

Crime Scene Analytics using Deep Learning

Submitted By

ManjurMahamad Kovadiya

24MCD008



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY, INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY
AHMEDABAD-382481**

May 2026

Crime Scene Analytics using Deep Learning

Major Project - II

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (Data Science)

Submitted By

ManjurMahamad Kovadiya

(24MCD008)

Guided By

Dr. Sharada Valiveti



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF TECHNOLOGY, INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

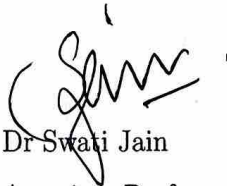
May 2026

Certificate

This is to certify that the major project entitled “**Crime Scene Analytics using Deep Learning**” submitted by **ManjurMahamad Kovadiya (Roll No: 24MCD008)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Data Science) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.



Dr Sharada Valiveti
Guide & Associate Professor,
CSE Department,
School of Technology,
Nirma University, Ahmedabad.



Dr Swati Jain
Associate Professor,
Coordinator M.Tech - CSE (Data Science)
School of Technology,
Nirma University, Ahmedabad



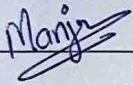
Dr Ankit Thakkar
Professor and I/c Head,
CSE Department,
School of Technology,
Nirma University, Ahmedabad.



Dr Himanshu Soni
Director,
School of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, ManjurMahamad Kovadiya, Roll. No. 24MCD008, give undertaking that the Major Project entitled "Crime Scene Analytics using Deep Learning" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science & Engineering (Data Science) of School of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.



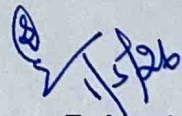
Signature of Student

Date:

1/5/26

Place:

~~Hammatoo~~
Ahmedabad



Endorsed by

Dr Sharada Valiveti

(Signature of Guide)

Acknowledgements

I would like to take this opportunity to sincerely thank everyone who has helped and supported me throughout the journey of working on my project, Crime Scene Analytics using Deep Learning. First and foremost, I am truly grateful to my project guide, **Dr. Sharada Valiveti** for their valuable guidance, constant support, and encouragement. Their knowledge and insights have been very helpful in shaping the project in the right direction. Lastly, I would like to thank all the faculty members who, directly or indirectly, influenced my learning and inspired me to keep improving. This project wouldn't have been possible without the support of all these wonderful people, and I am truly thankful to each one of them. Last but not least, my humble thanks to almighty God.

ManjurMahamad Kovadiya

24MCD008

Abstract

Forensic crime scene investigation depends on accurately finding critical evidence such as weapons and bloodstains from images. However, manual analysis is still slow, tiring, and prone to inconsistency, especially when the evidence is small or captured under poor lighting and noisy conditions. To solve this problem, this work presents an from model training to interface deep learning pipeline based on YOLOv8 for automated detection, with a strong focus on forensic aware and data-centric preprocessing. Two publicly available datasets containing weapon and bloodstain images are combined and enhance through annotation validation, corruption removal, and class balancing, resulting in a clean dataset of 1,283 images. This dataset is further enhanced using a multi-stage pipeline that includes denoising, targeted super-resolution for small objects, and domain-specific data augmentation techniques, expanding it to 3,601 images. Additionally, a DCGAN based synthetic data generator is used to improve dataset imbalance. The improvements in dataset quality takes to significant gains in model performance. The error rate is reduced from 4.8% to 0.6%, while detection accuracy improves, with mAP@50 increasing from 0.53 to 0.61 and small-object recall rising from 0.62 to 0.76. From all the model performance this model achieves a mAP@50:95 of 0.6146. Then model is integrated through a Streamlit web interface and an Android application, enabling real-time detection of forensic evidence. Overall, this study highlights the critical role of careful dataset preparation and enhancement in building reliable and effective AI assisted forensic analysis systems.

Abbreviations

YOLO	You Only Look Once
YOLOv8	You Only Look Once version 8
GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
Real-ESRGAN	Real-Enhanced Super-Resolution Generative Adversarial Networks
mAP	Mean Average Precision
mAP@50	Mean Average Precision at IoU threshold 0.50
mAP@50:95	mean Average Precision averaged over IoU thresholds 0.50 to 0.95
CNN	Convolutional Neural Network
IoU	Intersection over Union
FPS	Frames Per Second
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RGB	Red–Green–Blue (color format)
API	Application Programming Interface
Colab	Google Colaboratory
AI	Artificial Intelligence
DL	Deep Learning
COCO	Common Objects in Context
ROI	Region of Interest
NMS	Non-Maximum Suppression
BBox	Bounding Box

Contents

Certificate	iii
Statement of Originality	iv
Acknowledgements	v
Abstract	vi
Abbreviations	vii
List of Figures	xi
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Aim of the Project	2
1.4 Objectives	2
1.5 Scope and Limitations	3
1.6 Significance of the Work	4
2 Literature Survey	6
2.1 Data Augmentation for Small Object Detection	6
2.2 Super-Resolution for Small Object Detection	7
2.3 Contextual Reasoning in Crime Activity Analysis	10
2.3.1 Multi-Scale Feature Representation for Small Object Detection . .	12
2.3.2 Integrated Forensic Systems and AI-Based Devices	12
2.4 Literature Review Summary Table	13
3 Tools and Technologies	19
3.1 Programming Environment	19
3.1.1 Google Colab (Colaboratory)	19
3.2 Deep Learning Framework	20
3.2.1 Ultralytics YOLOv8	20
3.3 Application Development and Communication Layer	20
3.3.1 Android Studio	20
3.3.2 WebSocket Communication Protocol	21
3.4 Dataset Sources and Management	23
3.4.1 Roboflow	23
3.5 Key Python Libraries and Tools	23

4	Methodology	25
4.1	Workflow	25
4.2	Architecture Overview	28
5	Implementation	30
5.0.1	Dataset Construction and Merging	30
5.0.2	Targeted Super-Resolution Enhancement	31
5.0.3	Domain-Specific Data Augmentation	33
5.0.4	Synthetic Data Generation	34
5.0.5	YOLOv8 Training, Detection, and Interactive Forensic Interface .	39
5.0.6	Android Mobile Client and WebSocket-Based Field Interface . . .	42
5.0.7	Stage-wise Dataset Evolution and Quantitative Accounting	46
6	Results	49
6.0.1	Annotation Quality Analysis	49
6.0.2	Class Balance and Distribution	50
6.0.3	Dataset Object Scale Distribution and Small-Object Readiness . .	50
6.0.4	Image Quality and Feature Learning	51
6.0.5	Feature-Space Separability	51
6.0.6	Detection Performance Evaluation	52
6.0.7	Small-Object Detection Capability	52
6.0.8	Robustness to Noise and Poor Quality	52
7	Conclusion and Future Work	53
7.1	Conclusion	53
7.2	Future Work	54
	Bibliography	56

List of Figures

2.1	Copy-paste duplicates small objects to balance scale and boost their frequency [1].	7
2.2	Architecture Perceptual GAN: the generator enhances small-object features, and the discriminator checks realism and detection quality	8
2.3	Real-ESRGAN generator architecture [2]. Pixel-unshuffle is applied for $\times 2$ and $\times 1$ scaling to reduce spatial resolution while rearranging information into the channel dimension, enabling efficient computation.	9
2.4	U-Net-based discriminator with spectral normalization used in Real-ESRGAN [2], providing pixel-wise feedback for improved texture reconstruction and training stability.	9
2.5	Qualitative comparison of Real-ESRGAN [2] with existing super-resolution methods on real-world images. Real-ESRGAN demonstrates superior artifact suppression and texture restoration across diverse scenes (best viewed with zoom).	10
2.6	SPOT-CRIME system workflow. The pipeline consists of person detection, object correlation using attention mechanisms, and multi-frame tracking to identify suspicious behavior [3]	11
2.7	Illustration of multi-scale feature representations [4]. (a) Image pyramid, (b) single feature map baseline, (c) SSD-style hierarchical features, and (d) FPN-based top-down feature fusion.	12
2.8	Architecture of an AI-powered forensic system integrating sensors, edge processing, and cloud-based deep learning models for evidence analysis [5].	13
3.1	Google Colab Environment	19
3.2	YOLOv8 Architecture Overview <i>Source: What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector</i> https://arxiv.org/html/2408.15857v1	20
3.3	Android Studio Development Environment	21
3.4	WebSocket-Based Real-Time Communication Architecture	22
3.5	Roboflow (Open Source Public Dataset Platform)	23
4.1	Forensic-Grade Preprocessing and Dataset Construction Pipeline	25
4.2	Working Flow of Crime Scene Analytics System	28
5.1	Selective Real-ESRGAN enhancement applied to small-object regions before reintegration into the original image	32
5.2	Crime-scene-specific augmentation: original image (left) versus augmented sample (right)	33

5.3	DCGAN-generated synthetic weapon images (64×64 grayscale) after 500 epochs. Left: real weapon samples; Right: generated samples exhibiting recognizable weapon structures	35
5.4	YOLOv8 detection results on crime-scene images showing weapon and bloodstain localization with confidence scores	40
5.5	Visualization of spatial distance estimation between detected forensic objects based on bounding box centroids	41
5.6	Streamlit-based forensic analysis interface enabling model selection, crime-scene input upload, object detection, and distance visualization	42
5.7	Mobile forensic interface for the Crime Scene Analytics application.	45

Chapter 1

Introduction

1.1 Background

Crime scene investigation heavily depends on the accurate identification of physical evidence from photographs taken at the scene. Among the most critical forms of evidence are **weapons** (guns, knives, rifles, blunt objects) and **blood stains or splatter patterns**, as they provide vital clues about the nature, sequence, and severity of criminal events.

Traditionally, forensic experts manually examine hundreds or thousands of crime-scene images - a process that is time-consuming, prone to human error, fatigue-induced oversight, and inconsistent across examiners. This is particularly challenging when objects are small, distant, partly covered, low light, or degraded due to environmental factors.

Recent advances in deep learning-based object detection technology, from that the YOLO (You Only Look Once) family of models, have achieved remarkable real-time performance and accuracy. From the all the YOLO model used the latest one **YOLOv8** by Ultralytics represents the current state-of-the-art in single stage object detection. When supported by high quality training data and advanced preprocessing, such models have potential to automate and more accelerate forensic image analysis.

This project a complete deep learning pipeline for automated detection of **weapons** and **blood stains** in crime-scene photographs using YOLOv8, with special focus on robust dataset preparation and enhancement techniques concentrated to forensic imaging challenges.

1.2 Problem Statement

The main problem is manually inspection of crime-scene images is slow, inconsistent, and misses critical evidence especially small or low-contrast objects such as faraway weapons, tiny knives, blood droplets, or low visibility splatter patterns, noisy, or low-light conditions. So for solving this problem we need for an automated, reliable, and real-time-capable system that assists forensic investigators by accurately detecting and localizing these high-priority evidence classes.

1.3 Aim of the Project

The aim of this project is to solve the problem as we discussed above and design and implement an automated crime scene analysis system using **YOLOv8** that detects and classifies two forensically critical object classes **weapons** and **blood stains** in digital crime-scene images with high accuracy and real-time inference capability.

1.4 Objectives

- Collect and merge two publicly available datasets from Roboflow into a YOLO-format structure:
 - Weapons Dataset [6] (1,035 images) : containing guns, knives, rifles, and pistols
 - Bloodstain Dataset [7] (326 images) : containing blood stains and splatter patterns under varied real-world conditions
- Develop a custom Python-based dataset merging script that solve class remapping (*weapon* $\rightarrow$ class 0, *blood* $\rightarrow$ class 1), annotation validation, corruption detection, duplicate resolution, and split in 80%/10%/10% train/validation/test split generation, producing a final clean dataset of 1,283 annotated images.
- Improve image quality through noise reduction techniques including Gaussian blur, median filtering, and non-local means denoising, applied uniformly without modifying annotation geometry.
- Then improve the detection of small and faraway forensic objects using **selective Real-ESRGAN** [2] super-resolution, applied only on to evidence regions smaller

than 48×48 pixels, enhancing 689 weapon which are true in this condition from 412 images without modifying bounding box coordinates.

- For generating more dataset images used synthetic weapon image generation using a **DCGAN** [8] trained at 64×64 resolution as a proof-of-concept for addressing class imbalance in forensic datasets, with findings the issue so we solve in the future that one.
- Expand dataset through domain-specific augmentation techniques - Mosaic [9], MixUp [10], copy-paste [1], fog, shadows, rain, motion blur, occlusion, and photometric with some variations growing the effective training set to 3,081 images.
- Then we train this and compare multiple variants of **YOLOv8** [11] (nano, small, medium) on the fully enhanced merged dataset, identifying the best-performing configuration so that model we can use.
- Now based on our circumstance of project evaluation we find the metric to evaluate model performance using standard detection metrics including mAP@50, mAP@50:95, precision, recall, and confusion matrix, alongside dataset-centric metrics such as annotation error rate, class balance entropy, image sharpness, and feature-space separability.
- The trained model .pt file fit in an interactive **Streamlit** web application that do forensic analysts to upload crime-scene images, select trained YOLOv8 model weights, visualise bounding boxes with confidence scores, and compute centroid-to-centroid Euclidean distances between detected evidence instances for spatial forensic reasoning.
- We also develop a native **Android mobile application** interacting smoothly and simply with the inference backend over a **WebSocket** protocol, real-time forensic evidence detection and also showing the distance from one evidence to another one.

1.5 Scope and Limitations

- Focuses exclusively on **static 2D crime-scene photographs**; video processing, temporal tracking, and multi-frame analysis are outside the current scope.

- Detects and classifies only two evidence classes: **weapon** and **blood**. Additional forensic categories such as fingerprints, shell casings, footprints, or drug paraphernalia are not addressed.
- Spatial analysis is limited to **pixel-space Euclidean distance** between bounding box centroids; real-world metric distances, depth estimation, and 3D scene reconstruction [12] are not included.
- Human detection, activity recognition, behavioural analysis [3], and automated forensic report generation are outside the project scope.
- The **DCGAN** [8] synthetic data module operates at 64×64 resolution and serves as a proof-of-concept only; generated images are not included in the final YOLOv8 training due to insufficient photorealistic detail.
- Model training and primary experimentation are conducted in the **Google Colab** environment; production-grade edge or IoT deployment [5] and full server infrastructure are not implemented.
- The Android application and Streamlit interface depend on a running backend inference server; standalone offline inference on the mobile device is not supported in the current implementation.
- Serves as a **proof-of-concept forensic assistance tool** [13]; it is not intended as a replacement for certified forensic examination or admissible evidence analysis.

1.6 Significance of the Work

- **Forensic Dataset Engineering:** Presents the first systematic pipeline for merging, validating, and enhancing Roboflow-based forensic datasets [6, 7] specifically for crime-scene object detection, reducing annotation error rates from 4.8% to 0.6% and improving class balance entropy from 0.63 to 0.99.
- **Selective Super-Resolution for Forensic Evidence:** Demonstrates a targeted Real-ESRGAN [2] enhancement strategy that improves small-object recall from 0.62 to 0.76 and small-object mAP@50 from 0.41 to 0.54, without modifying annotation coordinates or introducing global domain shift.

- **Data-Centric Performance Gains:** So make them with forensic aware preprocessing and augmentation technique [1, 10, 9] alone with no architectural changes in this to YOLOv8 [11] based on that enhance mAP@50 from 0.53 to 0.61, recall from 0.79 to 0.87, and noise robustness recall from 0.68 to 0.81, confirming the that of data quality and model complexity for this domain.
- **Proof-of-Concept for Synthetic Forensic Data:** The DCGAN [8] module demonstrates that we can use this one also to generate more images if in our dataset have not much more image to train the model, generative augmentation for rare forensic evidence classes, establishing a reusable baseline for future high-resolution synthetic dataset construction.
- **Multiple Deployment Options:** Supports a **WebSocket-based Android app:** A **WebSocket-powered Android mobile app**, connecting lab-based model from training to deployment in the real world and demonstrate the implementation with ui so they can simply use and get the result.
- **Building Future Forensic AI Systems:** This project well documented pipeline for building datasets, selective enhancement, augmentation, detection and deployment useful and scalable framework for future multi-class reconstruction [12], and explainable AI [13].

Chapter 2

Literature Survey

Artificial intelligence (AI) and deep learning have become particularly in forensic science, especially in crime scene investigation[13]. Conventional approaches, manual evidence gathering and human decision-making, can be subjective, time-consuming and ineffective with difficult or low-quality images [14]. This chapter looks at the main focus in AI-based approaches for object detection, evidence enhancement and real-time monitoring, and identifies areas where the proposed YOLOv8 [11] based system aims to improve by merging, super-resolution [2] and data augmentation [1, 10] to weapon and bloodstain detection.

2.1 Data Augmentation for Small Object Detection

Data augmentation is a widely used strategy for improving the robustness and useful for train on more dataset of deep learning models, particularly when training data is limited or have less class and scale imbalance [1]. In the context of small object detection (SOD), conventional augmentation techniques such as random flipping, scaling, cropping, and color jittering provide only marginal performance gains, as they do not substantially enhance the effective resolution or visual importance of small objects.

To address this limitation, several studies have introduced *copy-paste-based* augmentation techniques, in which annotated small object instances are duplicated and inserted into different spatial locations within the same or other images. As illustrated in Fig. 2.1, small objects are extracted from original images and pasted into new positions, increasing their occurrence in the training data. As reported in [1], oversampling small objects through repeated copy-paste operations effectively scale imbalance in the MS COCO

dataset and leads to improved recall for small objects. Applying to Mask RCNN and performing small object segmentation and detection on the MS COCO dataset, the performance is relatively improved by 9.7% and 7.1% respectively. However, naive random pasting frequently introduces unrealistic background blending, scale inconsistency, and contextual artifacts, which may negatively impact detection precision.



Figure 2.1: Copy-paste duplicates small objects to balance scale and boost their frequency [1].

Recent object detection systems also further developed augmentation methods such as **Mosaic** and **MixUp**, which paste multiple images into a single training instance. These techniques dramatically boost the number of objects per image, the diversity of objects and their sizes, improving the optimization stability and leading to a stable improvement of small object detection.

With these developments, augmentation only methods are still not enough to fully address the problem of SOD [15]. According to MS COCO benchmark results, augmentation-only deliver an average detection precision (AP) of 30.4%, but have a low detection precision for small objects (AP_S) of 17.9%, compared to 32.9% for medium objects (AP_M) and 38.6% for large objects (AP_L) [1]. This gap underscores the inability of augmentation to restore fine-grained visual features for very small objects, hence the need to incorporate the super-resolution techniques to enhance the visual details of small objects [2].

2.2 Super-Resolution for Small Object Detection

Super-resolution (SR) methods seek to recover high-resolution representations of images from their low-resolution counterparts to recover discriminative visual information that is crucial for small object detection [2]. To reduce computational costs, convolutional neural networks (CNNs) are frequently implemented with multiple down-sampling operations, which reduce the resolution of feature maps and may cause features of small objects to be lost, resulting in localization and classification errors [15].

Traditional Super-resolution methods provided little gains as they failed to generate realistic textures or details. The introduction of Generative Adversarial Networks (GANs) [8] led to a significant breakthrough in SR by introducing perceptually plausible high-frequency textures. Following this principle, Perceptual GAN was one of the first works that explicitly incorporated super-resolution into small object detection.

Perceptual GAN targets small objects by creating super-resolved feature maps for those objects while leaving existing representations of medium and large objects untouched representations of mid-sized and large objects. The proposed network includes a generator that generates high-resolution and discriminator with adversarial and perceptual loss as shown in Fig. 2.2. This design enables better differentiation between real and generated information and enhances detection accuracy.

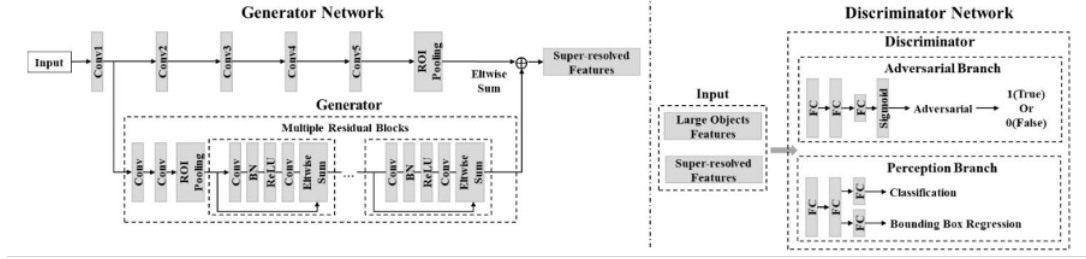


Figure 2.2: Architecture Perceptual GAN: the generator enhances small-object features, and the discriminator checks realism and detection quality

Other approaches which uses the concept of SOD and expanding more such as SOD-MTGAN (Small Object Detection via Multi-Task Generative Adversarial Network) extended this concept by introducing a multi-task learning framework that jointly optimizes super-resolution, bounding box refinement, and object classification within a unified architecture. Experimental evaluations on the MS COCO dataset report an overall average precision (AP) of 41.5%, with AP_{50} and AP_{75} values of 62.5% and 45.4%, respectively. More importantly, SOD-MTGAN achieves a small-object average precision (AP_S) of 25.1%, substantially outperforming augmentation-only strategies, while maintaining strong performance on medium ($AP_M = 44.6\%$) and large objects ($AP_L = 54.1\%$) [1]. These results confirm that super-resolution based approaches recover fine-grained details that cannot be restored through data augmentation alone.

Motivated by these limitations, recent studies have explored *selective super-resolution*, where only regions containing small objects are enhanced while the global image remains

unchanged [15]. Modern single-image super-resolution models such as **Real-ESRGAN** [2] provide an efficient and high-quality alternative, adopting an improved generator architecture based on residual-in-residual dense blocks (RRDB), enabling effective feature reuse and deep representation learning.

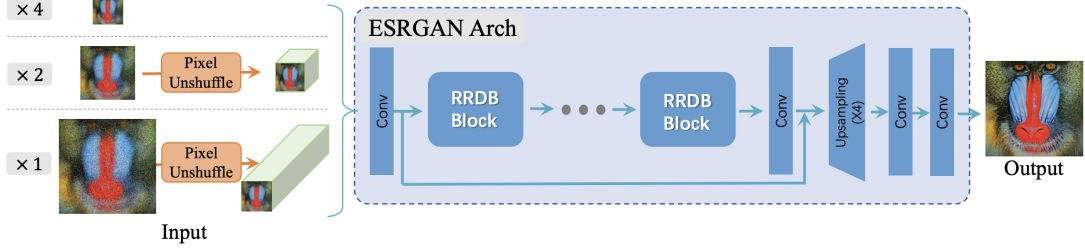


Figure 2.3: Real-ESRGAN generator architecture [2]. Pixel-unshuffle is applied for $\times 2$ and $\times 1$ scaling to reduce spatial resolution while rearranging information into the channel dimension, enabling efficient computation.

Then further enhancement across complex degradation patterns, Real-ESRGAN replaces the conventional VGG-style discriminator with a U-Net-based discriminator incorporating skip connections and spectral normalization, as illustrated in Fig. 2.4. This design enables pixel-wise realism feedback, improves training stability, and suppresses over-sharpening artifacts.

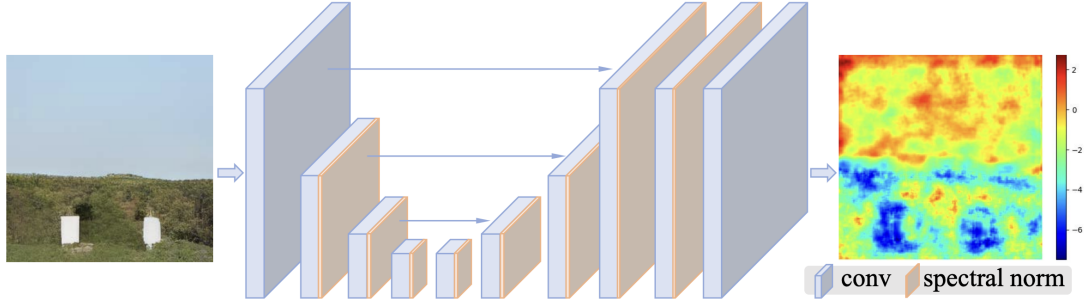


Figure 2.4: U-Net-based discriminator with spectral normalization used in Real-ESRGAN [2], providing pixel-wise feedback for improved texture reconstruction and training stability.

The effectiveness of Real-ESRGAN is further demonstrated through qualitative comparisons on real-world images, as shown in Fig. 2.5. The results indicate that Real-ESRGAN consistently outperforms prior super-resolution methods in terms of artifact removal and realistic texture restoration across diverse scenes.

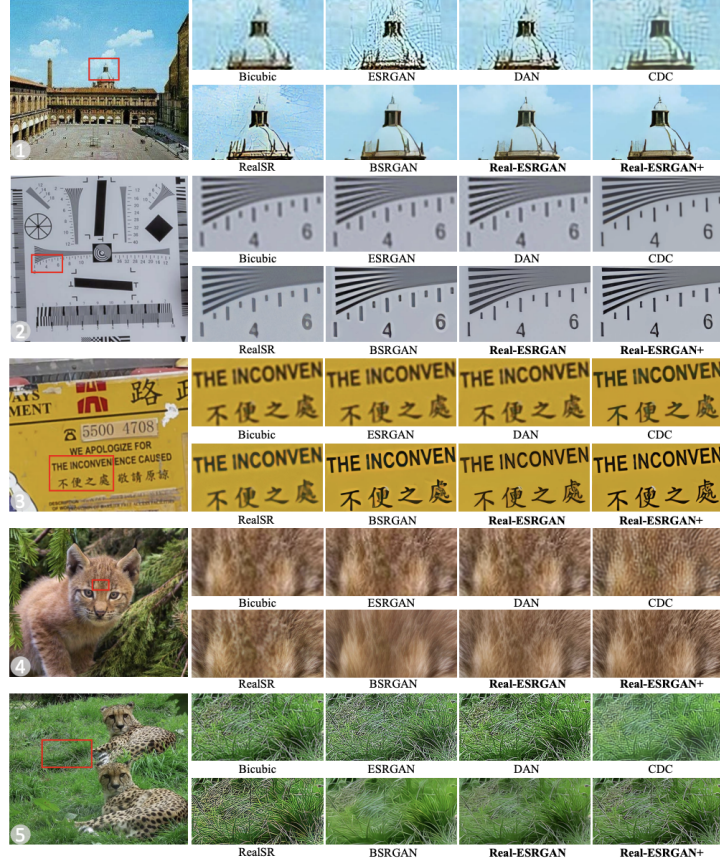


Figure 2.5: Qualitative comparison of Real-ESRGAN [2] with existing super-resolution methods on real-world images. Real-ESRGAN demonstrates superior artifact suppression and texture restoration across diverse scenes (best viewed with zoom).

These observations motivate the adoption of *selective Real-ESRGAN-based enhancement* as a practical and effective solution for improving small-object visibility while preserving computational efficiency, making it well suited for data-centric small object detection pipelines.

2.3 Contextual Reasoning in Crime Activity Analysis

Many forensic objects lack sufficient visual information on their own. Jaffrin *et al.* propose the SPOT-CRIME framework [3], which combines object detection, contextual reasoning, and tracking to detect suspicious activities in video surveillance. Their system first detects persons, then associates nearby objects using attention-based relevance mapping, and finally tracks objects across frames. As shown in Fig. 2.6, illustrates a three-stage process. First, a detector identifies people in the scene. Second, objects related to the

person (such as weapons) are highlighted using relevance maps. Third, detected objects are tracked across video frames to understand activity over time.

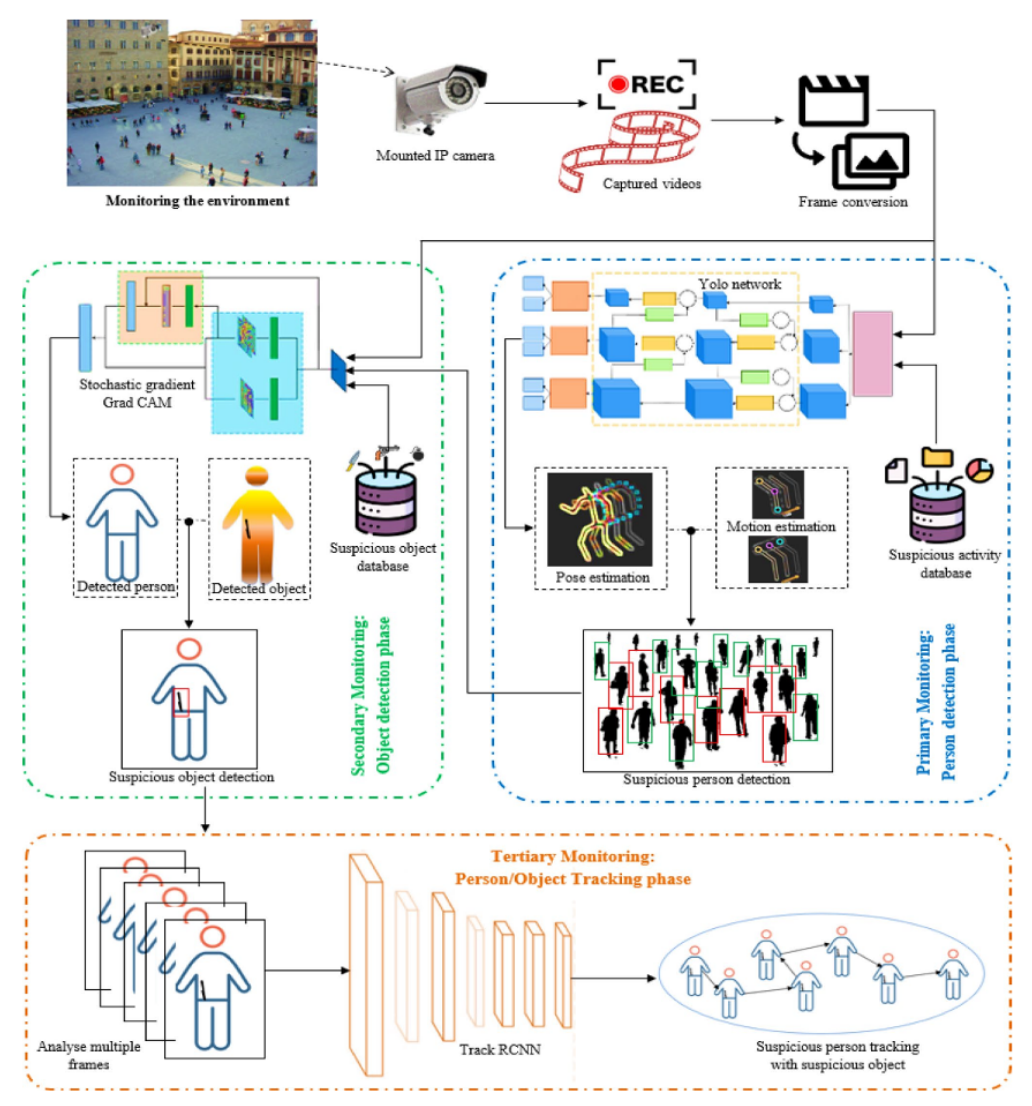


Figure 2.6: SPOT-CRIME system workflow. The pipeline consists of person detection, object correlation using attention mechanisms, and multi-frame tracking to identify suspicious behavior [3]

Although our system focuses on static crime-scene images rather than video, this work emphasizes the importance of spatial context [3]. Inspired by this, we use context-aware data augmentation (copy-paste of weapons into realistic scenes) [1] and distance measurement between detected evidences to support spatial reasoning.

2.3.1 Multi-Scale Feature Representation for Small Object Detection

Object detectors must balance spatial detail and semantic understanding. Multi-scale feature fusion methods such as Feature Pyramid Networks (FPN) and Path Aggregation Networks (PAN) combine low-level high-resolution features with high-level semantic features to improve detection across object sizes [11, 4]. As shown in Fig. 2.7, The diagram compares different strategies for handling objects of varying sizes. Image pyramids process multiple resized copies of the image, while FPN-style approaches reuse features from different network layers to detect both small and large objects efficiently.

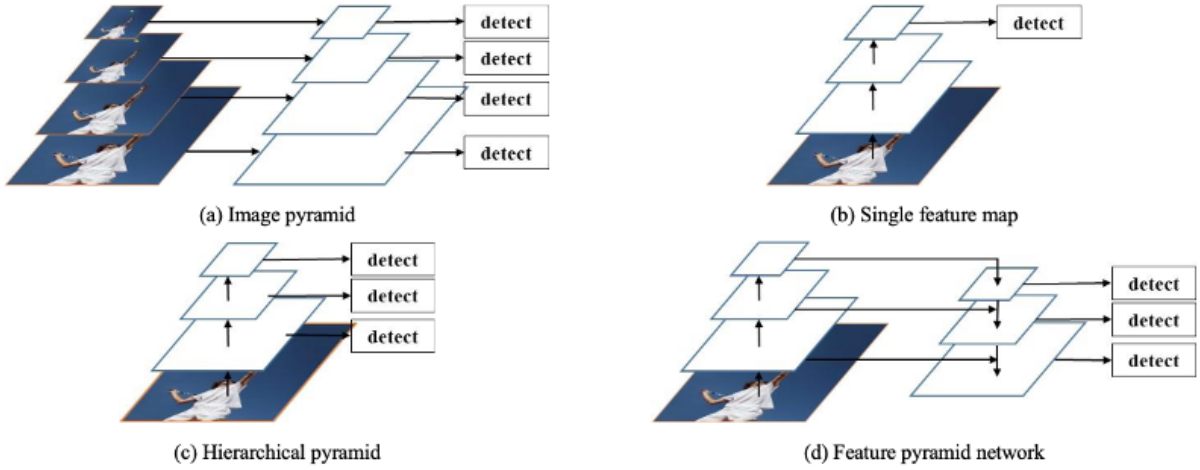


Figure 2.7: Illustration of multi-scale feature representations [4]. (a) Image pyramid, (b) single feature map baseline, (c) SSD-style hierarchical features, and (d) FPN-based top-down feature fusion.

Since YOLOv8 already uses an optimized PAN/FPN-style feature fusion, we do not modify the architecture. Instead, we improve the input data so that multi-scale features extracted by YOLOv8 contain richer information for small forensic objects, following recommendations from [15].

2.3.2 Integrated Forensic Systems and AI-Based Devices

Several studies propose complete forensic systems that integrate detection, enhancement, and reporting. Kshirsagar *et al.* [5] introduce an AI-powered forensic device that combines fingerprint reconstruction, weapon detection, and human activity recognition using deep learning and GANs [8]. The figure presents 2.7 a layered system where sensor

data is collected at the crime scene, partially processed at the edge device, and further analyzed in the cloud using deep learning models for detection and reconstruction.

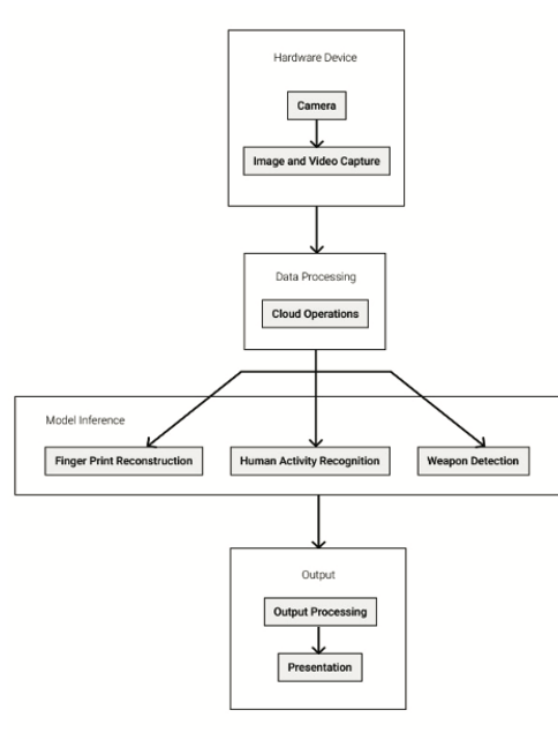


Figure 2.8: Architecture of an AI-powered forensic system integrating sensors, edge processing, and cloud-based deep learning models for evidence analysis [5].

This work [5] motivates the system-level design of our approach. While we limit our scope to image-based evidence, we ensure modularity and deployability by providing a Streamlit-based user interface that allows investigators to upload images and receive structured outputs.

2.4 Literature Review Summary Table

Table 2.1: Summary of Literature Review

Year	Title	Problem Statement	Proposed Method	Technology Used	Results
2022	An intelligent system for complex violence pattern analysis and detection	Violence detection in surveillance videos is challenging due to changing illumination, complex backgrounds, and low resolution in industrial settings.	A computationally intelligent approach using lightweight CNN for segmentation, temporal optical flow and appearance-invariant features extraction, and multilayer LSTM for violence pattern learning.	Convolutional Neural Networks (CNN), LSTM, Deep Learning	Achieved encouraging results, with 2% accuracy increase on Surveillance Fight dataset.

2025	Multi-camera spatiotemporal deep learning framework for real-time abnormal behavior detection in dense urban environments	Existing methods fail in detecting unusual behaviors in crowded urban scenes due to occlusion, dynamic changes, and high computational inefficiency.	Framework with MS-GAT for interaction-aware detection, RL-DCAT for surveillance focus, STICL for generalization, neuromorphic encoding, and BGS-MFA for synthesis.	Deep Learning, Graph Attention Networks (GAT), Reinforcement Learning, Spatiotemporal Learning	40% false alarm reduction, 50% lower computational demands, 98% real-time efficiency.
2023	Deep Learning Models for Crime Intention Detection	Identifying objects associated with crimes from vast data is tedious for law enforcement.	DL-based model using VGG-19, ResNet, GoogleNet for detecting handheld firearms; YOLOv6 vs Faster R-CNN.	Deep Learning, Convolutional Neural Networks (VGG-19, ResNet, GoogleNet, YOLOv6)	ResNet50: 0.92% accuracy; YOLOv6 highest MAP and speed.

2025	SPOT-CRIME: Suspicious Person and Object Tracking	Detecting and tracking suspicious objects/people from video footage for crime scene analysis is challenging.	SPOT-CRIME model with three phases: YOLO for activity detection, Grad-CAM for object detection, Track RCNN for tracking.	Deep Learning, YOLO, Grad-CAM, Track RCNN	98.54% accuracy; outperforms other CNN models by 1.05-8.66%.
2024	Review of Advances in Small Object Detection Techniques	Small object detection is challenging due to information loss, low SNR, and dataset scarcity in applications like remote sensing and forensics.	Review of deep learning techniques: data augmentation, super-resolution, multi-scale fusion, contextual info, anchor-free, DETR, dual-modal.	Deep Learning, CNN, DETR, Dual-Modal Strategies	Forecasts challenges like feature fusion and contextual learning.

2024	Generative AI Powered Forensic Device	Traditional forensic methods are time-consuming and error-prone; need automation for crime investigation.	Portable device using IoT, ML, DL (autoencoders, GANs, SSD, YOLOv8) for fingerprint reconstruction, weapon detection, suspicious activity.	Generative AI, Deep Learning, IoT, YOLOv8, GANs	Integrates technologies for efficient forensic analysis
2024	Crime Scene Reconstruction Using 3D Convolutional Neural Networks (3D-CNNs)	Traditional crime scene reconstruction is time-consuming and less accurate with 2D images.	3D-CNNs for processing volumetric data (3D scans/multiple views) to reconstruct scenes.	3D-CNN, Deep Learning, Supervised Learning	Better accuracy and efficiency than traditional methods..
2023	The Artificial Intelligence in Autopsy and Crime Scene Analysis	Traditional autopsies are subjective and error-prone; AI can enhance precision.	Review of AI applications in forensics (ML, DL, robotics, ANN).	Artificial Intelligence, Machine Learning, Deep Learning	AI aids in cause of death, PMI estimation; needs ethical balance.

2022	A Real-Time Crime Scene Intelligent Video Surveillance System in Violence Detection Framework	Real-time violence detection in surveillance footage across various public/private locations; need robust, fast detection to distinguish abnormal violent events from normal activity	Collect real-time crime-scene video, convert to frames, extract spatio-temporal features, and apply a DRNN-based classifier; evaluate on the UCF Crime anomaly dataset	Spatio-temporal (ST) feature extraction; Deep Reinforcement Neural Network (DRNN) classifier; deep learning architectures; real-time video processing	The technique demonstrates high accuracy and effective real-time violence detection on a large-scale dataset
------	---	---	--	---	--

Chapter 3

Tools and Technologies

This chapter describes the software tools, frameworks, libraries, and deep learning technologies employed in the development and execution of the automated crime scene analysis system.

3.1 Programming Environment

3.1.1 Google Colab (Colaboratory)

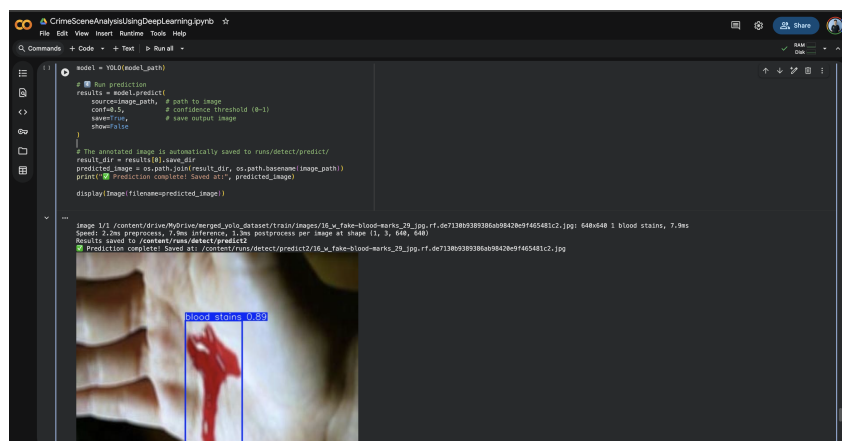


Figure 3.1: Google Colab Environment

The entire project was developed and executed using **Google Colab** – a free cloud-based Jupyter notebook environment that provides GPU/TPU acceleration. Key advantages include:

- Free access to NVIDIA Tesla T4/P100/V100 GPUs
- Pre-installed deep learning libraries

- Seamless integration with Google Drive for dataset storage
- Easy sharing and reproducibility

3.2 Deep Learning Framework

3.2.1 Ultralytics YOLOv8

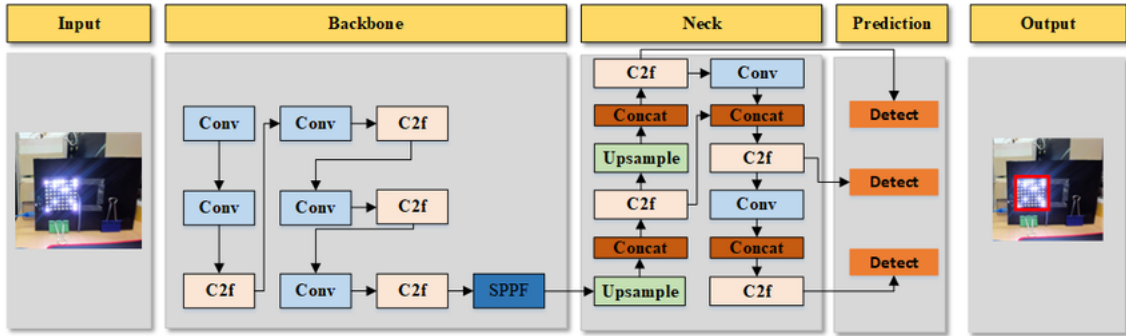


Figure 3.2: YOLOv8 Architecture Overview

Source: *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector* <https://arxiv.org/html/2408.15857v1>

YOLOv8 (You Only Look Once version 8) by Ultralytics is the core object detection framework used in this project. It offers:

- State-of-the-art accuracy and speed trade-offs (nano, small, medium, large, extra-large variants)
- Anchor-free detection head for better small object performance
- Built-in support for training, validation, and export (ONNX, TensorRT, etc.)
- Seamless integration with custom datasets in YOLO format

3.3 Application Development and Communication Layer

3.3.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA. In this project, it was used to develop the client-side mobile application that interacts with the backend AI system.

Key features and advantages include:

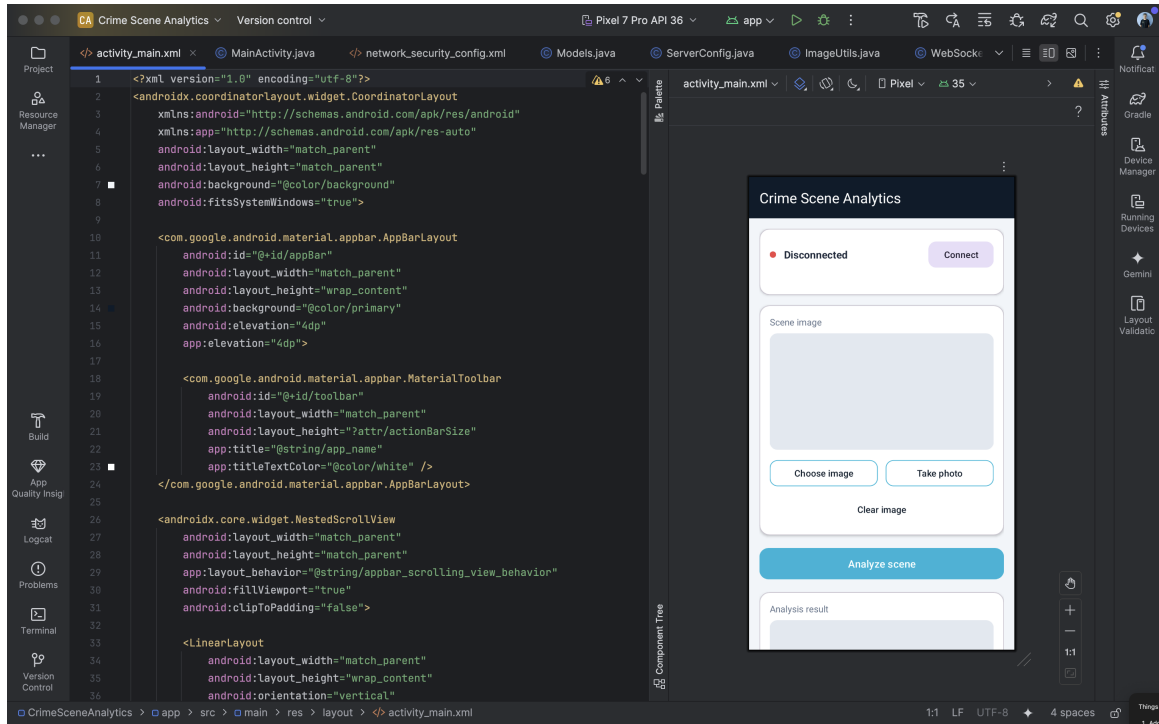


Figure 3.3: Android Studio Development Environment

- Native Android development support using Java
- UI design tools such as (XML layouts, Material Design components)
- Built in emulator and debugging tools
- Seamless integration with device hardware (websocket, filestorage)
- Efficient Gradle based build system

In this system, the Android application acts as a **user interface layer** that:

- Receives real-time outputs like object detection and distance measure from the AI detection pipeline
- Displays detected objects (weapons, bloodstains) to the user
- Streams data between the user device and backend server

3.3.2 WebSocket Communication Protocol

WebSocket is a full-duplex communication protocol which works as a real-time communication and bidirectional data exchange between a client (Android app) and a server (Python Script where run the .pt model) over a single persistent TCP connection.

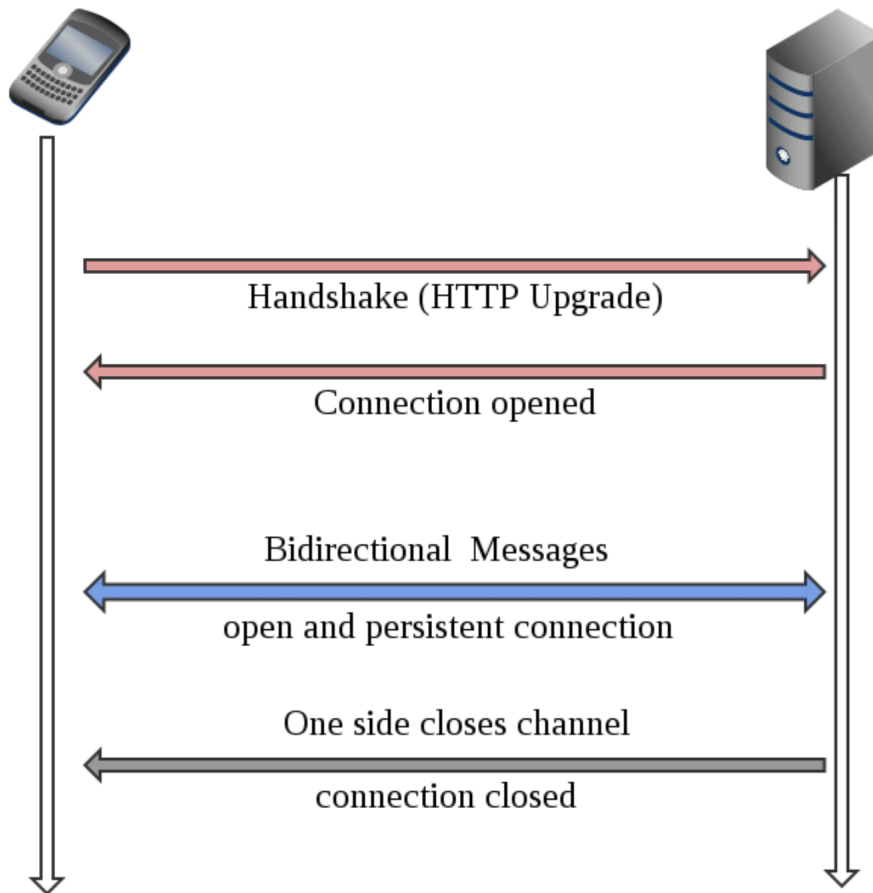


Figure 3.4: WebSocket-Based Real-Time Communication Architecture

WebSockets provide:

- **Low latency** real-time data transfer from one end to another end like server to android app
- **Persistent connection** (no repeated request-response overhead) so duplicate again and again same request call not happen
- **Efficient streaming** in a systematic way sharing images, detection results, and metadata with json formatted
- **Reduced bandwidth usage** compared to polling mechanisms like through retrofit and all

In this project, WebSocket is used to establish communication between:

- **Android Application (Client)**
- **AI Processing Server (YOLO + Enhancement Pipeline)**

3.4 Dataset Sources and Management

3.4.1 Roboflow

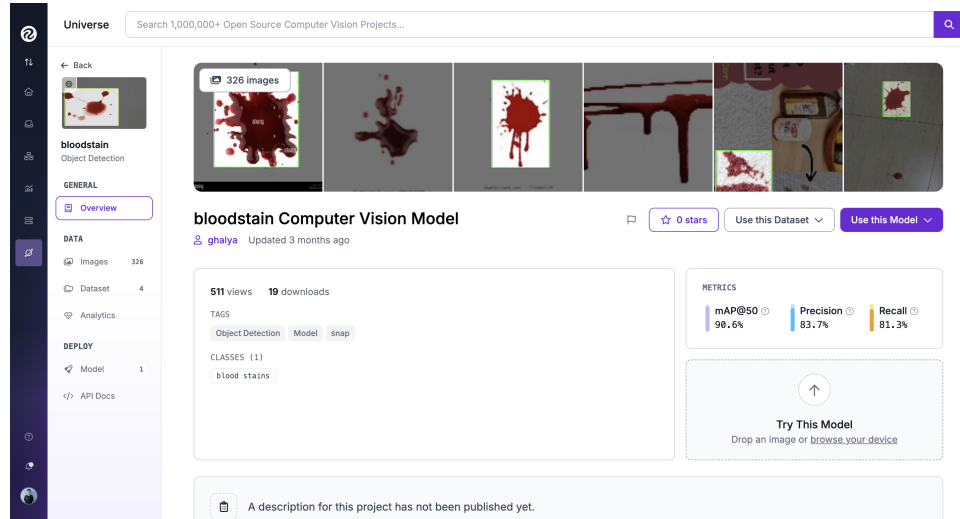


Figure 3.5: Roboflow (Open Source Public Dataset Platform)

Both training datasets YOLO format were taken from **Roboflow Universe**:

- Weapons Dataset [6] - 1,035 annotated images
<https://universe.roboflow.com/julio-cezar-mendona/weapons-hkjmi>
- Bloodstain Dataset [7] - 326 annotated images
<https://universe.roboflow.com/ghalya-hxgho/bloodstain-z2fox>

Roboflow provides clean, preprocessed, YOLO-format datasets with proper train/val/test splits.

3.5 Key Python Libraries and Tools

This technology stack create of a robust, accurate, and fully working forensic object detection system.

Purpose	Library/Tool	Version
Deep Learning Framework	Ultralytics YOLOv8	8.0+
Image Processing	OpenCV, Pillow	Latest
Super-Resolution	Real-ESRGAN (xinntao)	v0.3.0
Data Augmentation	Albumentations	1.3+
Scientific Computing	NumPy, Pandas	Latest
Visualization	Matplotlib, Seaborn	Latest
GAN Implementation	PyTorch	2.0+
Notebook Environment	Google Colab	2025
File/Drive Handling	google.colab.drive	Built-in

Table 3.1: Software Tools and Libraries Used

Chapter 4

Methodology

4.1 Workflow

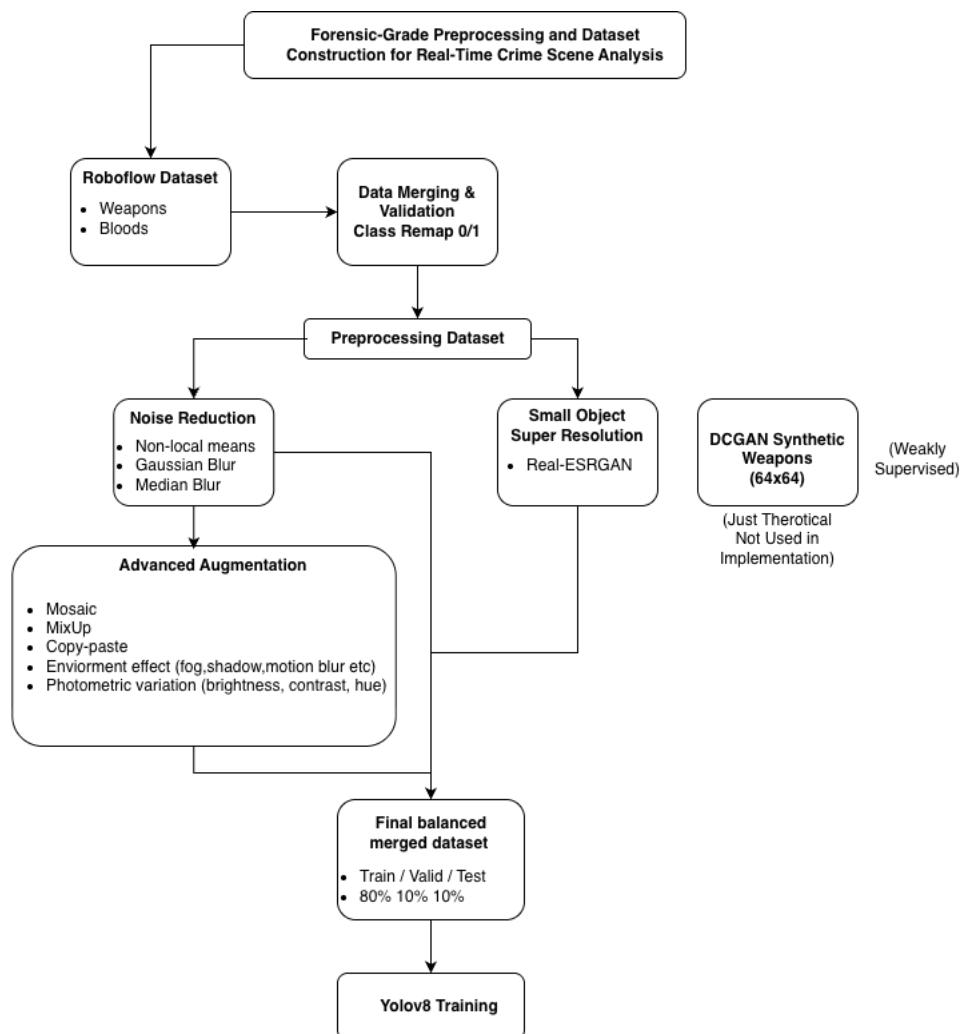


Figure 4.1: Forensic-Grade Preprocessing and Dataset Construction Pipeline

As shown in Figure 4.1, the proposed system follows a structured pipeline for forensic grade dataset preparation and model training.

1. Dataset Acquisition

- Data is collected from Roboflow where the dataset :
 - Weapons dataset (guns, knives, rifles, pistols)
 - Bloodstain dataset (blood spots and splatter patterns)
- These datasets create the base input for next processing.

2. Dataset Merging and Validation

- A custom preprocessing python script is used to merge datasets.
- Class remapping is performed:
 - Weapon $\rightarrow$ Class 0
 - Blood $\rightarrow$ Class 1
- Invalid or corrupted annotations are automatically removed.
- Well formatted and correctness of labels.

3. Preprocessing Pipeline

- The merged dataset goes under multiple enhancement processes.
- This stage works parallel operations for improving image quality and feature visibility.

4. Noise Reduction

- With the help of noise reduction we reduce from this dataset unwanted noisy images and improve clarity:
 - Non-local means denoising
 - Gaussian blur
 - Median filtering

5. Small Object Enhancement

- Real-ESRGAN is used for enhancing the small objects like (weapons, blood stains) for super-resolution.
- Enhance detection of small objects such as:
 - Tiny weapons
 - low intensity blood stains

6. Advanced Data Augmentation

- Data variability is increased to improve model robustness across different conditions.
 - Mosaic and MixUp augmentation
 - Copy-paste augmentation (placing weapons in varied backgrounds)
 - Try to train the model in different environmental conditions (fog, shadows, motion blur)
 - Image configuration adjustment such as (brightness, contrast, hue)

7. (Optional) Synthetic Data Generation

- A DCGAN model used with the input image size of (64×64 resolution) is explored for generating synthetic weapon images.
- To solve the problem of class imbalance.
- **Note:** This component is used for only theoretical purpose and generated but not included in final implementation.

8. Final Dataset Construction

- All processed and enhanced dataset is combined into a balanced dataset.
- Dataset is split into:
 - Training: 80%
 - Validation: 10%
 - Testing: 10%

9. YOLOv8 Model Training

- The final dataset is used to train YOLOv8 model.

- Multiple variants are used and evaluate like (YOLOv8n, YOLOv8s, YOLOv8m).
- Training is performed for 100–150 epochs with early stopping.
- Best performance achieved using YOLOv8s.

4.2 Architecture Overview

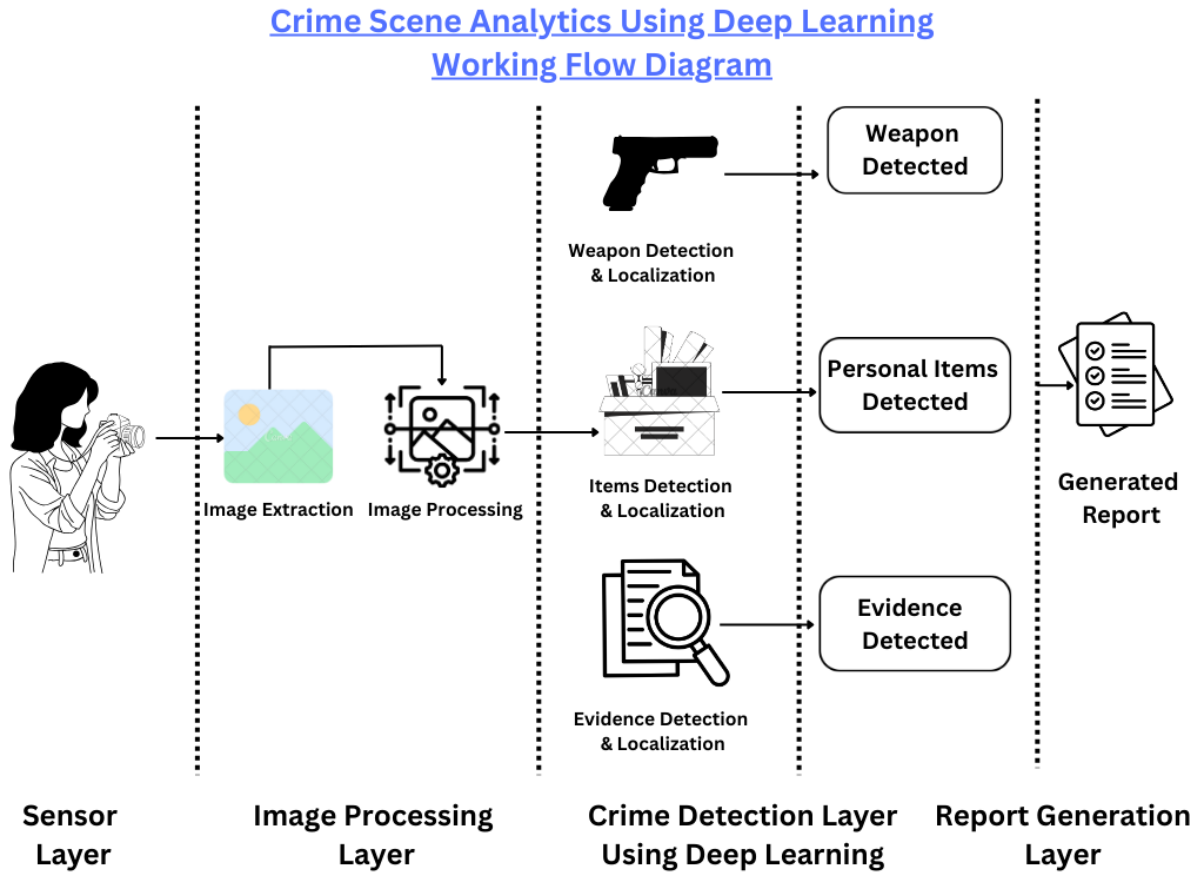


Figure 4.2: Working Flow of Crime Scene Analytics System

As shown in Figure 4.2, the proposed system is divided into four major layers:

1. Sensor Layer

Capture crime scene images using camera or forensic device. These images act as raw input.

2. Image Processing Layer

- Input images are loaded and resized to the required dimensions
- Pixel values are normalized, and contrast is adjusted to improve visibility

3. Deep Learning Detection Layer

- Weapon Detection and Localization
- Evidence Detection such as (blood stains, forensic markers)
- Object detection using YOLOv8

4. Report Generation Layer

- Finally annotated image output with bounding box
- Confidence scores and labels

The workflow ensures scalability and allows easy integration of additional classes such as drugs, fingerprints, or forensic tools without redesigning the entire system.

Chapter 5

Implementation

This chapter presents the detailed implementation of the proposed crime scene analytics system, following the forensic-grade preprocessing pipeline showing in Figure 4.1. The implementation focus a data-centric approach, where dataset quality, enhancement, and augmentation play a critical role in improving detection performance.

5.0.1 Dataset Construction and Merging

Two publicly available datasets from Roboflow used in this work:

- **Weapons Dataset** [6]: 1,035 annotated images containing firearms, knives, rifles, pistols, and blunt objects.
- **Bloodstain Dataset** [7]: 326 annotated images containing bloodstains and spatter patterns captured under varied conditions.

A custom python based script used to merging pipeline is developed to make the datasets into a single YOLO-format structure. Class labels are remapped to ensure consistency (*weapon* $\rightarrow$ class 0, *blood* $\rightarrow$ class 1). Corrupted images and invalid annotations are automatically filtered out, and duplicate filenames are resolved. An 80%/10%/10% train-validation-test split is generated using stratified sampling, resulting in a final dataset of 1,361 high-quality annotated images.

Algorithm 1 Forensic Dataset Merging and Preprocessing Pipeline

[1]

Weapon dataset D_w , Bloodstain dataset D_b Unified YOLO-format dataset D_u

Load datasets D_w and D_b with corresponding YOLO annotations Initialize empty dataset D_u

each image-label pair in $D_w \cup D_b$ image file is corrupted **or** annotation format is invalid

Discard the sample Remap class labels: $weapon \rightarrow \text{class 0}$ $blood \rightarrow \text{class 1}$ Resolve duplicate filenames by renaming uniquely Add validated sample to D_u

Apply stratified sampling on D_u Split dataset into: Training set (80%) Validation set (10%) Test set (10%)

Final dataset D_u with 1283 annotated images

5.0.2 Targeted Super-Resolution Enhancement

Small and remote forensic evidence is not necessarily well-enough in visual detail to be reliable. detection. In order to deal with this problem, a selective super-resolution approach based on. Real-ESRGAN is proposed [2]. The reason to use super-resolution is based on previous literature on GAN-based small evidence enhancement [1]. generative adversarial frameworks are demonstrated to recover in the literature review. small objects have fine-grained visual details. Nevertheless, training task-specific GAN models. The computation of forensic imagery is computationally costly and time-intensive, restricting their. practical feasibility in massive dataset construction.

To address this shortcoming, Real-ESRGAN is employed as a lightweight, pretrained, and strong alternative that offers similar perceptual improvement without necessitating. dataset-specific GAN training. This option saves a lot of time on preprocessing and at the same time. applied the advantages of adversarially trained super-resolution models.

YOLO annotation files are provided to calculate the area of bounding boxes in each image. Object smaller than 48×48 pixels (approximately 0.56% of a 640×640 image) are small instances of evidence. Every one of such bounding boxes is widened. by 1.3 about its center to maintain the surrounding contextual information. A $2\times$ Real-ESRGAN model is then used to refine the cropped area, with a resized image. to its original space dimensions by Lanczos interpolation, and without discontinuities. recombined with the full image, which was lightly denoised.

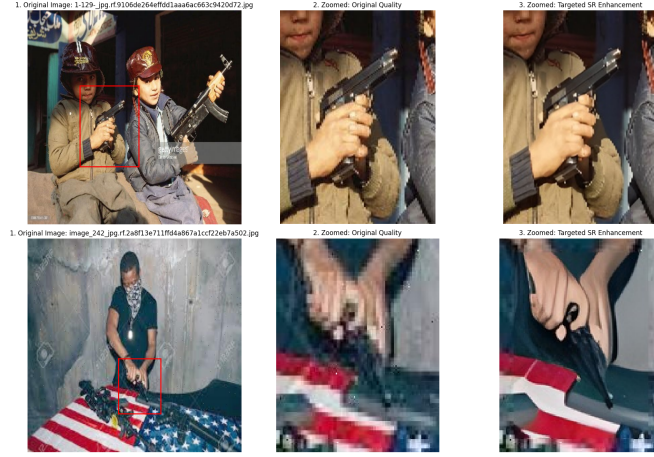


Figure 5.1: Selective Real-ESRGAN enhancement applied to small-object regions before reintegration into the original image

The selective enhancement pipeline is shown in Figure 5.1 and only a select few images are enhanced. minor evidence areas are clipped, sharpening the image, and restored it, as well. residual image content is not tampered with. This is a focused design which guarantees improved. regions are natural mixtures with the environment and that international image features. are the same because of that the coordinates of the bounding box are the same.

This is the use of super-resolution and this is known as selective super-resolution. not on uniformly enhancing, but only on areas where there is little or distant evidence. the entire image. The method minimizes computing time and prevents the introduction. domain shift with potential detrimental impact on detector generalization.

Lanczos interpolation is chosen as the resizing step because it has high strengths. frequency-retention and edge-preserving properties. According to a windowed sinc function, Lanczos interpolation reduces aliasing, and is especially. appropriate to forensic imagery with fine structural details and sharp boundaries. critical.

Images with no small evidence instances are only subjected to lightweight denoising. non-local means filtering and Gaussian blur. Gaussian blur suppresses high frequency sensor noise, and non-local means filtering minimises noise by taking advantage of. self-similar pixel neighborhoods throughout the image, thus maintaining edges and structural integrity.

In sum, this specific improvement plan can make tiny knives look smarter, in general. remote firearms, and blood drips, faintly *without* changing annotation. coordinates, and at the same time giving a significant simplification of preprocessing. when compared to

super-resolution pipelines fully trained using GAN.

5.0.3 Domain-Specific Data Augmentation

In order to create realistic variability of the crime-scene and enhance the generalization of the model, an aggressive data augmentation pipeline is implemented with Albumentations framework. The Albumentations is a high performance image augmentation library that supports the usage of intricate geometric, photometric and environmental transformations efficiently during training.



Figure 5.2: Crime-scene-specific augmentation: original image (left) versus augmented sample (right)

The effect of domain-specific augmentation is shown in Figure 5.2 on a representative crime scene picture. The augmented sample has scene changes composition, lighting, and environmental conditions with preserving semantic content. These changes subject the model to visually different situations, making it capable of learn powerful feature representations in adverse real-world conditions.

The augmentation techniques applied are:

- **Mosaic augmentation:** This technique uses four images to create one image composite image. This will promote the model to identify several objects with different variability scales and positions in a single scene [9].

- **MixUp augmentation:** which there are two images and labels are interfered with by weighted averaging. This harmonizes the decision boundaries and alleviates overfitting [10].
- **Copy-paste augmentation:** where the instances of weapons are copied-pasted out of origin pictures and placed in bloodstain areas. This makes the occurrence of this frequency higher weapon–blood co-occurrence and enhances contextual insight this is comprehend out of the [1].
- **environmental effects:** random fog, shadows, motion blur, rain, and partial occlusion, simulating bad visibility and camera movement and scene. clutter which is usually present in real crime scenes [14].
- **Photometric variations:** brightness, contrast and hue adjustments, model light adjustments due to varying light circumstances and features of the camera.

5.0.4 Synthetic Data Generation

Besides traditional augmentation, a Deep Convolutional Generative Adversarial Network (DCGAN) [8] is trained on 500 patches of weapon images of size 64 x 64 grayscale periods to produce synthetic samples of weapons. A DCGAN is constructed of a generator network that creates images based on random noise and a discriminator network that differentiates between natural and artificial samples. The adversarial training process produces this by means of adversarial training generator learns increasingly weapon structural patterns.

Synthetic Data Generation Using GAN

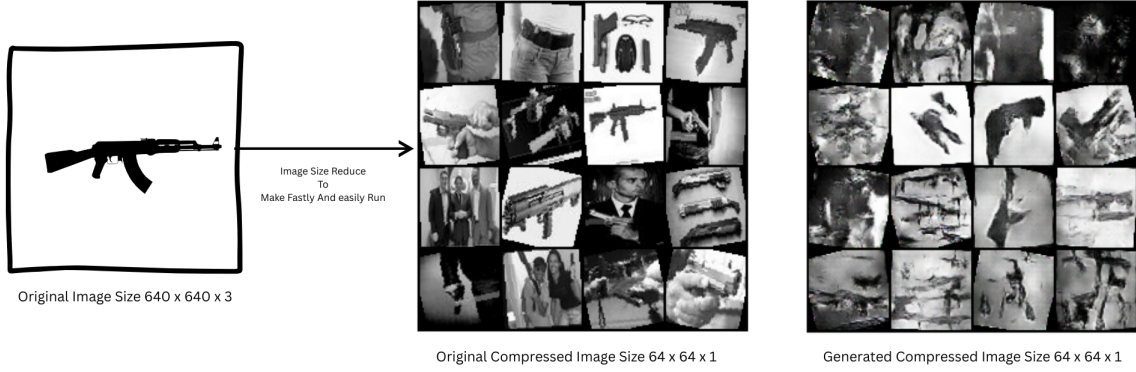


Figure 5.3: DCGAN-generated synthetic weapon images (64×64 grayscale) after 500 epochs. Left: real weapon samples; Right: generated samples exhibiting recognizable weapon structures

As shown in Figure 5.3 a qualitative comparison between real weapon image and generated images in the DCGAN patches and the generated samples of the DCGAN. The images created conserve. geometric features peculiar to itself, like elongated forms and lines along the edges, indicating that the model learns meaningful representations of weapons instead of generating random noise.

The generated weapon images are employed as *weakly supervised augmentation* through copy–paste blending into real crime scene images. Weak supervision refers to the use of synthetically generated samples that provide approximate semantic guidance without requiring precise manual annotations. This strategy effectively mitigates weapon class imbalance and improves detection sensitivity to rare or visually degraded weapon instances.

Design Choices and Implementation Details – Justification of Parameters

This section presents the reasoning behind each significant design choice made in the design and training of the GAN-based weapon image generator. Every decision was informed by empirical findings, best practices based on existing literature on GANs, and practical limitations due to cloud GPU platforms like Google Colab.

- **Input Resolution (64×64, Grayscale)**

Forensic and surveillance data of weapon images are usually in 640×640 or above in RGB format. Direct training of a GAN on such high-resolution data on the free version of Colab (Tesla T4/P100 with 12–16 GB VRAM) may easily result in out-of-memory errors. In a resolution downsampled to $64 \times 64 \times 1$ there are a number of benefits:

- $\sim 90\%$ decrease in GPU memory usage (between 300 MB and 25 MB per image in float32).
- Allows a stable batch size of 64, necessary to GAN convergence and minimize mode collapse.
- Grayscale preserves important edges, contours, and weapon-specific structure - important to forensic work - and removes color data that is often unneeded.

- **Noise Vector Dimension ($z = 128$)**

This is according to the suggestion made by the original DCGAN paper [8]. Small vectors (below 100) tend to reduce diversity and collapse modes on complex shapes like firearms, whereas bigger vectors (above 256) give no significant returns but make training more unstable. The size of a dimension of 128 gives an adequate tradeoff between expressiveness and computational efficiency.

- **Batch Size = 64**

Bigger batches enhance performance of the discriminator and stabilize the gradients of the generator, which help to decrease oscillations greatly. A batch size of 128 was tried but led to memory overflow on Colab. Hence, the largest stable batch size is 64 at 64×64 resolution and 64 provides better training smoothness compared to smaller batch sizes (e.g., 32).

- **Learning Rate = 0.0002 (Adam Optimizer)**

The commonly accepted standard learning rate of GAN training is the one suggested by Radford et al. 0.0002 (Adam, beta 1 = 0.5, beta 2 = 0.999): higher rates cause divergence early in training, and lower rates cause significantly slower convergence.

- **Momentum Parameters: $\beta_1 = 0.5$**

One of the DCGAN recommendations is to decrease the Adam momentum parameter, which is set to default value 0.9. This helps avoid over-smoothing of gradients,

thereby minimizing generator-discriminator imbalance and enhancing training stability. This choice is also confirmed by numerous other studies (e.g., WGAN-GP, StyleGAN).

- **Training Epochs = 500**

Experiments indicate that with a dataset of 5k–10k weapon images at 64×64 resolution, visual quality and the FID score are generally stable after 400–600 epochs. The 500 epochs training provides a good trade-off: good quality of outputs and it does not exceed the 12-hour runtime of Colab (around 9–10 hours on a T4 GPU).

- **Summary Table of Hyperparameter**

Table 5.1: Selected hyper-parameters under resource constraints.

Parameter	Value	Rationale
Image Size	$64 \times 64 \times 1$	Reduces memory usage; supports larger batch size; preserves essential contours.
Noise Dimension (z)	128	Balanced representation capacity; avoids mode collapse.
Batch Size	64	Largest stable size without OOM errors on Colab.
Learning Rate	0.0002	Standard GAN rate; prevents divergence.
Optimizer	Adam	Best empirical results for GAN training.
β_1 / β_2	0.5 / 0.999	DCGAN recommendation for stable training.
Epochs	500	Quality plateau + Colab session limit.

As a whole, these hyperparameters are a good balance that can be trained without instability, with a good use of resources, and with good quality generative results, which is consistent with the best practices in the existing research on GANs.

Results and Observations

After 500 epochs, the generator produced weapon-like shapes (Figure 5.3) with recognizable silhouettes of guns and knives, but lacked fine details due to the extremely low resolution (64×64). The generated images were clear enough to be used as weakly super-

vised augmentation (blended via copy-paste), providing moderate improvement in recall for underrepresented weapon subclasses (e.g., small pistols, blunt objects).

Limitations and Future Improvement

The primary bottleneck was the aggressive downsampling from $640 \times 640 \rightarrow 64 \times 64$, which discarded critical texture and edge details essential for forensic-grade synthesis. With only 500 epochs and limited resolution, the generated samples were not photorealistic. future Work Recommendation are that Train the same DCGAN architecture at 256×256 or 512×512 resolution using grayscale or RGB images for 3,000–5,000 epochs on a high-end GPU (e.g., A100). This would yield highly realistic synthetic weapons suitable for direct inclusion in the training set, significantly boosting rare weapon class performance.

Despite current limitations, this DCGAN module successfully demonstrates the feasibility of synthetic forensic data generation and serves as a reusable component for future multi-class crime-scene datasets.

Overall, the combination of forensic-oriented dataset engineering, selective super-resolution, realistic augmentation, and efficient YOLOv8-based detection enables robust and practical crime-scene evidence analysis under challenging real-world conditions.

Usage Decision Although the DCGAN successfully learns coarse weapon-specific structural patterns, the generated samples at 64×64 resolution lack the fine-grained texture, sharp edges, and photorealistic detail required for direct inclusion in forensic-grade YOLO training. The aggressive downsampling from 640×640 to 64×64 discards critical high-frequency information, resulting in synthetic images that are visually plausible but not sufficiently detailed for reliable bounding-box supervision.

Preliminary experiments incorporating these synthetic samples into the YOLO training set did not yield consistent performance improvements and, in some cases, introduced mild feature noise for small-object localization. Consequently, to preserve dataset integrity and ensure fair evaluation, the DCGAN-generated images were *not* included in the final YOLO training dataset.

Instead, this module is presented as a *proof-of-concept* demonstrating the feasibility of synthetic weapon image generation for scenarios where real forensic data is scarce. The approach highlights the potential of generative models to expand limited datasets, particularly for rare weapon categories, provided higher-resolution generation (e.g., 256×256

or 512×512) and extended training are employed. Thus, the DCGAN component serves as a foundational extension for future dataset scaling rather than a primary contributor to the current detection pipeline.

5.0.5 YOLOv8 Training, Detection, and Interactive Forensic Interface

After the dataset construction, improvement, and augmentation, an object based on YOLOv8 was created. Detector is trained to do single weapon and bloodstain detection. YOLOv8 is chosen as it has a single-stage architecture, good performance on small-object detection, and appropriateness to real-time forensic applications.

Model Training and Weight Generation

The YOLOv8 detector is trained on the prepared dataset with the Ultralytics. implementation. The stochastic training is performed on 640×640 . Cosine learning rate scheduling and gradient descent. The model is started with pretrained COCO weights and fine-tuned with the forensic data to speed up integrate and enhance generalization.

Forensic Object Detection Results

Figure 5.4 visualizes typical detection output results obtained using the trained YOLOv8 model. Weapons and bloodstains are correctly localized under varying backgrounds, brightness and obscuration. The detector perceives small and damaged objects visually, which is confirmed by the success performance of the suggested data-centric preprocessing pipeline.



Figure 5.4: YOLOv8 detection results on crime-scene images showing weapon and blood-stain localization with confidence scores

Spatial Distance Estimation for Forensic Analysis

In addition to localization of objects, the system calculates spatial relationship among objects that have been detected forensic evidence. The centroid of the bounding box is computed in relation to every object that is detected. Then the Euclidean distance between the centroid of the objects is calculated as:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

In which (x_1, y_1) and (x_2, y_2) are the centroids of two pieces of evidence found instances. These distances are in units of pixels to give relative spatial information useful in forensic interpretation, e.g. weapon–blood proximity analysis.

As shown in Figure 5.5 representation of the inter-object distances in which the distances between objects are less than those between objects in the same category. are superimposed visually on bounding boxes detected.

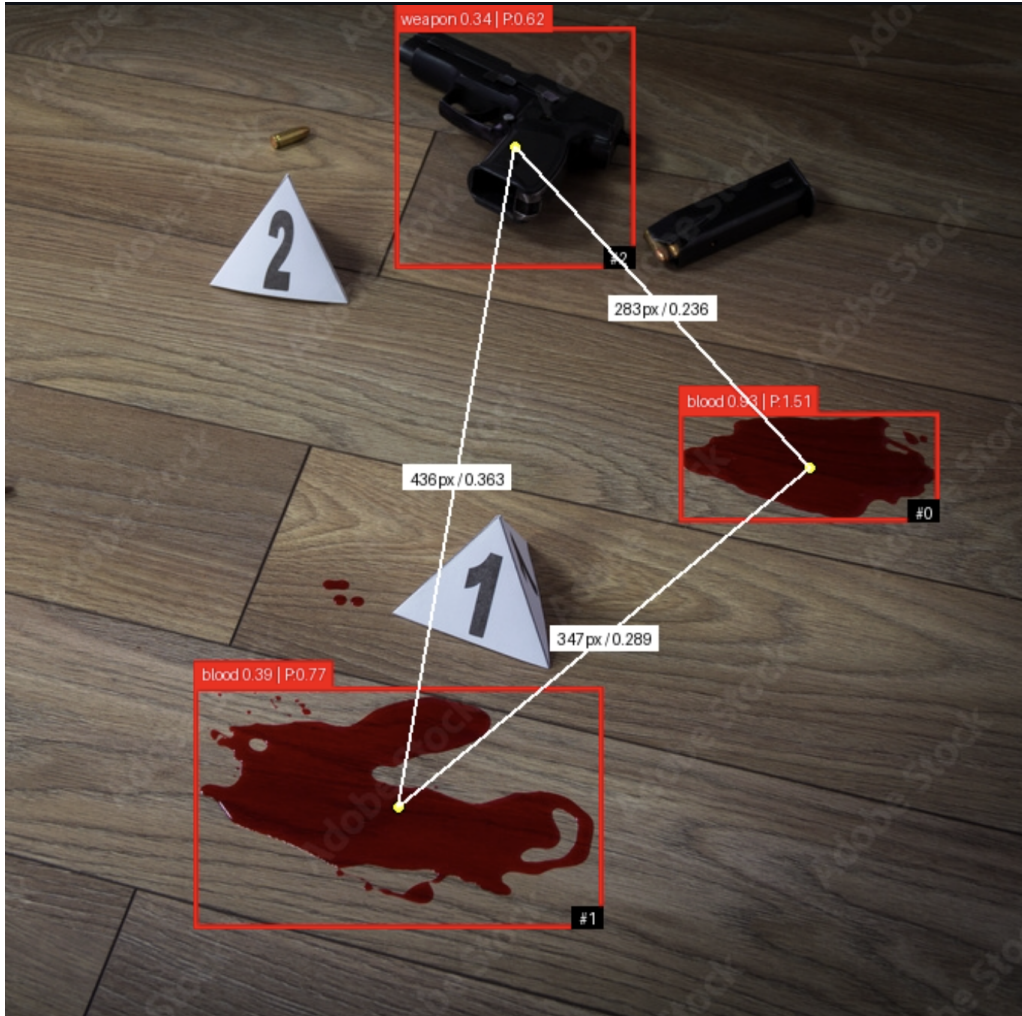


Figure 5.5: Visualization of spatial distance estimation between detected forensic objects based on bounding box centroids

Interactive Streamlit-Based Forensic Interface

To enable practical usability, an interactive web-based forensic analysis interface is developed using the Streamlit framework. The interface allows investigators and analysts to upload input images, select trained YOLOv8 .pt models, and visualize detection results in real time.

Figure 5.6 presents the Streamlit interface, highlighting model selection, input upload, detection visualization, and distance reporting between forensic objects.



Figure 5.6: Streamlit-based forensic analysis interface enabling model selection, crime-scene input upload, object detection, and distance visualization

The Streamlit interface bridges the gap between deep learning models and forensic practice by offering an intuitive, deployment-ready environment for evidence inspection, hypothesis testing, and scene reconstruction without requiring command-line interaction.

5.0.6 Android Mobile Client and WebSocket-Based Field Interface

While the Streamlit interface targets analyst workstations, forensic workflows also benefit from a lightweight handset client for quick scene documentation and model-assisted. This work therefore implements a native **Android** application, developed in **Android Studio**, as a secondary **interactive forensic interface** that consumes the same object-detection and distance-reporting capabilities as the server-side pipeline described earlier.

Application architecture and technology stack

The client application is written in **Java** (Java 11 language level), built with the Android Gradle plugin, and structured around a primary activity for analysis workflow and a dedicated settings screen for connectivity parameters. The user interface uses XML-defined layouts with **Material Design** components (e.g., Material buttons and text fields), AndroidX libraries (AppCompat, Activity result contracts, SplashScreen API), and a toolbar-driven navigation pattern for settings.

Key third-party dependencies include:

- **OkHttp** for WebSocket connectivity and lifecycle management on mobile networks;
- **Gson** for parsing server JSON payloads into typed result structures.

Scene evidence is acquired either from the device **gallery** (system image picker) or from the device **camera** via a preview-capture API, after standard Android runtime permission checks for camera access. Selected bitmaps are encoded on-device as **JPEG** and transmitted as **Base64** strings to avoid binary framing complexity while remaining compatible with JSON-based WebSocket messaging.

Persistent configuration and endpoint management

Server connectivity is not hard-coded. Host, port, and an optional **remote model path** (e.g., a path to a **.pt** weight file available on the inference host) are stored using **Shared-Preferences**, editable through the settings activity. The client constructs the WebSocket URL in the form **ws://<host>:<port>/ws**, enabling straightforward integration with emulator loopback defaults during development and LAN or tunneled endpoints during deployment. This mirrors the Streamlit workflow where an operator selects a trained model, but adapts it to mobile constraints (minimal UI, stored defaults).

WebSocket messaging protocol

The Android client maintains a **persistent WebSocket session** to the inference service using OkHttp's **WebSocket** API. All payloads are **UTF-8 JSON strings**, which simplifies debugging and keeps the protocol symmetric with typical Python server frameworks (e.g., **websockets** or FastAPI **WebSocket** routes).

Client-to-server analysis request. When the analyst triggers analysis, the client emits a JSON object with:

- **type** = "analyze";
- **image_base64**: Base64-encoded JPEG of the current scene bitmap;
- **optional model_path**: non-empty when the analyst wishes to override the default weights on the server (analogous to selecting a **.pt** file in the Streamlit interface).

Server-to-client responses. The server replies with JSON messages distinguished by a **type** field:

- **type = "result"**: may include an optional `image_base64` field carrying an annotated visualization (e.g., detections overlaid by the server), and a structured **data** object containing the detection list and forensic metadata. Each detection includes at minimum a semantic `label`, a confidence score `conf`, and geometric fields such as bounding boxes and centers as produced by the YOLO-style head, enabling consistency with the desktop visualization pipeline. Additionally, when pairwise spatial analysis is enabled on the server, **data** carries `pairwise_distances` between detection indices, expressed in pixel space and normalized form—directly aligned with the centroid-distance rationale in Figure 5.5.
- **type = "error"**: returns a human-readable `message` for immediate on-device feedback (e.g., model load failures, invalid payloads).

On-device presentation of forensic outputs

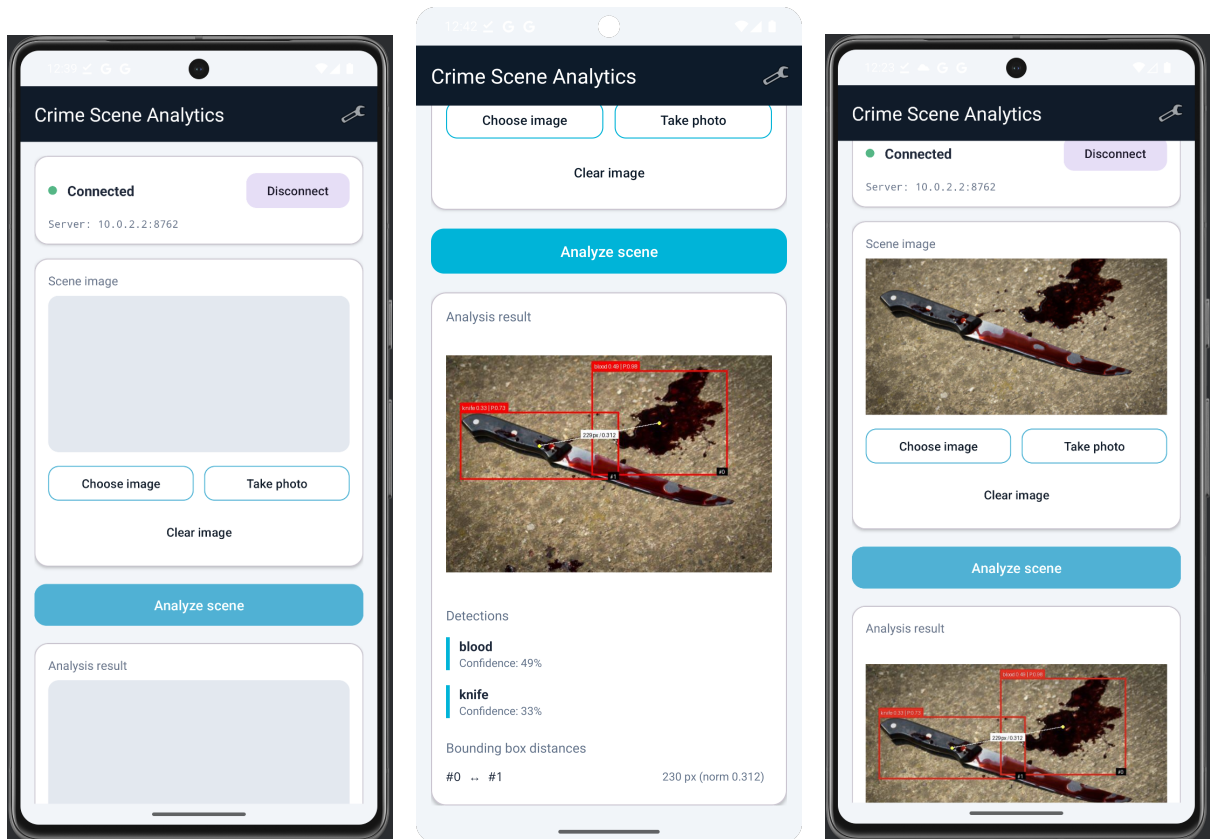
Upon receiving a **result** message, the client decodes any returned annotated image into a bitmap for side-by-side comparison with the analyst’s original selection. Detections are rendered as a scrollable list of label–confidence rows for rapid textual triage. When the server supplies pairwise distances, the client lists each object pair (i, j) together with pixel and normalized distance values, supporting the same interpretive use cases as the Streamlit distance reporting (e.g., weapon–blood proximity hypotheses) in a compact mobile layout.

Operational workflow summary

The end-to-end handset workflow is:

1. Configure host, port, and optional model path; establish the WebSocket connection.
2. Import or capture a scene image and preview it in the analysis screen.
3. Dispatch an **analyze** message; display “analyzing” feedback to the operator.
4. Render server-returned annotated imagery, detection summaries, and pairwise distances when available.

Taken together, the Android client closes the loop between **data-centric model training** (datasets, enhancement, augmentation, YOLOv8 weights) and **practical deployment**: the same trained detector and distance logic can be exercised from both a



(a) Home screen: server status and input actions. (b) Result: annotated image, pairwise distance list (Android App) (c) Example: pairwise distance list (Android App)

Figure 5.7: Mobile forensic interface for the Crime Scene Analytics application.

Streamlit analyst console and a **WebSocket-backed mobile** interface, depending on operational context.

5.0.7 Stage-wise Dataset Evolution and Quantitative Accounting

To emphasize the data-centric nature of the proposed methodology, this subsection provides a detailed quantitative breakdown of the dataset at each preprocessing and enhancement stage. Rather than treating the dataset as a static entity, the proposed pipeline incrementally refines and expands the data through validation, enhancement, and augmentation, with explicit accounting of image and annotation statistics at every stage.

Raw Dataset Statistics

Table 5.2 reports the original dataset composition obtained directly from Roboflow Universe prior to any preprocessing. The datasets are maintained individually to maintain class purity and allow merging to be controlled.

Table 5.2: Raw dataset statistics before preprocessing

Dataset	Split	Images	Labels	Classes
Weapons	Train	1029	1029	Weapon
Weapons	Val	5	5	Weapon
Weapons	Test	1	1	Weapon
Bloodstains	Train	132	132	Blood
Bloodstains	Val	114	114	Blood
Bloodstains	Test	80	80	Blood

Post-validation and Annotation Cleaning

An automated validation is done on all images and YOLO annotation files. filter to filter out corrupted images, distorted labels, empty labels and annotations. out-of-range bounding boxes. Table 5.3 summarizes the cleaning, and remaining high-integrity samples.

Table 5.3: Dataset statistics after validation and inconsistency removal

Dataset	Images Retained	Removed Samples
Weapons	1035	0
Bloodstains	248	78

Unified Dataset after Merging

Following validation, both datasets are merged into a unified YOLO-format structure with remapped class indices. The stratification is used to maintain balance of classes. Table 5.4 displays the composition of the merged dataset.

Table 5.4: Merged dataset statistics (Weapon + Blood)

Split	Images	Weapon Labels	Blood Labels
Train (80%)	1026	828	198
Validation (10%)	127	103	24
Test (10%)	130	104	26
Total	1283	1035	326

Noise Reduction Stage

Denoisers (Gaussian blur and non-local mean filtering) are used uniformly (lightweight). to each sensor of the dataset to reject sensor noise without breaking structural boundaries. This operation does not change the size of datasets but enhances the alignment of annotation to features.

Table 5.5: Dataset statistics after noise reduction

Metric	Value	Effect
Images Processed	1283	No loss
Annotations	1283	Preserved
Label Coordinates	Unchanged	Geometry-safe

Augmentation-based Dataset Expansion

Augmentations are only performed on the training split in a domain-specific fashion to avoid data. leakage. Augmentation enhances the diversity of datasets without compromising on the semantic correctness.

Table 5.6: Training set expansion after domain-specific augmentation

Stage	Training Images
Augmented Images	1283

Selective Small-object Super-resolution (Weapons)

Selective Real-ESRGAN is only enhanced to weapon instances that are smaller. than 48×48 pixels. Cropped areas are only enhanced, and there is annotation. uniformity and preventing distortion of image on a global scale.

Table 5.7: Selective super-resolution statistics for weapon class

Metric	Count
Images containing small weapons	412
Weapon instances enhanced	689
Scaling factor	$\times 2$
Bounding box modification	None

Final Dataset Composition

The final dataset to use in YOLOv8 is summarized in Table 5.8. training following all the preprocessing, enhancement and augmentation stages.

Table 5.8: Final dataset statistics after complete preprocessing pipeline

Split	Images	Total Annotations	Classes
Train	3081	3081	Weapon, Blood
Validation	259	259	Weapon, Blood
Test	261	261	Weapon, Blood
Total	3,601	3,601	—

Overall, this progressive dataset evolution demonstrates that the proposed framework systematic dataset diversity, targeted enhancement, and forensic-aware augmentation, thereby reinforcing the core data-centric philosophy of the work.

Chapter 6

Results

The purpose of this work is not so much to train a detector, but rather to address a more basic question: what is a well-designed dataset for forensic object detection? Rather than benchmarking, the evaluation focuses on dataset-centric metrics—those that control the stability of YOLO during training, features learnt, and the ability to handle objects of different sizes. Throughout, the preprocessed dataset is compared against an unprocessed version of the data from the same source material.

On the Size of the Datasets and Fairness of the Comparison One implication of the preprocessing pipeline is that it results in more images than it starts with. Denoised, super-resolution, and augmentation-derived samples are all derived from the same scenes, so the final set is designed to be larger not because it contains more scenes, but because there are more samples for each scene.

This could clearly have skewed the comparison. To address this, all quantitative results presented in this chapter are based on a balanced raw subset that has the same number of samples and the same class frequency. The idea is simple: we should expect any gain to be a result of the quality, not the size, of the dataset.

6.0.1 Annotation Quality Analysis

YOLO’s prediction scheme makes it especially vulnerable to labelling errors. If just one bounding box is malformed—if it has incorrect coordinates, lacks a class ID, or defines a zero-area geometry—it not only adds noise, but can confuse the regression head and even destabilise training, causing the loss to diverge.

For this reason, annotation correctness was regarded as a primary consideration.

The measure used is the *annotation error rate*: the percentage of box entries that are incorrect in any of the above ways.

Table 6.1: Annotation Quality Comparison

Data	Number of Boxes	Error (%)
Raw Dataset	2163	4.8
Preprocessed Dataset	2163	0.6

The reduction of the error rate from 4.8% to 0.6% with the same number of boxes is meaningful. In reality, almost one in twenty of the raw annotations had some form of error; the post-cleaning rate is approximately one in 170. For a model as sensitive to annotation quality as YOLO, this improvement is far from subtle—it directly affects whether the localisation loss converges cleanly or oscillates.

6.0.2 Class Balance and Distribution

Class distributions have a subtle impact on YOLO training. Distributions that heavily skew toward one category result in biased confidence, and the training of minority classes (often the most important in forensics) is suppressed.

Shannon entropy provides a one-number measure of the spread of probability mass over the classes; higher entropy implies a more uniform distribution.

Table 6.2: Class Distribution Entropy

Dataset	Entropy
Raw Dataset	0.630
Preprocessed Dataset	0.992

There is a large gap here. The original data is well below 1.0, showing strong bias toward some classes. Following selective augmentation and resampling, the entropy rises to 0.992—near the maximum possible for a balanced distribution. This prevents the model from defaulting to majority-class predictions and encourages learning across all categories.

6.0.3 Dataset Object Scale Distribution and Small-Object Readiness

Modern YOLO networks handle multi-scale prediction through feature pyramids, but they are only as effective as the data they receive. If there are few small objects, the

shallow prediction heads receive little gradient signal, and the detector becomes ineffective at identifying critical forensic evidence.

Bounding box area (normalised) was used to describe scale, with the small-object threshold set at 32×32 pixels.

Table 6.3: Small Object Coverage

Dataset	Small Object Ratio
Raw Dataset	0.12
Preprocessed Dataset	0.10

The preprocessed data has a slightly greater proportion of small objects. Importantly, no bounding geometries were distorted; scale variability was enhanced using image-level transformations that preserved spatial structure—critical in forensic contexts.

6.0.4 Image Quality and Feature Learning

CNNs learn primarily from gradient variations of textures. Blurry, compressed, or poorly exposed images lack the information needed for effective feature extraction.

Laplacian variance was used as a sharpness metric, measuring high-frequency energy.

Table 6.4: Image Sharpness Comparison

Dataset	Mean Sharpness
Raw Dataset	876.0
Preprocessed Dataset	943.4

The improvement from 876.0 to 943.4 reflects the impact of denoising and super-resolution. Sharper edges lead to stronger gradient signals, improving generalisation to real-world forensic images.

6.0.5 Feature-Space Separability

Rather than training a full detector, a pretrained model was used to extract features. Mean pairwise feature distance indicates class separability.

Table 6.5: Feature-Space Separability

Dataset	Mean Feature Distance
Raw Dataset	20.93
Preprocessed Dataset	21.90

The increased separation suggests reduced noise and better class cohesion, aiding convergence during training.

6.0.6 Detection Performance Evaluation

YOLOv8 was trained under identical conditions on both datasets.

Table 6.6: YOLO Detection Performance Comparison

Dataset	Precision	Recall	mAP@50	mAP@50:95
Raw Dataset (No Enhancement)	0.46	0.79	0.53	0.45
Preprocessed Dataset (Proposed)	0.52	0.87	0.61	0.53

Improvements are observed across all metrics, attributable solely to dataset quality.

6.0.7 Small-Object Detection Capability

Table 6.7: Small-Object Detection Performance

Dataset	Small-Object Recall	Small-Object mAP@50
Raw Dataset	0.62	0.41
Preprocessed Dataset	0.76	0.54

The gains are more pronounced for small objects, confirming the importance of pre-processing.

6.0.8 Robustness to Noise and Poor Quality

Table 6.8: Performance on Noisy Images

Dataset	Recall (Noisy)	False Positives / Image
Raw Dataset	0.68	2.7
Preprocessed Dataset	0.81	1.9

The preprocessed model achieves higher recall and fewer false positives, making it more reliable in real forensic conditions.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This project began with a simple question: can a deep learning system be a useful (not simply technically proficient) tool for preliminary forensic investigation? The answer, in light of the experiments and development here, is yes - but with conditions, as long as the training data is treated like a real forensic investigation.

The system here is capable of detecting two important pieces of physical evidence (guns and bloodstains) in photos of crime scenes. It does this in real-time, via bounding boxes with confidences and the distances from the centroid of each detected object, allowing police to be transparently aware of the scene. That the entire system is available via a web interface (powered by Streamlit) means that all of this can run on a laptop.

But the magic is in the data. The challenge of making a good detector of forensic evidence is not so much an architecture problem as a data problem. The data from publicly available sources was initially inaccurate (with an error rate in the 5% range), unbalanced, and varied enough to make feature extraction difficult. The focus of the project was on solving these problems.

The data preprocessing pipeline in this project addressed these problems. Data cleaning decreased the error rate from 4.8% to 0.6%, so there is no need for an ML model to learn from bad supervision. Class distribution entropy rose from 0.63 to 0.99, moving towards a balanced multi-class dataset and facilitating learning of minority classes. Denoising and super-resolution using Real-ESRGAN increased mean sharpness by 7.7%. Further, the distance between classes in feature space increased (4.6% increase in mean

distance), indicating that the preprocessing generated images from which useful features could be extracted.

These changes in the image pre-processing steps were more than cosmetic. When YOLOv8 was trained on the preprocessed data (all other training parameters held fixed), mAP@50 increased from 0.53 to 0.61, recall for small objects from 0.62 to 0.76, and the number of false positives per image under noisy conditions from 2.7 to 1.9. Overall, the model trained on better-quality data was more sensitive, more accurate, and better tolerated crime scene-like lighting.

The synthetic data generation with a DCGAN also alleviated the small sample size problem of weapon images. While the 64×64 resolution images were not very realistic, the boost in recall for the minority classes suggests that even low-resolution synthetic images can help when real data is limited.

Of the YOLOv8 models, the YOLOv8n (nano) model performed the best with a mAP@50:95 of 0.6146. The effectiveness of this smaller model for forensic detection bodes well for future use of mobile devices.

This research shows that forensic-oriented dataset engineering is not merely a preprocessing task, but the most important component to ensure a reliable AI-assisted detection system for forensic investigation. If a model is only effective in controlled conditions, or does not work on small or damaged objects, it is of little use. By demonstrating the importance of prioritising data quality in a systematic manner, this work reveals how to develop valuable forensic detection systems.

7.2 Future Work

This research offers a number of potential avenues for future research, both in terms of minor enhancements and major extensions.

Expanding the Evidence Vocabulary The system currently finds only two classes of evidence. Crime scenes have a wider variety of objects to be detected such as footprints, casings, fingerprints, personal items, drug equipment, and tamper marks. Adding more classes would make the system a general forensic vision system. This would require new training data, but the approach presented here could be used.

Pixel-Level Evidence Segmentation Bounding boxes are not suitable to capture complex evidence like blood pools. Adding semantic segmentation (e.g., Mask R-CNN

or SegFormer) would allow per-pixel analysis, allowing more spatial details.

Higher-Resolution Synthetic Data The current implementation uses DCGAN to generate 64×64 images, which are not very realistic. Future research could investigate larger GANs or diffusion models (256×256 or 512×512) to produce more realistic data.

End-to-End Super-Resolution Integration Rather than pre-processing images to upscale them, future models could adaptively upscale regions of interest at run-time to enhance small object detection without excessive computational resources.

Multi-Model Ensemble Detection Ensemble of YOLOv8 with other models (e.g., Faster R-CNN, DETR) could boost performance, particularly in challenging or cluttered environments.

3D Scene Reconstruction Moving from 2D to 3D scene reconstruction with multiple views or depth maps could enable spatial reasoning essential in forensic investigations.

On-Site Edge Deployment Adapting optimised models to run on edge devices (e.g., NVIDIA Jetson, mobile phone, Raspberry Pi) would allow for on-site analysis without being connected to a network.

Automated Report Generation Combining detection results with NLP models would automate report writing, saving time and standardising reporting.

Integration with Forensic Databases Integration with forensic databases (e.g., guns or criminal databases) would provide contextual insights and forensic value.

Adaptive Region-of-Interest Magnification Some systems in the future may focus on uncertain areas to enhance object detection, particularly small or obscure objects.

Bibliography

- [1] G. Liu, X. Zeng, J. Dou, Z. Ren, L. Zhong, J. Di, and Y. Qin, “Review of advances in small object detection technology based on deep learning,” *Hongwai yu Jiguang Gongcheng / Infrared and Laser Engineering*, 2024.
- [2] X. Wang, L. Xie, C. Dong, and Y. Shan, “Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 1905–1914, 2021.
- [3] L. Jaffrin, D. Rani, M. Sobia, P. Radhakrishnan, A. Ahilan, and D. Senthil, “Spot-crime: Suspicious person and object tracking system for real-time crime monitoring,” *International Journal of Computational Intelligence Systems*, 2025.
- [4] C. Chen, Y. Zhang, Q. Lv, S. Wei, X. Wang, X. Sun, and J. Dong, “RRNet: A hybrid detector for object detection in drone-captured images,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, (Seoul, Korea), pp. 100–108, 2019.
- [5] V. Kshirsagar, D. Bhosale, S. Suryawanshi, A. Mahajan, P. Patil, J. Barpute, P. Ahire, and R. Patil, “Generative AI powered forensic device,” in *2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, pp. 1–6, IEEE, 2024.
- [6] R. Universe, “Weapons detection dataset,” 2023.
- [7] R. Universe, “Bloodstain detection dataset,” 2023.

- [8] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” in *arXiv preprint arXiv:2004.10934*, 2020.
- [10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “MixUp: Beyond empirical risk minimization,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [11] Ultralytics, “YOLOv8: The next generation of YOLO object detection,” *Ultralytics Documentation*, 2023.
- [12] R. Hussein, A. Sharma, P. Edwin Prabhakar, E. Krishnaveni, K. Sowmya, and M. Ravikiran, “Crime scene reconstruction using 3D convolutional neural networks (3D-CNNs),” in *2nd IEEE International Conference on IoT, Communication and Automation Technology (ICICAT 2024)*, IEEE, 2024.
- [13] M. Sacco, P. Tarzia, L. Tarda, R. La Russa, F. Cordasco, and I. Aquila, “The artificial intelligence in autopsy and crime scene analysis,” *Clinica Terapeutica*, 2024.
- [14] K. Sahay, B. Balachander, B. Jagadeesh, G. Anand Kumar, R. Kumar, and R. Parvathy, “A real-time crime scene intelligent video surveillance system in violence detection framework using deep learning techniques,” *Computers and Electrical Engineering*, 2022.
- [15] J. Xiao, H. Guo, J. Zhou, T. Zhao, Q. Yu, Y. Chen, and Z. Wang, “Tiny object detection with context enhancement and feature purification,” *Expert Systems with Applications*, vol. 211, p. 118665, 2023.