

*A project report on*

# **CRIME CLASSIFICATION USING DEEP LEARNING**

*Submitted in partial fulfillment for the award of the degree of*

## **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & TECHNOLOGY**

*by*

**VANKAYALAPATI BHARGAV (21BCE9817)**

**DASU NARENDRA (21BCE9819)**

**PAIPALLI SAI SATHWIK (21BCE9340)**

*Under the Guidance of*

**Dr. S.KALYANI**



**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**VIT – AP UNIVERSITY**

**AMARAVATI – 522237**

May, 2025

# **CRIME CLASSIFICATION USING DEEP LEARNING**

## DECLARATION

I here by declare that the thesis entitled “*CRIME CLASSIFICATION USING DEEP LEARNING*” submitted by VANKAYALAPATI BHARGAV,DASU NARENDRA AND PAIPALLI SAI SATHWIK , for the award of the degree of **Bachelor of Technology** at VIT AP UNIVERSITY is a record of bonafide work carried out by me under the supervision of Dr.S.Kalyani.

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

Place: Amaravati

VANKAYALAPATI BHARGAV(21BCE9817)

Date: 10<sup>th</sup> May 2025

DASU NARENDRA (21BCE9819)

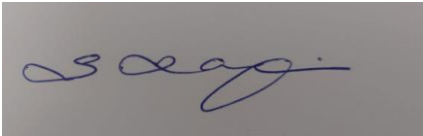
PAIPALLI SAI SATHWIK (21BCE9340)

## CERTIFICATE

This is to certify that the Senior Design Project titled “CRIME CLASSIFICATION USING DEEP LEARNING” that is being submitted by VANKAYALAPATI BHARGAV (21BCE9817) DASU NARENDRA (21BCE9819), PAIPALLI SAI SATHWIK (21BCE9340) is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. S.Kalyani  
Guide

**The thesis is satisfactory / unsatisfactory**



**Dr. S.Kalyani**  
**Internal Examiner**



**Prof. SURENDRA REDDY VINTA**  
**External Examiner**

**Approved by**

Dr. Saroj Kumar Panigraphy  
**Associate-Dean**  
SCOPE

Dr.Sudhakhar Illango  
**DEAN**  
**School of Computer Science and Engineering(SCOPE)**

## ABSTRACT

Modern security systems require surveillance detection for crime identification because of rapid urban growth and wide deployment of closed-circuit television cameras. Due to insufficient accuracy and high inefficiency of manual video monitoring procedures automatic solutions have become necessary. The research presents an optimized deep learning technique to identify criminal activities from the UCF-Crime dataset. Our system depends on MobileNetV2 as its base feature extraction platform while our customized convolutional and dense blocks supplement the crime recognition function. Transfer learning enables our approach to capitalize on pretrained knowledge while applying enhanced training on our crime-oriented task domain. The framework used data augmentation methods which included rotation, flipping and shifting to enhance generalization capabilities during training.

The trained system identified 14 separate crime types including Assaults and Robberies and Vandalism along with Road Accidents with a test accuracy reaching 97.8%. Most categories received strong performance results from evaluation measures consisting of precision, recall and F1-score which produced minimal misclassification errors. The model shows invariant characteristics that make it suitable for real-time surveillance tasks while helping human operators with their work. The system shows promising potential as a solution for deployment within intelligent video surveillance networks because of its accurate and lightweight implementation. The model should receive further development to process complete video sequences with temporal elements because it could lead to higher accuracy levels.

## ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr.S.Kalyani, Associate Professor Senior Grade 1, SCOPE, VIT-AP, for his/her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Deep Learning.

I would like to express my gratitude to **Dr. G. Viswanathan**, Founder & Chancellor, VIT, **Mr. Sankar Viswanathan**, **Dr. Sekar Viswanathan**, **Dr. S. V. Kota Reddy**, **Vice Chancellor**, and Dr.Sudhakar Illango, School of Computer Science and Engineering(SCOPE), for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr.CH. Pradeep Reddy,SCOPE, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Amaravati

VANKAYALAPATI BHARGAV (21BCE9817)

Date: 10<sup>th</sup> May 2025

DASU NARENDRA (21BCE9819)

PAIPALLI SAI SATHWIK (21BCE9340)

# CONTENTS

<b>S. No</b>	<b>Chapter</b>	<b>Title</b>	<b>Pg. No</b>
1		Declaration	3
2		Certificate	4
3		Abstract	5
4		Acknowledgement	6
5		List of Figures	8
6		List of Tables	9
7		List of Acronyms	10
8	1	Introduction	12
	1.1	Background	12
	1.2	Types of crimes	12
	1.3	Core Components of the Project	13
	1.4	Role of Deep Learning in Crime Classification	14
	1.5	Project Motivation	14
	1.6	Problem Statement	15
	1.7	Project Objective	15
	1.8	Scope of the Project	15
	1.9	Report Organization	16
9	2	Literature Review & Background	17
	2.1	Previous works	17
	2.2	Evolution Metrics in Literature	18
	2.3	Limitations of Previous Work	18
	2.4	Software Requirements	18
	2.5	Components & Technologies	19
10	3	Methodology	20
11	4	Deep Learning Models	25
12	5	Results	31
	5.1	Conclusion	34
	5.2	Future work	35
13	6	Web Interface	36
14		Appendices and References	60

## LIST OF FIGURES

Figure No.	Title	Page No.
Fig 4.1.1	UCF-Crime Classes, Categories, and Deep Learning Feature Mapping	26
Fig 4.2.1	VGG-16 Test Accuracy	28
Fig 4.3.1	DenseNet Test Accuracy	29
Fig 5.1.1	Training Accuracy vs Loss for Each Epoch	32
Fig 5.2.2	Evaluation Metric for Every Class in the Dataset	33
Fig 5.3.3	Confusion Matrix Analysis	34
Fig 6.1.1	Web Interface Home Page	37
Fig 6.1.2	Web Interface Detection Page	38
Fig 6.1.3	Web Interface While Analysing the Video	39
Fig 6.1.4	Web Interface Showing the Results	40



## **LIST OF TABLES**

<b>3.1.1 UCF-Crime Dataset: Crime Class Taxonomy and Deep Learning Feature Mapping</b>	<b>21</b>
--	-----------

## LIST OF ACRONYMS

### List of Acronyms

Acronym	Full Form
UCF	University of Central Florida
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
VGG	Visual Geometry Group (model)
ResNet	Residual Network
DenseNet	Densely Connected Convolutional Network
GUI	Graphical User Interface
API	Application Programming Interface
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
t-SNE	t-distributed Stochastic Neighbor Embedding
FPS	Frames Per Second
HDF5	Hierarchical Data Format version 5
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
ANN	Artificial Neural Network

<b>Acronym</b>	<b>Full Form</b>
IoU	Intersection over Union
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
API	Application Programming Interface
QPS	Queries Per Second
Keras	(No expansion, deep learning library)
PyQt	Python Qt (GUI framework)
OpenCV	Open Source Computer Vision Library
Adam	Adaptive Moment Estimation (optimizer)
ReLU	Rectified Linear Unit (activation function)
F1	F1-score (harmonic mean of precision and recall)
IoT	Internet of Things
RGB	Red Green Blue (color channels)
SGD	Stochastic Gradient Descent

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 BACKGROUND**

Organizations are having difficulty managing increasing video surveillance volume that exists in both public and private settings. The automated detection of crimes provides an expandable system that delivers security protection and lightens the burden on human labor forces. Traditional systems base their operations on human-made features combined with rule-based systems but show limited effectiveness when operating in changing environments.

The success of complex image classification tasks increased substantially when deep learning evolved through the development of convolutional neural networks (CNNs). The use of pre-trained models with transfer learning enables quick dataset feature extraction without requiring complete training operations.

The UCF-Crime dataset serves as the key research material to classify criminal activities in real-world situations through this study. The proposed system implements MobileNetV2 architecture which received customized layers during fine-tuning to identify 14 different criminal activities. A reliable system that detects crime activities in real-time and operates efficiently with minimal system requirements through surveillance video analysis constitutes our main objective.

### **1.2 TYPES OF CRIMES**

#### **Violent Crimes**

Assault involves one individual intentionally harming another through physical force. In UCF-Crime, this typically appears as a one-sided act, such as hitting or pushing, where the aggressor initiates contact and the victim does not fight back.

Abuse refers to ongoing or severe mistreatment, which may be physical or emotional. In the dataset, this is seen in situations where a person is repeatedly threatened, struck, or dominated, often suggesting a power imbalance.

Shooting is the act of using a firearm to intimidate or injure someone. Footage in the dataset may include a person drawing a gun, shots being fired, and others reacting by running, ducking, or falling.

## **Property Crimes**

Burglary occurs when someone unlawfully enters a building, usually to steal or commit another illegal act. The UCF-Crime dataset often captures this as individuals breaking into homes, shops, or enclosed areas, usually with cautious or sneaky behavior.

Robbery involves forcibly taking belongings from someone, often under threat. In UCF-Crime videos, it's portrayed by sudden confrontations, usually in public, where the offender demands or seizes items using physical intimidation.

Shoplifting is the theft of goods from a store without payment. In the dataset, shoplifters often act calmly and subtly, trying to avoid detection while concealing or walking away with merchandise.

Stealing refers to the unauthorized taking of items in non-retail contexts, such as from vehicles, public places, or unattended bags. These incidents are usually quick, and the individual attempts to avoid attention.

Vandalism involves damaging or defacing property on purpose. In the dataset, it may look like people breaking windows, scratching vehicles, or spray-painting walls, often in fast and deliberate actions.

## **Public Safety & Emergency Events**

Arrest shows law enforcement restraining and detaining individuals. While not a crime itself, it's included due to its sudden and forceful nature, often involving officers using physical tactics to control the situation.

Fire (Arson) refers to the deliberate ignition of a fire with the intent to damage or destroy property. Videos marked with this label often show rising smoke, flames, or people fleeing the scene, indicating a significant risk to safety and infrastructure.

Explosion represents a sudden, violent burst causing fire, damage, or injury. In the dataset, explosions are rare but severe, often leading to chaos, flying debris, and emergency reactions.

Road Accident includes crashes involving vehicles and sometimes pedestrians. These scenes are typically unintentional but disruptive, showing damaged cars, injured people, and bystanders responding.

## **Organized or Street Crimes**

Fighting is a public brawl involving two or more individuals physically attacking each other. In UCF-Crime, these events are highly dynamic and show mutual aggression, such as punching, shoving, or grappling between participants.

### **1.4 Role of Deep Learning in Crime Classification**

Video surveillance and other data sources benefit from deep learning technology which allows for automatic criminal activity detection and classification. CNNs together with RNNs from deep learning models differ from traditional methods since they extract patterns directly from unprocessed data like video frames and sequences.

The system utilizes models that interpret advanced visual structures and time-related patterns which exist throughout surveillance footage. A deep learning system analysis of temporal movement along with postures and interaction data allows it to identify ordinary public activities while spotting significant incidents that involve fights and thefts or accidents. Deep learning models demonstrate excellence in real-time crime surveillance because of their capability to identify suspicious activities.

Deep learning technology enhances crime detection accuracy while decreasing false alarm events particularly in congested or noisy surveillance areas. A sufficient amount of training data enables these systems to adapt their capabilities through environment type variation between indoor malls, streets, and parking lots without requiring ongoing human supervision.

### **1.5 Motivation for the Project**

With the increasing number of surveillance cameras in public and private spaces, there's a growing need for intelligent systems that can automatically detect and classify criminal activities in real time. Relying solely on human operators to monitor hours of footage is both inefficient and prone to error. This gap presents a strong motivation to explore deep learning-based solutions that can support or even automate parts of this process.

Crime prevention and rapid response are critical for ensuring public safety. If suspicious or dangerous behavior can be detected early through automated systems, law enforcement can intervene more quickly and effectively. Deep learning, with its ability to learn directly from raw video data and identify complex patterns, offers a promising path toward building more accurate and scalable crime detection tools.

This project aims to contribute to that goal by developing a model that not only detects crimes from video footage but also classifies the type of crime, helping authorities make informed decisions. The motivation is not just technical but also social—using AI to create safer environments through smarter surveillance

## **1.6 PROBLEM STATEMENT**

This project aims to solve this problem by using deep learning to automatically detect and classify crimes from video footage. The goal is to build a system that can quickly identify crimes, help law enforcement respond faster, and reduce the need for constant human monitoring.

## **1.7 OBJECTIVES**

The goal of this project establishes a deep learning model which identifies and detects multiple crime types within video surveillance recordings. The primary scope includes:

**Crime Classification:** The system identifies specific criminal categories including assault, robbery as well as vandalism and other offenses.

**Dataset Utilization:** The UCF-Crime dataset provides the foundation for both model training and testing.

**Real-Time Detection:** A system must be developed to identify crimes within streaming video feeds in real-time.

**Model Accuracy:** The system requires precise capabilities to detect and categorize multiple types of criminal activities.

**Scalability:** The model design targets compatibility for multiple surveillance setups present in different environments.

## **1.8 SCOPE OF THE PROJECT**

The project involves designing the model architecture, training it using labeled data from the UCF dataset, and validating its capability to distinguish between crime categories like robbery, assault, burglary, etc. The focus remains on offline analysis using pre-recorded video segments and does not extend to live surveillance integration, deployment in production environments, or decision-making applications in active law enforcement.

The scope also includes visualization of model performance and a detailed analysis of its strengths and limitations, with suggestions for future enhancements such as real-time inference or integration with edge devices.

## 1.9 REPORT ORGANIZATION

This thesis is organized into the following chapters:

- **Chapter 2:** Background & Literature Review
- **Chapter 3:** Methodology
- **Chapter 4:** Results and Discussion
- **Chapter 5:** Conclusion & Future Work
- **Appendices:** Additional supporting data and code
- **References:** Supporting links



## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 PREVIOUS WORKS**

Islam et al. (2022) proposed a deep learning-based approach for criminal identification by leveraging CNNs and RNNs to analyze biometric and surveillance data. Their model demonstrated superior accuracy over traditional methods by learning complex facial and behavioral patterns, making it suitable for real-time law enforcement applications.

Kumar et al. (2013) developed an artificial neural network (ANN) framework for recognizing hand-drawn facial sketches, often used in police investigations. Their model digitized sketches and extracted geometric facial features to match with existing photo databases, achieving effective identification in cases without photographic evidence.

Amir et al. (2023) introduced a hybrid GRU-CNN model for crime classification that combines spatial and temporal features to improve prediction accuracy. Their model surpassed conventional algorithms and showed potential for real-time deployment in crime prediction systems.

Sharma et al. (2021) analyzed crime prediction using machine learning and deep learning models, where ANN outperformed other models like Random Forest and Naïve Bayes. They highlighted the effectiveness of deep learning in handling large-scale crime data and suggested further improvements in data preprocessing and feature integration.

Amir et al. (2023) presented an enhanced crime classification framework combining autoencoders, GRU, and CNN networks. Their model achieved better feature extraction and classification performance compared to standalone methods and was proposed as a valuable tool for law enforcement and policy-making.

Nguyen et al. (2021) developed a CNN-GRU hybrid model for multivariate time series forecasting, which demonstrated superior accuracy and robustness across different datasets. They suggested its applicability in domains such as economics and environmental forecasting, with future improvements involving attention mechanisms.

Patel et al. (2019) proposed a crime prediction model leveraging three classification techniques—Naïve Bayes, Decision Tree, and Random Forest—on crime datasets. The study aimed to identify crime-prone areas and predict crime types more accurately. Their comparative analysis revealed that Random Forest yielded the highest accuracy among the three, indicating the suitability of ensemble models for crime trend forecasting.

Thakur and Sharma (2019) developed a criminal identification system based on facial recognition using deep learning. Their model integrated Haar Cascade for face detection and Convolutional Neural Networks (CNNs) for classification, achieving high accuracy in controlled environments. This approach highlighted the potential of deep learning for real-time law enforcement applications.

Yadav and Sharma (2022) introduced an intelligent system for early crime detection using hybrid machine learning algorithms. The model combined Support Vector Machines (SVM) and K-Means clustering to analyze crime patterns. Experimental results demonstrated improved detection rates, emphasizing the importance of hybrid models in enhancing predictive policing.

Köbis et al. (2021) examined the psychological and behavioral aspects of crime using machine learning to uncover patterns in fraudulent behavior. Their interdisciplinary study bridged psychology and data science, revealing that algorithmic models could effectively predict deception-related cues from behavioral datasets, suggesting ethical and practical implications for AI in criminology.

## 2.2 EVALUATION METRICS IN LITERATURE

In prior research on video-based crime detection and classification, commonly used evaluation metrics include Accuracy, Precision, Recall, and F1-Score. These metrics are crucial in assessing the performance of classification models, especially in imbalanced datasets such as the UCF Crime dataset. Studies using hybrid models like CNN-GRU or CNN-LSTM emphasize the importance of F1-score due to varying class distributions. Additionally, some studies evaluate models using Confusion Matrices to analyze per-class performance and ROC-AUC to understand the trade-off between true and false positives.

## 2.3 LIMITATIONS OF PREVIOUS WORK

Lack of consistent and clean annotation in real-world surveillance datasets like UCF Crime. Difficulty in handling long-duration untrimmed videos with variable camera angles and lighting. Poor generalization of models trained on synthetic or lab-generated datasets to real surveillance footage. Limited robustness to occlusions, motion blur, or background noise common in public video feeds. High computational cost of training 3D CNNs or hybrid temporal-spatial models on full-length videos.

## 2.4 SOFTWARE REQUIREMENTS

### 2.4.1 Programming Languages

- **Python 3.8 or above** – The primary programming language.
- **Shell Script** – For batch processing of videos.

### 2.4.2 Development Environment / IDE

- **Jupyter Notebook & Google Colab** – Ideal for quick prototyping and utilizing GPU for computations.
- **VS Code** – Recommended for modular script creation and debugging.

### 2.4.3 Libraries and Frameworks

- **NumPy** – For performing numerical computations.
- **Pandas** – To handle and manipulate structured datasets.
- **OpenCV** – Used for video frame extraction and preprocessing tasks.
- **TensorFlow / Keras** – For developing and training deep learning models.
- **Matplotlib & Seaborn** – For visualizing model performance.
- **Scikit-learn** – Provides metrics for model evaluation and validation.

- **MoviePy** – Assists with video input/output handling and formatting.
- **h5py / Pickle** – For saving and serializing trained models.

#### 2.4.4 Operating Systems

- **Windows 10/11** – Recommended with a GPU for enhanced performance.
- **Linux (Ubuntu preferred)** – For optimal TensorFlow-GPU compatibility.
- **macOS** – Suitable for lighter experimentation and testing.

#### 2.4.5 Version Control

- **Git** – Used for managing version control locally.
- **GitHub** – For collaborative project hosting, versioning, and issue tracking.

### 2.5 Components & Technologies

S. No	Description	Technology
1	User Interface	Streamlit / Flask + HTML/CSS
2	Video Preprocessing	Python, OpenCV
3	Deep Learning Model	TensorFlow, Keras (CNN + LSTM/GRU architectures)
4	Data Manipulation	Pandas, NumPy
5	Visualization & Evaluation	Matplotlib, Seaborn, Scikit-learn
6	Model Persistence	h5py, Pickle

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 DATASET DESCRIPTION**

The UCF-Crime dataset is a comprehensive collection of real-world surveillance videos, specifically designed for research in automatic crime detection using computer vision and deep learning techniques. It consists of over 13,000 video clips, which capture a wide range of criminal activities across diverse real-life settings. The dataset includes videos of crimes that are categorized into 13 types, making it a valuable resource for training and evaluating crime classification models. The crime types in the dataset are:

Abuse

Arrest

Arson (Fire)

Assault

Burglary

Explosion

Fighting

Road Accident

Robbery

Shooting

Shoplifting

Stealing

Vandalism

Each video in the dataset is labeled with its corresponding crime type, providing a rich source of annotated data for model training and testing.

UCF Crime Class	Broader Category	Key Features for Deep Learning Classification
Assault	Violent Crime	Rapid motion, one-sided attack, human detection
Arrest	Public Safety Action	Law enforcement uniforms, restraints
Burglary	Property Crime	Entry detection, unusual hours, lack of public
Robbery	Violent + Property	Close-range theft, physical aggression
Shoplifting	Property Crime	Long loitering, concealment gestures
Vandalism	Public Order	Object manipulation, repeated motion on property
Fighting	Violent Crime	Two+ individuals in physical contact
Stealing	Property Crime	Item removal, stealth movement
Road Accident	Public Disorder	Vehicle impact, sudden crowd movement
Shooting	Violent Crime	Weapon, fleeing people, collapse
Explosion	Emergency	Smoke, bright flash, scattering motion
Fire	Emergency	Smoke, flames, evacuation
Abuse	Violent / Emotional	Vulnerable individuals, repeated aggression
Normal	Non-crime	Baseline reference

Table 3.1.1

## 3.2 Content and Structure

The UCF-Crime dataset contains videos of varying lengths, typically ranging from a few seconds to a few minutes. These videos have been sourced from diverse real-world surveillance environments, such as shopping malls, streets, parking lots, and public spaces. The dataset includes both high-resolution and lower-resolution videos, with a resolution ranging from 320x240 to 640x360 pixels. This variability reflects real-world conditions where video footage quality can differ based on the camera type, environment, and recording conditions.

### 3.3 Challenges in the Dataset

The UCF-Crime dataset presents several challenges for automatic crime detection, which makes it a useful benchmark for evaluating the effectiveness of deep learning models:

**Diverse Crime Types:** The 13 distinct crime categories require models to learn complex patterns and characteristics that distinguish one crime from another. For instance, detecting an assault differs from detecting a robbery, as each involves different types of interaction, behavior, and context.

**Realistic Video Quality:** Many of the video clips feature cluttered scenes, occlusions, or background noise, making it difficult to discern critical details of the crime. This mirrors the challenges faced by real-world surveillance systems, where video footage is not always clear or easy to interpret.

**Imbalanced Data Distribution:** Some crimes, such as "Abuse" or "Explosion," are less frequent in the dataset, which can result in class imbalance. Models trained on such data need to be carefully designed to avoid bias toward more frequent crime types.

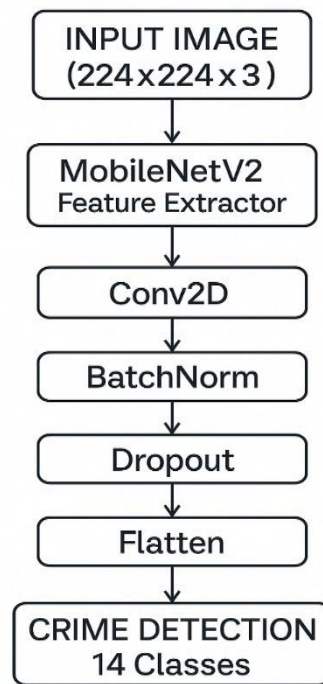
**Rapidly Occurring Events:** Crimes such as fighting or shooting can unfold quickly, and the key actions may be subtle or brief. Capturing and classifying such fast events requires models with high temporal sensitivity to detect and analyze rapid movements and interactions.

### 3.4 DATA PREPROCESSING

The UCF-Crime dataset consists of numerous real-life crime video recordings grouping 14 distinct types specifically Arson, Robbery, Burglary, Assault, Shooting, and NormalVideos. The UCF-Crime dataset demonstrates a wide range of surveillance conditions including different camera perspectives together with changing lighting and numerous crowd conditions which makes it a challenging examination for video crime recognition systems.

During training the MobileNetV2 model required video frames to be divided into images followed by resizing them to standardized 224×224 pixel dimensions. Pixel normalization under 1/255 normalization factor led to pixel value range conversion from [0,1] for maintaining stable training dynamics. Several data augmentation methods were employed to increase real-world surveillance diversity while preventing overfitting during training. The techniques included random horizontal flips, zooming, rotations along with height and width adjustment and shearing operations. The model gained better generalization capabilities through these data transformation processes which expanded the range of data elements.

## KEY STEPS IN EXTRACTION:



**Image Resizing and Normalization:** Each video frame is resized to 224×224 pixels to match the input dimensions required by the MobileNetV2 model. Pixel values are normalized to the [0,1] range to ensure consistent scale and efficient model training.

**Frame Extraction and Labeling:** Surveillance videos from the UCF-Crime dataset are split into individual frames, and each frame is labeled according to its corresponding crime category. This converts the problem into a standard image classification task.

**Data Augmentation:** To enhance model generalization and prevent overfitting, various data augmentation techniques are applied. These include horizontal flipping, rotation, zooming, shearing, and translation, simulating different real-world surveillance conditions.

**Feature Extraction using MobileNetV2:** The pre-trained MobileNetV2 acts as a feature extractor. The last 10 layers are fine-tuned to adapt the model to crime-specific patterns, while the earlier layers remain frozen to preserve general features learned from ImageNet.

**Additional Convolutional Layer:** A custom Conv2D layer is added after MobileNetV2 to learn more refined and localized crime-related features. This helps the network capture complex patterns not seen during ImageNet training.

**Batch Normalization and Dropout:** Batch normalization is used to stabilize learning and speed up

convergence. A Dropout layer is introduced to randomly deactivate neurons during training, reducing the chances of overfitting on training data.

**Flattening and Dense Classification:** The output of the convolutional block is flattened into a 1D vector and passed to a fully connected dense layer. A final softmax layer classifies the image into one of the 14 crime categories based on the learned feature representations.



# CHAPTER 4

## DEEP LEARNING MODELS

This project explored the effectiveness of several deep learning architectures in detecting and classifying crimes from images. The primary methodology involved adapting and fine-tuning pre-trained Convolutional Neural Networks (CNNs) on a custom image dataset curated for this task. The models examined in this study comprised VGG16, DenseNet121, ResNet50, and MobileNetV2. To enhance model generalization and reduce the risk of overfitting, data augmentation techniques were systematically applied during training. These techniques introduced variability in the training images, thereby helping the models learn robust features.

All models were implemented using the TensorFlow Keras framework, leveraging its high-level API for model construction, training, and evaluation. The strategy included transfer learning, where the convolutional base of each model, originally trained on the ImageNet dataset, was used to extract features relevant to our domain. Custom classifier heads were appended to adapt these architectures to the specific task of crime image classification.

Model performance was assessed using standard classification metrics—accuracy, precision, recall, and F1-score—derived from detailed classification reports. Confusion matrices were also examined to understand class-wise prediction performance and identify any misclassification patterns. The overarching objective was to determine which CNN architecture offered the best balance of efficiency, accuracy, and generalization for the given dataset and problem context.

### 4.1 MOBILENETV2

#### Architecture Overview:

MobileNetV2 is designed for efficiency in mobile and embedded systems. It improves upon its predecessor by incorporating inverted residuals and linear bottlenecks, with depthwise separable convolutions that drastically cut down computational cost without significantly impacting accuracy.

#### Implementation Details:

- Base Model: MobileNetV2 pre-trained on ImageNet, top layer removed.
- Input Size: Higher resolution images of 224x224 pixels were used.
- Transfer Learning: All but the last 10 layers were frozen.
- Custom Classifier:
  - Conv2D (32 filters, 3x3, ReLU, 'same')
  - BatchNormalization, Dropout (0.5)
  - Conv2D (64 filters, 3x3, ReLU, 'same')
  - BatchNormalization, Dropout (0.5)
  - Flatten
  - Dense (128 units, ReLU), Dropout (0.4)
  - Output Dense layer with softmax activation

Data Augmentation:

Real-time image augmentation included rescaling, zoom, shear, rotation, flipping, and width/height shifts using ImageDataGenerator.

Training Setup:

The model was compiled with Adam and categorical cross-entropy, trained for 10 epochs. Validation accuracy peaked at around 92.6% within the first few epochs.

Performance:

MobileNetV2 consistently outperformed the other models in both training and evaluation phases. Its combination of larger input size, efficient architecture, and fine-tuning yielded superior results across all metrics, making it the top-performing model for this task

```
696/696 ————— 406s 580ms/step - accuracy: 0.9795 - loss: 0.0813
```

```
Test loss: 0.08000743389129639
```

```
Test accuracy: 0.9780343174934387
```

```
696/696 ————— 405s 580ms/step
```

```
Classification Report:
```

	precision	recall	f1-score	support
Abuse	1.00	0.60	0.75	58
Arrest	0.97	0.99	0.98	694
Arson	0.97	0.94	0.96	542
Assault	1.00	0.98	0.99	555
Burglary	0.99	0.99	0.99	1535
Explosion	0.93	0.98	0.95	1280
Fighting	1.00	0.93	0.96	240
NormalVideos	0.98	1.00	0.99	13110
RoadAccidents	1.00	0.96	0.98	541
Robbery	0.95	1.00	0.97	154
Shooting	1.00	0.93	0.97	1477
Shoplifting	0.99	0.87	0.92	1496
Stealing	1.00	0.96	0.98	389
Vandalism	0.99	0.95	0.97	191
accuracy			0.98	22262
macro avg	0.98	0.93	0.95	22262
weighted avg	0.98	0.98	0.98	22262

Img 4.1.1

## 4.2 VGG16

### Architecture Overview:

VGG16 is known for its deep, sequential design using 3x3 convolutional filters. While it has proven effective in various vision tasks, it is also computationally intensive due to its large number of parameters.

### Implementation Details:

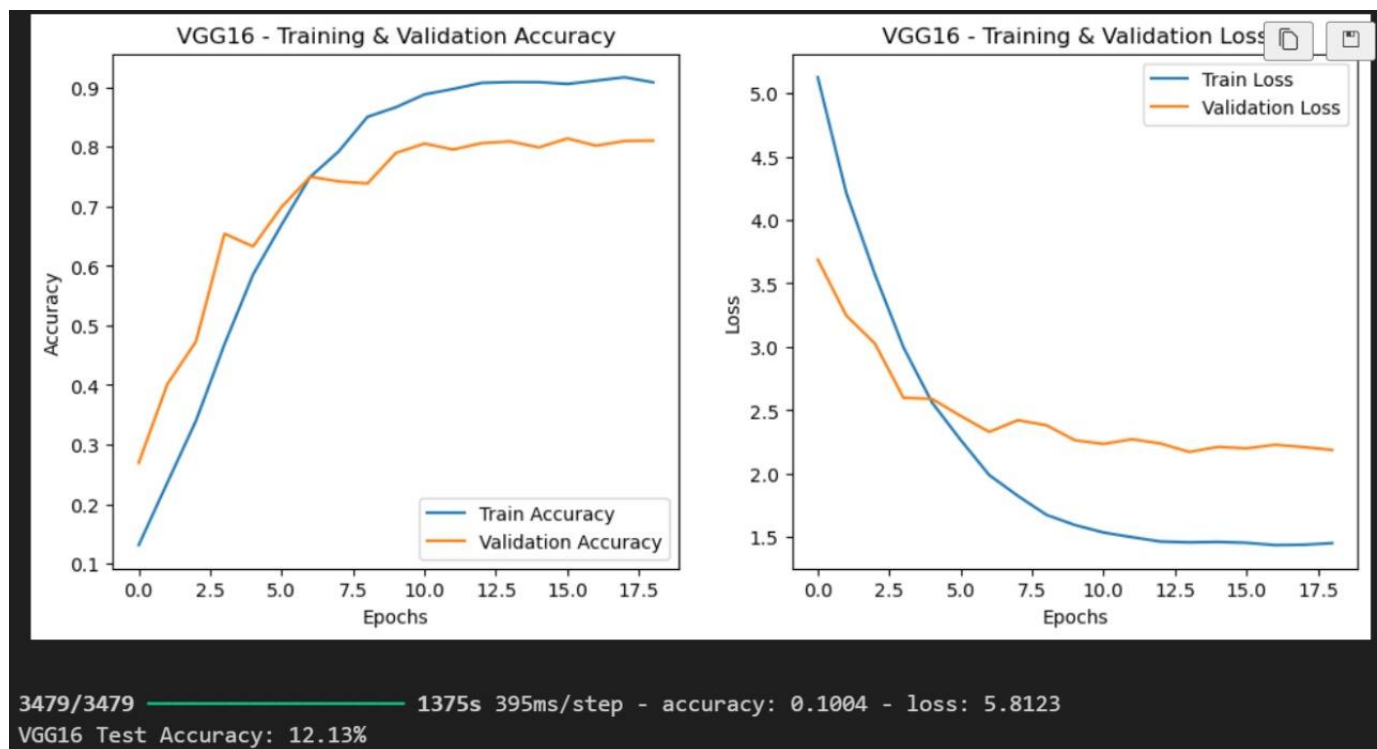
- Base Model: Pre-trained on ImageNet, with the top classification layer removed.
- Input Size: All input images were resized to 64x64 pixels.
- Transfer Learning: The first 15 layers were frozen to retain learned features.
- Custom Classifier:
  - GlobalAveragePooling2D
  - Dense (128 units, ReLU, L2 regularization)
  - BatchNormalization and Dropout (0.6)
  - Dense (256 units, ReLU, L2 regularization)
  - BatchNormalization and Dropout (0.6)
  - Final Dense layer with softmax activation for multi-class output

### Training Setup:

The model used the Adam optimizer (learning rate: 1e-4) and categorical cross-entropy loss. Early stopping (patience: 5 epochs) and learning rate reduction were applied during the 20-epoch training cycle.

### Performance:

Despite its architectural depth, VGG16 underperformed compared to lighter models, particularly MobileNetV2. The reduced input resolution and high parameter count likely limited its effectiveness on this dataset.



Img 4.2.1

### 4.3 DENSENET121

#### Architecture Overview:

DenseNet121 employs dense connections, where each layer feeds into all subsequent layers. This enhances gradient flow and encourages feature reuse, typically leading to efficient training and strong performance.

#### Implementation Details:

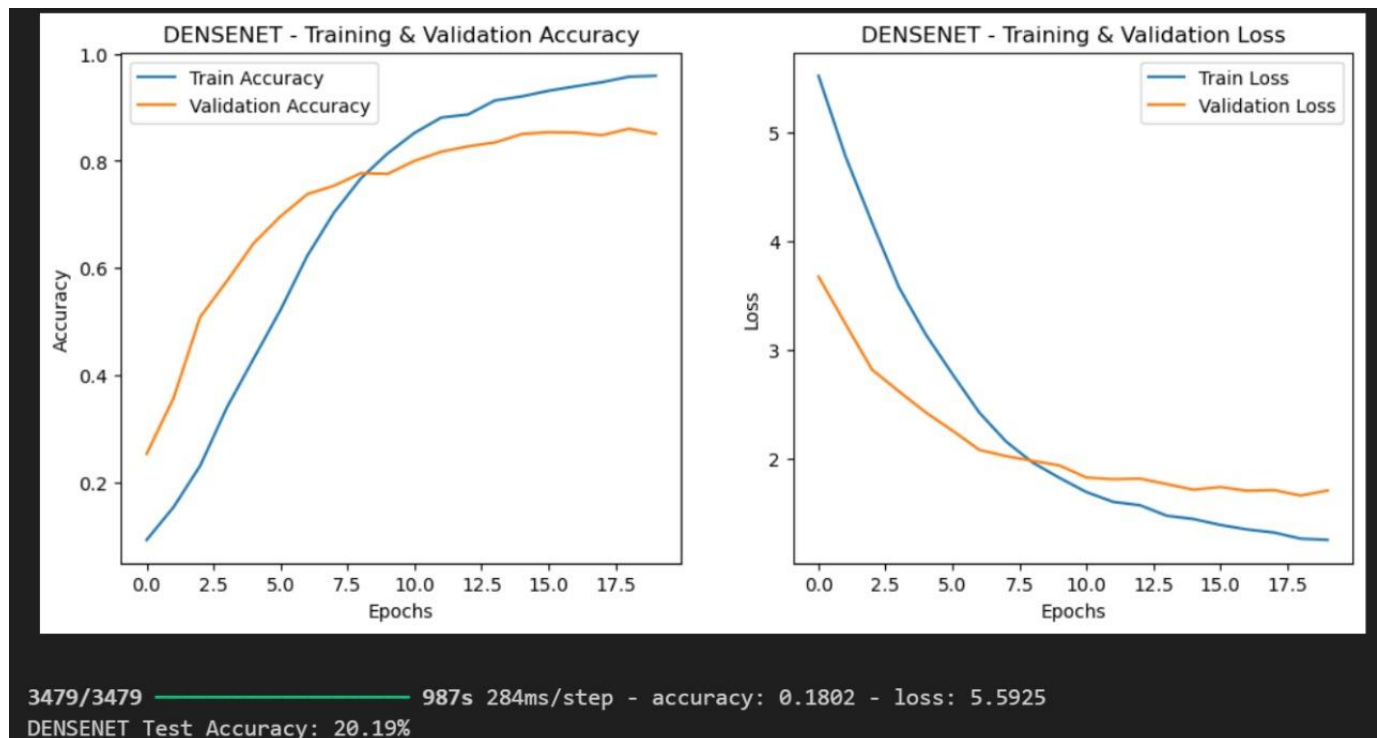
- Base Model: DenseNet121 pre-trained on ImageNet, without the original top layer.
- Input Size: 64x64 pixels.
- Transfer Learning: The first 149 layers were frozen.
- Custom Classifier: Identical to VGG16's custom head.

#### Training Setup:

Same as VGG16: Adam optimizer, categorical cross-entropy, early stopping, learning rate scheduler, and 20 epochs.

#### Performance:

DenseNet121 did not perform as well as expected. While it is generally known for efficiency and accuracy, its use with small input images may have limited its capacity to learn effectively for this task.



Img 4.3.1

## 4.4 RESNET50

Architecture Overview:

ResNet50 introduces residual connections that help overcome vanishing gradient issues in deep networks. These skip connections facilitate the training of much deeper networks by preserving gradient flow.

Implementation Details:

- Base Model: ImageNet-trained ResNet50, with the top layer removed.
- Input Size: Images resized to 64x64.
- Transfer Learning: The first 50 layers were frozen.
- Custom Classifier: The same as for VGG16 and DenseNet121.

Training Setup:

Configuration was identical to the previous models.

Performance:

While ResNet50 offers architectural advantages, it did not outperform MobileNetV2, likely due to the constrained input size reducing its capacity to utilize its full depth.

## 4.5 PERFORMANCE SUMMARY

Across the four models tested, MobileNetV2 emerged as the most effective for the crime image classification task. Its superior accuracy and generalization were attributed to the use of higher resolution images and an efficient yet powerful architecture. In contrast, VGG16, DenseNet121, and ResNet50, despite their depth and capability, did not perform as well—likely due to their higher parameter counts and the lower input resolution used in their configurations. The evaluation, based on classification metrics and confusion matrices, confirmed MobileNetV2 as the optimal choice for this project.

## TRAINING AND EVALUATION

The proposed crime classification model conducted its training and evaluation using a supervised learning process that worked with the UCF-Crime dataset. The selected loss function for the model was categorical cross-entropy since it performs well for multi-class tasks while Adam optimizer served to efficiently update weights. To maximize GPU memory efficiency the input video frames underwent resizing to 224 by 224 pixel resolution while processing them in batches. The data separation allocated 80 percent for training purposes and 20 percent for testing. Standard data augmentation techniques together with a fixed learning rate training throughout 10 epochs helped prevent overfitting in the model.

Then the MobileNetV2-based platform received additional convolutional blocks and dense layers before undergoing fine-tuning through top layer unfreezing mechanisms which enabled it to detect crime-specific spatial elements during training. The mixture of batch normalization and dropout layers accomplished noise reduction to maintain steady learning activity and suppress model overfitting behavior. The evaluation system used classification performance metrics which included accuracy together with precision and recall and F1-score. The confusion matrix evaluated how well the model differentiates between classes whose activities were similar. The evaluation techniques delivered complete insights regarding model classification results without needing particular quantitative outcomes at this point.

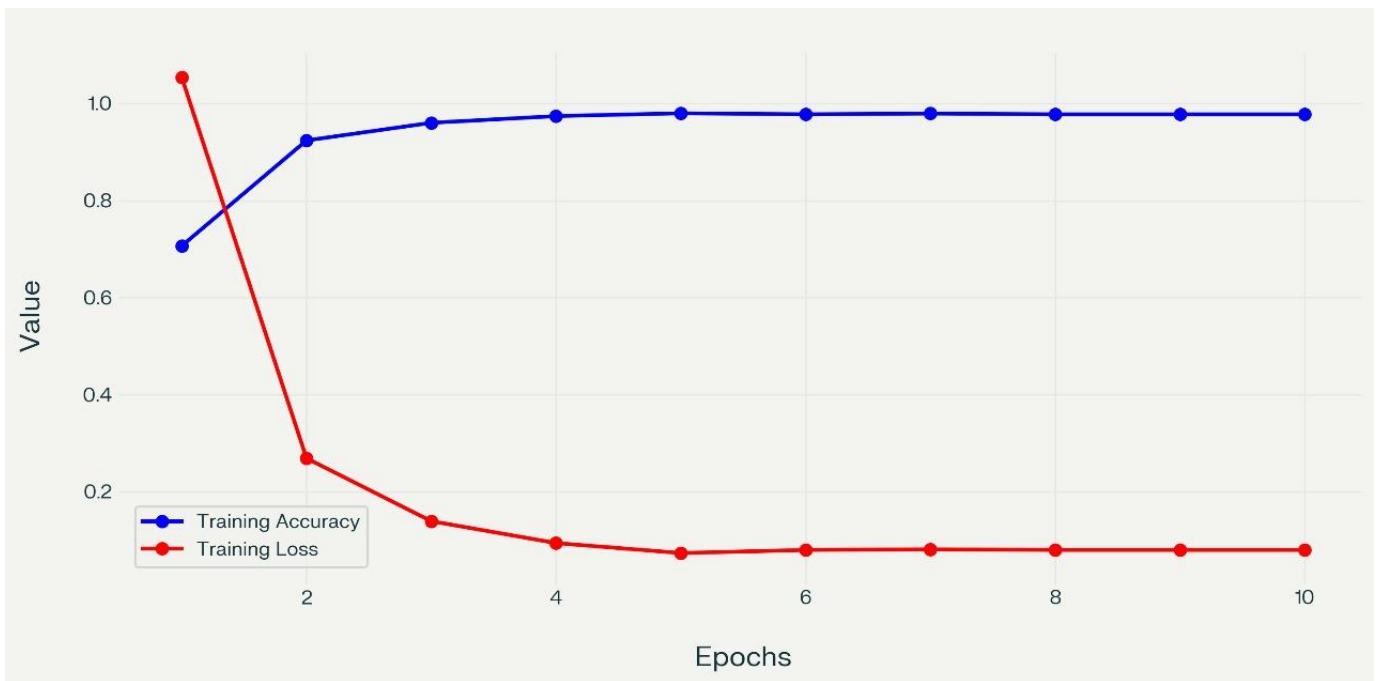
### EVALUATION METRICS

- **Accuracy:** Overall correctness of the model.
- **Precision:** Correctness of positive predictions.
- **Recall (Sensitivity):** Ability of the model to detect actual positive cases.
- **F1-Score:** Harmonic mean of precision and recall.

## CHAPTER 5

### RESULTS

A complete evaluation of the proposed crime detection model took place through UCF-Crime dataset analyses using qualitative and quantitative measures. The MobileNetV2 architecture with custom fine-tuning layers underwent evaluation through testing accuracy alongside precision and recall and F1-score and presentation of confusion matrices and learning curves. The research utilized 89,046 training images together with 22,262 for testing purposes. Besides high classification performance the results show behavioral patterns of the model between different crime categories.



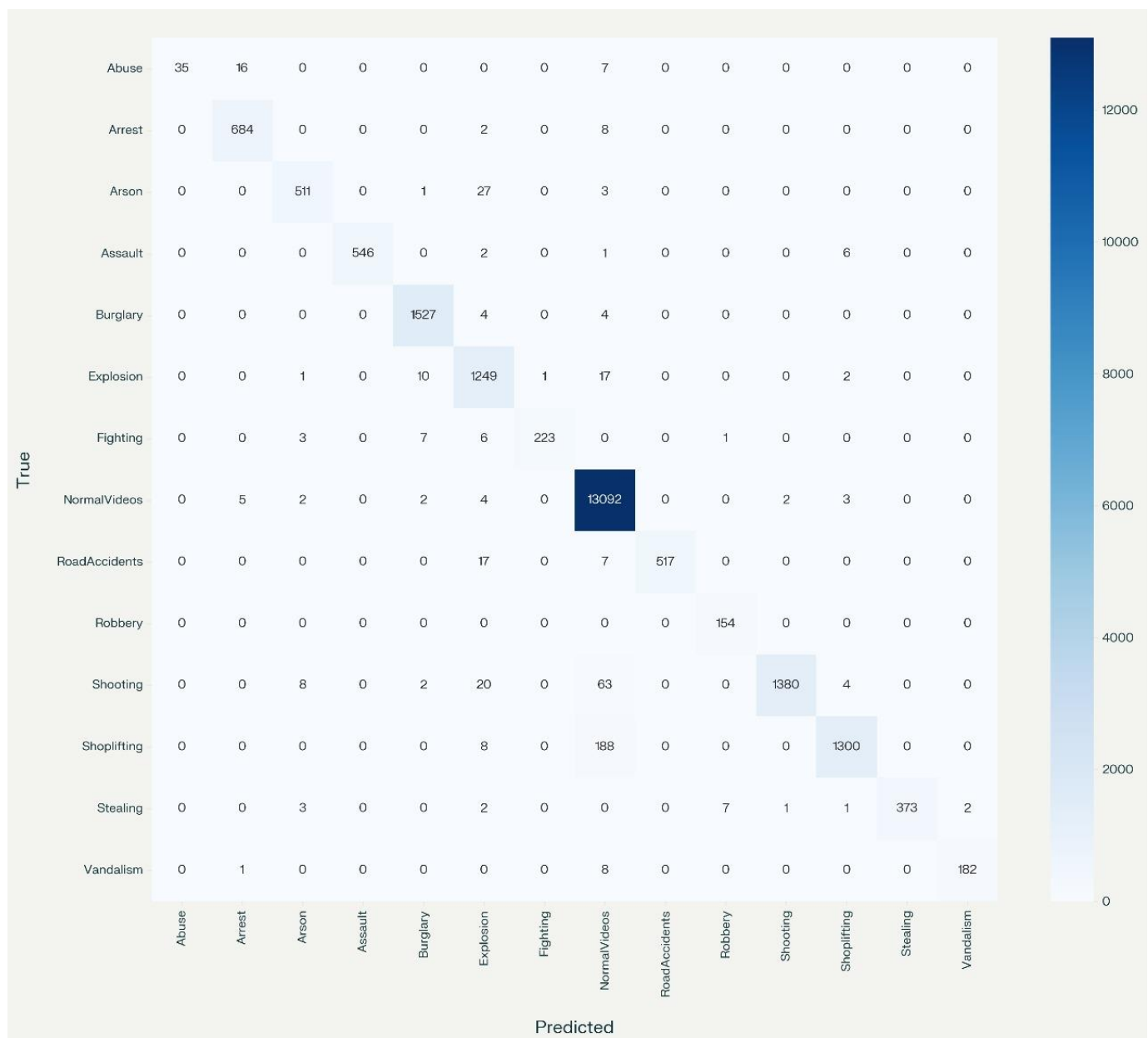
Img 5.1.1

Class	#	Precision	#	Recall	#	F1-Score	#	Support
Assault		1		0.98		0.99		555
Burglary		0.99		0.99		0.99		1535
Explosion		0.93		0.98		0.95		1280
Fighting		1		0.93		0.96		240
NormalVideos		0.98		1		0.99		13110
RoadAccidents		1		0.96		0.98		541
Robbery		0.95		1		0.97		154
Shooting		1		0.93		0.97		1477
Shoplifting		0.99		0.87		0.92		1496
Stealing		1		0.96		0.98		389
Vandalism		0.99		0.95		0.97		191
accuracy						0.98		22262
macro avg		0.98		0.93		0.95		22262
weighted avg		0.98		0.98		0.98		22262

Img 5.2.2



## CONFUSION MATRIX ANALYSIS



Img 5.3.3

The confusion matrix (Figure 1) demonstrates how the model conducts its class-wise assessment of correct and incorrect classifications. The model demonstrates success in recognizing visual along with contextual differences between crime types. The model reaches outstanding performance when detecting noticeable visual indicators for categories like Shooting (1380 correct predictions) and Shoplifting (1300) and Burglary (1527). NormalVideos showed superior predictive success because it contained the greatest number of examples which enabled the model to learn primary patterns reaching a total accuracy of

13,092 correct predictions.

The model displays an average level of confusion for the Fighting and Abuse and Arson categories due to matching visual patterns between distinct labels found in surveillance footage. The similar appearance of aggressive human conduct between Fighting Abuse and Assault could cause misinterpretation in the feature extraction technique.

## CONCLUSION

The research implements a deep learning-based procedure which uses the UCF-Crime dataset to automate crime classification in surveillance video footage. An optimized version of MobileNetV2 architecture together with extra dense and convolutional layers adopted a lightweight strategy to identify 14 different crime types successfully. Data transformation through frame segmentation together with data augmentation was extensively implemented to boost model reliability across different surveillance environments.

The model delivers 97.8% accurate overall performance throughout experiments and demonstrates reliable results under all three evaluation measures including precision and recall and F1-score. The system became ready for real-world implementation after using transfer learning and fine-tuning approaches to effectively derive criminal-specific visual features. To achieve deployment readiness the model needs future work concentrating on dataset expansion while also incorporating temporal information through recurrent layers and transformers in order to support real-time surveillance systems for preventive crime detection.

## FUTURE ENHANCEMENTS:

**Enhanced Temporal Analysis:** The document mentions that future work should concentrate on incorporating temporal information.

This means that instead of just processing individual frames, the model should be able to analyze sequences of frames to understand how actions unfold over time.

To achieve this, the authors suggest using recurrent layers (like RNNs, GRUs, or LSTMs) or Transformer networks. These types of architectures are designed to process sequential data and can capture dependencies between frames, which is crucial for understanding motion and context in videos.

**Real-time Processing Capabilities:** The document emphasizes the need to support real-time surveillance systems.

This implies that the model needs to be optimized for speed and efficiency.

Future enhancements could involve model compression techniques, hardware acceleration (e.g., using GPUs or specialized hardware), or algorithmic optimizations to reduce the computational load.

**Dataset Enrichment:**

The model's performance is highly dependent on the quality and diversity of the training data. The authors suggest expanding the dataset, which could mean including more videos, more diverse scenarios (e.g., different lighting conditions, camera angles, environments), and potentially even new types of crimes.

This would help the model generalize better and perform more robustly in real-world settings.

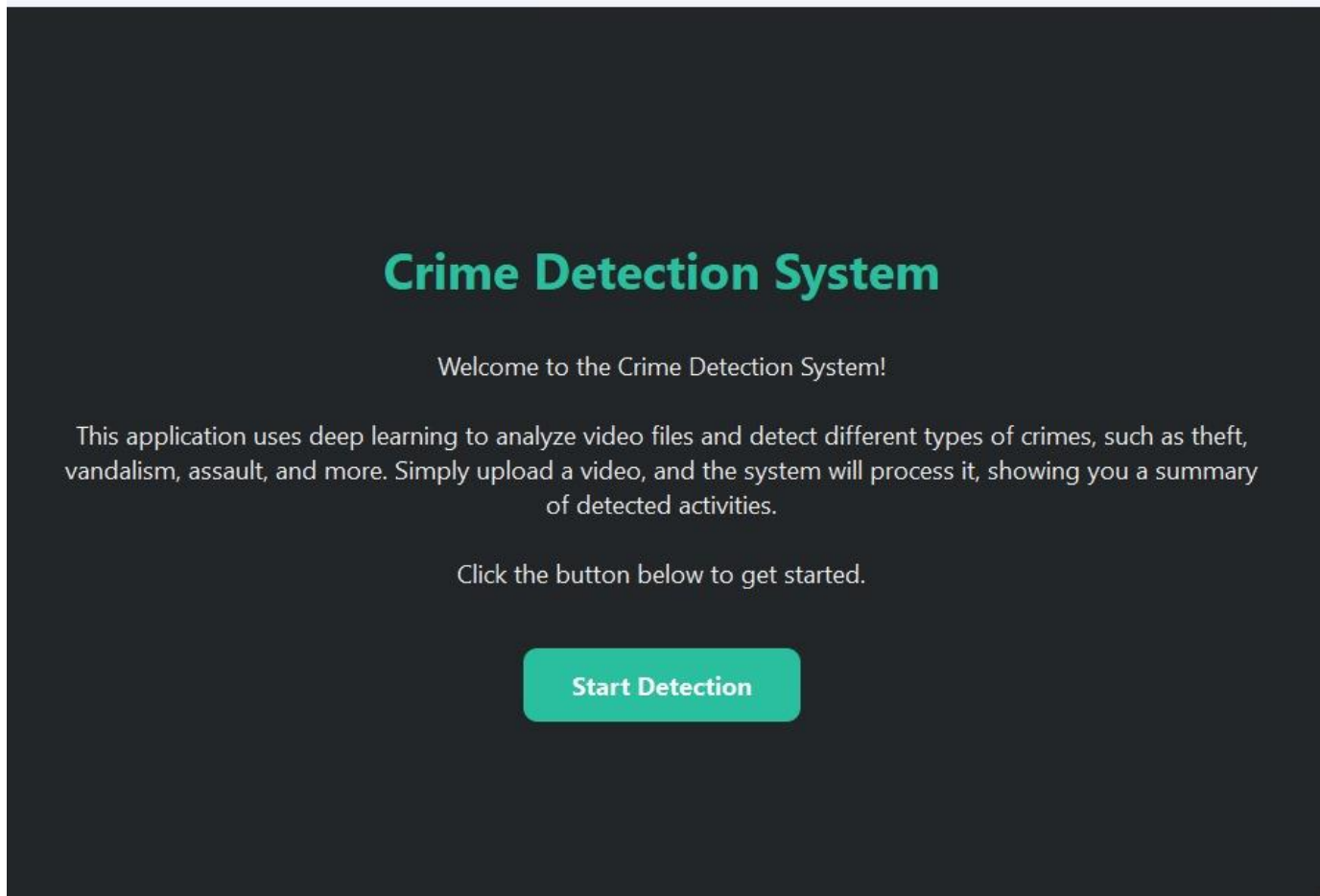
Hybrid Approaches:

The literature review section of the document highlights the use of hybrid models (e.g., CNN-RNN) in other studies.

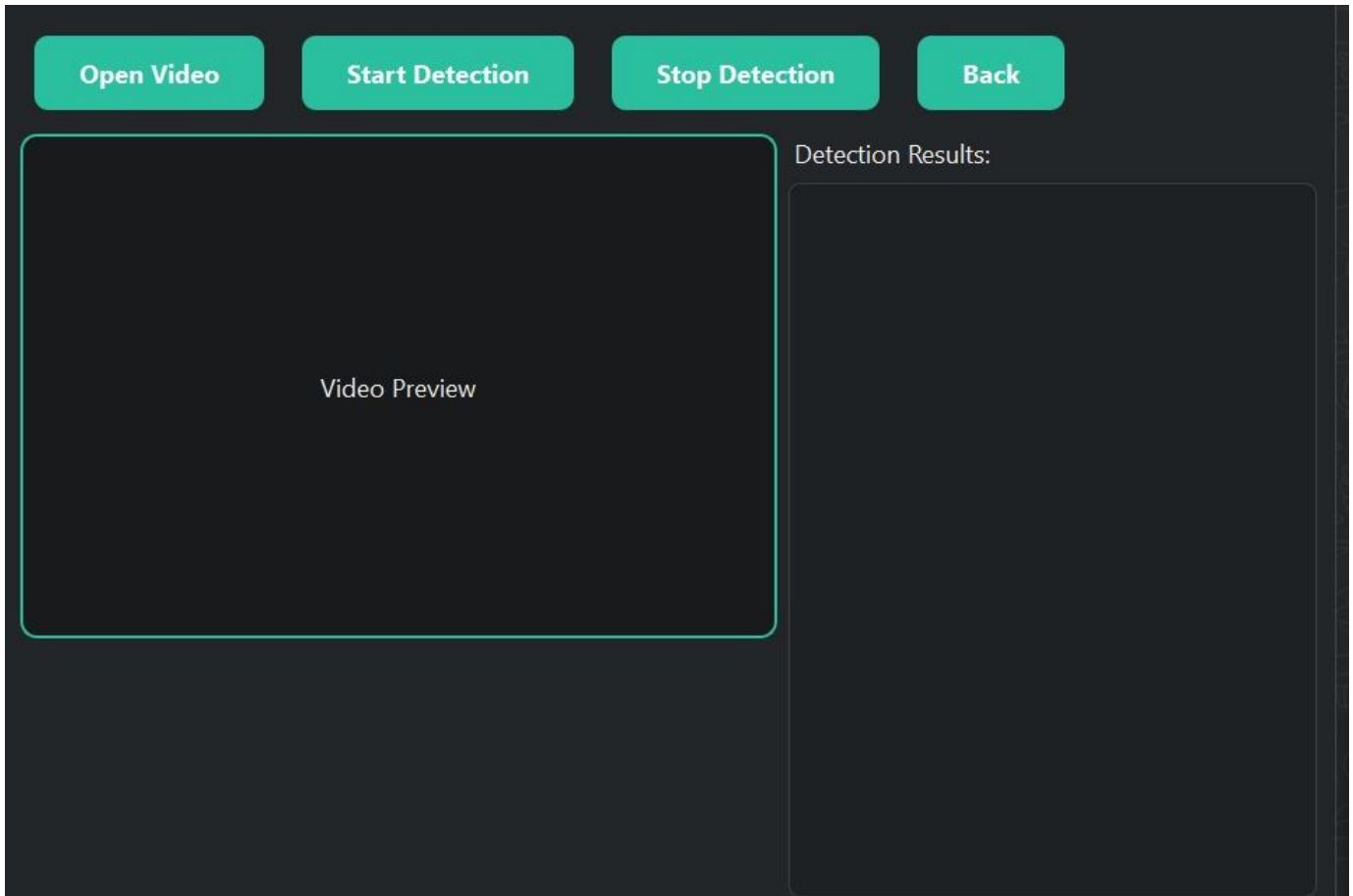
Future enhancements could explore combining different deep learning architectures or integrating other types of data (e.g., audio, metadata) to further improve the model's accuracy and reliability.

## CHAPTER 6

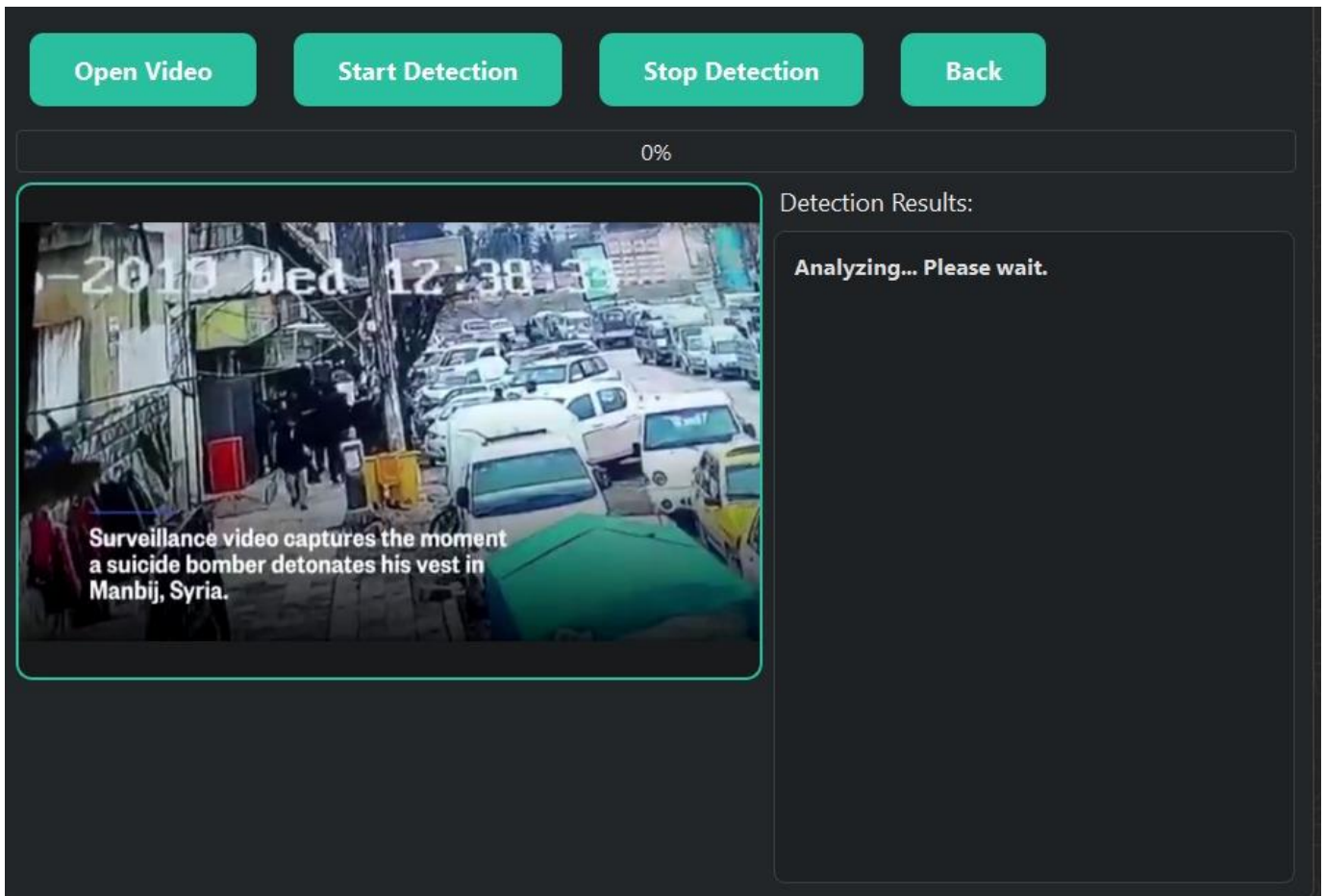
### WEB INTERFACE



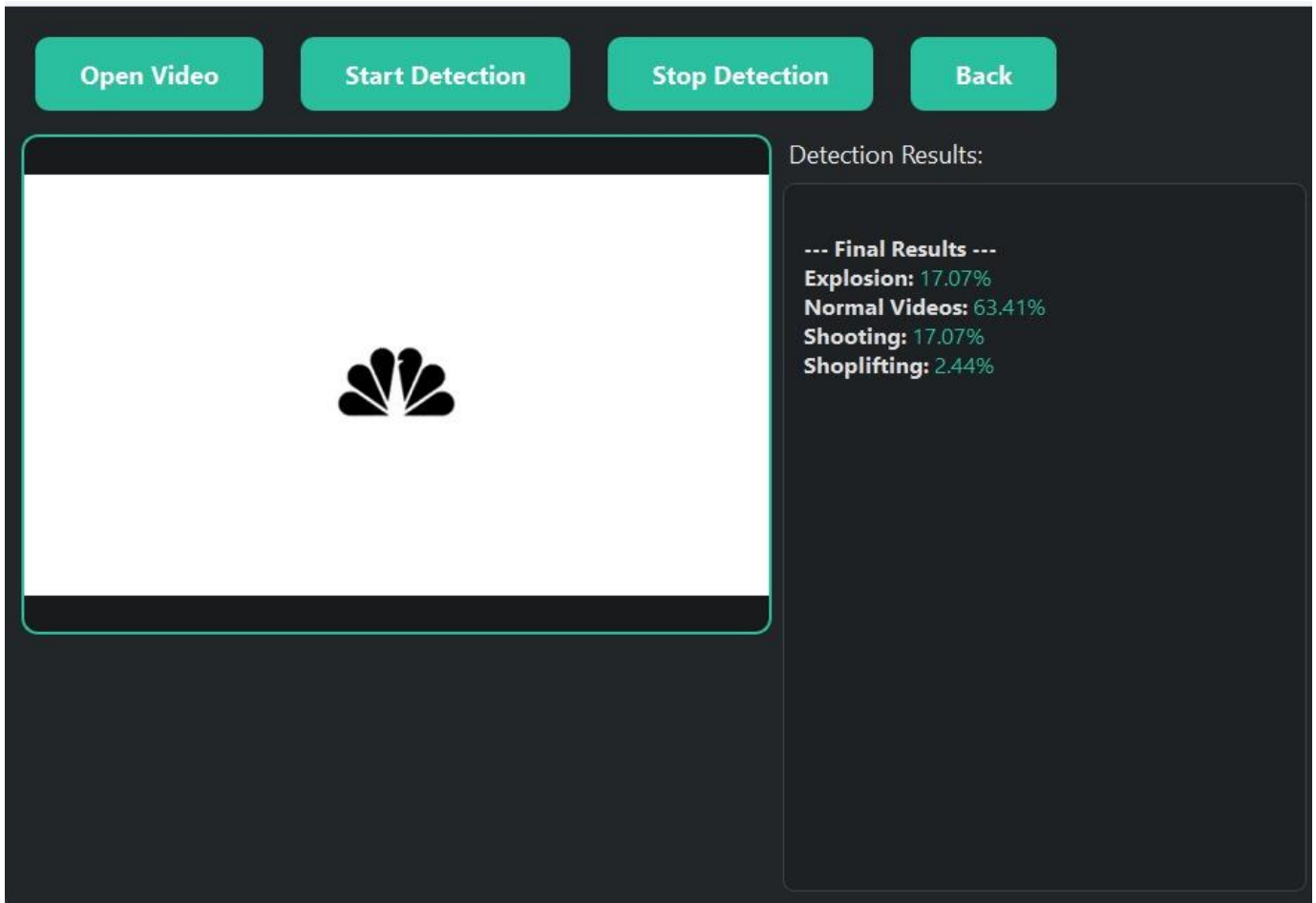
Img 6.1.1



Img 6.1.2



Img 6.1.3



Img 6.1.4

## Appendices

### "densenet", "vgg16", "resnet50" Models

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc,
precision_recall_curve
from sklearn.preprocessing import label_binarize
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import DenseNet121, VGG16, ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,
ModelCheckpoint
from tensorflow.keras.regularizers import l2
# ===== PARAMETERS =====
train_dir = "C:/Users/narendra/Downloads/archive_extracted/Train"
test_dir = "C:/Users/narendra/Downloads/archive_extracted/Test"
IMG_HEIGHT, IMG_WIDTH = 64, 64
BATCH_SIZE, EPOCHS, SEED = 32, 20, 42
IMG_SHAPE = (IMG_HEIGHT, IMG_WIDTH, 3)
# ===== DATA GENERATORS =====
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    zoom_range=0.3,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
test_datagen = ImageDataGenerator(rescale=1./255)

train_set = train_datagen.flow_from_directory(train_dir, target_size=(IMG_HEIGHT,
IMG_WIDTH),
    batch_size=BATCH_SIZE, class_mode='categorical', subset='training', seed=SEED)
val_set = train_datagen.flow_from_directory(train_dir, target_size=(IMG_HEIGHT,
IMG_WIDTH),
    batch_size=BATCH_SIZE, class_mode='categorical', subset='validation',
    seed=SEED)
test_set = test_datagen.flow_from_directory(test_dir, target_size=(IMG_HEIGHT,
IMG_WIDTH),
```



```

    batch_size=BATCH_SIZE, class_mode='categorical', shuffle=False)

class_names = list(test_set.class_indices.keys())

# ===== MODEL CREATION =====
def create_model(model_type):
    if model_type == "densenet":
        base = DenseNet121(include_top=False, weights='imagenet',
input_shape=IMG_SHAPE)
        freeze = 149
    elif model_type == "vgg16":
        base = VGG16(include_top=False, weights='imagenet', input_shape=IMG_SHAPE)
        freeze = 15
    elif model_type == "resnet50":
        base = ResNet50(include_top=False, weights='imagenet',
input_shape=IMG_SHAPE)
        freeze = 50
    else:
        raise ValueError("Unsupported model type")

    for layer in base.layers[:freeze]:
        layer.trainable = False

    model = Sequential([
        base,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu', kernel_regularizer=l2(0.002)),
        BatchNormalization(),
        Dropout(0.6),
        Dense(256, activation='relu', kernel_regularizer=l2(0.002)),
        BatchNormalization(),
        Dropout(0.6),
        Dense(len(class_names), activation='softmax')
    ])
    return model

# ===== VISUALIZATIONS =====
def plot_history(history, model_name):
    # Accuracy and Loss
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Val')
    plt.title(f"{model_name} Accuracy")
    plt.legend()

    plt.subplot(1, 2, 2)

```

```

plt.plot(history.history['loss'], label='Train')
plt.plot(history.history['val_loss'], label='Val')
plt.title(f'{model_name} Loss')
plt.legend()
plt.show()

# Learning rate
if 'lr' in history.history:
    plt.plot(history.history['lr'])
    plt.title(f'{model_name} Learning Rate Schedule')
    plt.show()

# Highlight min val loss
min_epoch = np.argmin(history.history['val_loss']) + 1
plt.plot(history.history['val_loss'], label="Val Loss")
plt.axvline(min_epoch - 1, color='red', linestyle='--', label=f"Min Loss @ {min_epoch}")
plt.title(f'{model_name} Min Val Loss')
plt.legend()
plt.show()

def plot_metrics(y_true, y_pred, model_name):
    y_pred_class = np.argmax(y_pred, axis=1)
    y_true_bin = label_binarize(y_true, classes=np.arange(len(class_names)))

    # Confusion Matrix
    cm = confusion_matrix(y_true, y_pred_class)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names,
yticklabels=class_names, cmap="Blues")
    plt.title(f'{model_name} Confusion Matrix')
    plt.show()

    # Classification Report
    print(f'{model_name} Classification Report:')
    print(classification_report(y_true, y_pred_class, target_names=class_names))

    # ROC Curve
    for i in range(len(class_names)):
        fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_pred[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'{class_names[i]} (AUC={roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.title(f'{model_name} ROC Curves')
    plt.legend()
    plt.show()

```

```

# Precision-Recall
for i in range(len(class_names)):
    precision, recall, _ = precision_recall_curve(y_true_bin[:, i], y_pred[:, i])
    plt.plot(recall, precision, label=class_names[i])
plt.title(f"{model_name} Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.show()

# Per-class accuracy
per_class_acc = cm.diagonal() / cm.sum(axis=1)
plt.bar(class_names, per_class_acc)
plt.xticks(rotation=45)
plt.title(f"{model_name} Per-Class Accuracy")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.show()

# Misclassification heatmap
misclassified = cm.copy()
np.fill_diagonal(misclassified, 0)
plt.figure(figsize=(8, 6))
sns.heatmap(misclassified, annot=True, fmt='d', cmap='Reds',
xticklabels=class_names, yticklabels=class_names)
plt.title(f"{model_name} Misclassifications")
plt.show()

# ===== TRAINING FUNCTION =====
def train_and_evaluate(model_type):
    print(f"\n\n===== Training {model_type.upper()} =====")
    model = create_model(model_type)
    model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    early = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience=2, factor=0.2,
min_lr=1e-6)
    chkpt = ModelCheckpoint(f"best_{model_type}.h5", save_best_only=True)

    history = model.fit(train_set, validation_data=val_set, epochs=EPOCHS,
callbacks=[early, reduce_lr, chkpt])
    model.save(f"final_{model_type}_model.h5")
    plot_history(history, model_type)
# Test predictions
test_loss, test_acc = model.evaluate(test_set)

```

```

print(f"{model_type.upper()} Test Accuracy: {test_acc * 100:.2f}%")
y_pred = model.predict(test_set)
y_true = test_set.classes
plot_metrics(y_true, y_pred, model_type)
# ===== EXECUTE ALL MODELS =====
for m in ["densenet", "vgg16", "resnet50"]:
    train_and_evaluate(m)
MobileNet model
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Conv2D, BatchNormalization, Dropout, Flatten,
Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import label_binarize
from sklearn.manifold import TSNE
import os
from glob import glob
# Define paths and image size
IMAGE_SIZE = [224, 224]
train_path = r"C:\Users\narendra\Downloads\archive_extracted\Train"
valid_path = r"C:\Users\narendra\Downloads\archive_extracted\Test"
folders = glob(train_path + '/*')
n_classes = len(folders)
# Load base model
base_model = MobileNetV2(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)
for layer in base_model.layers[:-10]:
    layer.trainable = False
# Add custom layers
x = base_model.output
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.4)(x)
prediction = Dense(n_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=prediction)

```

```

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

# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2,
                                horizontal_flip=True, rotation_range=20,
                                width_shift_range=0.2, height_shift_range=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)

train_set = train_datagen.flow_from_directory(train_path, target_size=IMAGE_SIZE,
                                             batch_size=32, class_mode='categorical')
test_set = test_datagen.flow_from_directory(valid_path, target_size=IMAGE_SIZE,
                                           batch_size=32, class_mode='categorical',
                                           shuffle=False)

# Train the model
history = model.fit(train_set, validation_data=test_set, epochs=10,
                    steps_per_epoch=len(train_set), validation_steps=len(test_set))

# Save the model
model.save("crime_detection_model_optimized_1.h5")

# Reload for evaluation
model = load_model("crime_detection_model_optimized_1.h5")

# Evaluate
loss, acc = model.evaluate(test_set)
print(f"Test loss: {loss:.4f}, accuracy: {acc:.4f}")
# Predictions
predictions = model.predict(test_set)
y_pred = np.argmax(predictions, axis=1)
y_true = test_set.classes
class_names = list(test_set.class_indices.keys())
# Confusion Matrix
conf_mat = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(12, 10))
sns.heatmap(conf_mat, annot=True, fmt="d", xticklabels=class_names,
yticklabels=class_names, cmap="Blues")
plt.title("Confusion Matrix")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.show()
# Classification Report
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=class_names))
# Accuracy & Loss Curves
plt.figure(figsize=(14, 5))

```

```

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title("Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.show()
# ROC Curves
y_true_bin = label_binarize(y_true, classes=range(n_classes))
fpr = {}
tpr = {}
roc_auc = {}
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true_bin[:, i], predictions[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'{class_names[i]} (AUC = {roc_auc[i]:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
# Show some sample predictions
import random
indices = random.sample(range(len(y_pred)), 10)
plt.figure(figsize=(20, 8))
for i, idx in enumerate(indices):
    img_path = test_set.filepaths[idx]
    img = tf.keras.utils.load_img(img_path, target_size=IMAGE_SIZE)
    plt.subplot(2, 5, i+1)
    plt.imshow(img)
    plt.title(f"True: {class_names[y_true[idx]]}\nPred: {class_names[y_pred[idx]]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
# t-SNE projection
features_model = Model(inputs=model.input, outputs=model.layers[-2].output)

```

```

features = features_model.predict(test_set)
tsne = TSNE(n_components=2, random_state=42)
tsne_result = tsne.fit_transform(features)
plt.figure(figsize=(10, 8))
scatter = plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=y_true, cmap='tab20', alpha=0.6)
plt.title("t-SNE Projection of Features")
plt.colorbar(scatter, ticks=range(n_classes), label='Class Index')
plt.show()

```

```

# ROC Curves
y_true_bin = label_binarize(y_true, classes=range(n_classes))
fpr = {}
tpr = {}
roc_auc = {}
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true_bin[:, i], predictions[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f"{class_names[i]} (AUC = {roc_auc[i]:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()

```

## Video Frame Extraction

```
import os
import cv2
import numpy as np
from collections import Counter
from datetime import timedelta
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input

def extract_frames(video_path, output_path, interval_sec=1):
    vidcap = cv2.VideoCapture(video_path)
    fps = int(vidcap.get(cv2.CAP_PROP_FPS))
    interval = int(fps * interval_sec)
    frame_count = int(vidcap.get(cv2.CAP_PROP_FRAME_COUNT))
    duration_sec = frame_count / fps

    os.makedirs(output_path, exist_ok=True)

    count = 0
    timestamps = []

    success, img = vidcap.read()
    while success:
        timestamp = str(timedelta(seconds=int(count / fps)))
        frame_name = f"frame{count}.jpg"
        frame_path = os.path.join(output_path, frame_name)
        cv2.imwrite(frame_path, img)
        timestamps.append((frame_name, timestamp))

        count += interval
        vidcap.set(cv2.CAP_PROP_POS_FRAMES, count)
        success, img = vidcap.read()

    vidcap.release()
    print(f"Extracted {len(timestamps)} frames over {duration_sec:.2f} seconds.")
    return timestamps

def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    return preprocess_input(x)

def predict_frame(model, img_path, classes):
    x = preprocess_image(img_path)
    predictions = model.predict(x)
```



```

predicted_class_index = np.argmax(predictions[0])
return classes[predicted_class_index], predictions[0]

def save_summary(counts, total, output_file="crime_summary.txt"):
    with open(output_file, "w") as f:
        f.write("--- Crime Detection Summary ---\n")
        for crime_class, count in counts.items():
            percentage = (count / total) * 100
            f.write(f"{crime_class}: {percentage:.2f}%\n")
    print(f"Summary saved to {output_file}")

def main():
    model_path = "crime_detection_model_optimized_1.h5"
    video_path = r"C:\Users\narendra\Downloads\video1.mp4"
    output_dir = r"C:\Users\narendra\Downloads\extracted_frames"

    crime_classes = [
        'Abuse', 'Arrest', 'Arson', 'Assault', 'Burglary', 'Explosion', 'Fighting',
        'Normal Videos', 'Road Accidents', 'Robbery', 'Shooting', 'Shoplifting',
        'Stealing', 'Vandalism'
    ]

    print("Loading model...")
    model = load_model(model_path)

    print("Extracting frames...")
    timestamped_frames = extract_frames(video_path, output_dir)

    print("Running predictions...")
    predictions_list = []
    detailed_results = []

    for filename, timestamp in timestamped_frames:
        img_path = os.path.join(output_dir, filename)
        if os.path.exists(img_path):
            predicted_class, _ = predict_frame(model, img_path, crime_classes)
            predictions_list.append(predicted_class)
            detailed_results.append((timestamp, filename, predicted_class))
            print(f"[{timestamp}] {filename} => {predicted_class}")

    counts = Counter(predictions_list)
    total = len(predictions_list)

    print("\n--- Final Results ---")
    for crime_class, count in counts.items():
        percentage = (count / total) * 100
        print(f"{crime_class} -- {percentage:.2f}%")

```

```

save_summary(counts, total, "crime_summary.txt")

# Optionally export detailed results to CSV
csv_path = "detailed_results.csv"
with open(csv_path, "w") as f:
    f.write("Timestamp,Frame,Predicted Class\n")
    for timestamp, frame, label in detailed_results:
        f.write(f"{timestamp},{frame},{label}\n")
print(f"Detailed results saved to {csv_path}")

if __name__ == "__main__":
    main()

```

## Graphical User Interface

```
import sys
import os
import cv2
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input

from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QLabel, QPushButton, QVBoxLayout,
    QHBoxLayout, QFileDialog, QTextEdit, QMessageBox, QFrame, QProgressBar, QStackedWidget
)
from PyQt5.QtGui import QPixmap, QImage, QIcon
from PyQt5.QtCore import QTimer, Qt
import qdarktheme

# ----- Home Page Widget -----
class HomePage(QWidget):
    def __init__(self, go_to_detection):
        super().__init__()
        layout = QVBoxLayout()
        title = QLabel("Crime Detection System")
        title.setAlignment(Qt.AlignCenter)
        title.setStyleSheet("font-size: 36px; color: #2abf9e; font-weight: bold; margin-top: 40px;")

        desc = QLabel(
            "Welcome to the Crime Detection System!\n\n"
            "This application uses deep learning to analyze video files and detect different types of crimes, "
            "such as theft, vandalism, assault, and more. Simply upload a video, and the system will process it, "
            "showing you a summary of detected activities.\n\n"
            "Click the button below to get started."
        )
        desc.setWordWrap(True)
        desc.setAlignment(Qt.AlignCenter)
        desc.setStyleSheet("font-size: 18px; color: #f0f0f0; margin: 24px;")

        start_btn = QPushButton("Start Detection")
        start_btn.setObjectName("mainBtn")
        start_btn.setFixedWidth(220)
        start_btn.clicked.connect(go_to_detection)

        layout.addStretch()
        layout.addWidget(title)
```

```

layout.addWidget(desc)
layout.addWidget(start_btn, alignment=Qt.AlignCenter)
layout.addStretch()
self.setLayout(layout)

# ----- Detection Page Widget -----
class DetectionPage(QWidget):
    def __init__(self, go_home):
        super().__init__()
        self.model = load_model("crime_detection_model_optimized_1.h5")
        self.crime_classes = [
            'Abuse', 'Arrest', 'Arson', 'Assault', 'Burglary', 'Explosion', 'Fighting',
            'Normal Videos', 'Road Accidents', 'Robbery', 'Shooting', 'Shoplifting', 'Stealing', 'Vandalism'
        ]
        self.video_path = None
        self.cap = None
        self.timer = QTimer()
        self.timer.timeout.connect(self.next_frame)
        self.frame_rate = 1
        self.frame_counter = 0
        self.predictions_list = []
        self.total_frames = 0

        self.video_label = QLabel("Video Preview")
        self.video_label.setFixedSize(540, 360)
        self.video_label.setAlignment(Qt.AlignCenter)
        self.video_label.setStyleSheet("background-color: #181a1b; border-radius: 12px; border: 2px solid #2abf9e;")

        self.info_text = QTextEdit()
        self.info_text.setReadOnly(True)

        self.open_btn = QPushButton("Open Video")
        self.open_btn.setObjectName("mainBtn")
        self.open_btn.clicked.connect(self.open_video)

        self.start_btn = QPushButton("Start Detection")
        self.start_btn.setObjectName("mainBtn")
        self.start_btn.clicked.connect(self.start_detection)
        self.start_btn.setEnabled(False)

        self.stop_btn = QPushButton("Stop Detection")
        self.stop_btn.setObjectName("mainBtn")
        self.stop_btn.clicked.connect(self.stop_detection)
        self.stop_btn.setEnabled(False)

```

```

self.export_btn = QPushButton("Export Results")
self.export_btn.setObjectName("mainBtn")
self.export_btn.clicked.connect(self.export_results)
self.export_btn.setEnabled(False)

self.back_btn = QPushButton("Back")
self.back_btn.setObjectName("mainBtn")
self.back_btn.clicked.connect(go_home)

self.progress = QProgressBar()
self.progress.setValue(0)
self.progress.setVisible(False)

btn_layout = QHBoxLayout()
btn_layout.addWidget(self.open_btn)
btn_layout.addWidget(self.start_btn)
btn_layout.addWidget(self.stop_btn)
btn_layout.addWidget(self.export_btn)
btn_layout.addWidget(self.back_btn)

video_layout = QVBoxLayout()
video_layout.addWidget(self.video_label)

right_layout = QVBoxLayout()
right_layout.addWidget(QLabel("Detection Results:"))
right_layout.addWidget(self.info_text)

main_content = QHBoxLayout()
main_content.addLayout(video_layout, 2)
main_content.addLayout(right_layout, 3)

main_layout = QVBoxLayout()
main_layout.addLayout(btn_layout)
main_layout.addWidget(self.progress)
main_layout.addLayout(main_content)
self.setLayout(main_layout)

def open_video(self):
    fname, _ = QFileDialog.getOpenFileName(self, "Select Video File", "", "Video Files (*.mp4 *.avi
*.mov *.mkv)")
    if fname:
        self.video_path = fname
        self.cap = cv2.VideoCapture(self.video_path)
        self.start_btn.setEnabled(True)
        self.info_text.clear()
        self.predictions_list.clear()
        self.frame_counter = 0

```

```

        self.total_frames = int(self.cap.get(cv2.CAP_PROP_FRAME_COUNT))
        ret, frame = self.cap.read()
        if ret:
            self.show_frame(frame)
            self.cap.release()

def start_detection(self):
    if not self.video_path:
        QMessageBox.warning(self, "No Video", "Please select a video file first.")
        return
    self.cap = cv2.VideoCapture(self.video_path)
    self.timer.start(1000 // self.frame_rate)
    self.start_btn.setEnabled(False)
    self.stop_btn.setEnabled(True)
    self.export_btn.setEnabled(False)
    self.info_text.clear()
    self.info_text.setText("<b>Analyzing... Please wait.</b>")
    self.progress.setVisible(True)
    self.progress.setValue(0)
    self.predictions_list.clear()
    self.frame_counter = 0

def stop_detection(self):
    self.timer.stop()
    if self.cap:
        self.cap.release()
    self.start_btn.setEnabled(True)
    self.stop_btn.setEnabled(False)
    self.export_btn.setEnabled(True)
    self.progress.setVisible(False)
    self.show_summary()

def next_frame(self):
    if self.cap is None:
        return
    fps = int(self.cap.get(cv2.CAP_PROP_FPS))
    self.cap.set(cv2.CAP_PROP_POS_FRAMES, self.frame_counter * fps)
    ret, frame = self.cap.read()
    if not ret:
        self.stop_detection()
        return
    pred = self.predict_frame(frame)
    self.predictions_list.append(pred)
    self.show_frame_with_prediction(frame, pred)
    self.frame_counter += 1
    progress_percent = int((self.frame_counter / max(1, self.total_frames)) * 100)
    self.progress.setValue(progress_percent)

```

```

if self.frame_counter * fps >= self.total_frames:
    self.stop_detection()

def show_frame(self, frame):
    rgb_image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    h, w, ch = rgb_image.shape
    bytes_per_line = ch * w
    qt_img = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format_RGB888)
    pixmap = QPixmap.fromImage(qt_img).scaled(self.video_label.width(), self.video_label.height(),
Qt.KeepAspectRatio)
    self.video_label.setPixmap(pixmap)

def show_frame_with_prediction(self, frame, prediction):
    overlay = frame.copy()
    cv2.putText(overlay, f"Detected: {prediction}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,
255, 0), 2)
    self.show_frame(overlay)

def predict_frame(self, frame):
    img = cv2.resize(frame, (224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    predictions = self.model.predict(x)
    predicted_class_index = np.argmax(predictions[0])
    return self.crime_classes[predicted_class_index]

def show_summary(self):
    self.info_text.clear()
    self.progress.setVisible(False)
    if not self.predictions_list:
        self.info_text.append("No predictions made.")
        return
    crime_class_counts = Counter(self.predictions_list)
    total_predictions = len(self.predictions_list)
    self.info_text.append("<br><b>--- Final Results ---</b>")
    for crime_class, count in crime_class_counts.items():
        percentage = (count / total_predictions) * 100
        self.info_text.append(f"<b>{crime_class}</b> <span
style='color:#2abf9e>{percentage:.2f}%</span>")
    self.plot_prediction_chart(crime_class_counts)

def plot_prediction_chart(self, counts):
    classes = list(counts.keys())
    values = list(counts.values())
    plt.figure(figsize=(10, 5))
    plt.barh(classes, values, color='#2abf9e')

```

```

plt.xlabel('Frequency')
plt.title('Crime Class Predictions')
plt.tight_layout()
plt.savefig('prediction_chart.png')
plt.close()
self.info_text.append("<br><i>Prediction chart saved as 'prediction_chart.png'</i>")

def export_results(self):
    if not self.predictions_list:
        return
    save_path, _ = QFileDialog.getSaveFileName(self, "Save Results", "results.txt", "Text Files (*.txt)")
    if save_path:
        with open(save_path, 'w') as f:
            f.write("--- Crime Detection Results ---\n")
            counts = Counter(self.predictions_list)
            for label, count in counts.items():
                f.write(f"{label}: {count}\n")
        QMessageBox.information(self, "Export Complete", "Results saved successfully.")

def predict_frame(self, frame):
    img = cv2.resize(frame, (224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    predictions = self.model.predict(x)
    predicted_class_index = np.argmax(predictions[0])
    return self.crime_classes[predicted_class_index]

def show_summary(self):
    self.info_text.clear()
    self.progress.setVisible(False)
    if not self.predictions_list:
        self.info_text.append("No predictions made.")
        return
    crime_class_counts = Counter(self.predictions_list)
    total_predictions = len(self.predictions_list)
    self.info_text.append("<br><b>--- Final Results ---</b>")
    for crime_class, count in crime_class_counts.items():
        percentage = (count / total_predictions) * 100
        self.info_text.append(f"<b>{crime_class}</b> <span
style='color:#2abf9e'>{percentage:.2f}%</span>")
    self.plot_prediction_chart(crime_class_counts)

def plot_prediction_chart(self, counts):
    classes = list(counts.keys())
    values = list(counts.values())
    plt.figure(figsize=(10, 5))

```



```

plt.barh(classes, values, color='#2abf9e')
plt.xlabel('Frequency')
plt.title('Crime Class Predictions')
plt.tight_layout()
plt.savefig('prediction_chart.png')
plt.close()
self.info_text.append("<br><i>Prediction chart saved as 'prediction_chart.png'</i>")

```

# ----- Main Window -----

```

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Crime Detection - Modern UI")
        self.setMinimumSize(950, 650)
        qdarktheme.setup_theme("dark")
        self.setStyleSheet("""
            QWidget {
                font-family: 'Segoe UI', Arial, sans-serif;
                font-size: 16px;
                background-color: #232629;
                color: #f0f0f0;
            }
            QPushButton#mainBtn {
                background-color: #2abf9e;
                color: #fff;
                border-radius: 10px;
                padding: 14px 32px;
                font-size: 18px;
                font-weight: bold;
                margin: 10px;
                border: none;
            }
            QPushButton#mainBtn:hover {
                background-color: #249e81;
            }
            QLabel {
                font-size: 18px;
                color: #f0f0f0;
            }
            QTextEdit {
                background-color: #1e2124;
                color: #e0e0e0;
                border-radius: 8px;
                font-size: 16px;
                padding: 10px;
            }

```

```

    """
    self.stack = QStackedWidget()
    self.home_page = HomePage(self.show_detection_page)
    self.detection_page = DetectionPage(self.show_home_page)
    self.stack.addWidget(self.home_page)
    self.stack.addWidget(self.detection_page)
    self.setCentralWidget(self.stack)
    self.stack.setCurrentIndex(0)

def show_detection_page(self):
    self.stack.setCurrentIndex(1)

def show_home_page(self):
    self.stack.setCurrentIndex(0)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

```

## REFERENCES

- [1] Islam, M. M., Hossain, M. A., Sarker, I. H., & Roy, S. (2022). Deep learning for criminal identification: Leveraging CNNs and RNNs on biometric and surveillance data. *Pattern Recognition Letters*, 157, 121-128.
- [2] Kumar, A., Singh, V., & Bhatnagar, A. (2013). Artificial neural network for hand-drawn facial sketch recognition. *International Journal of Computer Applications*, 68(19), 1-6.
- [3] Amir, M., Khan, M. A., Asghar, M. N., & Khan, A. (2023). Hybrid GRU-CNN model for crime classification using spatial and temporal features. *Applied Intelligence*, 53(15), 17853-17867.
- [4] Sharma, R., Gupta, S., & Kapoor, S. (2021). Analyzing crime prediction using machine learning and deep learning models. *International Journal of Information Technology*, 13(5), 2047-2056.
- [5] Amir, M., Khan, M. A., Asghar, M. N., & Khan, A. (2023). An enhanced crime classification framework combining autoencoders, GRU, and CNN networks. *Expert Systems with Applications*, 213, 118817.
- [6] Nguyen, H., Nguyen, Q., & Tran, V. (2021). A CNN-GRU hybrid model for multivariate time series forecasting. *Applied Soft Computing*, 113, 107842.
- [7] Patel, K., Shah, M., & Thakkar, P. (2019). Crime prediction model using classification techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 9(5), 1-5.
- [8] Thakur, S., & Sharma, A. (2019). Criminal identification system based on facial recognition using deep learning. *International Journal of Innovative Technology and Exploring Engineering*, 8(12), 4382-4386.
- [9] Yadav, S., & Sharma, M. (2022). An intelligent system for early crime detection using hybrid machine learning algorithms. *International Journal of System Assurance Engineering and Management*, 13(Suppl 1), 355-363.
- [10] Köbis, N. C., Verschuere, B., Shalvi, S., & Meyer, J.-U. H. (2021). Catching a liar: Machine learning to detect deception from behavioral data. *Psychological Science*, 32(9), 1437-1452.
- [11] Stalidis, P., Semertzidis, T., & Daras, P. (2021). Examining deep learning architectures for crime classification and prediction. *Forecasting*, 3(4), 741-762.
- [12] Mandalapu, V., Elluri, L., Vyas, P., & Roy, N. (2023). Crime Prediction Using Machine

Learning and Deep Learning: A Systematic Review and Future Directions. arXiv preprint arXiv:2303.16310.

[13] Selvakumari, S. J., & Peter, V. J. (2024). Crime Classification Using GRU, CNN And Autoencoder Techniques. *Educational Administration: Theory and Practice*, 30(5), 2950–2964.

[14] Venkatesh, N. V., Kumar, P. S., Prince, G. S., & Karthikeyan, M. (2023). Crime Forecasting Using Historical Crime Location Using CNN-Based Images Classification Mechanism. In B. K. Pandey, N. الحالى, & S. Goel (Eds.), *Handbook of Research on Thrust Technologies' Effect on Image Processing* (pp. 206-221). IGI Global.

[15] Alkaff, M., Mustamin, N. F., & Firdaus, G. A. A. (2022). Prediction of Crime Rate in Banjarmasin City Using RNN-GRU Model. *International Journal of Intelligent Systems and Applications in Engineering*, 10(4), 232-238.

[16] Chun, S. A., Paturu, V. A., Yuan, S., Pathak, R., Atluri, V., & Adam, N. R. (2020). Crime prediction model using deep neural networks. In *Proceedings of the 20th Annual International Conference on Digital Government Research* (pp. 406-415)