Project Documentation

1. INTRODUCTION

1.1 Project Overview

Project Title: Crime Identification using Deep Learning

Objective:
Develop a system that utilizes deep learning techniques to analyze images and video footage of crime scenes or incidents, enabling the identification and classification of different types of crimes. The goal is to provide valuable insights to law enforcement agencies for crime scene investigation, forensic analysis, and surveillance, ultimately aiding in the development of effective crime prevention strategies.
Components:

1. **Deep Learning Model:**
   - Choose or develop a deep learning model suitable for image and video analysis.
   - Train the model on a large dataset of crime-related images and videos to recognize patterns and features associated with different types of crimes.

2. **User Interface:**
   - Develop a user-friendly web interface for interaction.
   - Create pages for home and prediction using HTML, CSS, and Flask (a web framework for Python).
   - The home page should provide an introduction to the system, and the prediction page should allow users to upload images or videos for crime prediction.

3. **Integration with Flask:**
   - Use Flask to handle routing, rendering HTML templates, and integrating the deep learning model for predictions.
   - Implement routes for home and prediction pages.

4. **File Upload and Processing:**
   - Implement file upload functionality on the prediction page.
   - Process the uploaded images or videos and pass them to the deep learning model for classification.

5. **Prediction Display:**
   - Display the results of crime predictions on the web interface.
   - Provide relevant information, such as the predicted type of crime.

6. **Security Measures:**
   - Implement security measures to handle file uploads securely.
   - Validate and sanitize user inputs to prevent potential security vulnerabilities.

7. **Deployment:**
   - Deploy the application on a server to make it accessible to users.
   - Consider using platforms like Heroku, AWS, or others for deployment.

8. **Documentation:**
   - Document the project, including the deep learning model architecture, training process, and how to use the web interface.

Technologies:

- **Deep Learning Framework**: TensorFlow, PyTorch, or a framework suitable for your model.
- **Web Framework:** Flask for backend development.
- **Frontend:** HTML, CSS for the user interface.
- **Deployment:** Choose a platform based on your preference and requirements.

Future Enhancements:

- **Real-time Surveillance:** Extend the system to handle real-time surveillance for immediate crime identification.
- **User Authentication**: Implement user authentication for secure access to the system.
- **Data Analytics:** Incorporate data analytics to identify trends and patterns in crime data over time.

This project overview provides a high-level roadmap for implementing crime identification using deep learning. Depending on your specific goals and requirements, you may need to adapt and extend this outline.

1.2 Purpose

The purpose of the "Crime Identification using Deep Learning" project is to leverage advanced deep learning techniques to analyze images and video footage related to crime scenes or incidents. The primary objectives are:

1. **Enhanced Crime Classification:** Develop and deploy a deep learning model capable of accurately identifying and classifying various types of crimes based on visual cues present in images and videos.

2. **Law Enforcement Support:** Provide a valuable tool for law enforcement agencies, aiding them in crime scene investigation, forensic analysis, and surveillance. The system aims to assist in making informed decisions by automating the process of identifying potential criminal activities.

3. **Preventive Strategies:** Enable law enforcement to identify trends and patterns in crime data through the analysis of large datasets, particularly in surveillance footage. This information can be used to formulate effective crime prevention strategies and interventions.

4. **User-Friendly Interface:** Create a user-friendly web interface that allows users, including law enforcement personnel, to easily interact with the system. The interface should facilitate the upload of images and videos for crime prediction and provide clear and interpretable results.

5. **Educational Documentation:** Document the deep learning model's architecture, training process, and the overall functionality of the system. This documentation serves as a resource for understanding and utilizing the system effectively.

6. **Technological Integration:** Integrate cutting-edge technologies such as deep learning frameworks (e.g., TensorFlow, PyTorch) and web development tools (e.g., Flask) to create a seamless and efficient solution for crime identification.

7. **Scalability and Deployment:** Develop the project with scalability in mind, allowing for future enhancements and accommodating larger datasets. Deploy the system on a server to make it accessible, providing a practical tool for real-world applications.

8. **Security Measures:** Implement robust security measures to ensure the confidentiality

and integrity of data, particularly during the file upload process. Validate and sanitize user inputs to prevent potential security vulnerabilities.

9. Future Enhancements: Provide a foundation for future enhancements, including real-time surveillance capabilities, user authentication for secure access, and the incorporation of data analytics to identify long-term crime trends.

By achieving these objectives, the project aims to contribute to the advancement of crime detection and prevention methods, ultimately assisting law enforcement in creating safer communities.

2. LITERATURE SURVEY

2.1 Existing problem

Existing Problems in Crime Detection and Prevention:

1. Manual Processes and Delay:
- Traditional crime identification relies heavily on manual processes, causing delays and subjectivity.

2. Underutilization of Surveillance Data:
- Despite generating vast data, surveillance systems lack tools for efficient data analysis and trend identification.

3. Inaccuracy in Classification:
- Human errors in crime classification can lead to misinterpretations and misjudgments.

4. Resource-Intensive Forensic Analysis:
- Forensic analysis demands significant time and resources, impacting investigation timelines.

5. Lack of Predictive Analytics:
- Traditional methods focus on retrospective analysis, lacking predictive capabilities for proactive interventions.

6. Security and Privacy Concerns:
- Integration of technology raises concerns about data security and privacy, requiring

robust measures.

7. Technological Barriers in Law Enforcement:
- Adoption challenges due to budget constraints, lack of expertise, and resistance to technological change.

8. Non-User-Friendly Tools:
- Existing systems may lack intuitive interfaces, hindering broader adoption by non-technical users.

Addressing these problems through the proposed "Crime Identification using Deep Learning" project can revolutionize crime detection, making it more efficient, accurate, and accessible for law enforcement.

2.2 References

2.3 Problem Statement Definition

In contemporary crime detection and prevention efforts, there exists a critical need for more efficient, accurate, and technologically advanced methods. The current landscape faces several challenges that hinder the effectiveness of traditional crime identification processes. Manual approaches are often time-consuming, prone to errors, and may lack the capability to analyze the increasing volume of surveillance data generated. Additionally, the classification of crimes can be subjective, leading to potential misinterpretations and delays in investigative procedures.

Furthermore, the underutilization of technological advancements in deep learning poses a barrier to leveraging the full potential of available data for crime analysis. The absence of automated tools for predictive analytics and real-time surveillance hampers law enforcement agencies' ability to proactively address emerging crime trends.

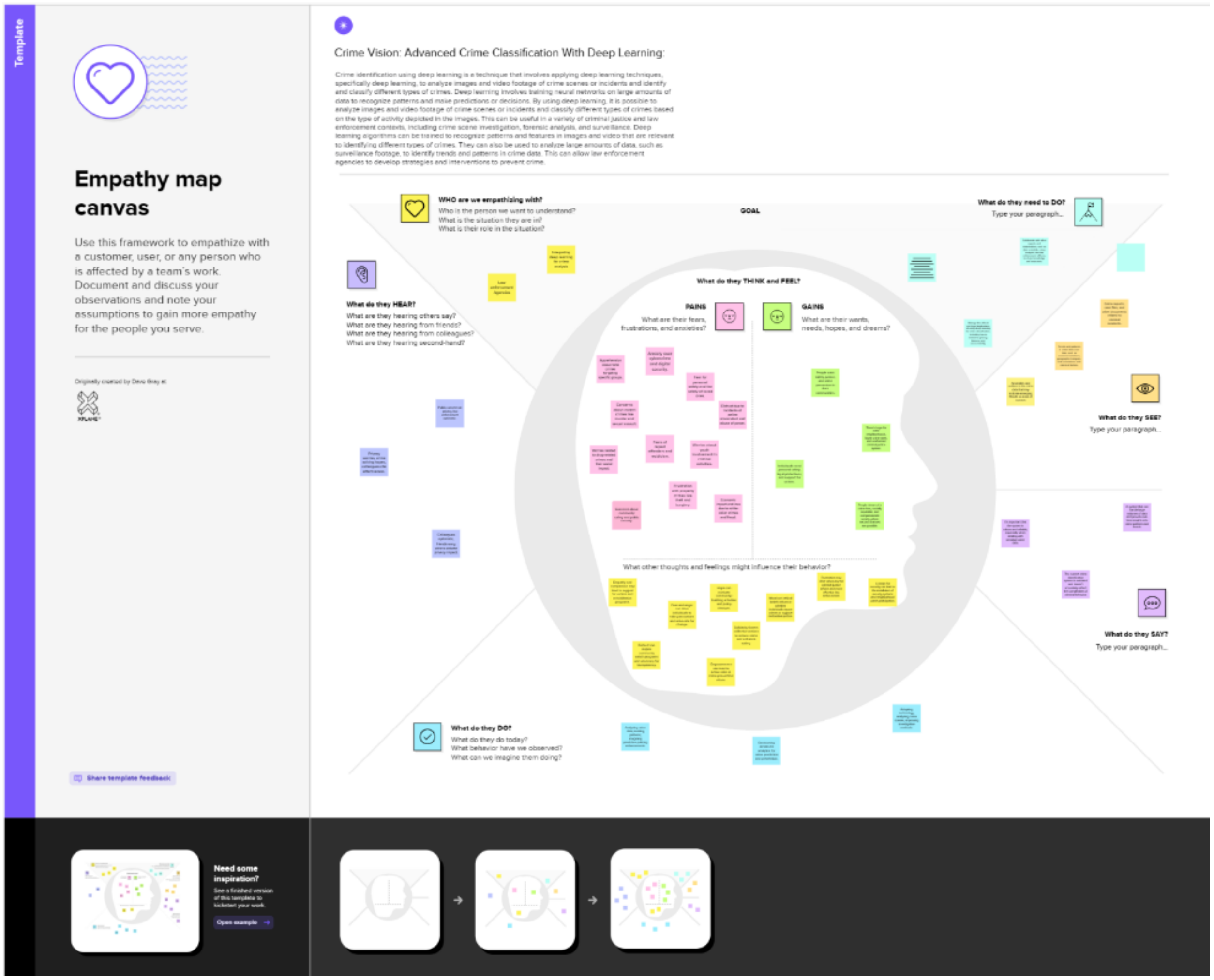
Security and privacy concerns associated with the integration of advanced technologies, especially in the context of handling sensitive image and video data, need to be adequately addressed. The adoption of these technologies may also be hindered by resource constraints, lack of expertise within law enforcement agencies, and resistance to technological change.

In light of these challenges, the proposed project, "Crime Identification using Deep Learning," aims to develop a solution that automates and enhances crime identification processes. By implementing deep learning techniques, the project seeks to overcome the limitations of traditional methods, providing a more accurate and timely means of classifying crimes from images and video footage. The user-friendly interface and integration with Flask aim to make this technology accessible to law enforcement personnel, facilitating a seamless transition to advanced crime detection methods. The project also places a strong emphasis on security measures to ensure the confidentiality and integrity of sensitive data.

Ultimately, the goal is to address the existing problems in crime detection, empower law enforcement with advanced technological tools, and contribute to the development of more effective crime prevention strategies.

3. IDEATION & PROPOSED SOLUTION

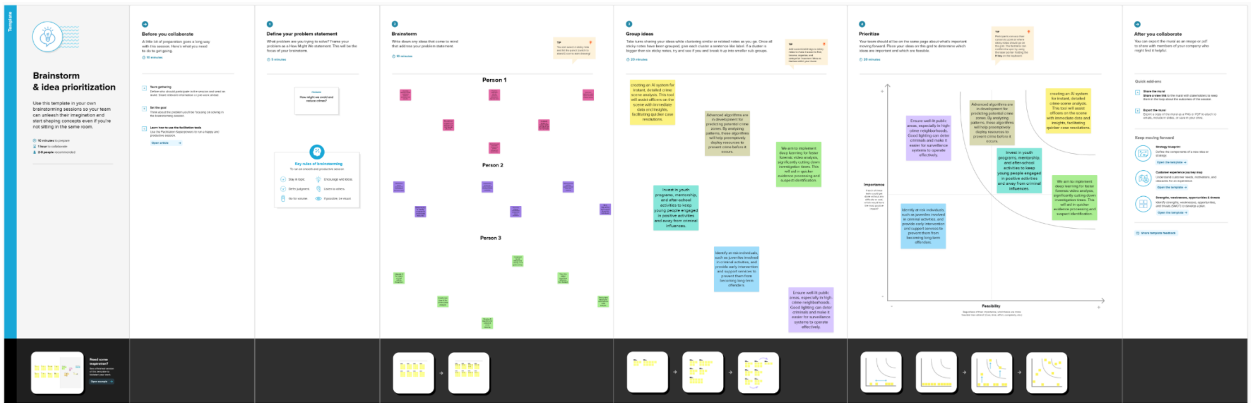
3.1 Empathy Map Canvas


Template
Empathy map canvas
Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.
Share template feedback
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example
WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
GOAL
What do they need to DO?
Type your paragraph...
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
What do they SEE?
Type your paragraph...
What other thoughts and feelings might influence their behavior?
What do they SAY?
Type your paragraph...
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?

Crime Vision: Advanced Crime Classification With Deep Learning:

Crime identification using deep learning is a technique that involves applying deep learning techniques, specifically deep learning, to analyze images and video footage of crime scenes or incidents and identify and classify different types of crimes. Deep learning involves training neural networks on large amounts of data to recognize patterns and make predictions or decisions. By using deep learning, it is possible to analyze images and video footage of crime scenes or incidents and classify different types of crimes based on the type of activity depicted in the images. This can be useful in a variety of criminal justice and law enforcement contexts, including crime scene investigation, forensic analysis, and surveillance. Deep learning algorithms can be trained to recognize patterns and features in images and video that are relevant to identifying different types of crimes. They can also be used to analyze large amounts of data, such as surveillance footage, to identify trends and patterns in crime data. This can allow law enforcement agencies to develop strategies and interventions to prevent crime.

3.2 Ideation & Brainstorming



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

1. User Authentication:

- Secure user authentication for law enforcement personnel.

2. File Upload and Processing:

- Allow users to upload image and video files for crime prediction.
- Process uploaded files using the deep learning model for crime classification.

3. Prediction Display:

- Display crime prediction results clearly.
- Show predicted crime types to users.

4. Real-Time Surveillance :

- Optionally support real-time surveillance for immediate crime identification.

5. Data Analytics :

- Optionally include functionality for analyzing crime data trends.

6. Security Measures:

- Implement data security measures.
- Validate and sanitize user inputs to prevent security issues.

7. User-Friendly Interface:

- Develop an intuitive web interface.

- Ensure easy navigation and user feedback.

8. Documentation:
- Provide comprehensive usage documentation.

9. Scalability:
- Design for scalability to accommodate future enhancements.

10. Integration with Flask:
- Use Flask for routing and integration of the deep learning model.

4.2 Non-Functional Requirements:

1. Performance:
- Ensure timely responses during file uploads and predictions.
- Handle a reasonable number of concurrent users effectively.

2. Security:
- Comply with industry security practices.
- Implement secure file upload mechanisms.

3. Usability:
- Create an intuitive user interface for law enforcement.
- Provide clear error messages for guidance.

4. Reliability:
- Maintain system reliability during critical times.
- Implement error handling to minimize downtimes.

5. Scalability:
- Design for horizontal scalability to handle user growth.

6. Maintainability:
- Design for ease of maintenance and updates.
- Ensure well-documented and modular code.

7. Compatibility:
- Ensure compatibility with common web browsers and devices.

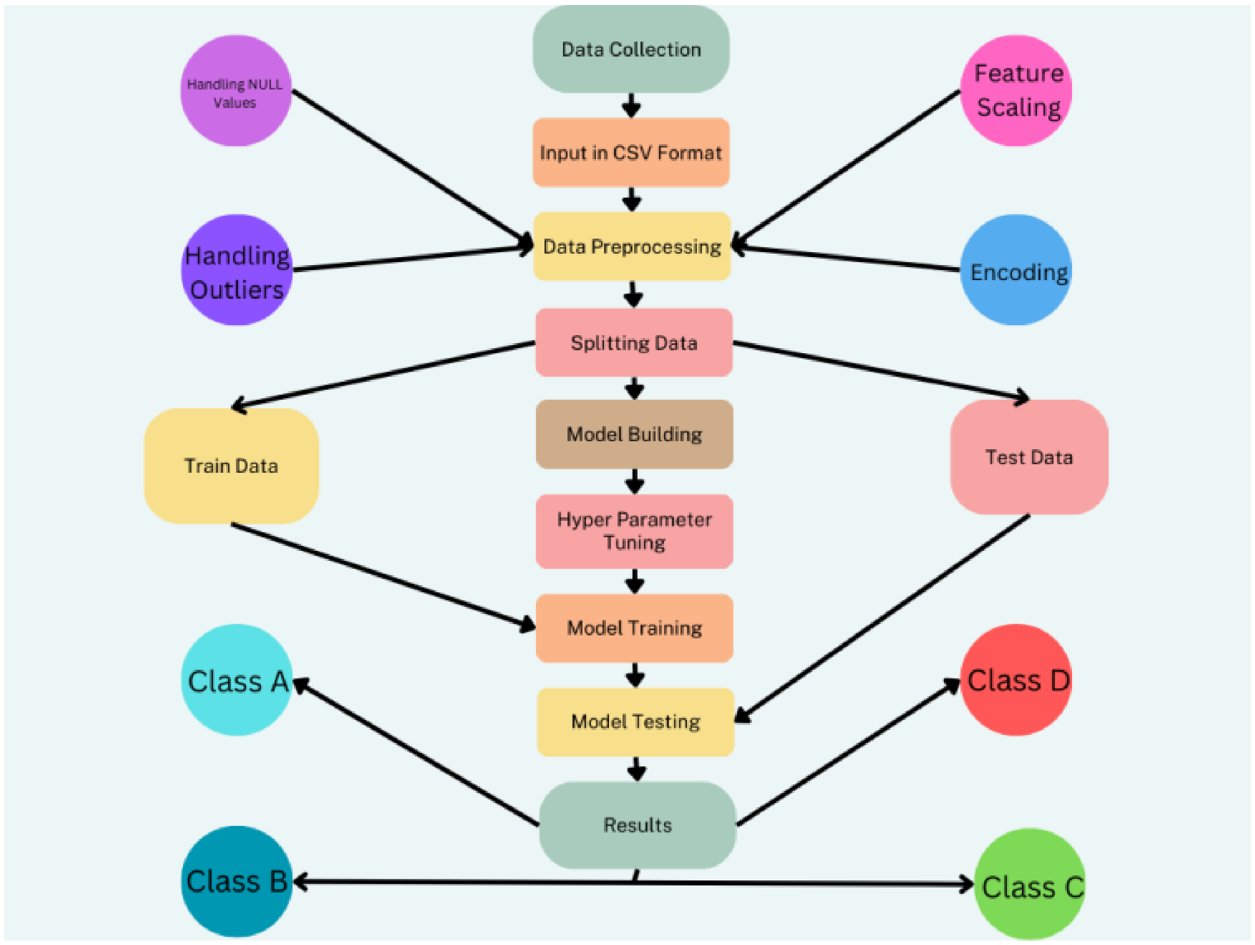
8. Ethical Considerations:
- Adhere to ethical data privacy and AI use in law enforcement.

9. Regulatory Compliance:
- Ensure compliance with relevant laws and regulations.

10. Deployment:
- Deploy on a secure server infrastructure, consider cloud-based solutions.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priorty	Release

Forensic Investigator	Utilize deep learning to identify crimes in crime scene media	US-01	Crime Scene Investigation	Accurately classify the crime type.	High	Sprint-1
Law Enforcement	Use deep learning to analyze surveillance data for crime trends	US-02	Surveillance Trend Analysis	Identify patterns and trends.	High	Sprint-1
Forensic Scientist	Apply deep learning for detailed analysis of forensic evidence	US-03	Evidence Analysis	Input forensic evidence data.	High	Sprint-1
Security Analyst	Implement real-time monitoring with deep learning for instant alerts	US-04	Real-time Crime Detection	Integrate model into real-time system.	Medium	Sprint-2

Law Enforcement Strategist	Leverage deep learning insights for targeted prevention strategies	US-05	Crime Prevention Strategies	Input historical crime data.	Medi um	Sprint-2

5.2 Solution Architecture

Building Blocks:

Data Collection: Gather crime scene images and videos.
Data Preprocessing: Normalize and compress data for consistency.
Deep Learning Model: Employ CNNs for training and inference.
Model Training: Train the model on labeled datasets.
Model Deployment: Implement the model for real-time processing.
Crime Identification: Apply the model to classify crimes.
Output Generation: Generate classifications and confidence scores.
Integration: Integrate results with law enforcement systems.
Feedback Loop: Collect feedback for continuous improvement.
Surveillance Analysis: Analyze data for patterns.
Strategy Development: Formulate crime prevention strategies.

Workflow of Data:

Data Collection: Collect crime scene data.
Data Preprocessing: Standardize and compress data.
Model Training: Train deep learning model.
Model Deployment: Deploy model for real-time processing.
Crime Identification: Identify and classify crimes.
Output Generation: Generate classifications.

Integration: Integrate results with law enforcement systems**.**
Feedback Loop: Gather feedback for model improvement.
Surveillance Analysis: Analyze data for patterns.
Strategy Development: Develop crime prevention strategies

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

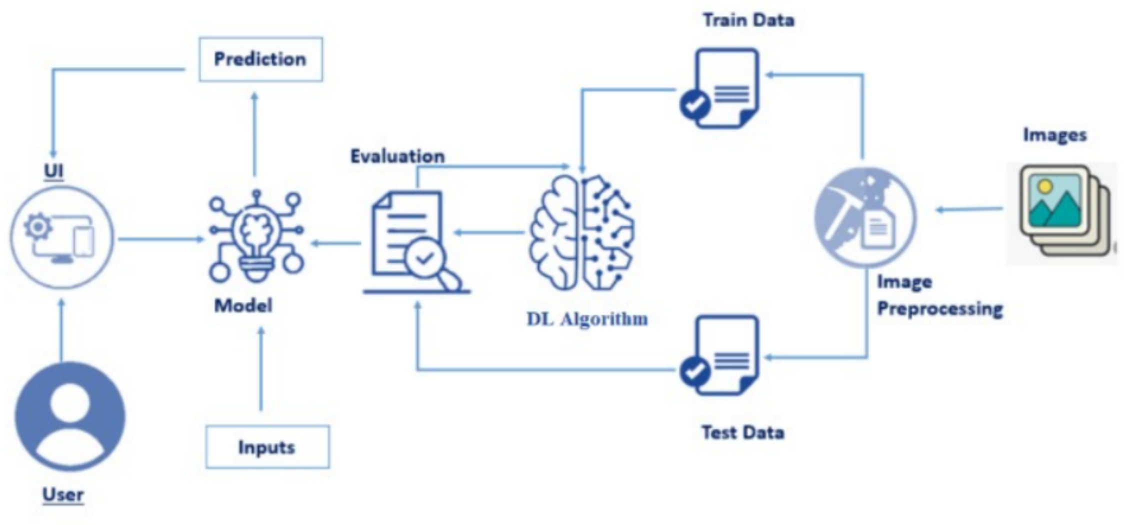


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	The logic of your application involves selecting a programming language compatible with your chosen deep learning framework.	Java / Python
3.	Application Logic-2	Consider the infrastructure needed for training your deep learning model.	IBM Watson STT service
4.	Application Logic-3	By incorporating IBM Watson Assistant into your project, you can enhance the user experience, improve accessibility, and streamline interactions with your crime detection system.	IBM Watson Assistant
5.	Database	Image Data:,Text Data, Neural Network Architecture,Integration with IBM Watson Assistant,Continuous Integration/Continuous Deployment (CI/CD)	MySQL, NoSQL, etc.
6.	Cloud Database	A cloud-based database service provides scalable and managed database solutions, allowing users to store, retrieve, and manage data efficiently over the internet	IBM DB2, IBM Cloudant etc.
7.	File Storage	Establish a clear organizational structure for your files and consider metadata requirements for efficient file retrieval and management	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Access to Third-Party Services,Social Media Integration,Machine Learning and AI Services	IBM Weather API, etc.
9.	External API-2	Data Aggregation,Data Retrieval,Customization and Extensibility	Aadhar API, etc.
10.	Machine Learning Model	Machine learning models are designed to generalize from existing data, allowing them to perform tasks without being explicitly programmed.	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Install server software relevant to your needs, such as web servers (e.g., Apache, Nginx), database servers (e.g., MySQL, PostgreSQL), and application servers. Cloud Server Configuration :Select a cloud service provider (e.g., AWS, Azure, Google Cloud) based on your project requirements, budget, and preferences.	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	TensorFlow is an open-source machine learning framework developed by the Google Brain team. It is widely used for building and	Keras is an open-source high-level neural networks API written in Python. It is often used as a

		training deep learning models, including those for image recognition and classification. Pandas is an open-source data manipulation and analysis library for Python. It is commonly used for cleaning, transforming, and analyzing datasets.	frontend for TensorFlow and other deep learning frameworks, providing a simple and intuitive interface for building and training models.
2.	Security Implementations	Firewalls act as a barrier between a trusted internal network and untrusted external networks (like the internet)..	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	The 3-tier architecture separates the application into three logical layers: presentation, application (business logic), and data. This separation allows each layer to scale independently based on specific requirements.Microservices architecture breaks down an application into small, independent services, each responsible for a specific business capability. This modularity allows for independent scaling of services based on their specific usage patterns and demands.	3-tier architecture

S.No	**Characteristics**	**Description**	**Technology**
4.	Availability	Ensuring high availability of an application is crucial to provide uninterrupted service to users. Various architectural patterns and technologies contribute to achieving high availability.	Content Delivery Networks (CDNs)CDNs distribute static content across servers strategically located around the world.
5.	Performance	Design the application architecture to be horizontally scalable, allowing it to handle an increasing number of requests.	Leverage Content Delivery Networks to reduce latency and accelerate content delivery to users.

6.2 Sprint Planning & Estimation

Sprint	Functional Requirement(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Image Recognition Model Development	US1	I want a secure campus environment.	5	High	SATYA PRANAY

Sprint 2	Crime Classification Algorithm	US2	I want automatic identification of security threats on campus.	8	Medium	Jaswanth Sai
Sprint 3	Surveillance Integration	US3	I want deep learning integrated with campus surveillance for prompt incident response.	6	High	Sai Sathwik

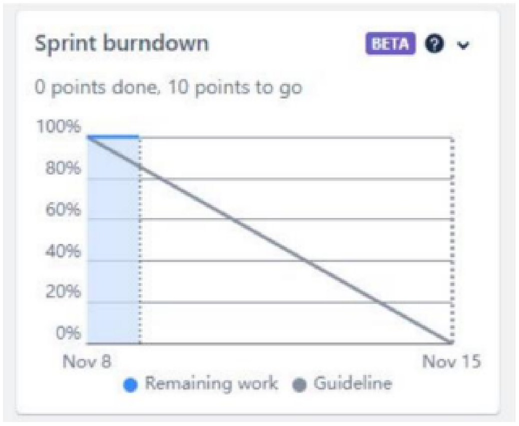
Project Tracker,Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)	Team Members
Sprint - 1	3	5 Days	8 Nov 2023	13 Nov 2023	3	13 Nov 2023	SATYA PRANAY
Sprint – 2	4	3 Days	14 Nov 2023	17 Nov 2023	4	17 Nov 2023	Jaswanth Sai
Sprint - 3	3	2 Days	18 Nov 2023	20 Nov 2023	3	20 Nov 2023	Sai Sathwik

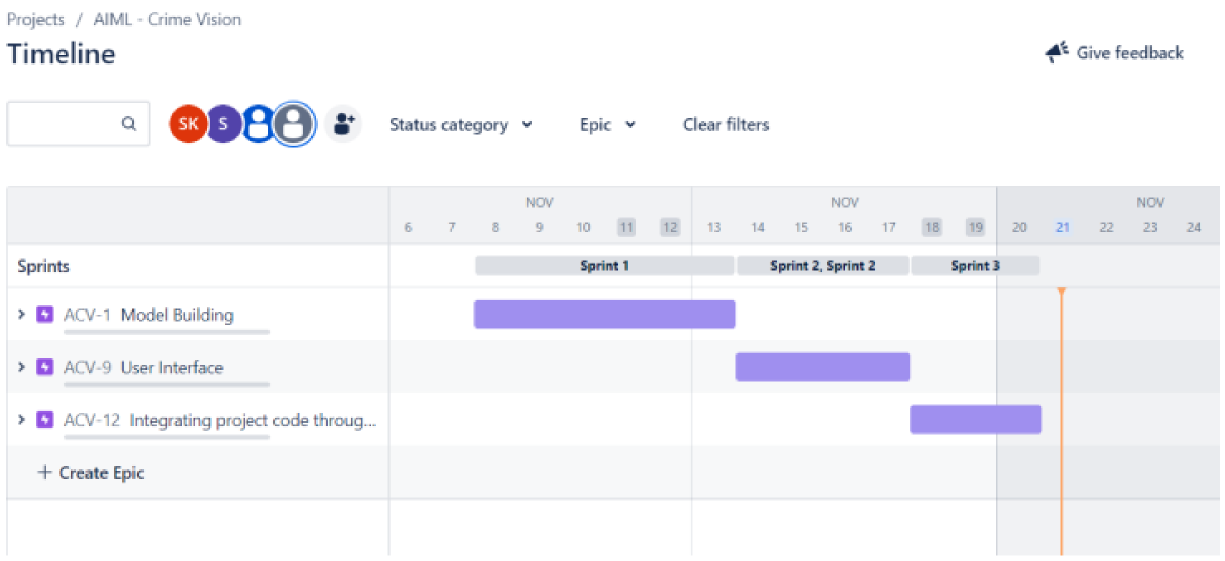
Velocity:

AV = Sprint Duration / Velocity = 10 / 10 = 1

Burndown Chart



Sprint Delivery Schedule:

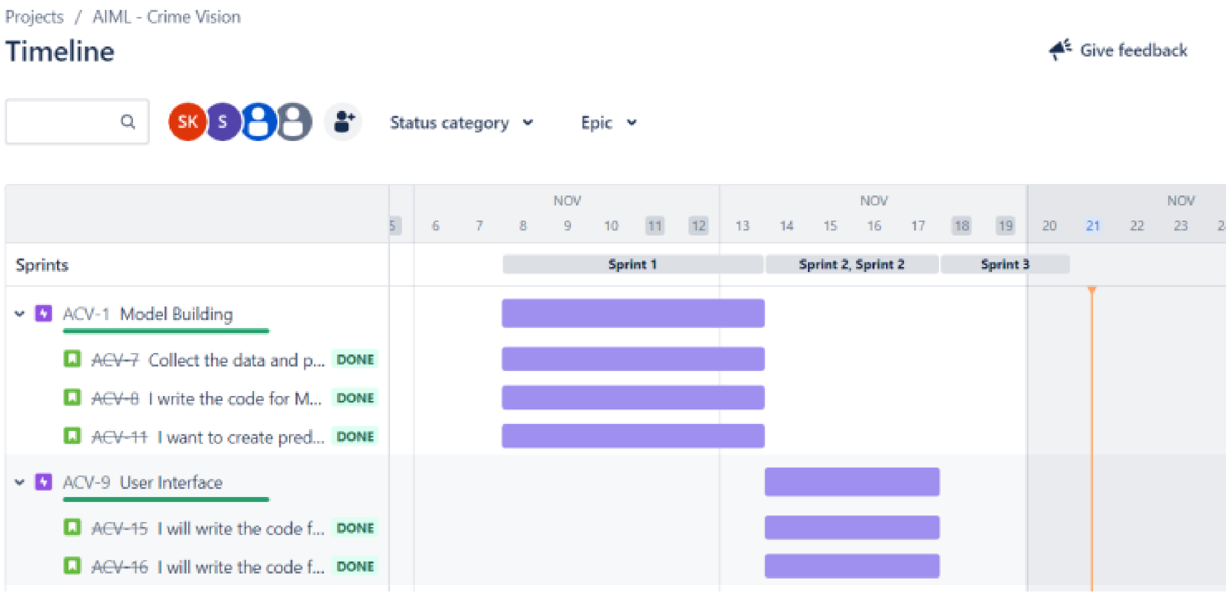

Projects / AIML - Crime Vision

Timeline

ACV-1 Model Building

ACV-9 User Interface

ACV-12 Integrating project code throug...



Projects / AIML - Crime Vision

Timeline

ACV-1 Model Building

~~ACV-7~~ Collect the data and p... DONE

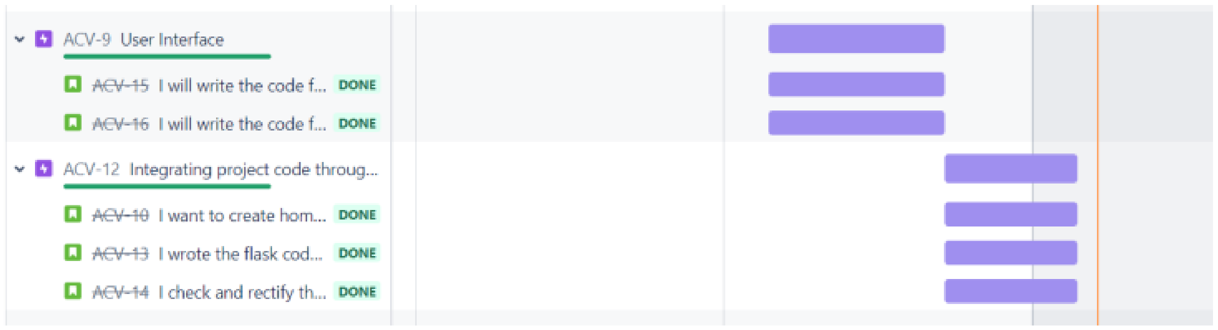
~~ACV-8~~ I write the code for M... DONE

~~ACV-11~~ I want to create pred... DONE

ACV-9 User Interface

~~ACV-15~~ I will write the code f... DONE

~~ACV-16~~ I will write the code f... DONE



ACV-9 User Interface

~~ACV-15~~ I will write the code f... DONE

~~ACV-16~~ I will write the code f... DONE

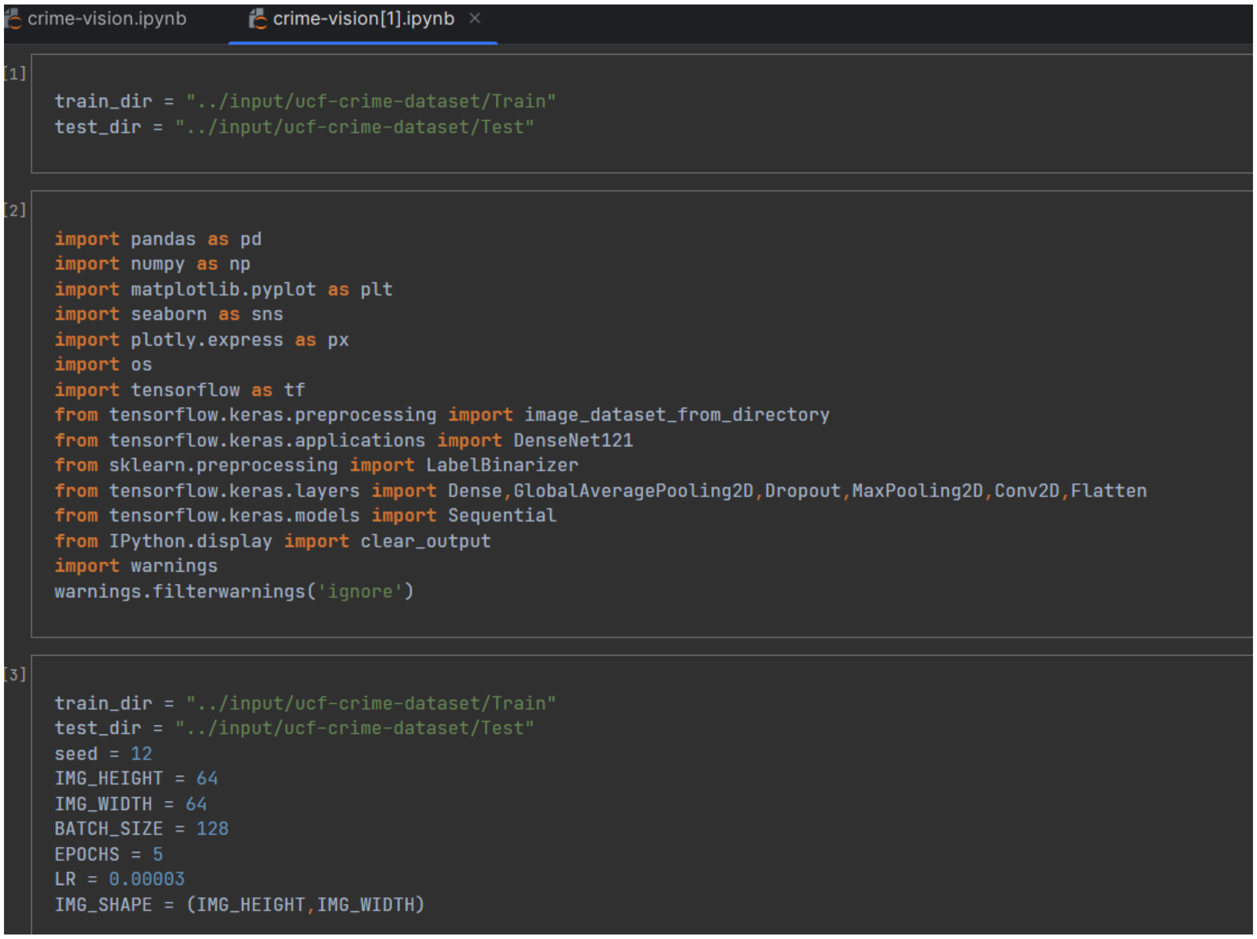
ACV-12 Integrating project code throug...

~~ACV-10~~ I want to create hom... DONE

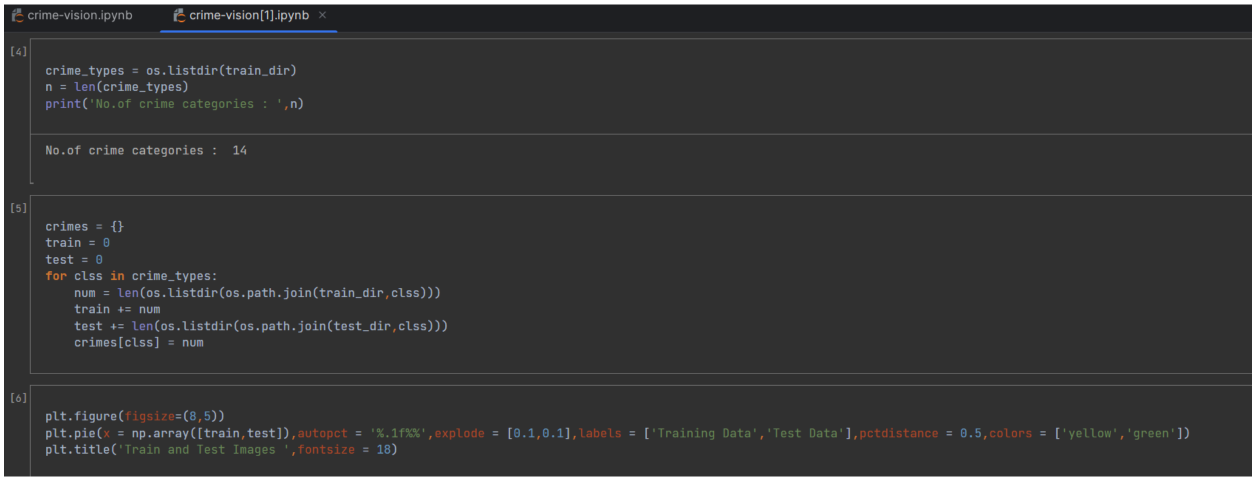
~~ACV-13~~ I wrote the flask cod... DONE

~~ACV-14~~ I check and rectify th... DONE

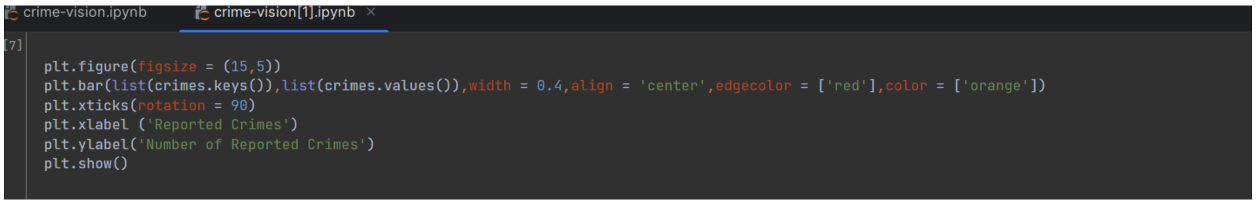
7. CODING & SOLUTIONING



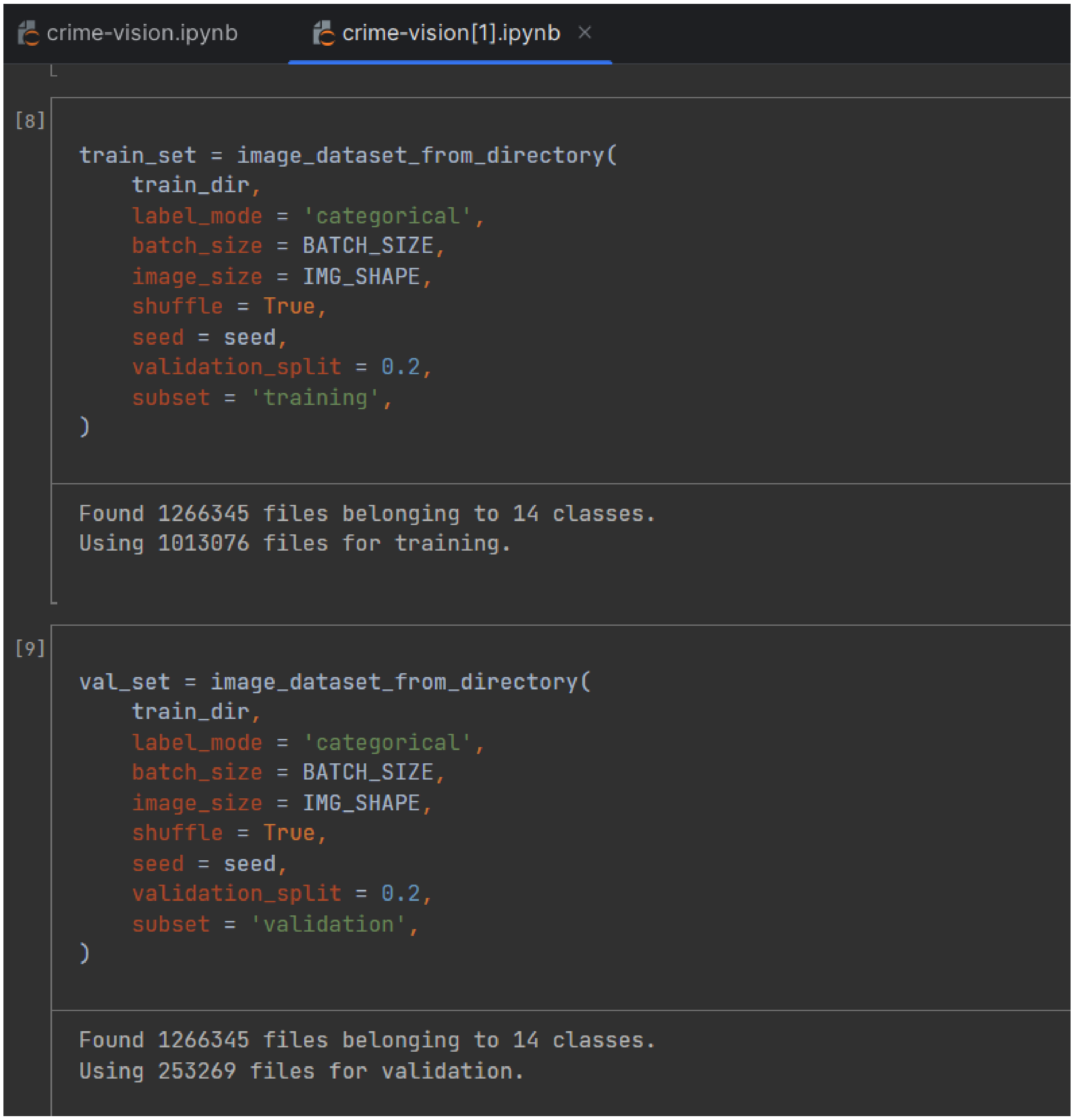
```
[1]
train_dir = "../input/ucf-crime-dataset/Train"
test_dir = "../input/ucf-crime-dataset/Test"

[2]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import os
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.applications import DenseNet121
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.layers import Dense,GlobalAveragePooling2D,Dropout,MaxPooling2D,Conv2D,Flatten
from tensorflow.keras.models import Sequential
from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')

[3]
train_dir = "../input/ucf-crime-dataset/Train"
test_dir = "../input/ucf-crime-dataset/Test"
seed = 12
IMG_HEIGHT = 64
IMG_WIDTH = 64
BATCH_SIZE = 128
EPOCHS = 5
LR = 0.00003
IMG_SHAPE = (IMG_HEIGHT,IMG_WIDTH)
```



```
[4]
crime_types = os.listdir(train_dir)
n = len(crime_types)
print('No.of crime categories : ',n)

No.of crime categories :  14

[5]
crimes = {}
train = 0
test = 0
for clss in crime_types:
    num = len(os.listdir(os.path.join(train_dir,clss)))
    train += num
    test += len(os.listdir(os.path.join(test_dir,clss)))
    crimes[clss] = num

[6]
plt.figure(figsize=(8,5))
plt.pie(x = np.array([train,test]),autopct = '%.1f%%',explode = [0.1,0.1],labels = ['Training Data','Test Data'],pctdistance = 0.5,colors = ['yellow','green'])
plt.title('Train and Test Images ',fontsize = 18)
```



```
[7]
plt.figure(figsize = (15,5))
plt.bar(list(crimes.keys()),list(crimes.values()),width = 0.4,align = 'center',edgecolor = ['red'],color = ['orange'])
plt.xticks(rotation = 90)
plt.xlabel ('Reported Crimes')
plt.ylabel('Number of Reported Crimes')
plt.show()
```

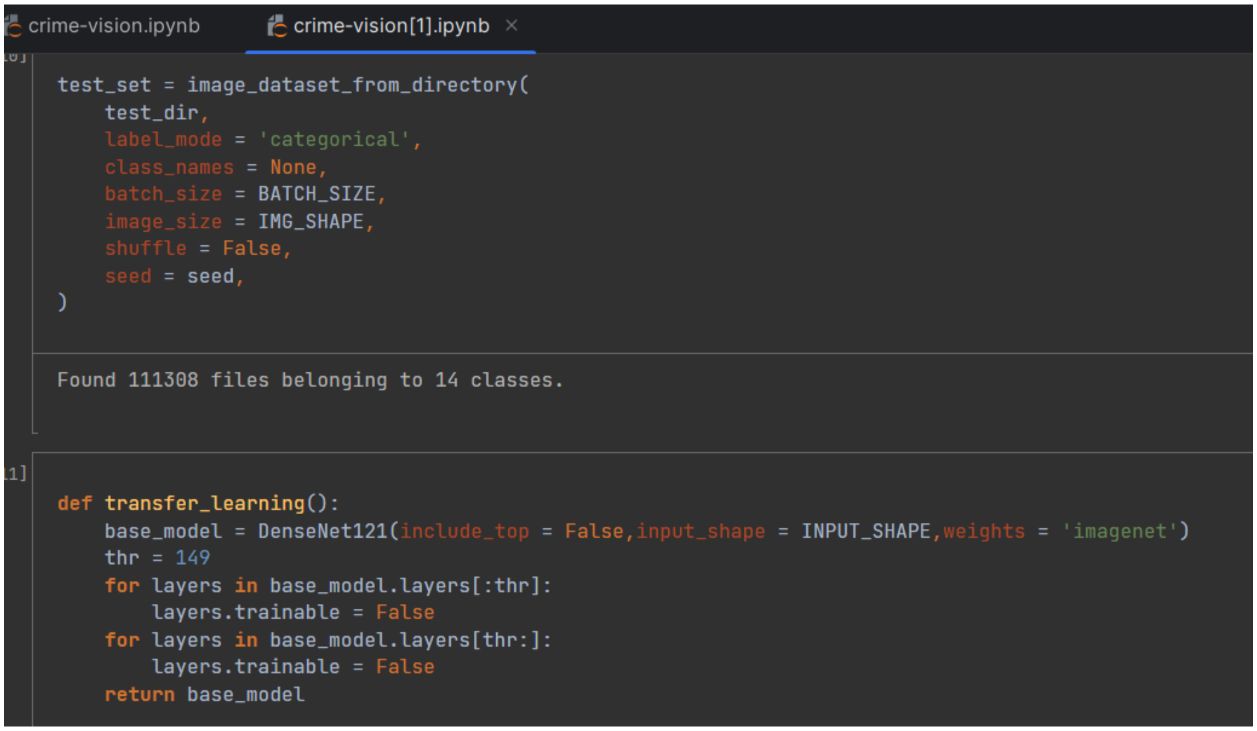


```
[8]

train_set = image_dataset_from_directory(
    train_dir,
    label_mode = 'categorical',
    batch_size = BATCH_SIZE,
    image_size = IMG_SHAPE,
    shuffle = True,
    seed = seed,
    validation_split = 0.2,
    subset = 'training',
)
```

```
Found 1266345 files belonging to 14 classes.
Using 1013076 files for training.
```

```
[9]

val_set = image_dataset_from_directory(
    train_dir,
    label_mode = 'categorical',
    batch_size = BATCH_SIZE,
    image_size = IMG_SHAPE,
    shuffle = True,
    seed = seed,
    validation_split = 0.2,
    subset = 'validation',
)
```

```
Found 1266345 files belonging to 14 classes.
Using 253269 files for validation.
```

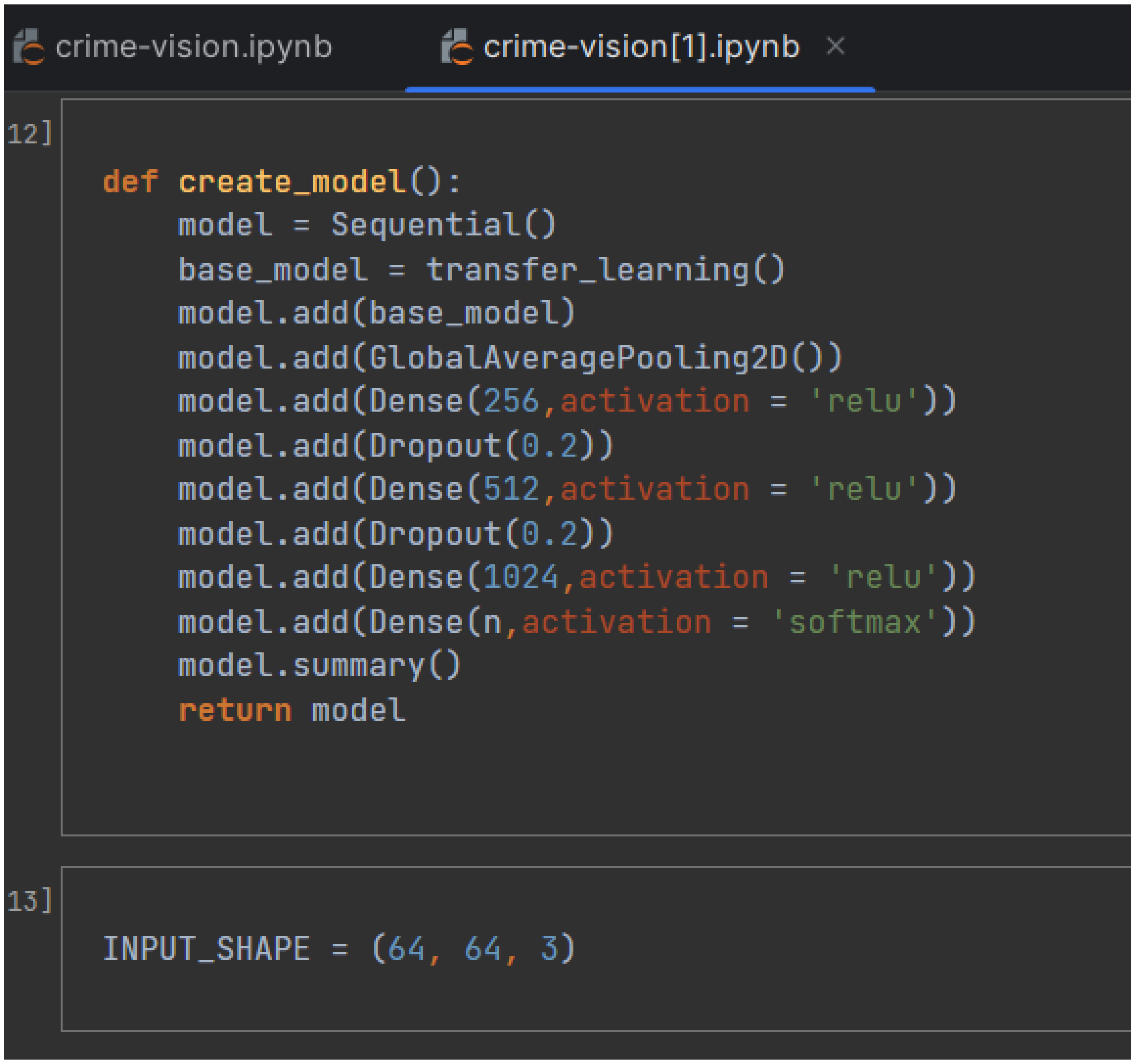


crime-vision.ipynb   crime-vision[1].ipynb ×

```
test_set = image_dataset_from_directory(
    test_dir,
    label_mode = 'categorical',
    class_names = None,
    batch_size = BATCH_SIZE,
    image_size = IMG_SHAPE,
    shuffle = False,
    seed = seed,
)
```

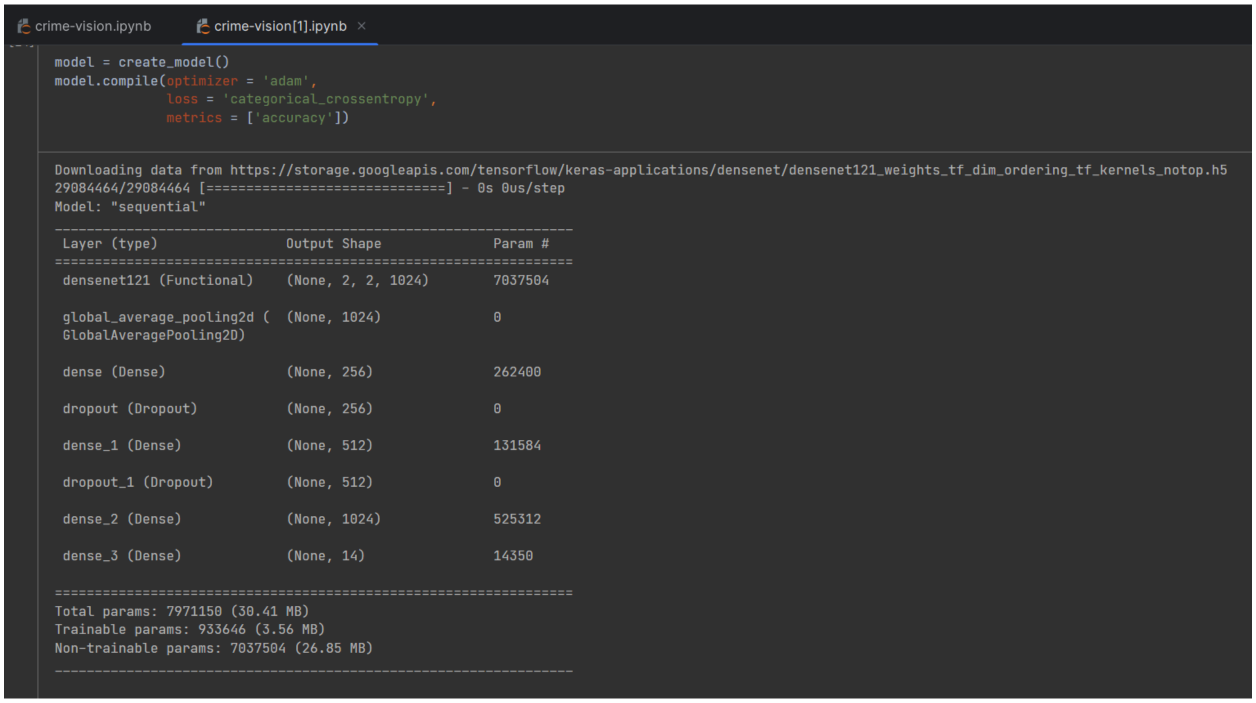
```
Found 111308 files belonging to 14 classes.
```

```
def transfer_learning():
    base_model = DenseNet121(include_top = False,input_shape = INPUT_SHAPE,weights = 'imagenet')
    thr = 149
    for layers in base_model.layers[:thr]:
        layers.trainable = False
    for layers in base_model.layers[thr:]:
        layers.trainable = False
    return base_model
```



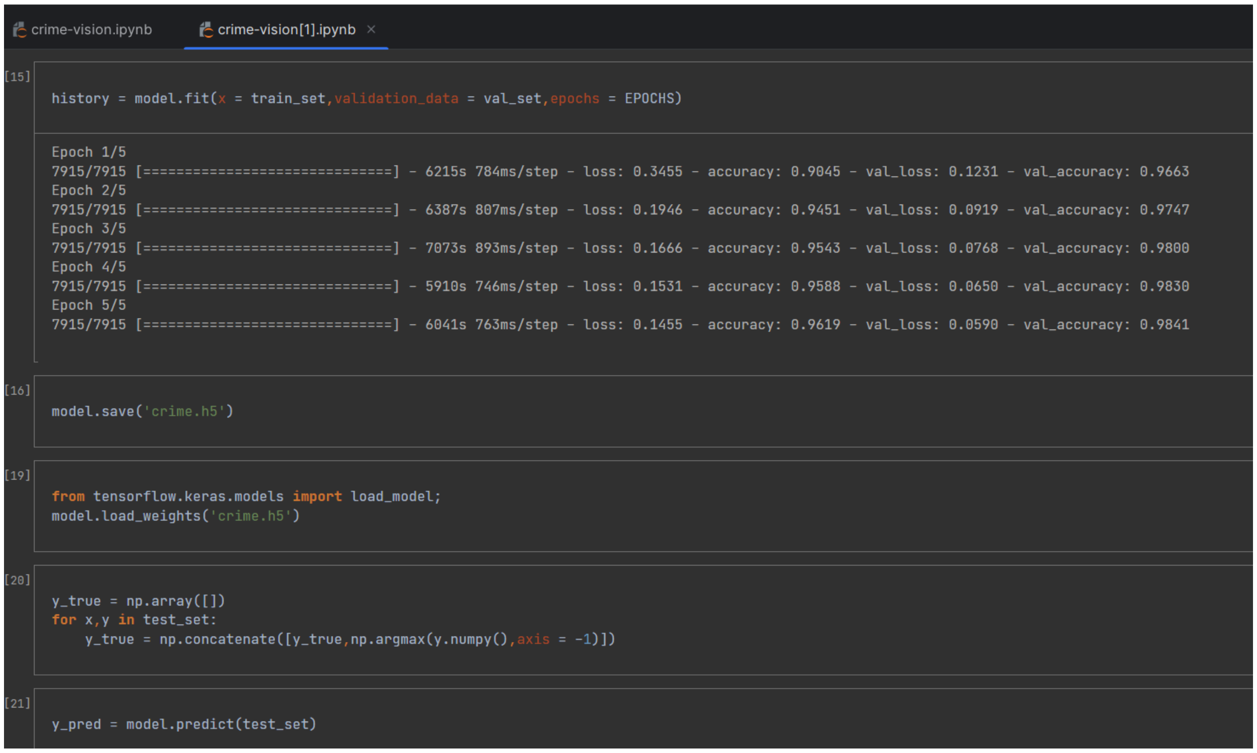
12]
```python
def create_model():
    model = Sequential()
    base_model = transfer_learning()
    model.add(base_model)
    model.add(GlobalAveragePooling2D())
    model.add(Dense(256,activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(Dense(512,activation = 'relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1024,activation = 'relu'))
    model.add(Dense(n,activation = 'softmax'))
    model.summary()
    return model
```

13]
```python
INPUT_SHAPE = (64, 64, 3)
```



```
model = create_model()
model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
29084464/29084464 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d (  (None, 1024)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               262400

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 512)               131584

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 1024)              525312

 dense_3 (Dense)             (None, 14)                14350

=================================================================
Total params: 7971150 (30.41 MB)
Trainable params: 933646 (3.56 MB)
Non-trainable params: 7037504 (26.85 MB)
_________________________________________________________________
```



```
[15]
history = model.fit(x = train_set,validation_data = val_set,epochs = EPOCHS)
```

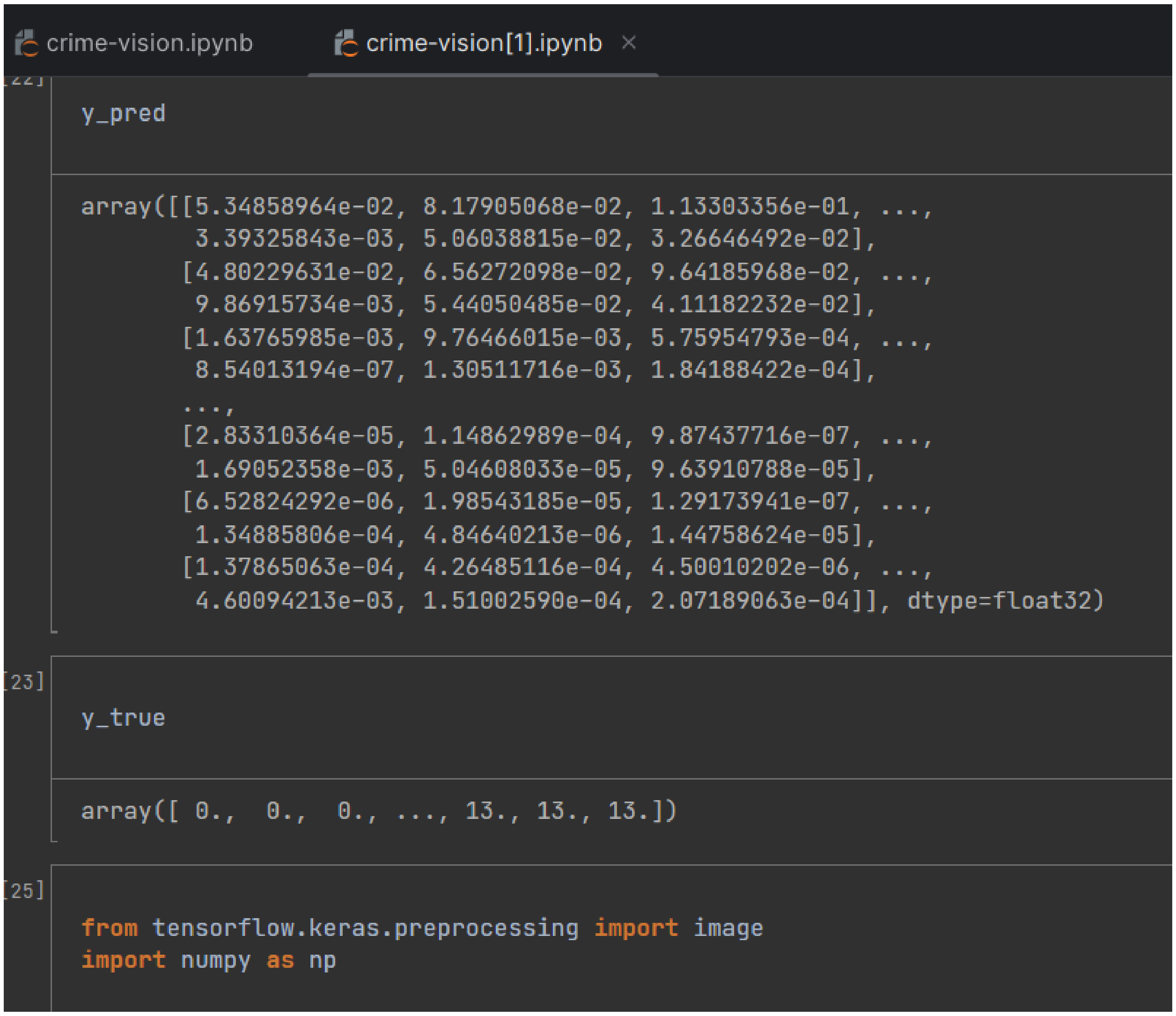
```
Epoch 1/5
7915/7915 [==============================] - 6215s 784ms/step - loss: 0.3455 - accuracy: 0.9045 - val_loss: 0.1231 - val_accuracy: 0.9663
Epoch 2/5
7915/7915 [==============================] - 6387s 807ms/step - loss: 0.1946 - accuracy: 0.9451 - val_loss: 0.0919 - val_accuracy: 0.9747
Epoch 3/5
7915/7915 [==============================] - 7073s 893ms/step - loss: 0.1666 - accuracy: 0.9543 - val_loss: 0.0768 - val_accuracy: 0.9800
Epoch 4/5
7915/7915 [==============================] - 5910s 746ms/step - loss: 0.1531 - accuracy: 0.9588 - val_loss: 0.0650 - val_accuracy: 0.9830
Epoch 5/5
7915/7915 [==============================] - 6041s 763ms/step - loss: 0.1455 - accuracy: 0.9619 - val_loss: 0.0590 - val_accuracy: 0.9841
```

```
[16]
model.save('crime.h5')
```

```
[19]
from tensorflow.keras.models import load_model;
model.load_weights('crime.h5')
```

```
[20]
y_true = np.array([])
for x,y in test_set:
    y_true = np.concatenate([y_true,np.argmax(y.numpy(),axis = -1)])
```

```
[21]
y_pred = model.predict(test_set)
```



crime-vision.ipynb   crime-vision[1].ipynb

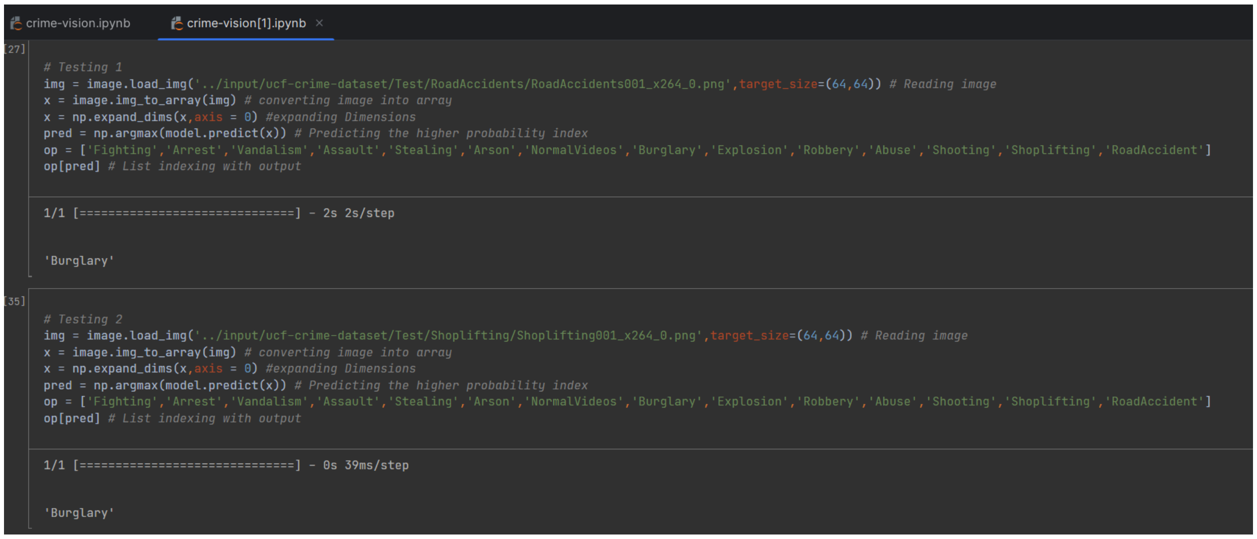
[22]
```
y_pred
```

```
array([[5.34858964e-02, 8.17905068e-02, 1.13303356e-01, ...,
        3.39325843e-03, 5.06038815e-02, 3.26646492e-02],
       [4.80229631e-02, 6.56272098e-02, 9.64185968e-02, ...,
        9.86915734e-03, 5.44050485e-02, 4.11182232e-02],
       [1.63765985e-03, 9.76466015e-03, 5.75954793e-04, ...,
        8.54013194e-07, 1.30511716e-03, 1.84188422e-04],
       ...,
       [2.83310364e-05, 1.14862989e-04, 9.87437716e-07, ...,
        1.69052358e-03, 5.04608033e-05, 9.63910788e-05],
       [6.52824292e-06, 1.98543185e-05, 1.29173941e-07, ...,
        1.34885806e-04, 4.84640213e-06, 1.44758624e-05],
       [1.37865063e-04, 4.26485116e-04, 4.50010202e-06, ...,
        4.60094213e-03, 1.51002590e-04, 2.07189063e-04]], dtype=float32)
```

[23]
```
y_true
```

```
array([ 0.,  0.,  0., ..., 13., 13., 13.])
```

[25]
```
from tensorflow.keras.preprocessing import image
import numpy as np
```



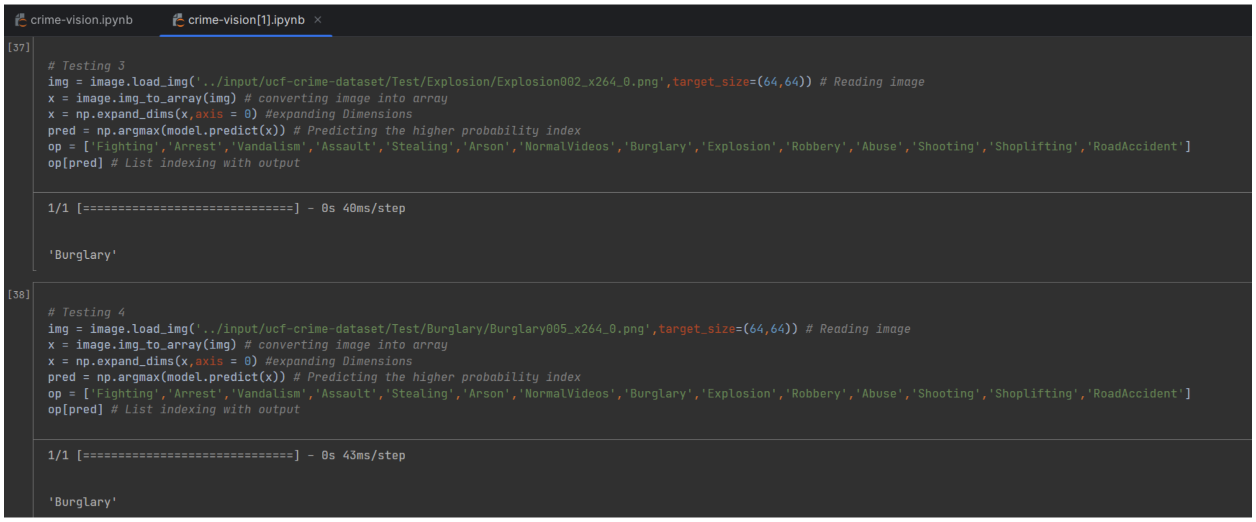
crime-vision.ipynb   crime-vision[1].ipynb

[27]
```
# Testing 1
img = image.load_img('../input/ucf-crime-dataset/Test/RoadAccidents/RoadAccidents001_x264_0.png',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # converting image into array
x = np.expand_dims(x,axis = 0) #expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probability index
op = ['Fighting','Arrest','Vandalism','Assault','Stealing','Arson','NormalVideos','Burglary','Explosion','Robbery','Abuse','Shooting','Shoplifting','RoadAccident']
op[pred] # List indexing with output
```

```
1/1 [==============================] - 2s 2s/step


'Burglary'
```

[35]
```
# Testing 2
img = image.load_img('../input/ucf-crime-dataset/Test/Shoplifting/Shoplifting001_x264_0.png',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # converting image into array
x = np.expand_dims(x,axis = 0) #expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probability index
op = ['Fighting','Arrest','Vandalism','Assault','Stealing','Arson','NormalVideos','Burglary','Explosion','Robbery','Abuse','Shooting','Shoplifting','RoadAccident']
op[pred] # List indexing with output
```

```
1/1 [==============================] - 0s 39ms/step


'Burglary'
```

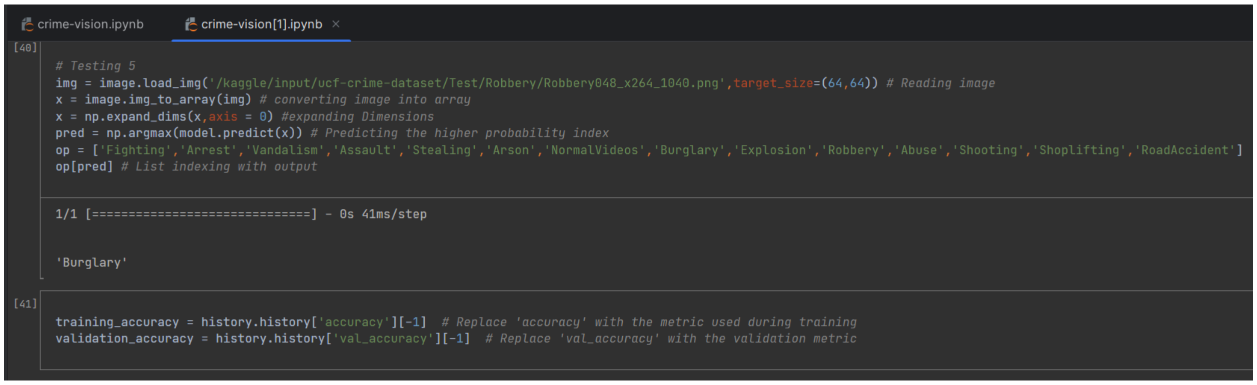


```
[37]
# Testing 3
img = image.load_img('../input/ucf-crime-dataset/Test/Explosion/Explosion002_x264_0.png',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # converting image into array
x = np.expand_dims(x,axis = 0) #expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probability index
op = ['Fighting','Arrest','Vandalism','Assault','Stealing','Arson','NormalVideos','Burglary','Explosion','Robbery','Abuse','Shooting','Shoplifting','RoadAccident']
op[pred] # List indexing with output
```

```
1/1 [==============================] - 0s 40ms/step


'Burglary'
```

```
[38]
# Testing 4
img = image.load_img('../input/ucf-crime-dataset/Test/Burglary/Burglary005_x264_0.png',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # converting image into array
x = np.expand_dims(x,axis = 0) #expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probability index
op = ['Fighting','Arrest','Vandalism','Assault','Stealing','Arson','NormalVideos','Burglary','Explosion','Robbery','Abuse','Shooting','Shoplifting','RoadAccident']
op[pred] # List indexing with output
```

```
1/1 [==============================] - 0s 43ms/step


'Burglary'
```



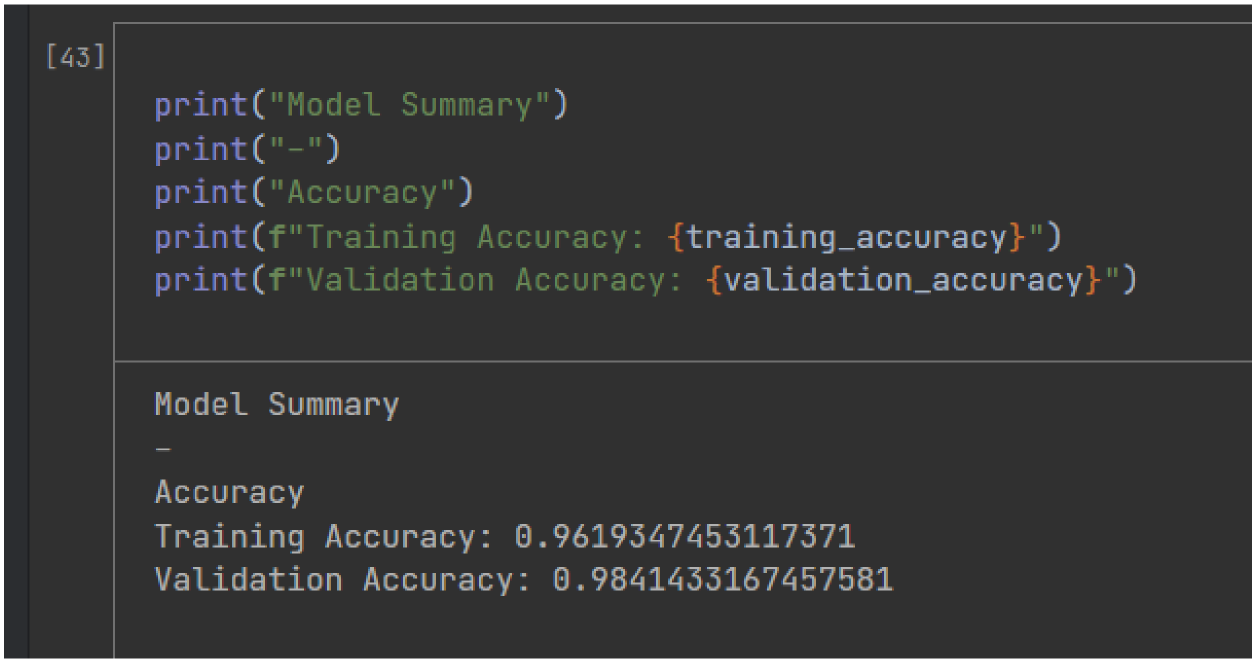
```
[40]
# Testing 5
img = image.load_img('/kaggle/input/ucf-crime-dataset/Test/Robbery/Robbery048_x264_1040.png',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # converting image into array
x = np.expand_dims(x,axis = 0) #expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probability index
op = ['Fighting','Arrest','Vandalism','Assault','Stealing','Arson','NormalVideos','Burglary','Explosion','Robbery','Abuse','Shooting','Shoplifting','RoadAccident']
op[pred] # List indexing with output
```

```
1/1 [==============================] - 0s 41ms/step


'Burglary'
```

```
[41]
training_accuracy = history.history['accuracy'][-1]  # Replace 'accuracy' with the metric used during training
validation_accuracy = history.history['val_accuracy'][-1]  # Replace 'val_accuracy' with the validation metric
```

8. PERFORMANCE TESTING

8.1 Performace Metrics

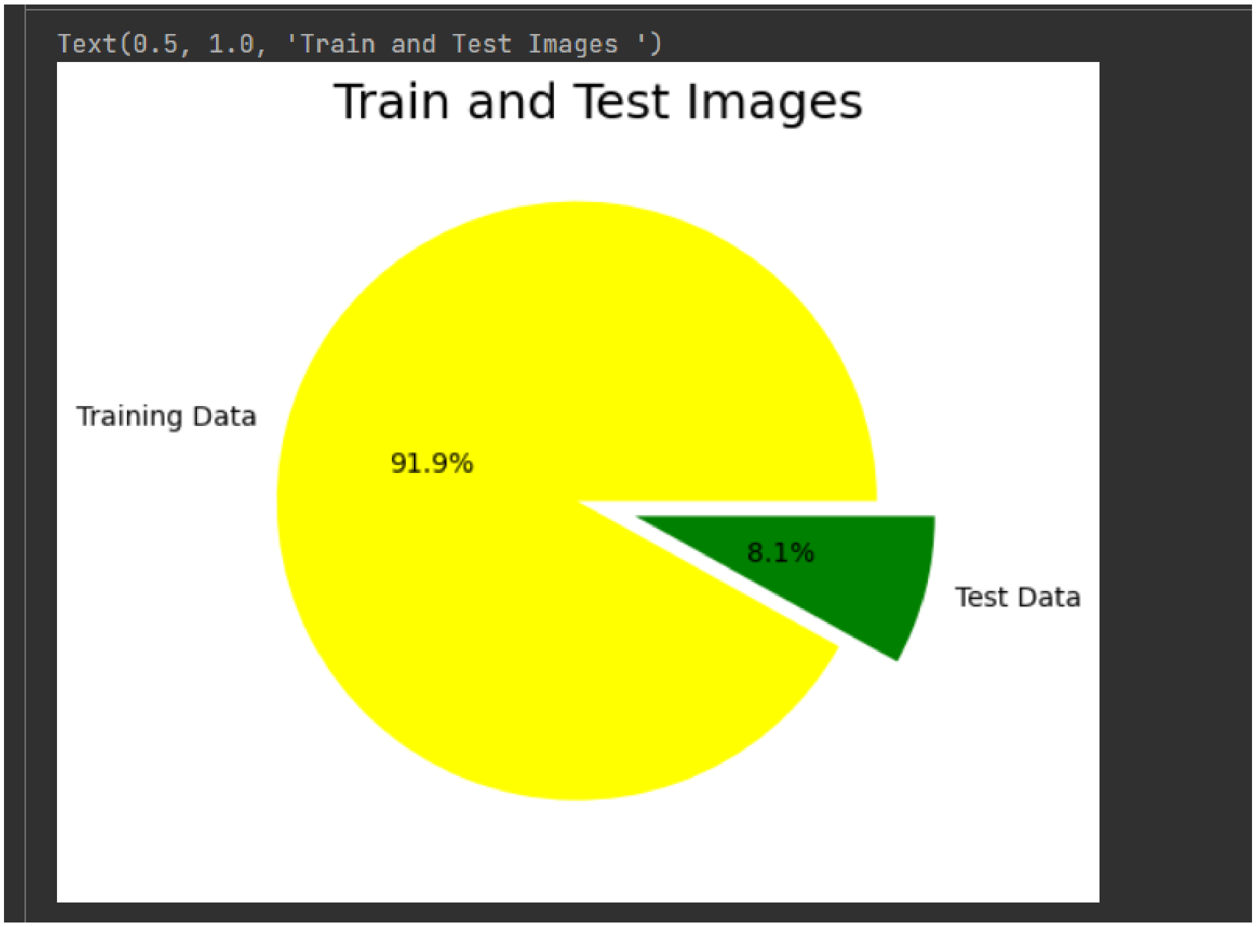


```
[43]
print("Model Summary")
print("-")
print("Accuracy")
print(f"Training Accuracy: {training_accuracy}")
print(f"Validation Accuracy: {validation_accuracy}")
```

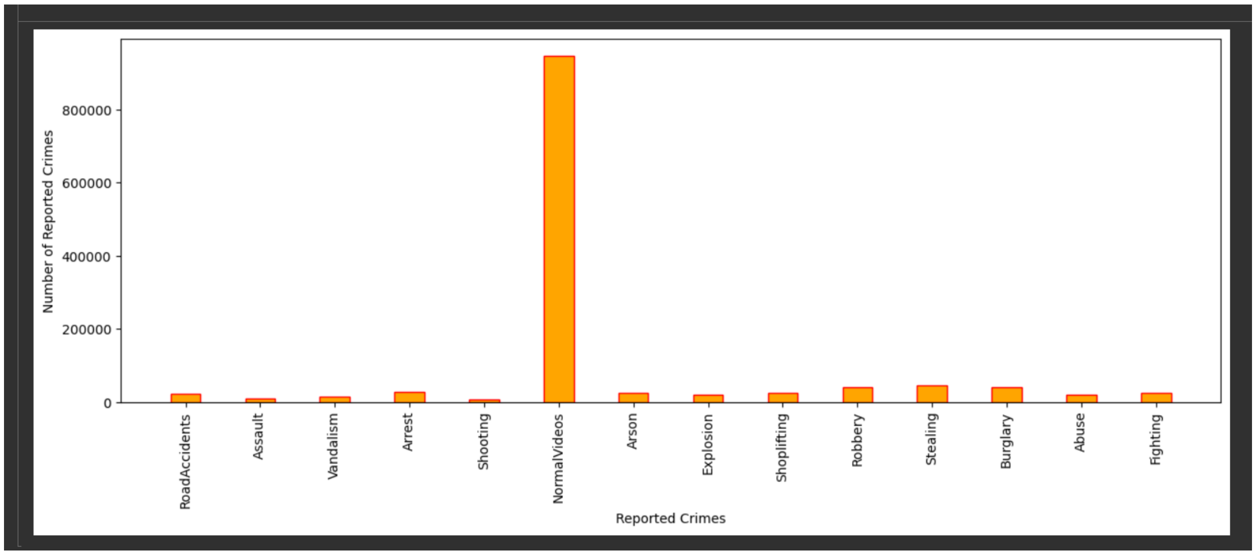
```
Model Summary
-
Accuracy
Training Accuracy: 0.9619347453117371
Validation Accuracy: 0.9841433167457581
```

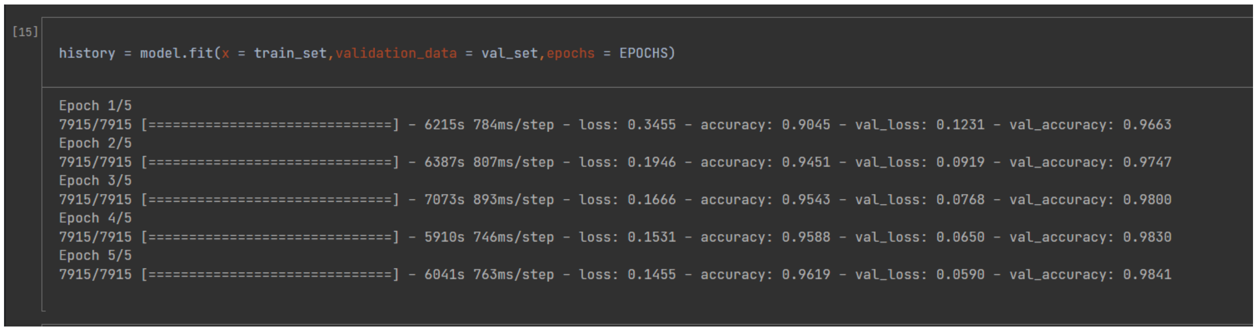
9. RESULTS

9.1 Output Screenshots



```
Text(0.5, 1.0, 'Train and Test Images ')
```





```
[15]
history = model.fit(x = train_set,validation_data = val_set,epochs = EPOCHS)

Epoch 1/5
7915/7915 [==============================] - 6215s 784ms/step - loss: 0.3455 - accuracy: 0.9045 - val_loss: 0.1231 - val_accuracy: 0.9663
Epoch 2/5
7915/7915 [==============================] - 6387s 807ms/step - loss: 0.1946 - accuracy: 0.9451 - val_loss: 0.0919 - val_accuracy: 0.9747
Epoch 3/5
7915/7915 [==============================] - 7073s 893ms/step - loss: 0.1666 - accuracy: 0.9543 - val_loss: 0.0768 - val_accuracy: 0.9800
Epoch 4/5
7915/7915 [==============================] - 5910s 746ms/step - loss: 0.1531 - accuracy: 0.9588 - val_loss: 0.0650 - val_accuracy: 0.9830
Epoch 5/5
7915/7915 [==============================] - 6041s 763ms/step - loss: 0.1455 - accuracy: 0.9619 - val_loss: 0.0590 - val_accuracy: 0.9841
```

10. ADVANTAGES & DISADVANTAGES:

Advantages:

1. Efficiency in Crime Identification:
- Deep learning enables quicker and more efficient identification of criminal activities, reducing the time required for manual analysis.

2. Accuracy in Classification:
- Deep learning models can learn intricate patterns, leading to more accurate crime classification compared to traditional methods.

3. Real-Time Surveillance:
- If implemented, real-time surveillance capabilities allow for immediate detection and response to criminal activities.

4. Data Analysis for Trends:
- Deep learning can analyze large datasets, helping law enforcement identify trends and patterns in crime data over time.

5. Automation Reduces Workload:
- Automated processes reduce the workload on law enforcement personnel, allowing them to focus on strategic and complex tasks.

6. Proactive Crime Prevention:
- Predictive analytics based on deep learning can help law enforcement agencies be more proactive in preventing crime.

7. Enhanced Forensic Analysis:
- Deep learning can expedite the forensic analysis process, providing timely insights for

investigations.

8. User-Friendly Interface:
- A well-designed interface makes the technology accessible to law enforcement personnel, even those without advanced technical skills.

Disadvantages:

1. Data Privacy Concerns:
- The use of deep learning in crime identification raises concerns about the privacy of individuals, particularly when analyzing sensitive data.

2. Bias in Training Data:
- If the training data used for the deep learning model is biased, it can lead to biased predictions, potentially impacting certain demographics unfairly.

3. Resource Intensive:
- Implementing and maintaining deep learning systems can be resource-intensive, requiring significant computational power and expertise.

4. Security Risks:
- The integration of technology poses security risks, and any vulnerabilities in the system could be exploited.

5. Resistance to Technology:
- Some law enforcement agencies or personnel may resist the adoption of new technologies due to a lack of familiarity or concerns about reliability.

6. Ethical Considerations:
- The use of deep learning in law enforcement raises ethical questions, including the responsible use of AI and potential misuse.

7. Complex Implementation:
- Implementing a deep learning system requires expertise in both machine learning and web development, making it complex and challenging for some organizations.

8. Dependency on Data Quality:
- The effectiveness of deep learning models is highly dependent on the quality and representativeness of the training data, which may not always be readily available.

11. CONCLUSION

In conclusion, the "Crime Identification using Deep Learning" project presents a promising avenue for revolutionizing crime detection and prevention. The implementation of deep learning techniques offers significant advantages in terms of efficiency, accuracy, and the potential for proactive crime prevention. The system's real-time surveillance capabilities, automated analysis, and user-friendly interface contribute to its potential impact on law enforcement processes.

However, it's crucial to acknowledge the challenges associated with the project, including data privacy concerns, potential bias in training data, and the resource-intensive nature of implementing and maintaining deep learning systems. Striking a balance between leveraging advanced technologies and addressing ethical considerations is paramount.

The advantages of this project extend beyond the immediate gains in crime identification; they encompass the empowerment of law enforcement personnel, reduction of manual workload, and the facilitation of more strategic and data-driven decision-making. The system's scalability and potential for future enhancements position it as a valuable asset in the evolving landscape of crime detection technologies.

In the face of these opportunities and challenges, a thoughtful and ethical approach to the implementation of the "Crime Identification using Deep Learning" project is essential. Collaboration between technology experts, law enforcement professionals, and ethical considerations should be central to the project's development and deployment. Through continuous improvement, adherence to best practices, and a commitment to ethical standards, this project has the potential to significantly contribute to the enhancement of public safety and the effectiveness of law enforcement efforts.

12. Future Scope:

1. Enhance Real-Time Surveillance:
- Improve real-time surveillance capabilities for quicker crime identification.

2. Continuous Model Improvement:
- Implement strategies for ongoing deep learning model improvement.

3. Expand Crime Types:
- Extend the system to identify a broader range of crimes.

4. Integrate External Databases:
- Explore integration with external databases for enhanced contextual understanding.

5. Refine Predictive Analytics:

- Improve predictive analytics for more accurate forecasting of crime hotspots.

6. User Authentication and Access Controls:
- Implement advanced user authentication and access controls.

7. Collaborate with Experts:
- Collaborate with AI and forensic experts to enhance capabilities.

8. Mobile Application Development:
- Develop a mobile application version for remote access.

9. IoT Device Integration:
- Explore IoT device integration for additional data sources.

10. Emphasize Ethical AI Practices:
- Address concerns related to bias and privacy with a focus on ethical AI practices.

13. Appendix:

The appendix section includes additional information and supplementary materials that support the "Crime Identification using Deep Learning" project. This may include:

1. Technical Specifications:
- Detailed technical specifications of the deep learning model, algorithms used, and system architecture.

2. Code Snippets:
- Relevant code snippets, particularly those related to the integration of Flask, deep learning model implementation, and user interface development.

3. Data Privacy Policy:
- A comprehensive data privacy policy outlining how user data is handled, stored, and protected within the system.

4. User Manuals:
- User manuals providing step-by-step guidance on using the system, uploading files, interpreting predictions, etc.

5. Training Data Information:
- Details about the training data used for the deep learning model, including its sources, size, and any preprocessing steps.

6. Ethical Guidelines:
- A document outlining ethical guidelines followed during the development and deployment of the system, addressing issues such as bias, fairness, and responsible AI use.

7. Testing and Validation Reports:
- Reports on testing procedures, validation results, and any benchmarks used to assess the system's performance.

8. Feedback Forms:
- Forms used for collecting feedback from law enforcement users during testing phases, helping to refine the system based on user experience.

Including these materials in the appendix enhances the transparency, completeness, and credibility of the "Crime Identification using Deep Learning" project documentation.