all pixels on the boundaries of foreground regions (*i.e.* foreground points that have at least one background neighbor). Delete any such point that has more than one foreground neighbor, as long as doing so does not *locally disconnect* (*i.e.* split into two) the region containing that pixel. Iterate until convergence.

This procedure erodes away the boundaries of foreground objects as much as possible, but does not affect pixels at the ends of lines.

This effect can be achieved using morphological thinning by iterating until convergence with the structuring elements shown in Figure 1, and all their *90°* rotations (*4×2 = 8* structuring elements in total).

In fact what we are doing here is determining the *octagonal skeleton* of a binary shape --- the set of points that lie at the centers of octagons that fit entirely inside the shape, and which touch the boundary of the shape at at least two points. See the section on skeletonization for more details on skeletons and on other ways of computing it. Note that this skeletonization method is guaranteed to produce a connected skeleton.

5.2.2.2 Harris Corner

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness.[3] Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation and illumination. Although corners are only a small percentage of the image, they contain the most important features in restoring image information, and they can be used to minimize the amount of processed data for motion tracking, image stitching, building 2D mosaics, stereo vision, image representation and other related computer vision areas.

In order to capture the corners from the image, researchers have proposed many different corner detectors including the Kanade-Lucas-Tomasi (KLT) operator and the Harris operator which are most simple, efficient and reliable for use in corner detection. These two popular methodologies are both closely associated with and based on the local structure matrix. Compared to the Kanade-Lucas-Tomasi corner detector, the Harris corner detector provides good repeatability under changing illumination and rotation, and therefore, it is more often used in stereo matching and image database retrieval. Although there still exists drawbacks and limitations, the Harris corner detector is still an important and fundamental technique for many computer vision applications.

```
In [4]: harris_corners = cv2.cornerHarris(img, 3, 3, 0.04)
        harris_normalized = cv2.normalize(harris_corners, 0, 255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32FC1)
        threshold_harris = 125
```

Fig.5.4. Harris Corner

Commonly, Harris corner detector algorithm can be divided into five steps.

1. Color to grayscale
2. Spatial derivative calculation
3. Structure tensor setup
4. Harris response calculation

5. Non-maximum suppression

Color to grayscale

If we use Harris corner detector in a color image, the first step is to convert it into a grayscale image, which will enhance the processing speed.

The value of the gray scale pixel can be computed as a weighted sums of the values R, B and G of the color image.

5.2.2.3 Extract KeyPoints

A key point is a region of an image which is particularly distinct and identifies a unique feature.

Key points are used to identify key regions of an object that are used as the base to later match and identify it in a new image. Image of the same object can be taken in varying conditions like the varying lighting conditions, angle, scale, and background. A good keypoint is one which is invariant to all these conditions.

```
In [6]: keypoints = []
        for x in range(0, harris_normalized.shape[0]):
        for y in range(0, harris_normalized.shape[1]):
        if harris_normalized[x][y] > threshold_harris:
        keypoints.append(cv2.KeyPoint(y, x, 1))
```

Fig.5.5. Extract KeyPoints

A keypoint is calculated by considering an area of certain pixel intensities around it. Keypoints are calculated using various different algorithms, ORB(Oriented FAST and Rotated BRIEF) technique uses the FAST algorithm to calculate the keypoints. FAST stands for ***Features from Accelerated Segments Test.*** FAST calculates keypoints by considering pixel brightness around a given area. Consider a pixel area in an image and lets test if a sample pixel p becomes a keypoint.

5.2.2.4 Define and Compute Descriptors

Local Feature representation of images are widely in use for recognition and matching in the context of recovering geometry, dealing with occlusion, tracking, etc. One would want these features to be both repeatable as well as invariant to various viewing conditions like lighting, viewpoint and object orientation. Depending on the

task in hand various invariances become important. For the purposes of stereo matching where there is a projective (approximated to affine if the objects are far away from the camera) a descriptor which is very discriminative is useful. Also these features should be affine invariant. On the other hand if we look at examples of images within the same class the same local feature can have much larger variance. For e.g. lip corners of all human faces). The feature descriptor should be able to accommodate the intraclass variation but also give good inter class discriminativeness. Instead of complete rotation invariance we would like the features to be robust to small transformations of the object.

```
In [7]: orb = cv2.ORB_create()
        des = orb.compute(img, keypoints)
        return (keypoints, des);
```

Fig.5.6. Define and Compute Descriptors

5.3 Feature Engineering

What is a feature and why do we need the engineering of it? Basically, all machine learning algorithms use some input data to create outputs. This input data comprise features, which are usually in the form of structured columns. Algorithms require features with some specific characteristics to work properly. Here, the need for feature engineering arises. I think feature engineering efforts mainly have two goals:

- Preparing the proper input dataset, compatible with the machine learning algorithm requirements.
- Improving the performance of machine learning models.

```
In [8]: def removedot(invertThin):
            temp0 = numpy.array(invertThin[:])
            temp0 = numpy.array(temp0)
            temp1 = temp0/255
            temp2 = numpy.array(temp1)
            temp3 = numpy.array(temp2)

            enhanced_img = numpy.array(temp0)
            filter0 = numpy.zeros((10,10))
            W,H = temp0.shape[:2]
            filtersize = 6
```

Fig.5.7. Feature Engineering

One of the features of the dataset in which maximum number of values in that row are Zeros so they has to be normalised and are set to some value. So, mean of the entire column of data_visibilty is calculated and the value is set to that mean value.

This make all the Zeros of the column to particular value and accuracy of the model can be made more efficient.

5.4 Evaluation of Model

5.4.1 Matching Descriptors

```
In [9]: bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = sorted(bf.match(des1, des2), key= lambda match:match.distance)
```

Fig.5.8. Matching Descriptors

Features or **key-points** of an image are corners which are unique in the image. Harris and FAST are two different corner detectors, we have discussed later. Corner detectors are invariant for translation, illumination and rotation. But it is variant for scaling.

5.4.2 Plot KeyPoints

Keypoint detection consists of locating key object parts. For example, the key parts of our faces include nose tips, eyebrows, eye corners, and so on. These parts help to represent the underlying object in a feature-rich manner. Keypoint detection has applications that include pose estimation, face detection, etc

```
In [10]: img4 = cv2.drawKeypoints(img1, kp1, outImage=None)
         img5 = cv2.drawKeypoints(img2, kp2, outImage=None)
         f, axarr = plt.subplots(1,2)
         axarr[0].imshow(img4)
         axarr[1].imshow(img5)
         plt.show()
```

Fig.5.9. Plot KeyPoints

5.4.3 Plot Matches

```
In [11]: img3 = cv2.drawMatches(img1, kp1, img2, kp2, matches, flags=2, outImg=None)
         plt.imshow(img3)
         plt.show()
```

Fig.5.10. Plot Matches

Pattern matching is the technique to find existence of a pattern within an image. To localize a given pattern ‘w’ in the image ‘f’, concept of mask is used. An image matrix of pattern ‘w’ is the mask. This mask is placed over all possible pixel locations in the image ‘f’ and contents of image mask and image ‘f’ are compared. As a result of this comparison, a factor matching score is computed. If this matching score is greater than a predefined threshold value, the pattern ‘w’ is said to be matched with some portion of image ‘f’. For the comparison of pattern ‘w’ and image ‘f’ various techniques have been proposed. the pattern ‘w’ is said to be matched with some portion of image ‘f’. For the comparison of pattern ‘w’ and image ‘f’ various techniques have been proposed.

5.5 Model Prediction

In this prediction the entire database is trained with three harris corner models in which each model provides different output values of different accuracies .All the models are compared and the conclusions are made.

5.5.1 Modelling with Harris Corner

Step1: Load the database.

Step2: Divide the database.

Step3: Assign the Harris corner algorithm.

Step4: Fit the model using algorithm.

Step5: Detect the images for the testing dataset using a trained model.

Step6: Check the accuracy of the model.

```
In [4]: harris_corners = cv2.cornerHarris(img, 3, 3, 0.04)
        harris_normalized = cv2.normalize(harris_corners, 0, 255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32FC1)
        threshold_harris = 125
```

Fig.5.11. Modelling with Harris Corner

Get Descriptors

Step1: Load the database.

Step2: Divide the database.

Step3: Assign the Harris corner algorithm to a database.

Step4: Fit the database using given algorithm.

Step5: Detect the crime for the testing dataset using a trained model.

Step6: Check the accuracy of the model.

```
In [12]: def main():
         image_name = sys.argv[1]
         img1 = cv2.imread(r"C:\Users\srico\PycharmProjects\FRCD\venv\FingerPrintBasedCrimeDetection/DataBase1/" + image_name, cv2.IMREAD_
         kp1, des1 = get_descriptors(img1)

         image_name = sys.argv[2]
         img2 = cv2.imread(r"C:\Users\srico\PycharmProjects\FRCD\venv\FingerPrintBasedCrimeDetection/DataBase1/" + image_name, cv2.IMREAD_
         kp2, des2 = get_descriptors(img2)
```

Fig.5.12. Get Descriptors

In computer vision, visual descriptors or image descriptors are descriptions of the visual features of the contents in images, videos, or algorithms or applications that produce such descriptions. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others. These descriptors have a good knowledge of the objects and events found in a video, image or audio and they allow the quick and efficient searches of the audio-visual content.

Visual descriptors are divided in two main groups:

- General information descriptors
- Specific domain information descriptors

These are useful in their own ways.

CHAPTER - 6

RESULT ANALYSIS

6.1 Prediction of Outputs

After training all the models the output values are loaded into the Excel files and can be viewed according to our requirements. The outputs of the algorithm will be as follows:

The below screenshot is while using the run command of our project:

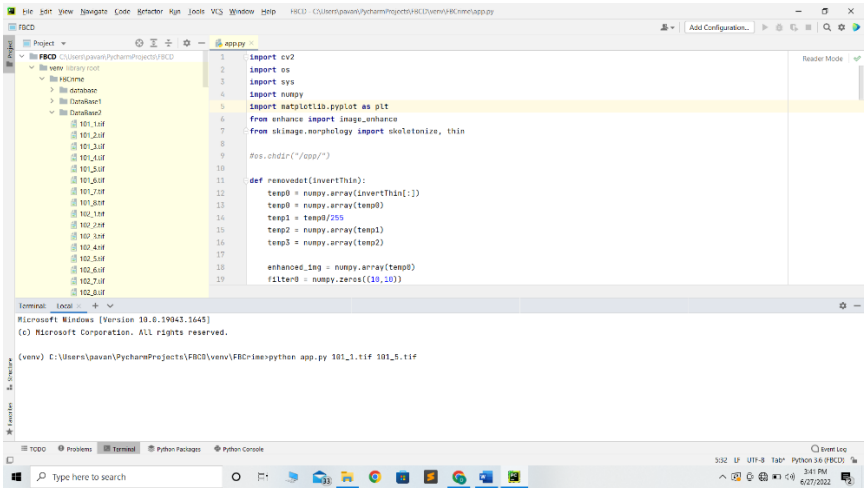


Fig.6.1 Running the code

The below screenshot is while plotting the keypoints of the fingerprint images:

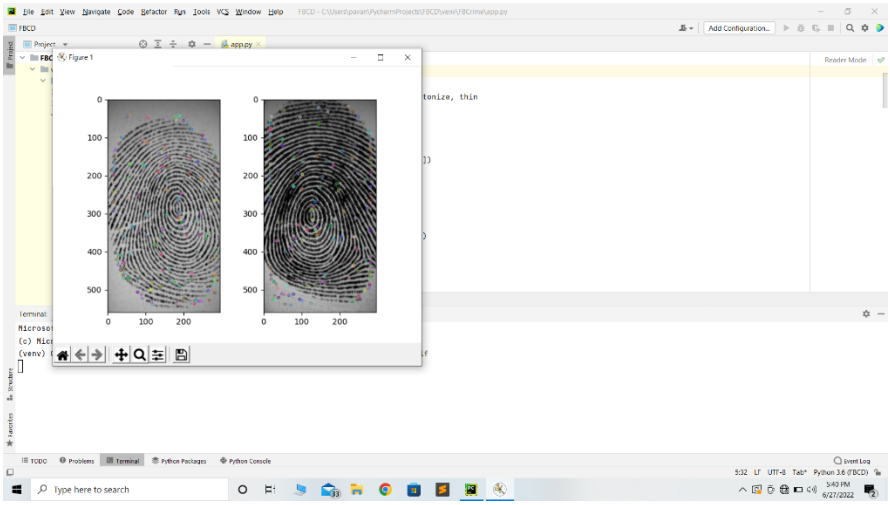


Fig.6.2 Fingerprint KeyPoints Plotting

The below screenshot is while matching the keypoints of the fingerprint images:

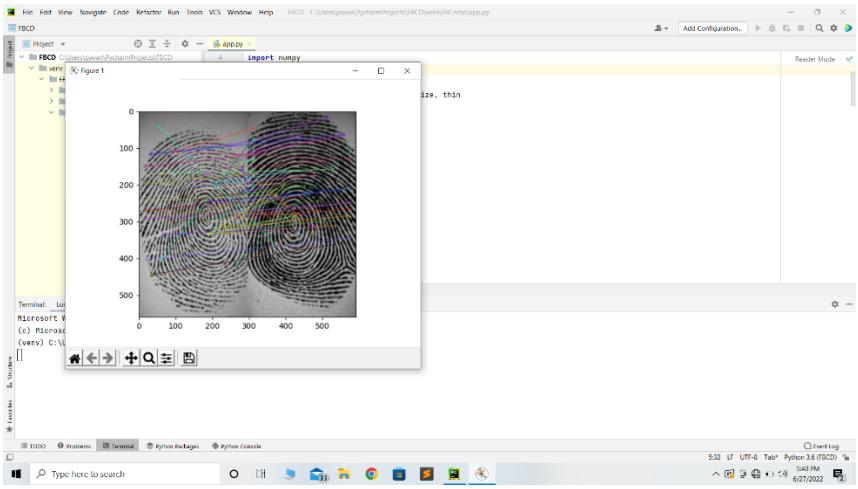


Fig.6.3 Fingerprint KeyPoints Matching

The below screenshot is the output of our project:

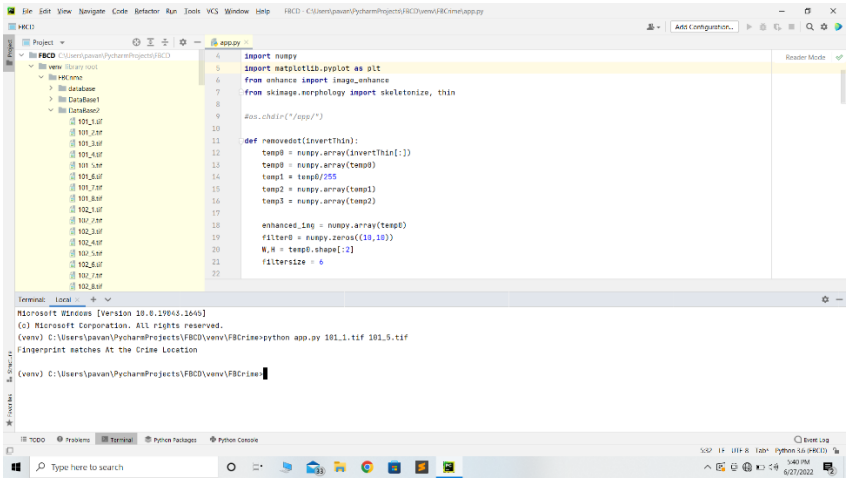


Fig.6.4 Output Screenshot

CHAPTER - 7

TESTING

7.1 INTRODUCTION

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

Testing is a process of executing a program with the intent of finding an error. A successful test is one that uncovers an as yet undiscovered error. A good test case is one that has a high probability of finding error, if it exists.

The first approach is what is known as Black box testing and the second approach is White box testing. We apply white box testing techniques to ascertain the functionalities top-down and then we use black box testing techniques to demonstrate that everything runs as expected.

7.2 Black-Box Testing

This technique of testing is done without any knowledge of the interior workings of the application. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining the outputs without knowing how and where the inputs are worked upon.

- Well suited and efficient for large code segments
- Code access is not required
- Clearly separates user's perspectives from the developer's perspective through visibly defined roles.

7.3 White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. It is also called "glass testing" or "open-box testing". In order to perform white box testing on an application, a tester needs to know the internal workings of the code.

The tester needs to look inside the source code and find out which part of the code is working inappropriately.

In this, the test cases are generated on the logic of each module. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent modules have been executed.
- Execute all logical decisions and loops.
- Execute through proper plots and curves.

7.4 Performance Evaluation

Corners are usually defined as interest points for which intensity variation in the principal directions is locally maximised, as response from a filter given by the linear combination of the determinant and the trace of the autocorrelation matrix. The Harris corner detector, in its original definition, is only rotationally invariant, but scale-invariant and affine-covariant extensions have been developed.

As one of the main drawbacks, corner detector performances are influenced by two user-given parameters: the linear combination coefficient and the response filter threshold. The main idea of the authors' approach is to search only the corners near enhanced edges and, by a z -score normalisation, to avoid the introduction of the linear combination coefficient.

```
In [15]: score = 0;
         for match in matches:
             score += match.distance
             score_threshold = 33
         if score/len(matches) < score_threshold:
             print("Fingerprint matches At the Crime Location")
         else:
             print("Fingerprint does not match At the Crime Location")
```

Fig.7.1. Performance Evaluation

Combining these strategies allows a fine and stable corner selection without tuning the method. The new detector has been compared with other state-of-the-art detectors on the standard Oxford data set, achieving good results showing the validity of the approach. Analogous results have been obtained using the local detector evaluation framework on non-planar scenes by Fraundorfer and Bischof.

CONCLUSION

Fingerprint identification system used to identify the given fingerprint as a crime person or non-crime person. Pre-processing was executed with Otsu thresholding, fingerprint thinning, and minutiae extraction is executed with the Cross-Number method. Harris corner detector takes the differential of the corner score into account with reference to direction directly, instead of using shifting patches for every 45-degree angles, and has been proved to be more accurate in distinguishing between edges and corners. The identification of fingerprints is a significant technique for various types of security authentication. Hence, we have advanced outlines to make the identification process simpler and accurate. The experimental output clearly says that, as compared to other methods, this method gives better performance. Matlab software is used for training and testing of the data to get accurate result.

Fingerprint identification system used for identifies the criminal who involved in the crime helps to automate fingerprint identification process. Pre-processing was performed with fingerprint thinning and minutiae extraction. The idea is to consider a small window around each pixel p in an image. We want to identify all such pixel windows that are unique. Uniqueness can be measured by shifting each window by a small amount in a given direction and measuring the amount of change that occurs in the pixel values.

REFERENCES

Journals

[1] O. I. Abiodun et al., "Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition," in IEEE Access, vol. 7, pp. 158820-158846, 2019.

[2] Anil K. Jain, and Jianjiang Feng, "Latent fingerprint matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, pp. 2452-2465.

[3] Mohammad Mogharen Askarin , KokSheik Wong and Raphael C-W Phan "Reduced contact lifting of latent fingerprint", In Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2017, pp. 1406-1410.

[4] Cao, Kai, Eryun Liu, and Anil K. Jain, "Segmentation and enhancement of latent fingerprints: A course to fine ridge structure dictionary," In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, pp. 1847-1859.

[5] Ramaiah, N. Pattabhi, A. Tirupathi Rao, and C. Krishna Mohan, "Enhancements to latent fingerprints in forensic applications," In 79th IEEE International Conference on Digital Signal Processing, 2014, pp. 439-443.

[6] Mouad.M.H.Ali, Vivek H.Mahale, Pravin Yannawar A Gaikwad, "Fingerprint Recognition For Person Identification and Verification Based On Minutiae Matching," In IEEE 6th International Conference on Advanced Computing, 2016, pp. 332-339.

[7] Wahid Zafar,Tasweer Ahamad and Muhammad Hassan, "Minutiae Based fingerprint Matching Techniques," In 17th IEEE International Multi Topic Conference, 2014, pp. 411-416.

Conference papers

[8] Pavithra. R and K.V. Suresh, “Fingerprint Image Identification for Crime Detection”, IEEE International Conference on Communication and Signal Processing, April 4-6, 2019, pp. 0797-0800.

[9] B. Wenxuan, F. Zhihong, P. Min and W. Pu, "Research on Indoor Edge Location Based on Location Fingerprint Recognition," International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Qiqihar, China, 2019, pp. 302-306.

[10] P.B.S Serafim, "A Method based on Convolutional Neural Networks for Fingerprint Segmentation," International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-8.

[11] K. Han, Z. Wang and Z. Chen, "Fingerprint Image Enhancement Method based on Adaptive Median Filter," 2018 24th Asia-Pacific Conference on Communications (APCC), Ningbo, China, 2018, pp. 40-44.