ENHANCING CRIME PREDICTION WITH MACHINE VISION: CONVOLUTIONAL NEURAL NETWORK APPROACH

Taiwo Marcus Akinmuyisitan

A thesis submitted in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy (PhD) to:

The Department of Electronic and Electrical Engineering

College of Engineering, Design and Physical Sciences

Brunel University London (BUL)

September, 2024.

Principal Supervisor: Prof. John Cosmas

Supervisor: Professor Tatiana Kangalova

ACKNOWLEDGEMENTS

I want to especially thank my principal supervisor, **Prof. John Cosmas** for his guidance and supports towards completing this thesis. I would also like to express my sincere gratitude to **Prof. Tatiana Kangalova** for their generous supports. I would not have completed this thesis without their experience and guidance. My profound gratitude goes to my wife, **Mrs Akinmuyisitan Febisola** and my Brother **Benard Akinmuyisitan** for their encouragements, moral and financial supports throughout the programme. Finally, I will like to appreciate my children, **Flourish, Praise and Treasure Akinmuyisitan** for their moral support during the program.

DECLARATION

I declare that this thesis is my own work and submitted for the first time to the Post-Graduate Research Office. The study originated, composed, and reviewed by myself and my supervisors in the Department of Electronic and Computer Engineering, College of Engineering, Design and Physical Sciences, Brunel University of London UK. All the information derived from other works duly referenced and acknowledged as required.

SIGNED: .... DATE: ....

Table of Contents

List of Tables

LIST OF FIGURES

ACRONYMS AND ABBREVIATIONS

A-CNN	Advance Convolution Neural Network
AI	Artificial Intelligence
ANN	Artificial Neural Network
AWS	Amazon Web Services
BFLOPs	Billion Floating Operation Per Seconds
BN	Batch normalization
CDC	Centre for Disease Control
CNN	Convolution Neural Network
COD	Categories-Specific Object Detection
ConvNet	Convolutional Neural Network
CSP	Common Spatial Patterns
DNN	Deep Neural Network
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
IDP	Internally Displayed Person
IoU	Intersect Over Union
IPOB	Indigenous People of Biafra
KDE	Kernel Density Estimate
KNN	K-Nearest Neighbours
LDA	Linear Dimensional Analysis
mAP	Mean Average Precision
OCR	Optical Character Recognition
OD	Object Detection
R-CNN	Recurrent Convolutional Neural Network
RoI	Region of Interest
SOD	Salient Object Detection
SSD	Single Shot Multiple Detection
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UCF	University of Central Florida
VOC	Video Object Classes
XGBoost	Extreme Gradient Boosting
YOLO	You Only Look Once

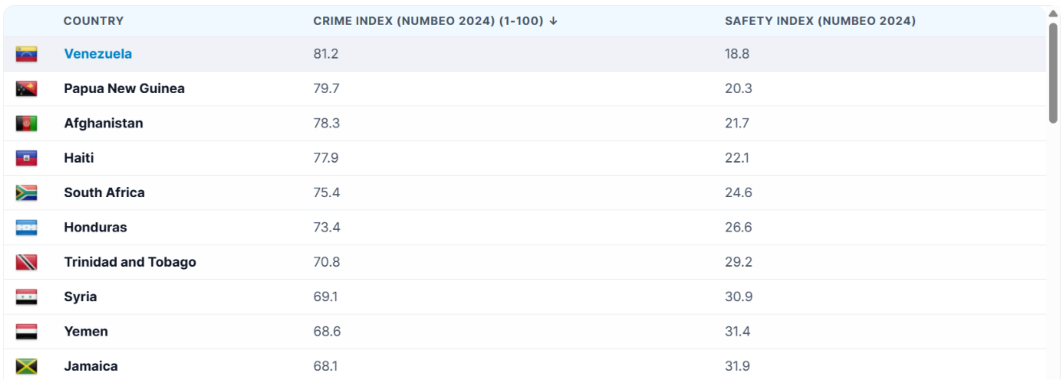
ABSTRACT

Crime significantly challenges socio-economic development, hindering societal progress and stability. This PhD Thesis focuses on training and comparing YOLOv4 with SSD models for classifying high-risk perpetrators. The annotated training dataset comprises approximately 3,118 images, including common weapons such as shotguns, handguns, rifles, and knives. It was sourced from the UCF and the NIST criminal databases. The secondary sources are trusted, compliant with data privacy laws, and personal identifiable information was anonymized before used for training. The models' validation shows effective object detection, with YOLOv4 achieving a mean Average Precision (mAP) of 90.58%. It achieves a 75.99 % precision for identifying persons of interest between 4,000 and 10,000 epochs. This model excelled most at detecting knives, achieving a precision of 96.11% with 80 instances predicted correctly (TP) with 2 incorrect prediction (FP). The recall rate of 74.5% for YOLOv4 indicates the model identified about 74.5% of actual positive instances. Moreover, with an F1 score of 77.2%, our findings highlight a balance between precision and recall. However, there is room for improvement due to 259 (FN) missed objects out of the 1205 total predictions. In contrast, the SSD model achieved 66.5% precision for persons, with its highest precision for handguns at 81.42 %. But adjusting configuration variables (hyperparameter tuning) improved the model's mAP to 84.19%, indicating better generalization and prediction across classes. Both architectures performed well by reducing misclassification including maintaining relatively high true positive and low false positive rates across all classes. To enhance pre-crime analysis, the YOLOv4 model was integrated with the Deep-SORT tracker to maintain object identities over time and improved the model's ability to identify weapon sub-objects on individuals and categorize potential perpetrator as high-risk.

CHAPTER ONE

Introduction

The recent rise in global security has become a major concern for stakeholders, especially in the context of global governance. Criminal groups are constantly coming up with new strategies that security forces struggle to curtail all over the world. Yearly, governments, corporations, independent groups, and military forces expend vast amounts of resources to fight against the universal threat to peace and harmony. But there is a downside to using antiquated methodologies. Fig 1.0 shows the recently released 2024 world crime index and ranks, and we can see how Venezuela occupies the highest crime index of 81.2 and with low safety index of 18.8. By contrast, Jamaica has a crime index of 68.1 and a safety index of 31.9. Every country in the world has varying degree of crime indices. This implies no country in the world that is without security issues.



COUNTRY	CRIME INDEX (NUMBEO 2024) (1-100) ↓	SAFETY INDEX (NUMBEO 2024)
Venezuela	81.2	18.8
Papua New Guinea	79.7	20.3
Afghanistan	78.3	21.7
Haiti	77.9	22.1
South Africa	75.4	24.6
Honduras	73.4	26.6
Trinidad and Tobago	70.8	29.2
Syria	69.1	30.9
Yemen	68.6	31.4
Jamaica	68.1	31.9

Figure 1.0. Top 10 Crime Index [1]

According to Agaga in [2], while developed countries are deploying technologies such as facial recognition and intruder detection systems to detect crimes, there remains a need for systems that utilize machine learning and computer vision to predict crimes before they occur. As a result, researchers are making efforts to develop innovative technologies that can enable society to effectively combat the rising wave of crime all over the world. With the rapid advancements in artificial intelligence, this area of research has significant promise applications prospect.

1.1 . The Transformative Power of Artificial Intelligence in Crime Prediction

Artificial intelligence (AI) is an ever-growing field looking to mimic the human brain. It is a field that thinks rationally and acts radically on notions of problem solving and innovation. Artificial intelligence (AI) is among the leading trends of modern technology, whose goal is to replicate human brain cognitive functions. With AI growing in remarkable pace, its impact is felt across many sectors; from manufacturing and communication technologies, to education, business and agriculture, as well as social media, health and national security. One of the growing areas of study at present is the potential of AI to predict crime before it happens. Through the application of machine learning, computer vision, and deep learning methods we can analyze, detect and predict possible criminal behaviors in the society. Figure 1.1 depicts the fundamental elements that define the components of artificial intelligence.

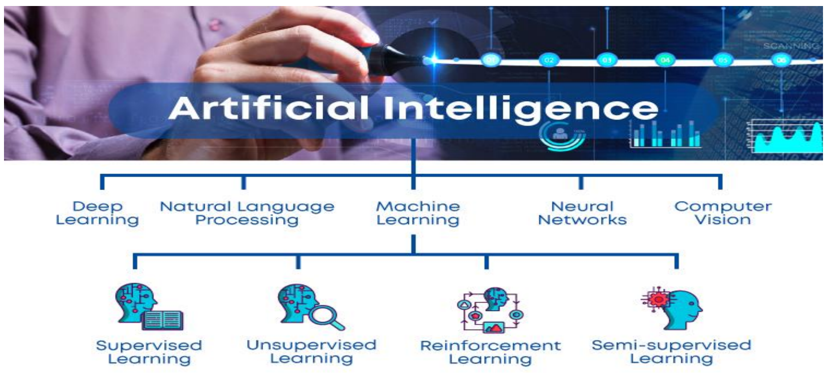


Figure 1.1. Spectrum of Artificial Intelligence [3]

Computer vision is concerned with machine interpretation and understanding of visual information. Object recognition, scene analysis, and anomaly detection are techniques employed in this field. AI systems can process images and videos for relevant patterns by these means. Deep learning, on the other hand, means training algorithms to tackle substantial amounts of data, enabling them to learn from experience and therefore make intelligent choices. The combined effect of these technologies delivers the ability of machines to perform such tasks as image retrieval, object tracking and behaviour classification, all of which are essential for accurate crime prediction.

In crime prediction, AI systems process big datasets to detect patterns and correlations that suggest criminal behaviour, this is the term referred to as Machine Learning. For example, machine learning-based algorithms may exploit historical crimes, social media posts, and environmental data to anticipate where crimes and criminals will emerge. Through such techniques as pattern recognition and binary classification, these algorithms are able to differentiate between normal from abnormal behaviours. This knowledge is an essential intelligence to law enforcement agencies. Feature extraction is a key part of Machine Vision applications. It involves the identification and isolation of image attributes that are important for decision making. These attributes may be low-level one such as edges and corners. It could also be a high-level, one in which domain-specific knowledge related to the objects and activities analysis is necessary.

Therefore, the performance of image-processing and machine learning algorithms evaluated with metrics such as mean average precision (mAP), a measure that shows the accuracy of crime predictions. Within AI and computer vision research, one of the most established areas is that which concerns Convolutional Neural Networks (CNNs). Unlike traditional binary classification methods, CNNs can process temporally dynamic spatial data like video information from surveillance cameras. It is this feature which makes it possible to understand an object's movement and interaction over time, giving more precise predictions for criminal behaviour. Consequently, AIs entry into crime prediction could be beneficial in the public safety technology- By combining machine-learning with computer vision, law enforcement agencies can predict impending felonies, assign resources more efficiently and essentially make neighborhoods more secure. As AI technology advances, crime prediction will increasingly become an integral component of its capabilities.

1.2. The Effects of Machine Vision on Crime Prediction and Public Safety

Today, machine vision is one of the arms of artificial intelligence that could improve crime prediction and enhance public safety. It integrates the concepts of advanced computer vision, deep and machine learning with neural network technologies to recognize and predict crime in crime scenes. In the context of this research, the system uses the capabilities of HD (High-Definition video) and real-time crime video of surveillance networks as input datasets. As shown in Figure 1.2, the process begins with image capturing from camera, image preprocessing, feature extraction, pattern recognition and decision making. Machine Vision can recognize crime weapon on human subjects in pre-crime video or real-life scenario. Also, it has the potential to study behavior, discern anomalies and predict criminals in the environment.

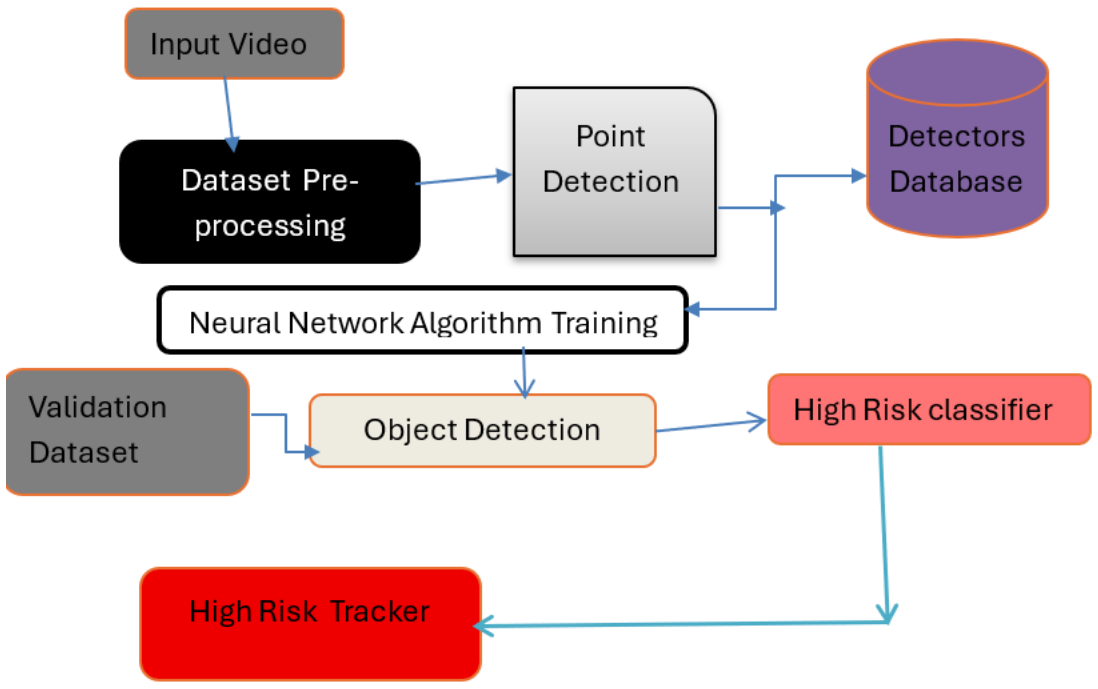


Figure 1.2. Machine Vision for Crime Prediction

1.2.1 The Advanced Analytic Functions of Machine Vision

Machine vision advanced systems utilize machine learning algorithms to analyze visual information and detect objects of interest, such as suspicious human movement, crime weapons in the pre-crime video. This technology can allow law enforcement agencies to improve their coverage of high-risk networks and thus improve operational efficiency. Additionally, with combining it with predictive analytics, machine vision systems find application in prediction analytics and criminal hot spots, helping to monitor crime in those areas from resources and interventions.

1.2.2 Adaptation and Lifelong Learning through Machine Learning Algorithm

In furtherance, the complete implementation of machine learning in computer vision enhances the accuracy and robustness of machine vision instrument. These algorithms evolve when exposed to a new dataset, so that they appropriately adapt to predict new patterns of crime. It is this ongoing learning curve that keeps security agencies ahead of future crimes-constantly adjusting to the new landscape of criminal activity. Figure 1.3 depicts the machine learning pipeline that illustrates visual design of the predictive instrument. The diagram shows key stages involved in developing a Machine Learning Model, capable of predicting crime in the surroundings.

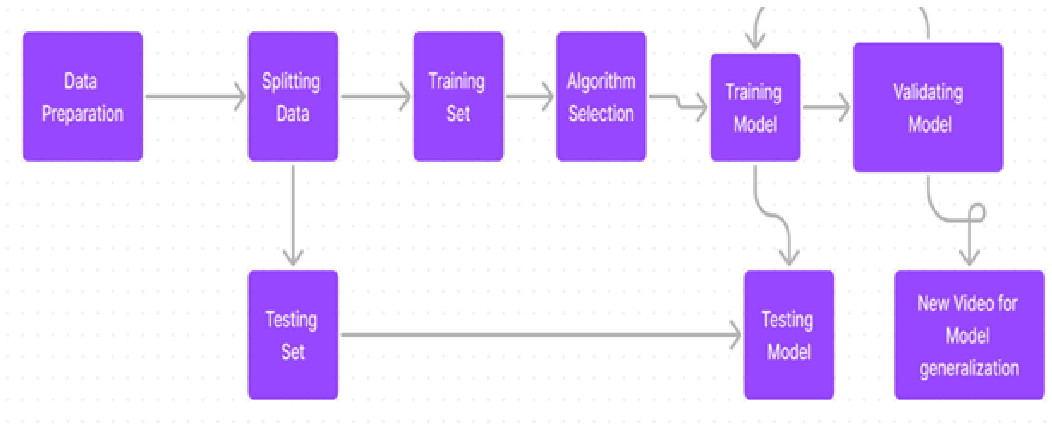


Figure 1.3. Simplified Machine Learning Pipeline for Crime Prediction

1.3 Research Background

Data from research conducted by Jha in [4] demonstrates that economies and populations grow crime rates are likely to increase in geometric progression, reinforcing the need for initiative-taking crime prediction systems. The rising crime rates, particularly in technologically advancing economies, have long made analytical pre-crime modelling a subject of interest. However, predicting every crime actor accurately using computer vision remains a significant challenge.

The implementation of these technologies has the potential to reduce damage from terrorism and crime. The Millennium Development Goals highlight the need for innovative technologies to secure lives, including AI-based forecasting systems. These systems aim to provide early crime warnings used in global perspectives. Countries such as the U.S., U.K., Germany, and Switzerland have adopted crime control measures supported by advanced technologies [5]. This development reduces dependency on traditional policing. While law enforcement traditionally relied on historical data, the emergence of analytical and predictive modelling has revolutionized crime mapping.

The first attempt at using computer vision for crime prediction dates to 1998, but early systems limited by computational power. With modern high-performance computing, Computer Vision with deep machine learning systems can now detect, recognize, and forecast criminal activities more effectively.

As illustrated in Fig. 1.1, the field of computer vision is a subdomain of artificial intelligence. It is concerned with enabling machines to interpret, analyze, and understand visual information from the world, simulating aspects of human vision [6]. Rooted in early research on image processing and pattern recognition in the 1960s and 1970s, the theoretical foundations of computer vision build upon mathematical models of image formation, filtering, segmentation, and feature extraction [7]. Fundamental techniques such as edge detection, corner detection, and region-based segmentation derived from signal processing and linear systems theory.

Object detection, a core problem within computer vision, involves not only recognizing the presence of objects but also determining their precise locations within images or video streams. Researchers like Lowe in [8] and Dalal in [9] demonstrated in their respective studies, how traditional methods for object recognition often relied on handcrafted features. Notable among these are the Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG), which extract key local features invariant to changes in scale and illumination. These techniques laid the groundwork for machine learning-based classifiers such as Support Vector Machines (SVMs) and Random Forests, which widely used in early detection pipelines.

The change in basic assumptions came with the rise of deep learning, particularly Convolutional Neural Networks (CNNs), which automate hierarchical feature extraction and significantly outperform traditional approaches in image classification and detection tasks as demonstrated by Krizhevsky in [10].

CNNs leverage spatial locality through convolutional layers, allowing for scalable and end-to-end learning of visual representations. Architectures like Region-Based CNN (R-CNN), YOLO (You Only Look Once), and SSD (Single Shot Multibox Detector) further extended CNN capabilities by enabling real-time object detection and localization, as evident in the work published by the authors in [11], [12] and [13] respectively.

These advances have led to the integration of computer vision and object detection across various domains such as autonomous driving, facial recognition, medical diagnostics, and surveillance systems, where robust visual interpretation is critical [14]. Theoretical frameworks from statistical learning theory, information theory, and Bayesian inference continue to underpin modern developments in this field.

1.3.1 Artificial Neural Network Paradigm

Yasar in [15] describes neural networks as a computational system inspired by how biological networks operate. It is a term sometimes related to neurons according to the paper published by Zhang in [16]. The study by Ou in [17], refers to neuron computing as the computing systems that attempt to function like the human brain. The brain's essential functions, such as pattern algorithm, motor control, and ability to perceive objects, flexibility in inference, insightfulness, and judgment in any circumstance outlined to replicate in the design of such a neuro-computing system. To achieve those goals, neuro-computing adopts various algorithms that help analyze problems, learn from experience, and make decisions based on the data gained from the environment. Artificial neural network (ANNs) developed to solve problems in pattern recognition, prediction, optimization, and memory associative problems.

1.3.2 The Purpose of Artificial Neural Network

Learning regarding artificial neural networks includes updating network connection weights to allow a network to function as designed. The network saddled with learning effectively and sufficiently for the model to generalize on unseen dataset. It must learn with sufficient weight to map the input-output relationships for the new scenario. This primes the neural network over the conventional learning processes. Since the networks do not biologically engineer therefore, all neural networks in the field of artificial intelligence regarded in a broader context as artificial neural networks.

The artificial neural network has successfully helped scientists and researchers to achieve computational and prediction challenges. They are volume parallelism, Adaptivity, Generalization capabilities, Object learning, Large-volume information processing, and Fault tolerance consumption of lower energy. Parallelism has aided modern-day computer systems to compete with the human brain [18].

1.3.3 The Deep Learning Revolution

The rise of deep learning in the early 2010s and its ascent to dominance in AI can be seen as a paradigm change in modern computing history. There had only been theoretical hints of such networks for many years, instantiated for the first time into a set of applications for Automatic Speech Recognition (ASR). As shown by [19], DNNs reached record-breaking ACC in ASR tasks, and DNNs have promptly become the new standard in both academic and industry research [20].

The notable change was achieved when [10] introduced AlexNet, a CNN with multiple layers that achieved good performances on the ImageNet classification challenge. The pair's efforts had a good impact - they were able to bring down error rate from 26% to 16% in a year, far surpassing the previous annual reduction rate of 2%. This discovery was globally recognized and the catalyst for the deep learning revolution.

The rapid acceptance of DNNs was facilitated by the main technological breakthroughs. The transformation also occurred with the reuse of Graphic Processing Units (GPUs) that were initially architected for game graphics. These parallel computers were able to speed up the matrixial calculations that are essential to the training of neural networks, making practical what before was heavily time consuming, and ideally quite impossible. Equally important was the emergence of large, annotated datasets like ImageNet's 1.3 million images. According to Russakovsky in their work, the datasets provided the essential training for these models [21].

It was these pieces fitting together that created a market shift across the industry. Acknowledging the empirical success of DNNs, large technology companies made significant investment into the field. The advances presented by Abadi, Paszke, and Chen in in their distinct papers show how this improvement contributed to the rise of open-source frameworks, such as TensorFlow[22], PyTorch[23], and MXNet[24], which abstracted much of the additional layers of complexity away with features such as automatic differentiation, and scalable optimization. The culmination of these two advances have spawned a cycle of innovation significant to the advancement of artificial intelligence and computer vision projects to this day.

1.3.4 Deep Neural Networks in Structured Data

Deep Neural Network (DNN) is a generic class of architecture, which includes Differential Function Networks (DFNs) with generalized DAG connectivity structure. Such a formalized arrangement used to map inputs to their corresponding outputs, providing a framework for various learning tasks. One of the ways to increase the flexibility of linear models is to apply a feature transformation, substituting the input x with a transformed version $\varphi(x)$. For example, in one dimension (1d), we could use a polynomial expansion as shown in equation 1.0

$$\varphi(x) = [1, x, x^2, x^3, \dots] \tag{1.0}$$

Bishop in his published work referred to this equation as the basic function expansion [25]. The resulting model gives:

$$f(x; \theta) = W\varphi(x) + b \tag{1.1}$$

As shown above, the equation (1.1) remains linear in parameters $\theta = (W, b)$, preserving convexity in optimization as described by Goodfellow in [14]. However, manually designing ϕ(x) is restrictive. It may be worth noting that θ in this respect represents the feature index for each class and the corresponding parameters threshold values.

A more powerful approach is for the system to learn the feature transformation by introducing other parameters like θ_2: In this regard, the function becomes.

$$f(x;\ \theta) = W\varphi(x;\ \theta_2) + b \tag{1.2}$$

Here, w and b are learnable parameters and $\theta = (\theta_1, \theta_2)$ and $\theta_1 = (W, b)$. By recursively stacking such transformations, we obtain increasingly complex functions.

If we compose L (the loss function over a region)

We derive equation 1.3 as follows:

$$f(x;\ \theta) = f_L\,(f_{L-1}\,(\ldots f_1\,(x) \ldots)) \tag{1.3}$$

Where each f_ℓ(x) = f (x; θ_ℓ) is a layer of the model. This is the essence of deep neural networks (DNNs), which this research builds on to achieve the object detection in the video datasets used.

The term DNN encompasses models where differentiable functions are composed into any directed acyclic graph (DAG) [14].

Murphy in 2022, in their book published on probabilistic Machine Learning describes equation (1.2) as the simplest case chain-structured DAG known as a feedforward neural network (FFNN) or multilayer perceptron (MLP) [26].

DNN structure has the capacity to forward data layer by layer in nested nodes. The simplest illustration of this given in equation (1.0), where the DAG is just a linear chain. This architecture is typically known as a Feedforward Neural Network (FFNN) or Multilayer Perceptron (MLP). This type of MLP assumes a fixed-dimensional input in the form of x ∈ $\mathbb{R}^D$, referred to as single layer. The downside of it is its limitation to structured data alone. It typically stored in an N × D matrix according to Bishop in [25].

Here, D= the dimensionality of the vector, equivalent to the number of the input features, X = The input x is the feature covariates, sometimes called the predictors. In this research, it could be the height, weight of an objects of a class person, shotguns, long gun, knife or rifle as depicted in equation 1.4.

N = The data sample sizes.

Mathematically;

$$f(x;\, \theta) = I(w^{\mathrm{T}}x + b \geq 0) = H(w^{\mathrm{T}}x + b) \tag{1.4}$$

Where:

- *H* is the Heaviside step function.
- *w* and *b* are learnable parameters.

1.3.5 Multilayer Perceptron (MLPs)

From equation (1.4) we can see that Perceptron considered as a deterministic variant of logistic regression, defined as:

$$f(x;\, \theta) = I(w^{\mathrm{T}}x + b \geq 0) = H(w^{\mathrm{T}}x + b) \tag{1.5}$$

Where:

- *H* is the Heaviside step function (a linear threshold function),
- *w* and *b* are learnable parameters.

Unfortunately, this linear model has limitations in data science as it cannot predict output for nonlinear cases like detecting static and motion objects. This makes it not effective for our research.

To make the equation applicable to our research, we introduced hidden layers between input and output, allowing the network to learn nonlinear decision boundaries, as depicted in the study conducted by Bishop in [25]. By stacking perceptron with nonlinear activation functions (e.g., sigmoid, ReLU), MLPs can approximate arbitrary functions [27] and [28]. This mostly achieved in practice by using a differentiable MLP, which makes our model training easier. But there is still a potential problem since the Heaviside function has zero gradient. Such models in practice are difficult to train, limiting their practical use as demonstrated by [14]. To enable gradient-based optimization in the research, we replace *H* with a differentiable activation function $\varphi : \mathbb{R} \rightarrow \mathbb{R}$. This allows the network to learn via backpropagation [29].

At each layer *l*, the hidden units z_1 computed as:

$$z_1 = f_1 \, (z_{1-1}\,) = \varphi_1(b_1 + W_1 \, z_{1-1}\,) \qquad (1.6)$$

In scalar form, the k'th neuron in layer *l* given by:

$$z_{k1} = \varphi_1 \left(z_1 + \, \sum_{j=1}^{z_1 - 1} W_{jklzjl-1}\right) \qquad (1.7)$$

where:

- $\boldsymbol{W_1}$ is the weight matrix at layer l,
- $\boldsymbol{L}$ *is the layer*
- $\boldsymbol{b_1}$ is the bias vector,
- $\boldsymbol{\varphi_1}$ is the activation function (in this research, we choose sigmoid activation function.

The pre-activation (weighted input) now represented as:

$$a_1 = b_1 + W_1 z_{1-1} \tag{1.8}$$

Also, the activation achieved by applying φ_1

$$z_1 = \varphi_1(a_1) \tag{1.9}$$

Today, the term *"MLP"* always refers to this differentiable version, trained with backpropagation, rather than the original non-differentiable perceptron model..

1.3.6 MLP for Image Classification

The evolution of neural networks tailored for image processing has transitioned from multilayer perception to advanced convolutional neural networks (**CNNs**) utilized today. As demonstrated by LeCun in [30], MLPs proved effective for image classification tasks by processing flattened versions of two-dimensional images across fully connected layers series leading up to remarkable accuracy rates exceeding 97% on NIST datasets.

However, this approach faced other fundamental limitations regarding visual data handling since transforming images into one-dimensional vectors resulted in loss of essential spatial relationships among pixels that delineate visual patterns.

This architectural choice not only hampered parameter efficiency but also obstructed capturing hierarchical features present within visuals from simple edges up towards complex object representations. The breakthrough emerged with CNN development first presented by LeCun, specifically focusing on backpropagation applied toward handwritten zip code recognition tasks which retained two-dimensional structuring through specialized convolutional operations unlike their MLP predecessors.

As articulated in Goodfellow et al.'s comprehensive text on deep learning published in 2016, CNN architecture leverages three key principles. They considered good choices in enhancing suitability for visual tasks: local receptive fields scanning features; shared weights which reduce parameter counts and spatial subsampling, fostering translation invariance.

This progression from MLP architecture towards CNN signifies more than incremental enhancement. It indicates substantial shifts concerning how neural networks process visual information collectively and converging toward achieving human-level performance against challenging image recognition benchmark. Following this development, this research fully utilizes Convolutional Neural Network for the Model developments.

1.3.7 The Convolutional Neural Networks Mechanics

The models that solve object detection and image classifications often use Convolutional Neural Networks these days. CNN manipulates high-dimensional inputs and inputs with number of features. Therefore, the research adopts CNN framework comparing the performances of two detectors, YOLOv4 and Single Shot Multi-Box Detection (SDD) Models.

The Convolutional Neural Network consists of the convolutional layer, the pooling layer, the activation layer, and fully connected layers. The first layer takes the image of the input and convolves it with multiple learned parameters that will give rise to new 2-D feature maps. In this work, the learned parameters referred to as the threshold and weight, sometimes called the kernels or filter. Each of the resulting kernels is a 2-D matrix and smaller in size than the image it applied. The filter comprises values equivalent to the learned weight. The pooling layer sandwiched between the successive Convolutional Neural Network (ConvNet) layers. It aims at reducing map resolutions, improving spatial invariances to minor shifts, and lowering the memory requirements during execution. Popular pooling techniques are maximum pooling, mean pooling, mixed pooling, and spatial pyramid pooling.

Another crucial element of CNN is the activation function (As seen in *equation 1.9*). The activation function simply helps state the output of a node from a given input or series of input values. It is like the binary action of the neuron to propagate or stop a potential output. Typically, examples are jump functions, sigmoid, and Rectilinear Linear Unit (ReLU). For image classification tasks, Sigmoid and Jump are better while ReLU used where unlimited output required. But this work has utilized Sigmoid for its advantage in model training and development.

The last layer referred to as the fully connected. Here, all the feature maps and filters have now become 1*1 as shown in Figure 1.4.

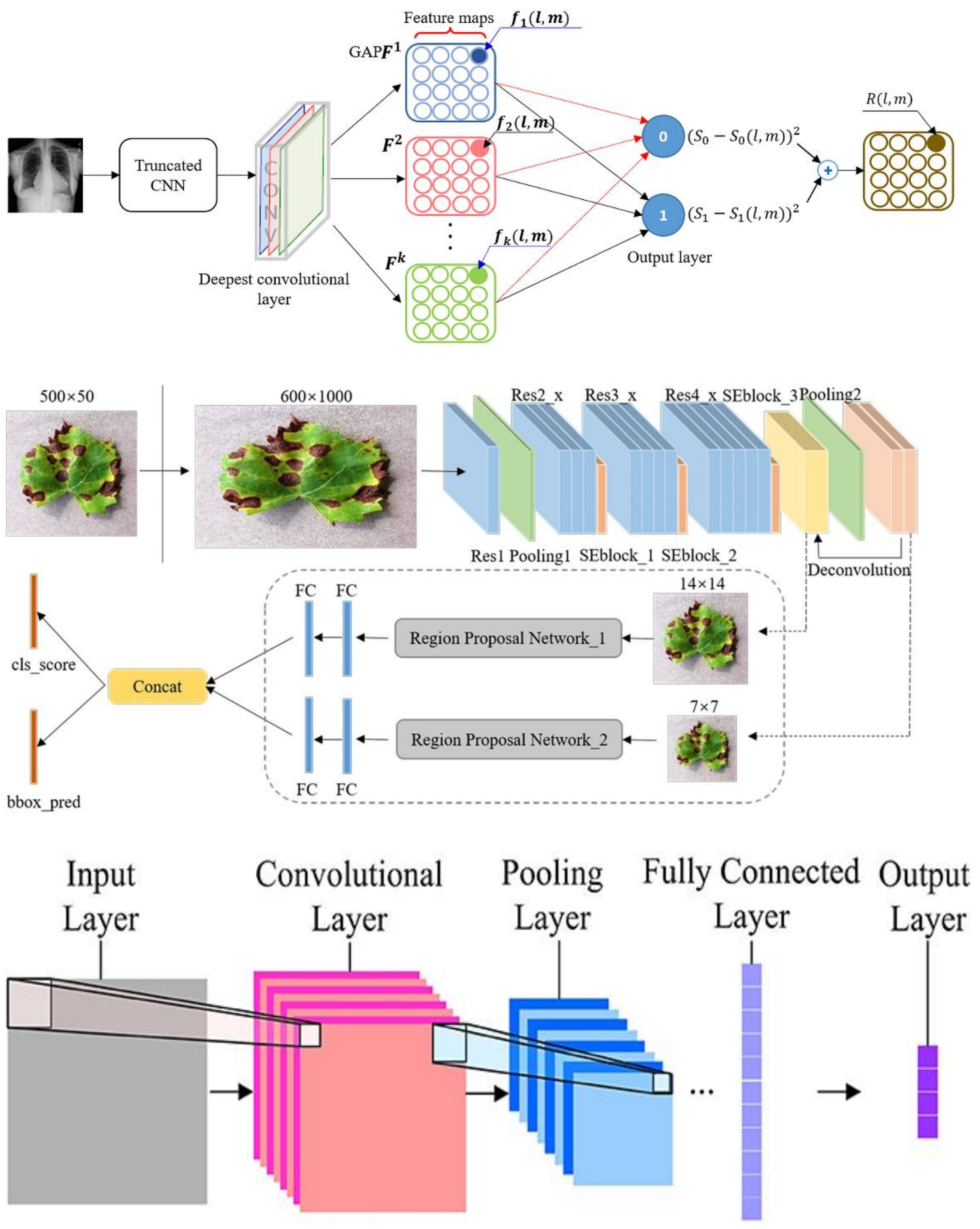


Figure 1.4. The Convolutional Neural Network Components [3]

1.3.7.1 The Key Components of Convolutional Neural Network

1. **Convolutional Layer:** The convolutional layers are responsible for applying learned filters (kernels) to the input image. This process helps detect important features like edges, corners, and textures. Each convolution operation involves

computing the dot product between the filter and a local region of the image .The result of this operation is a *feature map* in equation 1.9

The equation for the convolution operation is as follows:

$$Y\,(i,j) = (\,X * K)(i,j) = \; \sum_m \sum_n X\,(i+m, j+n).K(m,n) \qquad (1.10)$$

Where: $Y\,(i,j) =$ is the output value at position $(i,j), X$ is the input image or feature map, K is the kernel or the filter. Also, * denotes the convolution, $\sum_m \sum_n$: is the summation over the kernel dimensions. And the $(i+m, j+n)$ represents the indexing account for flipping in convolution.

1. **Pooling Layer:** Due to the equivariance phenomenon, Convolution would often preserve the information about the positions of input features. For instance, in this study, it is of interest to know if a combination of unacceptable behaviours of an actor is present anywhere in the video images. Pooling Max is one way to achieve this. It is a concept that helps to compute the maximum over its incoming values. A similar result reached with average pooling. Average pooling accomplishes the task by replacing the max with the mean value. In any case, the output neuron has a similar response no matter the positions of the input pattern occurrence within the receptive field as demonstrated by Murphy in [26]. Averaging this over all the locations in a feature map termed the global averaging pooling. Invariably, it passed the feature map of the image into a 1*1*D dimensional feature map, reshaped to a D-dimensional vector, passed its output to a fully connected layer that mapped it to C- C-dimensional, and finally passed the result to a SoftMax output. In this design, CNN created by alternating convolutional layers with max

pooling layers. The result followed by a linear classification layer at the end as shown in Fig. 1.4

2. **SoftMax Activation function used in the design:** The activation function used in the CNN Model design called the Softmask function. This is because as represented in the convolutional operation, equation 1.9, SoftMax takes a vector m of K real number and normalize to probability distribution proportional to m exponent such that each component sums to 1 or is in interval of (0,1) without any negative numbers.
3. **The Normalization Layers:** This is an extra layer added to CNN in this design to scale up models against vanishing or exploding gradients. The extra layers help to standardize the statistics of the hidden units. Batch normalization is the most popular normalization layer and was used in the system model training.
4. **Batch Normalization:** Batch normalization (BN) helps to ensure the distribution of all the activations within a layer is zero mean and unit variance. In this study, frozen batches of norm layers combined with the preceding layers to increase speed. One of the known advantages of using BN in the work is smoother optimization landscapes, which enhances the overall accuracy of our model.

1.3.7.2 Backpropagation Algorithm

Back propagation, an iterative algorithm used in deep learning training, especially in feed forward method. When assumed that the computation graph is a simple linear chain of layers, as in a Multi-Layer Perceptron (MLP). Backpropagation in this case is the repeated application of the chain rule of calculus. The method extends to more complex architecture, including arbitrary directed acyclic graphs as seen in equations 1.1, 1.2 and

1.3). This general procedure, often referred to as automatic differentiation, is the backbone of machine learning algorithms.

In this research, it helps us to compute the loss function gradient of the output network relative to the design parameter used in each of the convolutional layer [26] .The phenomenon minimized lost function. In addition, back propagation helps to pass down the gradient of the model loss function to the optimization algorithm where the values of the hyperparameters adjusted at the best weight, learning rate and biases to improve the performance of the model at validation.

Back propagation uses two processes, the pass forward and pass back. Based on equations 1.1, 1.2, and 1.4, we consider the inputs to the neural network as (a, b, c and d). It utilizes a specified activation function and a predefined hidden layer. Figure 1.5 illustrates back propagation process with diagram.

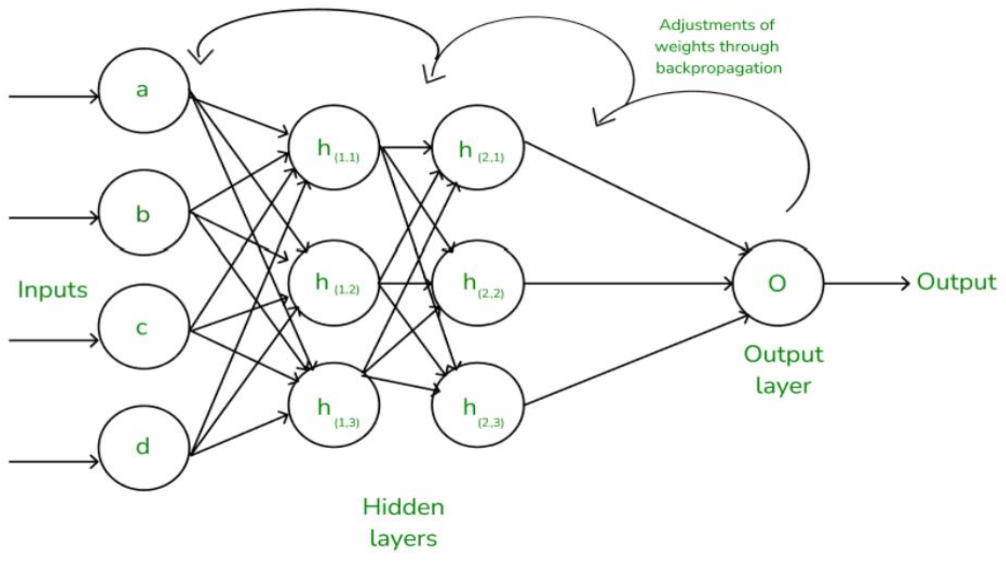


Figure 1.5. Backward Propagation Principle [31]

1.3.7.3 Forward vs Reverse Mode Differentiation

Forward mode differentiation efficiently computes derivatives by propagating tangent values during the forward pass, ideal for scenarios with fewer inputs than outputs. Conversely, reverse mode differentiation, used in backpropagation, calculates gradients by moving errors backward, optimizing neural networks by enabling effective weight updates based on gradients.

If we consider a function o = f(x), where $x \in \mathbb{R}^n$ and $o \in \mathbb{R}^m$. We can assume that f defined as a composition of functions $f = f_4 \circ f_3 \circ f_2 \circ f_1$ (1.11)

Where each f_i is a function that maps from one vector space to another.

Where $f_1 : \mathbb{R}^{n} \rightarrow \mathbb{R}^{m1}, f_2 : \mathbb{R}^{m1} \rightarrow \mathbb{R}^{m2}, f_3 : \mathbb{R}^{m2} \rightarrow \mathbb{R}^{m3}, and\ f_4 : \mathbb{R}^{m3} \rightarrow \mathbb{R}^{m}$. The intermediate steps needed to compute

$$o = f(x)\ are\ x_2 = f_1(x), x_3 = f_2(x_2), x_4 = f_3(x_3), and\ o = f_4(x_4). \qquad (1.12)$$

We can compute the Jacobian $J_f(x) = \frac{\partial o}{\partial x^T} \in \mathbb{R}^{mxn}$ using the chain rule method in calculus:

$$\frac{\partial o}{\partial x} = \frac{\partial o}{\partial x_4}\frac{\partial x_4}{\partial x_3}\frac{\partial x_3}{\partial x_2}\frac{\partial x_2}{\partial x} = \frac{\partial f_4(x_4)}{\partial x_4}\frac{\partial f_3(x_3)}{\partial x_3}\frac{\partial f_2(x_2)}{\partial x_2}\frac{\partial f_1(x)}{\partial x}$$

$$= J_{f4}(x_4)\, J_{f3}(x_3)\, J_{f2}(x_2)\, J_{f1}(x) \qquad (1.13)$$

To compute the Jacobian $J_f(x)$ efficiently.

We could recall from mathematical function that

$$J_f(x) = \frac{\partial f(x)}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{pmatrix} = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_1} \right) = \in \mathbb{R}^{m \times n} \quad (1.14)$$

Where $\nabla f_i(x)^T \in \mathbb{R}^{1\,x\,n}$ is the ith row (for $i = 1 : m$) and $\frac{\partial f}{\partial x_j} \in \mathbb{R}^m$ is the jth column (for $j = 1 : n$).

Note that, in our notation, when $m = 1$, the gradient, denoted $\nabla f(x)$, has the same shape as x. It is therefore a column vector, while $J_f(x)$ is a row vector. In this case, we therefore technically have

$$\nabla f(x) = J_f(x)^T \quad (1.15)$$

We can extract the ith row from $J_f(x)$ by using a vector Jacobian product (VJP) of the form $e_i^T J_f(x)$, where $e_i \in \mathbb{R}^m$ is the unit basis vector. Similarly, we can extract the jth column from $J_f(x)$ by using a Jacobian vector product (JVP) of the form $J_f(x)\, e_i$, where $e_i \in \mathbb{R}^n$. This shows that the computation of $J_f(x)$ reduces to either n JVPs or m VJPs. If $n < m$, it is more efficient to compute $J_f(x)$ for each column $j = 1 : n$ by using JVPs in a right-to-left manner. The right multiplication with a column vector v gives:

$$J_f(x)\, v = \underbrace{J_{f_4}(x_4)}_{m \times m_3} \quad \underbrace{J_{f_3}(x_3)}_{m_3 \times m_2} \quad \underbrace{J_{f_2}(x_2)}_{m_2 \times m_1} \quad \underbrace{J_{f_1}(x_1)}_{m_1 \times n} \quad \underbrace{v}_{n \times 1} \quad (1.16)$$

This can be computed using forward mode differentiation;

Also, we assume $m = 1$ and $n = m_1 = m_2 = m_3$, the cost of computing $J_f(x)$ is $O(n^3)$ as highlighted by the Bishop in [26].

This results in the creation of multiple Jacobian matrices, each representing the partial derivatives of a function with respect to its input video dataset. The Jacobian matrix is crucial for understanding how the function (f), which indicates the model classifier's output operation across the five distinct classes in the criminal dataset. It varies based on the detected class from the input video data. The understanding is fundamental for optimization and gradient-based learning approaches used in for the parameter tuning of the SDD Model in this research.

1.3.7.4 Model Fitting in Convolutional Neural Network

CNN model fitting consists of the training of a convolutional neural network on labelled data where data feature parameters are learned in order to minimize errors in prediction on images and make system to generalise well on unseen datasets.

Typically, we build up the tree one node at a time. The loss function $\mathcal{L}(\theta)$ in equation 1.16 needed minimised to achieve this objective.

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \ell(y_1, f(x_n;\ \theta)) = \sum_{j=1}^{J} \sum_{x_n \in R_i} \ell(y_i, w_j) \tag{1.17}$$

Here is the walkthrough,

Suppose we loop at node i and let

$$D_i = \{(x_n, y_n) \in N_i\} \tag{1.18}$$

Be the collection of input samples that arrive at node i. Based on a selected feature j and threshold t, we define the left and right splits of these samples as follows:

$$D_i^L\ (j, t) = \{(x_n, y_n)\} \in N_{i,j} \leq t\} \tag{1.19}$$

$$D_i^R\ (j, t) = \{(x_n, y_n)\} \in N_{i,j} > t\} \tag{1.20}$$

We then select the best features j_i and corresponding threshold t_i that optimally splits the data using a chosen impurity measures (e.g. Gini index, information gain, or mean squared error) as represented in equation 1.21

$$(j_i, t_i) = \arg \min_{j \in \{1, \dots, D\}} \min_{t \in T_j} \frac{|D_i^L (j,t)|}{|D|} c(D_i^L (j,t)) + \frac{|D_i^R (j,t)|}{|D|} c(D_i^R (j,t)) \quad (1.21)$$

The set of possible thresholds, T_i for feature j can be obtained by sorting the unique values of $\{x_{nj}$

Overfitting and *underfitting* are two issues considered when we trained our model in this research so that the model achieved is generalizable. Overfitting occurs when a model not only learns the underlying structure in the training set but also the noise in the data causing it to perform poorly on unseen video dataset [14]. It is a scenario that arises when models are too complex compared to the size of their training set or when training goes on for too many epochs. On the other hand, *underfitting* occurs when the model is too simple to capture the underlying structure of the data, which results from weakly-regularized neural networks [32].

Many methods have been developed to solve these issues. Regularization methods like L1 (Lasso) (Least Absolute Shrinkage and Selection Operator) and L2 (Ridge) curtail large coefficients for overfitting, further shrinking model complexity. Dropout, another form of regularization in deep learning, introduces randomness by randomly turning off neurons at learning time, which prevents the model from over-relying on some features [33]. Cross-validation is another key tool to test and prevent overfitting by testing on multiple subsets of data as demonstrated by the author in [34]. To address underfitting, more expressive model more features or deeper networks, can be used along with more training time. The research conducted by Zheng in [35] demonstrated how feature engineering contributes to model expressiveness and bias reduction .Finally, both bias and variance need to be balanced, frequently referred as bias-variance tradeoff, to obtain optimal model performance. Handling overfitting and underfitting is essential in practical deployment of machine learning models, which often require it to be as accurate as possible and to generalize well.

In training this model, reverse mode differentiation (backpropagation) is employed for both YOLOv4 and SSD models. This method is effective due to the vast number of input features, like image pixels, versus fewer output classes. Backpropagation efficiently calculates gradients for model weight updates, making it ideal for complex neural networks in image processing tasks. Also, measures are taken during data acquisition, preparation and training that ensure our CNN model do not underfit or overfit. This shall be discussed in the subsequent chapters.

1.4 Aim of the Study

The proposed system aims to leverage Machine Learning, Deep Learning, and Computer Vision concepts to detect and recognize common weapons on person, thereby predicting potential perpetrators in pre-crime scenarios.

1.5 Study Objectives

The following are the specific objectives of the proposed system:

1. To acquire crime dataset, implement data annotation with Darknet/labelImg to label large crime video containing person, shotgun, handgun, riffle and knife classes.
2. To design Custom SDD and Yolov4 Convolutional Neural Network Models.
3. Develop Python scripts that trains and validate custom YOLOv4 and SDD Convolutional Neural Network Models.

4. Optimize Yolov4 and SDD CNN Models for enhanced performance and detection accuracy where necessary.
5. Compare the mean average precisions and performance of SDD with Yolov4 detectors for high-risk behavior classifier system.
6. Develop high-risk python scripts for high risk behavioral classification.
7. Develop an AI model capable of tracking the trajectory movement of person of interest in pre-crime scene using Deep-Sort algorithm, with bounding boxes over the person of interest and the detected weapon.
8. Develop a functional and scalable cloud-based AI Safety Software with high mean average precision (mAP), high recalling and high confidence scores.

To achieve the objectives and give the research global visibility, at a point, sections of the research findings presented in international conference or published in academic journal. They are:

1. Akinmuyisitan T.M., and Cosmas John (2024), Convolutional Neural Network Paradigm: Comparison of VGG16 and Resnet50 in Criminal Face Prediction, World Academic of Science and Engineering Technology Conference, Oct 2023,Vol18(1) available at: https://publications.waset.org/10013462/advanced-convolutional-neural-network-paradigms-comparison-of-vgg16-with-resnet50-in-crime-detection
2. Akinmuyisitan T. M., and Cosmas John. Enhanced Crime Prediction with Computer Vision - YOLOv4 Approach. Global Journal of Computing and Technology, USA, Vol. 24 No. D1 (2024): GJCST-D Neural & AI: Volume 24 Issue D1 **DOI:** https://doi.org/10.34257/GJCSTDVOL24IS1PG55

3. Akinmuyisitan T.M., and Cosmas John Advancing Real-Time Detection and High-Risk Person Classification in Pre-Crime Scenes: A Comprehensive Machine Vision Approach Utilizing SSD Detector Technology, USA, Vol. 24 No. D1 (2025): GJCST-D Neural & AI: Volume 25 Issue D1

   DOI: https://doi.org/10.34257/GJCSTDVOL25IS1PG51

4. Akinmuyisitan T. M and Cosmas John Deep Neural Network: Predicting Future Prices of Cryptocurrency Using LSTM and GRU (Paper Accepted at Elsevier Journal)

1.6 Research Question

This study answers the research question of how we could develop an AI system that can analyze persons of interest and predict potential criminal in a precrime video or real world scenario.

1.7 Contribution to Knowledge in Artificial Intelligence and Computer Vision

This thesis presents significant contributions to the fields of artificial intelligence and computer vision, serving as a valuable resource for scholars and future research. The specific contributions to the existing literature are outlined as follows:

A. **Comprehensive Framework for Data Acquisition and Pre-processing**: This research provides a detailed methodology for acquiring large-scale criminal video datasets from secondary sources, including the UCF database, NIST via Kaggle. It outlines essential data pre-processing techniques for scrubbing data sourced from the internet, which is crucial for training deep learning models. The thesis explicitly details data augmentation and annotation processes for both static and motion videos, utilizing tools such as labelImg . Additionally, it includes a conversion process for transforming image datasets from the YOLO format (.TXT) to Pascal VOC (XML format), facilitating SDD data training. These contributions are intended to assist academics and researchers in the field of computer vision and deep learning.

B. **Development of a SDD Neural Network Model for Weapon Detection:** The thesis details the training and development of a Single Shot Detector (SSD) neural network model utilizing a criminal dataset to detect weapons. This aspect of the research serves as a reference for future studies focused on weapon detection applications. Furthermore, the hyperparameter tuning process that enhanced the model's performance to achieve a mean average precision of 84.19% is documented, providing valuable insights for practitioners in the field.

C. **Implementation of the YOLOv4 Model for Crime Prediction:** The research includes the training and development of a YOLOv4 model using a criminal dataset to detect common weapons and classify potential perpetrators as high-risk within a controlled environment. This contribution highlights the practical applications of advanced machine learning and computer vision techniques in enhancing security measures.

1.8 Thesis Structure Overview

The rest of the PhD Thesis structure organized as follows; Chapter Two provides the synthesis of the existing literature and further gives the definitions of related terms in the field of Computer Vision, Machine Learning, and Deep Learning. Chapter Three presents the methodology on data acquisition and pre-processing for the two methods used. In Chapter Four, the research expounds Single shot Multi Detection Method of weapon and person of interest detection. The results and interpretation of the finding was also included in the chapter. Chapter Five contains the YOLOv4 Technique and, the development steps, results and interpretation. Finally, Chapter Six provides the conclusion, Recommendation and Future direction for the research

CHAPTER TWO

Literature Review

2.0 Introduction

This section reviews the related work on topics of artificial intelligence, computer vision, machine learning, deep learning, and related fields. Synthesizing prior work, it logically integrates these collections of research and the overall dearth of similar works on the topic. The synthesis helps place the research in the larger context of the existing literature in crime prediction with computer vision. Additionally, the chapter describes theories and terminologies related to the technologies to further help the audience understand in an uncomplicated way.

The rest of this chapter organized as follows:

- The Evolution of Crime Prediction Technology
- Traditional Crime Prediction Methods
- The Roles of Machine Learning in Crime Prediction
- Overview of Computer Vision
- Computer Vision Algorithms and Framework
- Applications of Computer Vision in System Security and Surveillance
- Advancement in the Convolutional Neural
- The Roles of Compute
- Challenges and Limitations in the Literature- technical and ethical challenges
- Identifying Research Gaps

- Conclusion of the Literature Review

2.1. The Evolution of Crime Prediction Technologies

Crime prediction evolved from its traditional observation-oriented methods to the advanced algorithm-based models of today. Crime prediction and prevention were formerly founded on traditional criminological paradigms, including Chicago School ecological approaches and Cesare Beccaria's deterrence theory. The research by Sampson in [36] assert that these models focused on routine activity, social disorder, and rational criminal decision-making. To forecast criminal behavior, law enforcement agencies relied on local experience, intuition, and crime mapping. The past methods may have lacked forecasting and data precision necessary in real-time intervention.

Crime prediction started to depend more on data in the mid-1900s with the advent of computers and geographic information systems (GIS). The use of crime mapping tools enabled the spatial analysis of crime hotspots, which led to the development of predictive policing. Theories such as environmental criminology and the broken windows theory [37], which postulated that outward manifestations of disorder incite additional criminal activity, had a major impact on this change. Methods like CompStat, created in New York City in the 1990s, showed quantifiable effects on reducing crime by using historical crime data to more efficiently allocate police resources [38].

The development of artificial intelligence (AI) and machine learning has been the key drivers behind modern crime prediction. Such technologies process vast amounts of data, such as demographic data, crime reports, social media posts and weather patterns, to more accurately forecast possible criminal activity. Predictive policing software such as PredPol and Hunch-Lab that predict future crime hotspots from historic crime report data used to assist law enforcement in taking preventive actions [39]. According to the study conducted by Richardson, while these systems are capable of increasing efficiency at a greater scale, they have drawn criticism for amplifying racial prejudices due to the over policing of minority communities contained within the data they rely on [40].

Another developing element is the use of real-time data feeds and surveillance tools, like social network analysis, gunshot detection and facial-recognition systems, to predict crime scenes. These emerging innovations are part of the set of changes commonly referred to by academics as "predictive surveillance," in which "law enforcement practices that merge predictive, data analytics and criminalizing technologies" [41]. While promising, this raises ethical concerns regarding civil rights, consent, and privacy. Balancing the rights of the individual against the public good is a central issue when it comes to the application of predictive technologies.

To conclude this section, the evolution of crime prediction from criminological models to AI-based systems accords with wider technological and social trends. Before the widespread availability of such tools, compile-time checks were used to enforce such invariants, but are still not able to verify code with fairness guarantees and ethical requirements for policing. These approaches was perceived as validation, logistic regression models predicts [42] and also as a mandatory prerequisite for the future of crime forecasts. It could build on transparency, accountability and community engagement that will assure the fairness of outcomes and reliability of policing that are necessary for public trust in surveillance [43]. This research gives priority to individual right, privacy and minimize model biasness in its developments.

2.2 Traditional Crime Prediction Methods

Traditional crime prediction techniques established the foundation for current crime detection method. The systems used simple statistical models to track patterns and predict criminal behavior. One such method which has been used for well over a century is regression analysis. This technique analyses the relationship between one or more independent variables (such as unemployment, education, or population density) and the rate of recorded crime as the dependent. Linear regression is widely used to predict the likelihood of crimes depending on the geographical areas. As demonstrated in by Braga, these procedures gave policy makers and law enforcement the tools to target resources according to their own statistical evidence [44].

Another important crime prediction technique is hot spot analysis. This has the potential to identify areas of high rates of crime in the society. The mapping and visualization of crime patterns using spatial data, often with analysis of nearest neighbor or kernel density estimation to detect clusters, became possible for law enforcement agencies. This approach became increasing popular in the 1990s in high-crime cities and resulted in tactical responses such as community policing efforts and hot-spot policing initiatives [45]. The advantages of hot spot analysis include its user-friendliness and visualization, which may help the police to make decisions.

The traditional methods were effective, but they had their limitations because regression predictions often relied on finite, static data, they had limited capability to adapt to dynamic patterns of crime. Hot spot mapping technology, while successful in identifying clusters of crime, also had methodological limitations. It tended to concentrate in certain neighborhoods, but not suitable to address the root causes of crime or account for displacement (e.g. when criminal activity spreads to nearby neighborhoods when enforcement is increased). The study by Brantingham in [46] emphasized that the the slate of prior usage can also serve to perpetuate systemic bias, with historical high crime areas that have been over-policed continuing to be represented as high-risk locations even if they no longer currently carry that distinction. This risked branding communities, and it added to public distrust of police approaches.

Moreover, real-time processing and adaptability were usually missing in traditional crime prediction systems. The later were mostly updated only once per year or month in some cases, so they are not appropriate for short-term crime monitoring or rapidly evolving environments. They were inadequate and could not reflect broader societal or situational conditions likely to influence offending perpetrators, as they were insulated from other potential data sources, including weather, traffic, social media, or emergency calls [47]. This constrained their suitability for preventive crime monitoring. As a result, despite the fact that traditional crime prediction techniques such as regression analysis and hot spot mapping provided a basic knowledge of patterns in crime, they were constrained by their static nature, over focusing on the simplification of complex variables, and tendency to confirm existing biases.

2.3 The Role of Machine Learning in Crime Prediction

Crime prediction has been revolutionized by machine learning (ML), which enables the analysis of big datasets. It is widely accepted that although traditional statistical methods can describe the global crime trends and patterns, machine learning (ML) algorithms such as support vector machines (SVMs), decision trees, random forests, and neural networks, have the capability of revealing latent patterns and nonlinear relations in crime data.

By analyzing both structured and unstructured data, these models offer deeper insights into criminal behaviour. Brantingham in their study established that in order to forecast crime hotspots and increase law enforcement efficiency, predictive policing systems now incorporate geospatial analytics and real-time data feeds [46]. However, high-quality data and careful algorithmic design to prevent reinforcing preexisting biases are necessary for these models to be effective.

Support vector machines (SVMs) can handle high-dimensional data, hence, they have shown particular efficacy in identifying high-risk crime areas. SVMs performed better than conventional hotspot mapping methods in the mapping of violent crime hotspots in the United Kingdom in 2024, as shown by Chainey and Tompson in their research [47]. SVMs were used more recently by [48], to predict burglary patterns in urban settings, with an accuracy rate of 89%. However, SVMs' "black-box"(inexplainable logic of the model in a simple way to users) nature can make them difficult to interpret, which is a major disadvantage in policing situations where openness is essential. According to Johnson in his published paper in 2024, SVMs may also require careful feature selection and parameter tuning [49].

Analyzing time-series and spatiotemporal crime data has made neural networks in particular, deep learning models more popular. Neural networks outperformed linear regression in predicting crime trends in Canadian cities, according to the author in [10]. Identifying long-term dependencies in crime patterns, the recent research conducted by Wang in [50] reveals the current developments like transformer-based architectures, which have further increased forecasting accuracy. But the instrument comes with notable disadvantages. In addition, the model are computationally intensive and operates on large historical databases, despite the ability to accurately predict forecasted events. Moreover, there remains a risk of algorithmic bias as a neural network based on biased historical arrest data could inadvertently entrench policing disparities, as was highlighted by Ferguson in [51].

Early machine learning approaches to crime prediction focused on pattern recognition and clustering. For example, the Apiah in [52] introduced k-means clustering model for classifying crimes into five classes which are murder, theft, rape, kidnapping and burglary by adopting data from NCRB. Although it was claimed that the model reached 95% accuracy, this approach could not effectively deal with nonlinear and multimodal data and hence did not capture and train on non-linear dataset. This highlights the need for more robust models capable of capturing intricate feature like images and video data input interactions, especially in heterogeneous urban settings.

In contrast, the author in [53] introduced a feature-level data fusion approach leveraging Deep Neural Networks (DNN) with four learned layers: spatial, temporal, environmental context, and joint representation. This model integrated environmental data (Google Street View images processed via AlexNet) to demonstrate a correlation between urban disorganization and crime likelihood. Although promising, the model's dependence on image-derived environmental features raises scalability and generalizability concerns in dynamic urban environments.

2.3.1 Classification of Machine Learning Algorithms

The Figure 2.1 shows the broad divisions of machine learning. Supervised, Unsupervised, Semi supervised and Reinforcement learning.

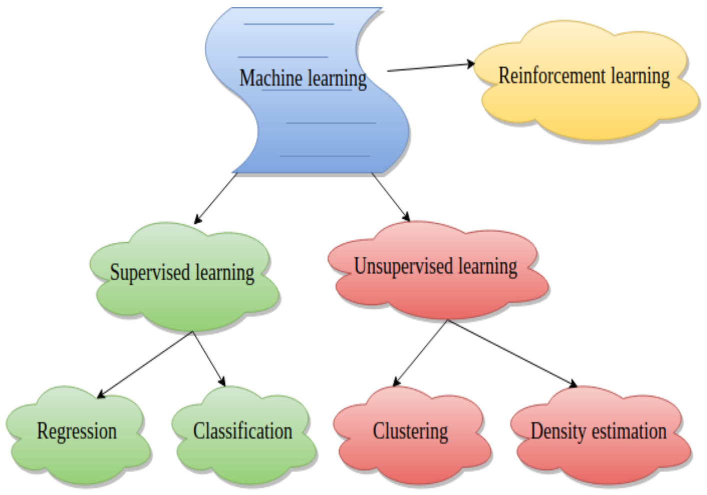


Figure 2.1. Machine Learning Division [3]

2.3.1.1 Supervised Machine Learning

In supervised learning, each input is paired with its corresponding output, and the model is trained on labelled data. In order to make precise predictions on unseen data, the model learns a mapping from inputs to outputs [14]. As shown in Figure 2.1, this paradigm includes Support Vector Machines (SVMs), Decision Trees, and Logistic Regression. In addition, Neural networks such as YOLO, SDD are example of supervised learning as they require labelled data to map input-output. They are better defined as model architectures trained using optimisation techniques like gradient descent and backpropagation, even though some of these are stand-alone algorithms [25]. The work of Sohn in [54] reveals that Supervised learning has demonstrated success across diverse domains. The number and quality of annotations directly affect model performance. Large labelled datasets are perfect for supervised learning. But overfitting, class imbalance, and label noise still exist, calling for regularization method and data augmentation [55]. Our research adopts this class of ML techniques and perform Data augmentation technique to increase the instances of the weapons and person classes in the training dataset.

2.3.1.2 Unsupervised Machine Learning

Unsupervised machine learning incline on approaches that analyze unlabeled data in order to discover hidden patterns or structures in data without having a prior knowledge about the output [54]. This approach is particularly effective in exploratory data analysis because the model works to uncover inherent correlations within complex statistical data, rather than predict results. Two commonly used approaches are dimensionality reduction and clustering techniques. With accelerated variants, large-scale datasets, K-means clustering is frequently used to separate data into discrete groups based on similarity. Hierarchical clustering is particularly effective for multi-level cluster analysis in fields such as the social sciences and biology [56]. Principal component analysis (PCA) is still crucial for dimensionality reduction in terms of eliminating noise and removing important features, even though advanced methods such as t-SNE and UMAP offer powerful tools for visualizing high-dimensional data [57].

Among the many notable applications of unsupervised learning are genomics analysis, anomaly detection, and customer segmentation. Retailers, for instance, use clustering to identify patterns in the purchases made by their customers, and anomaly detection techniques are essential for thwarting fraud and guaranteeing cybersecurity [58]. In genomics, unsupervised methods facilitate the analysis of single-cell RNA sequencing data. Unsupervised learning has drawbacks, including scalability problems when working with high-dimensional data and the continuous discussion about the optimal evaluation metrics for clustering outcomes. These difficulties show how much more research is required to improve the efficacy and efficiency of unsupervised learning strategies.

2.3.1.3Semi-Supervised Machine Learning

According to Belkin in [59], by combining a small amount of labelled data with a much larger pool of unlabeled data, semi-supervised learning builds upon the principles of unsupervised learning and capitalizes on the benefits of both methodologies This approach is particularly useful in domains such as medical imaging, where expert annotations demand substantial resources, where labelled data is scarce or expensive to acquire. Semi-supervised approaches can attain performance levels comparable to fully supervised models by first learning from the large unlabeled dataset and then refining with the few labelled examples, while significantly minimizing the need for manual labelling [60].

In a number of fields, semi-supervised learning has shown impressive results. Models trained with a large amount of unlabeled data and a small number of labelled MRI scans have demonstrated medical diagnosis accuracy comparable to fully supervised systems for tasks such as tumor detection [61]. Even with little labelled data, natural language processing techniques like MixText have produced better text classification results [62]. The combination of labelled and unlabeled satellite imagery has improved the accuracy of land-cover classification, which has also benefited remote sensing applications [63]. There are still issues associated with semi supervised. A few are class imbalance within the labelled subset and confirmation bias brought by inaccurate pseudo-labels [54]. To solve these problems and improve the dependability of semi-supervised models, recent developments have been made, such as the incorporation of active learning strategies and consistency regularization techniques like FixMatch.

2.3.1.4Reinforcement Learning

This method of machine learning is not supervised in that it does not require labeling of input-output pairs; the quality of learned models can even improve with propagation strength. In reinforcement-learning (RL), an agent learns by interacting with its environment and receiving feedback in the form of rewards or punishments. Over time, the agent can learn the optimal decision making strategies, known as policies by performing this process. Contrast this with unsupervised learning which infers patterns in data from a fixed dataset, and reinforcement learning which is constantly evolving since the agent must always react and learn from its actions. This is why RL is ideal for setting where the agent must react to real-time data and feedback from the environment, like autonomous driving, robotics, and gaming (AlphaGo for example). RL is more advanced and purpose-oriented than unsupervised learning as it specifically tries to optimize long-term combined rewards. This is illustrated in Figure 2.2

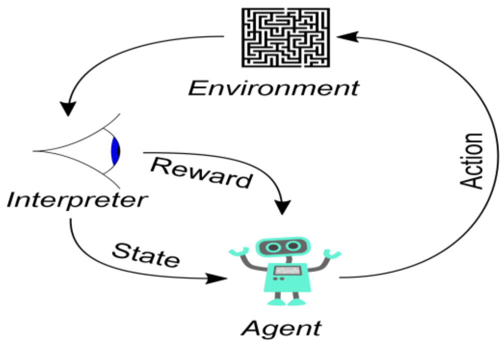


Figure 2.2. Reinforcement Learning

Instead of looking for hidden structures in a set of data, reinforcement learning aims to maximize the most yielding reward signals. This is different from supervised machine learning. Reinforcement learning is the third category of the machine learning paradigm, according to some authors. In addition to supervised and unsupervised learning approaches, reinforcement was taken into consideration as a learning style in their published book on reinforcement learning.

A computational process for comprehending goal-directed learning and decision-making is called reinforcement learning. An agent establishes a direct relationship with the environment through reinforcement learning. The state, agent, action, and signal reward are all defined. RL is a system that makes decisions based on the most lucrative outcomes in its surroundings and extrapolates from experience. Research in this crucial area of artificial intelligence has grown over time. Its wide interdisciplinary applicability in engineering, neuroscience, psychology, and other scientific domains may be the cause of this. A branch of artificial intelligence and machine learning, reinforcement learning has broad uses in optimisation, statistics, and other fields.

2.3.2 Ethical and Practical Challenges of Machine Learning

ML-based crime prediction models present serious ethical issues in spite of their benefits. Because models trained on historically skewed policing data may disproportionately target marginalized communities, algorithmic bias is still a serious problem. According to recent studies conducted by Emily and their team in 2023, over policing in minority neighborhoods has been strengthened by predictive policing tools in the United States [64]. Furthermore, the research by Rudin in [65] has previously justified how model accountability is complicated by the opaque nature of many machine learning models, which makes it challenging for law enforcement to defend predicted results.

Study conducted by Bauer in [66], supports fairness-aware machine learning, which integrates bias mitigation strategies and encourages algorithmic transparency, as a solution. While machine learning has effective crime prediction tools, its utilization guided by ethics, openness and ongoing review. Machine learning (ML) can be a tool for successful and fair law enforcement provided that it has access to diverse training data, that it is interpretable, and that fairness considerations are included. Our research prioritize on the development of transparent AI models.

2.4 Overview of Computer Vision

Computer vision is an area of artificial intelligence (AI) that helps machines interpret and understand the visual world, which is comprised of images and videos. It applies mathematical algorithms to reconstruct the three-dimensional (3-D) geometry, texture, and motion of objects in the scene, rendering computer systems capable of processing, rendering, and interpreting 3D models of objects embedded in space. Szeliski, in their research in [67], ascertain that this aspect of AI is complex because it is full of inverse problems. It tends to deduce hidden features from partial cross-sections of the visual world. This problem is challenging because visual data is typically noisy, ambiguous, and partial, which makes accurate discovery of object configurations hard.

Computer vision usually adopts the interplay of physics-based methods, probability models, machine learning and so on to solve problems. Forward models take into account the principles of radiometry, optics physics and sensor models and are used to simulate how light interacts with objects, as well as how images are created [68]. These models aims to take physical properties of the scene, such as shapes, textures, lighting conditions, etc., which are implicitly influenced in input images or videos. Machine

Learning approaches, particularly deep learning, have also transformed computer vision, where the system is trained to learn representations of objects from large datasets and to disambiguate among possible solutions by observing the patterns in the data.

Computer vision aims to 'understand' the world by recovering object properties - shapes, color attributes - from imagery. That forms the perceptual side of AI [7]. Computer vision develops due to the growth of both computational capacity and algorithmic techniques, demonstrating problem-solving capacities ranging from autonomous driving to medical imaging.

2.4.1 Computer Vision Applications in Security and Surveillance

Computer vision is fast becoming an invaluable asset in today's security and surveillance systems, with the ability to automatically and in real time monitor, detect, and interpret visual information.

The objective of computer vision in artificial intelligence (AI) is to provide machines with the ability to understand and interpret visual data captured by cameras placed in the real world. This includes images and videos. It allows computer systems to process, view and interpret 3D models of objects in space using mathematical methods for estimating location, appearance, and motion [67]. It often involves inverse problems where the computer infers unknowns about a scene from incomplete or uncertain explanations of static or dynamic objects that have been recreated in a virtual environment.

Computer vision systems is applicable in various fields. For example it is used to recognize suspects via facial recognition to be able to pick out suspicious behaviors and to analyze video to get evidence for law enforcement and public safety. It can be applied in assembly line quality control, and defect detection in industrial environments. Other intrinsic advantages of applications of computer vision is its potential to handle and understand large-scale data.

According to the research conducted by LeCun, the recent advancements in deep learning, particularly in convolutional neural networks (CNNs), have enabled computer vision systems to achieve remarkable progress on tasks such as object detection and image classification [68]. These capabilities are critical for applications such as autonomous driving, medical diagnosis, and surveillance of criminal activities in which rapid analysis of visual data achieved with minimum error. Thus, in the intersection of machine learning, image processing and pattern recognition, the computer vision provides disruptive potential in multiple sectors. The role of computer vision in image description predominant by extracting, analyzing, and respond to visual data. Computer vision strives to impact the world by inferring object properties, shapes, and color properties from images. Its algorithms have demonstrated great promise in various application fields such as medical imaging and autonomous driving. This literature considers its applications in the area of system security as applicable to our research.

A growing trend in the literature of this domain is the fusion of computer vision modalities such as activity recognition, object detection, and facial recognition, which is key for improving public safety and crime prevention and increasing the efficiency of investigations. Due to these applications, conventional surveillance has developed into a more intelligent system.

2.4.1.1 Facial recognition Application

This is among the most extensively researched and used security-related computer vision applications. It involves using a person's facial features to automatically identify or verify them. Zhao in [69] states that facial recognition systems compare captured images to databases of known people using feature extraction and pattern matching techniques. This computer vision applications systems are used for threat detection and identity verification in border control, airports, and public gatherings. However, ongoing discussions and research into equitable and responsible implementation spurred by concerns about accuracy in different lighting and pose conditions, as well as ethical concerns about privacy and mass surveillance [70].

2.4.1.2 Object detection

Object detection is a common term in machine vision. It is a process that describes how the instance of real time objects found in static or motion images. The object detected often labelled with a class name and put in bounding box with its correspondence confidence value. The object in view is both recognized and localized. These days, we use different object detectors like YOLO, Single Shot Multi-Detector (SDD), Mask R-CNN and so on, for object detection tasks. Some of the commonest challenges faced by machine vision researchers in solving object detection tasks are varying object distance from camera and object in motion. There is Possibility of object shape changing in time. In addition, the Object in motion poses blurriness. This is called Occlusion- state of an object in obscurity. It can be because of its cluttered background and Shallowness. Other advanced applications of machine learning network used for the behavioral classification exercise in this research are Semantic segmentation, Image diagnosing, Depth and motion estimation, deep sorting and Instant segmentation

2.4.1.2.1 Semantic Segmentation

In computer vision, Semantic segmentation has an important task that involves the classifications of each image pixel into a predefined category which enables a detailed understanding of scene. It is applicable and widely used in autonomous driving, satellite image analysis and in medical imaging. In the research conducted by Long in [71] and Raijpurka in [72], Fully Convolutional Networks (FCNs) and U-Net architectures techniques improves segmentation accuracy significantly by leveraging hierarchical features extracted from convolutional layers

2.4.1.2.2 Image Diagnosing

Image diagnosing refers to the use of image analysis, specifically in the medical field, where deep learning models are used to identify conditions like diabetic retinopathy, pneumonia, and tumors from medical scans. For example, in detecting pneumonia, CheXNet, a deep convolutional neural network trained on chest X-rays, has shown performance on with radiologists [73].

2.4.1.2.3 Depth and motion estimation

Depth and motion estimation is an important technique performed in object detection. It is applicable in augmented reality, robotic navigation, and 3D reconstruction. While motion estimation (optical flow) monitors pixel displacements over time to comprehend object and camera movement, depth estimation seeks to predict the distance of objects from a single or stereo image. In order to improve scalability, recent methods use self-supervised learning to estimate motion and depth without labelled data [74].

2.4.1.2.4 Instant Segmentation

Instance segmentation distinguishes individual object instances within a class instead of just labelling pixels, combining object detection and semantic segmentation. This field has advanced t and often used by neural networks models like Mask R-CNN, which produce object masks for every instance that is detected, enabling accurate classification and localization [75]. Together, these methods aid in the creation of intelligent systems for robotics, autonomous vehicles, healthcare, and surveillance.

With this computer vision techniques, real-time object detection is now feasible even in crowded and complex scenes [76]. Rapid object detection frameworks were first presented by Viola in [77]. The methods have later evolved into deep learning methods which significantly improve both the speed and the performance of the detection system. These technologies are deployed to monitor potential threats and anomalies in locations ranging from government buildings to retail centers to and transportation.

2.4.1.3 Activity recognition with Computer Vision

Analyzing human movement patterns to find patterns that might point to suspicious or criminal activity is part of surveillance application of computer vision. For instance, using datasets and models that assist in training systems to distinguish between normal and abnormal behavior. The Researchers in [78] classified human actions into simple gestures and complex interactions. Activity recognition is particularly useful for identifying theft, fighting, loitering, and unauthorized entry in secured areas

Deep neural models such as RNNs, LSTMs,YOLO improves the temporal modelling of these activities making the systems able to predict events before they further deteriorate. The development of intelligent predictive systems that include activity-based monitoring, object-directed analysis, facial recognition and more are all related to the integration of these technologies through converged surveillance solutions. They enable faster responses, reduce human monitoring fatigue and generate real-time alerts. However, scholars have also warned of the risk of bias and over-generalization, particularly in non-homogenous populations, by which misclassification can trigger false arrests and over-policed neighborhoods [79]. Such system need a robust algorithmic fairness and transparency to gain public trusts. The instruments also need additional human observations to overwrite prediction when error occur in operations. Therefore, due to advances in facial recognition, object detection, and activity recognition, the application field of computer vision in security and surveillance has been broadened.

To maintain credibility and trust in their use, these technologies may be adopted while taking ethical concerns, data security, and public accountability into the design consideration. This research considers the gaps in knowledge and account for them at planning and design. It aims to develop robust, fair, unbiased, transparent and unprejudiced predictive instrument, the proposed instrument planned to be proactive and preventive, detecting weapons in pre-crime scenario before casualties happen in real- life

2.5 Computer Vision Algorithms and Frameworks for Crime Predictions

Recent research explores the integration of machine learning and computer vision for crime prediction. For example, Wang in [80] conceptualized an AI-based crime forecasting system combining neural networks, heuristic engines, and computer vision techniques. Though largely theoretical, the study emphasized the importance of fusing multiple features, including spatial-temporal data and human behavior analytics, to predict criminal activities. Another research conducted by Yan in 2024 [81] , applied computer vision to public health with a system designed for multi-human fall detection using YOLO combined with temporal classification.

The method showed success using RGB images alone, improving accessibility compared to systems requiring RGB-D or sensor data. Kalman filtering was utilized to track each subject over time, enhancing visibility in crowded scenes. Further, Kulbacki in their study in [82], presented a human motion analysis framework using computer vision techniques, focusing on body part segmentation, joint localization, and action recognition in both 2D and 3D video streams. Such methods hold potential for recognizing abnormal behaviour indicative of criminal activity, though the study lacks specificity for detecting crime-related gestures or concealed objects.

Findings from the research of Kounaldi in [83], explores the integration of computer vision into crime forecasting, representing a change in thinking in policing and public safety. Traditional methods relying on statistical crime reports and historical data now augmented by AI-driven visual analytics. Modern systems leverage real-time surveillance footage to assess behavioral and environmental cues predictive of criminal activity. For instance, the research by Bappee in [84] demonstrated how recurrent vandalism patterns or unauthorized intrusions can be identified through spatiotemporal feature extraction. This enables law enforcement to deploy preventive measures.

Several research on intelligent surveillance systems that can identify and stop criminal activity have been developed with computer vision in response to the rising incidence of violent crimes in both public and private settings [51]. These computer vision crime detective instruments use machine learning, deep learning, and image processing concepts to automatically identify guns, knives, and other dangerous objects in real-time video feeds [85]. According to Alpaydin, 2020 in [86], human monitoring is a major component of traditional security systems, but it can be limited in scope and prone to errors. On the other hand, automated weapon detection with computer vision provides more accurate analysis of vast amounts of video data, faster reaction times, and all-round watchfulness. The ability of deep learning models to identify weapons from CCTV footage has been shown in a number of studies [68]. For example, Convolutional Neural Networks (CNNs) have demonstrated exceptional ability to extract spatial features from video frames in order to differentiate between various objects [87].

In another research conducted by Deqi in [88], a YOLOv3-based model was trained to identify handguns in school settings. The test scenarios showed an accuracy of 90%. This accuracy is acceptable but higher precision is possible as error loss needed to be narrowed because of the sensitivity of the application in security. In a similar research conducted in 2023 by Goudah in [89], a modified ResNet-50 architecture was used to detect knives in tube stations that were visible and hidden. These empirical results demonstrate how computer vision may serve as an early warning system in areas where crime is a problem [90]. However, these works constraints to detection of only one class of weapon, which could have been expansive to other common weapons. To make such devices robust, there may be need to include ‘persons’ to the dataset and train the model to recognize this class and classify perpetrators based on illegal harms possessions. This is the gap that this research fills in the field of computer vision.

Furthermore, in the use of computer vision in crime detections, research demonstrates the efficiencies of other real-time object detection frameworks like SSD (Single Shot Multibox Detector), higher YOLO families (YOLOv4, YOLOv5), and Faster R-CNN. This higher precision frameworks have emerged as key instruments in weapon detection. For example, these higher YOLO series is popular for trading the speed and accuracy with cut-off streaming capacity as needed in surveillance systems. The fast and precise counting and detection of the number of people in specific areas are essential for crowd management, security inspection and also public safety on certain events. In dynamic environments, traditional methods frequently fail to produce precise and timely results. To fill this void, Suguna in [91], investigated how to integrate YOLOv5 with alarm and motion tracking systems, revealing a remarkable 85% decrease in false positives during public security trials. Their work investigates the application of YOLOv5 for people counting and detection in crowded scenes. This state-of-the art object detection framework is famous for its efficiency, accuracy, and real-time application performance, and can substantially transform how people were observed at a given location, these tools are essential for law enforcement response because they can track suspects across frames in addition to detecting weapons. This mAP is high but needed to significantly improve for the instruments to be recommended for public use. In addition, the research needed to be expanded to pre-empt crime before the weapons used to perpetrate criminal acts.

Therefore, for a more accurate precision, there are obstacles to overcome before weapon detection systems implemented in practical environments [14]. The high degree of variation in lighting, occlusion, and camera angles is one significant problem that impacts detection models' accuracy. Furthermore, identifying partially hidden weapons or distinguishing between real and unreal weapons continue to be major challenges in the existing literature [92]. Additionally, the authors in [93] and [79] emphasized in their papers that the training datasets frequently lack diversity in terms of object types and backgrounds, which results in incomplete or biassed model performance

According to Zubboff in his published work on social theory in 2023, automated weapon detection systems' ethical and legal ramifications need considered in addition to their technical aspects [94]. Concerns regarding data security, privacy, and abuse is continuous with using surveillance technologies [95]. For instance, misuse of weapon detection software by unauthorized individuals or states can lead to unjust discrimination or unwarranted surveillance. Promoting transparency, accountability and the mitigation of bias in these systems, Wengi in their paper published in 2025 called for regulatory frameworks that ensure the responsible use of AI in public security [96].

Therefore, we could say that, when all these shortfalls accounted during system planning and developments, computer vision-based weapon detection is a promising area in crime prevention and public safety as supported in the paper published by Brundage and their team in [97]. The capacity to identify and address criminal threats enhanced by combining real-time surveillance systems with deep learning [98]. This is the improvement that this research aims to advance relative to the existing works in crime prediction. Researchers and legislators must work together to address the difficulties and moral issues that come with implementing the technology as it develops further [99].

We align with Tegmark in their paper on “Being human in the age of artificial intelligence”. The researcher revealed that as AI and computing power continue to advance, we anticipate that future weapon detection systems will become more precise, flexible, and socially acceptable, further solidifying their position as crucial instruments in contemporary law enforcement [100].

Further to these studies, in recent years, the increasing availability of surveillance footage, computing power, and deep learning advancements have made the integration of computer vision (CV) techniques with crime prediction a prominent area of study. Numerous studies have shown how well CV-based models work to predict or stop criminal activity, especially in the areas of object detection, facial recognition, and behavior analysis. For example, in their groundbreaking study "Real-world Anomaly Detection in Surveillance Videos," Sultani, Chen, and Shah (2018) concentrated on identifying anomalous activity in continuous video feeds. Their method made use of a deep learning model built on a deep multiple instances ranking framework, which was created especially for spotting anomalous activity in security footage. Notably, a sizable dataset comprising both commonplace scenes and unusual events like fights, thefts, and accidents was used to train the system. They were able to spot questionable trends using this approach without the need for in-depth frame-level annotations.

With an average frame-level AUC of 75.4% on the UCF-Crime dataset, the study's results were impressive. Compared to traditional models that employed handcrafted features, this performance was noticeably better. Their approach was particularly suited for dynamic and changing environments because it placed a strong emphasis on temporal localization or determining the exact moment of criminal acts through weakly supervised learning. The system's low need for human annotation stood out among its advantages, providing a workable answer for real-world uses like public transportation hubs, shopping malls, and urban surveillance. The study did, however, acknowledge certain limitations. For instance, a person falling could just be an accident rather than a suspicious act; not all anomalies in video data necessarily point to criminal intent. Furthermore, the poor quality of the video dataset and model training design had a direct impact on the model generalization.

According to Redmon in [85], they presented a research conducted with YOLOv3. An Incremental Improvement, a noteworthy advancement in real-time object detection that has been widely applied to surveillance tasks like detecting loitering people or identifying weapons, despite not being specifically created for crime detection. Their approach used a fully convolutional neural network with residual blocks and up sampling layers, which improved the network's speed and accuracy in identifying objects in changing environments.

YOLOv3 achieved an 80.2% recall and a mean Average Precision of 57.9% on COCO dataset. Its performance was good with multiple object detection and at 45 frames per second on GPU, and it balanced detection quality with the speed, which demonstrated its abilities in the real-time detection. Thus, YOLOv3 is suitable for applications such as automated monitoring systems and public security/surveillance systems, for which it is necessary to perform real-time accurate detection. However, YOLOv3 also has its drawbacks. It tends to struggle with small or occluded objects, which is especially challenging in occluded or cluttered scenes. Furthermore, YOLOv3 requires additional modules or behavior analysis (BA) adaptations to be applicable in criminal surveillance as it has not been designed for the detection of abnormal activities.

Nonetheless, it is a useful tool for public CCTV systems due to its resilience to changes in scale and illumination as well as its quick detection speed, particularly when it comes to tasks like spotting weapons or illegal intrusions. Due to this shortfall, this research has adopted the use of YOLOv4 methods, which can be an improvement to the findings of Redmon in the detection task.

By examining spatiotemporal motion templates without complete video decoding, Kang in [101] presented a reconstruction-free method for action inference in compressed surveillance footage. With a 78.2% F1-score on the PETS2007 dataset and over 90% accuracy for high-contrast abnormal actions like sprinting, their approach showed good performance in detecting suspicious activities like running or trespassing. The main advantage of the system is that it operates directly on the motion vectors in compressed video streams and is thus suitable for low complexity surveillance system. However, the sensitivity to camera-angles, backgrounds, and background noise, and its lack of ability to cover subtle and complex behaviors will limit its application to more complex surveillance situations.

In their work "Multi-Level Recurrent Residual Networks for Crime Forecasting," by Li *et al* in [102] developed a hybrid system that leverages the computer vision for object and person detection as well as RNNs for understanding spatiotemporal crime patterns. Through a thorough examination of crime logs and surveillance footage, the model was able to predict probable future crimes by integrating these data sources. With a reported recall of 85.1%, precision of 82.6%, and F1-score of 83.8%, the method showed good performance. This system's strength is its ability to combine historical crime trends with visual data to provide a thorough prediction framework that is effective at predicting reoccurring patterns of criminal activity. However, the precision could be optimised through advanced hyper parameter tuning for higher accuracy. This research adopts advanced optimisation techniques to ensure high true positives (correct detections of instances) and low false positives (incorrect detections).

Section Summary

From anomaly detection to real-time object recognition and behavioral forecasting, the application of computer vision to crime prediction has shown promise in a number of fields. Compared to conventional techniques, these systems offer greater automation, quicker reaction times, and wider coverage. But model generalizability, ethical use, and diversity of datasets also are often requirements for effectiveness. To ensure fair and responsible usage in law enforcement, more emphasis placed on explainability, bias reduction, and data privacy as technology develops in this work.

2.6 Real-life Applications of Computer Vision Models

This section takes a look at literatures and real life scenario where the computer vision model has been used to identify weapons and crime scenes linked to the violent crime in real time situation.

In the research by Sultani, the University of Central Florida's UCF-Crime Dataset Project significantly advances crime recognition in video analysis with over 1,900 real-world CCTV clips annotated for 13 criminal activities, such as fighting, robbery, and vandalism. This dataset has been used by researchers to train deep neural networks with weakly supervised learning, which enables anomaly detection without requiring annotations at the frame level. Studies using Multiple Instance Learning (MIL) frameworks and 3D convolutional networks have produced encouraging frame-level anomaly detection

AUCs of up to 75.4% [103], indicating the dataset's great potential to improve real-world surveillance applications. Its real-world footage contributes to generalizability and model relevance, but it also has limitations due to limited camera angles and scene diversity, which can degrade its performance when there are new scenes. In addition, since the dataset contains only anomalous patterns, other behaviours that are not criminal might be confused with crime, such as dancing. This further indicates the need for models to be able to have a better knowledge of context (the ability to differentiate between when someone is committing crime and when someone is just behaving normally).

To conclude this section, using Computer Vision Model, there is the possibility to pre-empt crimes and threats proactively. This could accelerate preparedness responses, and train predictive algorithms to explore the potential of strong machine learning based CCTV video analytics models. Each of the existing research, however, highlights important drawbacks like algorithmic bias, data privacy, and infrastructure requirements problems that was resolved for the implementation of this research.

2.7 Advancements in the Convolutional Neural Network Framework

Convolutional Neural Networks (CNNs) are a subset of deep learning models created especially for processing and analyzing visual data, including pictures and videos. In order to closely resemble the human visual system, their architecture uses layered operations that are excellent at capturing spatial hierarchies in data [104]. CNNs are now the basis for contemporary computer vision applications, such as object detection, facial recognition, and activity recognition in security systems as suggested by Lin in [105]. Convolutional, pooling, and fully connected layers make up a typical CNN architecture, and each one contributes differently to learning. Based on the visual input, these layers conduct classification or regression tasks, sequentially extract features, and reduce the dimensionality of the data.

The convolutional layer is the main part of a CNN. It applies learnable filters, also called kernels, to the input image or feature map to find local patterns such as edges, textures, or shapes. Each filter performs an element-wise multiplication and a summation on the input data to produce a feature map. This operation enables the network to detect spatial hierarchies and maintain the spatial relationship between pixels, making it ideal for image analysis.

The pooling layer pools the features maps to small size, which can reduce computation complexity and avoid overfitting in CNN. Popular pooling methods are the max pooling, taking the maximum activation in a region, and the average pooling, averaging the values in pools, respectively [106]. For instance, a 2×2 max pooling operation with stride of 2 effectively cuts the input feature map's width and height in half, preserving the most noticeable features while eliminating the less significant CNNs usually contain one or more fully connected layers, also known as dense layers, following a convolutional and pooling layer [107].

This section highlights how CNNs have evolved significantly over time, with deeper and more effective architectures emerging that have enhanced object detection and image classification capabilities.

2.7.1 AlexNet (2012)

Developed by [10], AlexNet demonstrated the efficacy of deep CNNs by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It brought GPU-based training, dropout for regularization, and ReLU activation. It is used for image classification with notable improvements over earlier models. However, the framework overfits without dropout. Also, the computational cost is high.

2.7.2 VGGNet (2014)

According to Simonya in [108] , the framework was created by Visual Geometry Group (VGG), this architecture used tiny 3x3 convolutional filters to increase depth (up to 19 layers). It is suitable in transfer learning, feature extraction, and classification. The feature extraction for VGGNet is simple and efficient. However, the detector has low inference time when used to train model.

2.7.3 ResNet (2015)

Introduced by He *et al* in [109] ResNet employed residual learning to overcome vanishing gradients, enabling the training of very deep networks (up to 152 layers). It is used for Image classification, object detection (e.g., in Faster R-CNN). The detector is scalable and excellent in generalization. But it is computationally expensive.

2.7.4 The Single Shot Multibox Detector (SSD) Architecture

Unlike previous methods and like R-CNN, which required multiple processing stages for detection, SSD (Single Shot MultiBox Detector) is a well-known object detection framework that predicts object classes and their corresponding bounding boxes in a single pass through the network, enabling significantly faster performance . SSD's efficiency and real-time performance have made it a preferred choice in applications that require accurate and fast object detection, especially in edge computing scenarios [110].

A base convolutional neural network (like VGG16) combined with additional convolutional layers intended for object localization in the SSD architecture. The network can produce feature maps at various scales. This makes it possible to detect objects of variable sizes [61]. SSD is especially well-suited for real-time applications since it uses multi-scale feature maps to strike the ideal balance between speed and accuracy [111]. However, recent studies show that SSD's built-in feature extraction may not be able to manage highly occluded objects, which is why attention mechanisms have been added to enhance detection performance [112]. Zhang in [113] reveals that despite SDD benefits, it has some disadvantages. For small objects, its detection accuracy is lower than that of region proposal techniques such as Faster R-CNN.

Furthermore, compared to more recent object detectors like YOLOv4, SSD's mean average precision (mAP) is typically lower [50]. Further research into more sophisticated architectures that preserve real-time performance while enhancing detection accuracy. This is the case especially for small and obscured objects. Because of this shortcomings, the detector draws attention to the trade-off between speed and precision that [61]. Recent study in 2024 by Liu in [112], reveal improvements of SSD such as SSD-Edge for optimized edge deployment and multi-scale context aggregation [61]. This demonstrate ongoing efforts to enhance SSD’s capabilities.

The author in [12] emphasizes that SSD and YOLOv4 leverage convolutional neural network architecture, an advancement in computer vision for image classification tasks.

In single-shot detection, the CNN base network truncated before the classification layers, enabling the extraction of crucial features from input layers.

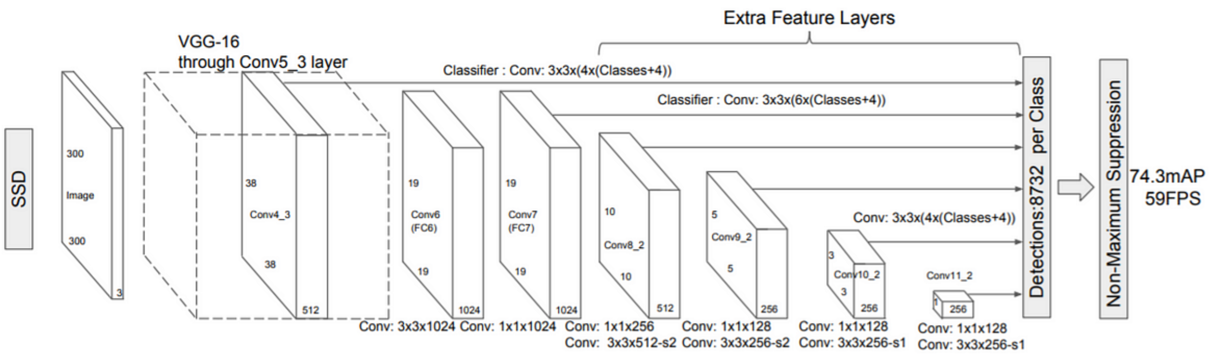


Figure 2.3. SSD Network Architecture [113]

Components of SDD Neural Network

(a) Multiscale Feature Maps

SSD uses a base convolutional neural network (CNN), often pre-trained on a large dataset for image classification (like VGG, ResNet, or MobileNet), there are however additional CNN layers added to reduce spatial dimensions whilst increasing the number of channels. This base network modified to extract feature maps at different scales. These feature maps capture information at various levels of abstraction.

(b) Default Boxes (Anchor Boxes)

Instead of using a fixed set of anchor boxes for the entire image, SSD utilizes default boxes that are associated with specific feature map locations. Each default box has a set of predicted offsets for refining its position and dimensions, making the detector more flexible in handling objects of varied sizes and aspect ratios [111].

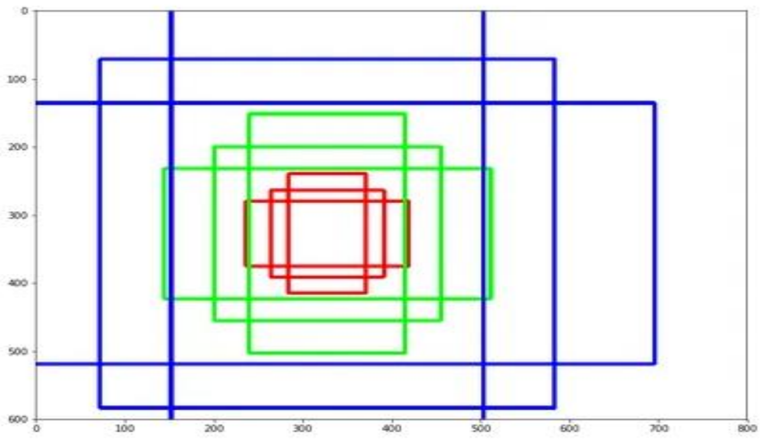


Figure 2.4. Anchor Boxes [111]

(c) Aspect Ratios and Multiple Feature Maps

SSD utilizes default boxes with varying aspect ratios, employing them across multiple feature maps at different resolutions. This strategy effectively captures a diverse range of potential object shapes and sizes. In contrast to alternative models, SSD forgoes the reliance on an intermediate fully connected layer for predictions and instead utilizes convolutional filters directly. To improve the representation of smaller objects, SSD combines feature maps from multiple scales. The algorithm fuses information from different layers to ensure that the detector can effectively capture objects of diverse sizes

2.7.5 The YOLO Algorithms

YOLO is a popularly used computer vision algorithm for object detection. The most well-known series of YOLO are YOLOV2, YOLOV3, YOLOV4, YOLOV5, YOLOV6, YOLOV7, YOLOv8, YOLOv9 and YOLOV10, which were released in late 2024.

YOLO is well known in deep learning for its molecular sizes; fast speed computations; incredible main average precisions (MAP) and efficient object detections. Originally, it contained 24 convolutional layers and two fully connected. YOLO processes images fast because it only puts frames into the network to get its output. YOLO is well-grounded in real-time video object detection and classification. The model identifies classes with bounding boxes around the object region.

2.7.5.1 The YOLOv4 Architecture

Study by Bochovskiy in [114] reveals how the introduction of YOLOv4 marks a major advancement in the YOLO family of real-time object detection algorithms. Through integrating further advanced features in its previous architecture, this version was designed to maintain real-time performance with high accuracy. It was designed to provide better detection accuracy and speed while also making tradeoffs to reduce the deficiencies from the previous YOLO models.

From the paper published by Redmon 2016. We can say that YOLO, in a single glance, takes the entire image and predict the bounding box coordinates and class probabilities [76]. YOLO's most significant advantage is its fast pace in detection. it is speedy, and it can manage an average of forty-five (45) frames per second. Among the earliest versions of YOLO, version 4 is one of the fastest and most accurate in detecting objects. But with optimization and hardware acceleration mechanisms, higher rate possible, the proposed algorithm as shown in fig 2.5 consists of fifty-three (53) convolution layers.

The architecture comprises of three different layer forms. Firstly, the residual layer formed when the activation easily forwarded to an inner layer neural network. In a residual setup, the result of layer one summed to the output of layer two. The second is the detection layer which performs detection at three different scales or stages. The size of the grids increased for detection. The third is the up-sampling layer which increases the spatial resolution. Here the image up sampled before scaled. Also, the concatenation operation used to concatenate the outputs of the previous layer to the presentation layer. The addition operation used to add previous layers.

In the YOLOv4 object detection module used in this research, the YOLO takes input frames first, and these frames divided into grids, say 3 x 3, and on every grid, image classification and localization applied.

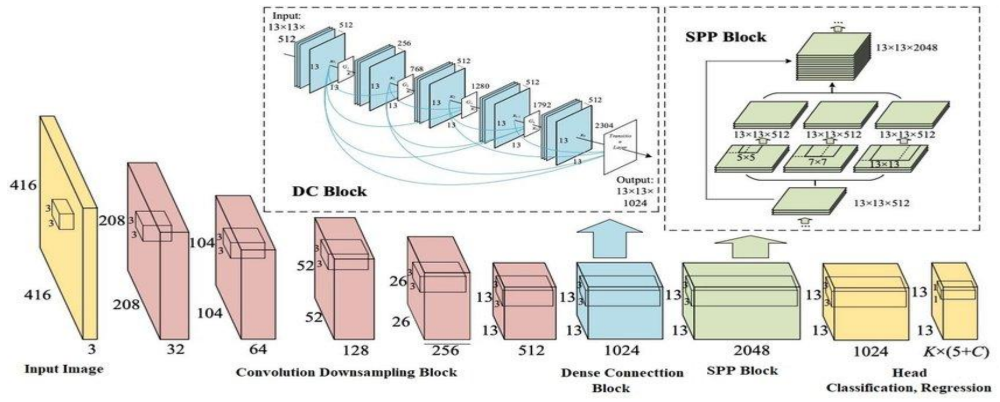


Figure 2.5. YOLOv4 computer vision architecture [76]

The bounding boxes and their equivalent class probabilities for objects are predicted. We then filter out the specific classes as required human, handgun, riffle, short gun, and knife instances in the input dataset. Meanwhile, there are more than 80 classes of the objects present in the COCO dataset used for the research hence, the need for class filtering.

To understand the YOLO algorithm, it is crucial to determine what we currently expect. It varies from the majority of the neural network models because it uses a single convolutional network that predicts bounding boxes and the resulting class probabilities. The bounding boxes weighted by the probabilities, and the model makes their detection dependent on the final weights. Thus, the end-to-end output of the model maximized, and, as a result, images can be produced and processed at a rapid pace as demonstrated by the author in [76].

The main benefits of YOLOv4 are its well-balanced performance, which offers high accuracy and real-time speed, making it applicable to a variety of real-world applications. It also makes use of several advanced training enhancements that improves learning outcomes, such as the activation function Freebies (like Drop Block regularizations and Mosaic augmentation). YOLOv4 demands high GPUs to be trained and if not further modified, its complexity is not friendly for running on edge devices with limited computing resources. This transition is clearly visible in the development from AlexNet to YOLOv4, moving from an accuracy-oriented classification towards real-time object detection with large trade-off between speed and complexity. Deep networks such as ResNet are still necessary in cases with high requirements on accuracy rather than speed.

The YOLO family of real-time object detection algorithms aims to simplify prediction tasks [76]. YOLO splits an image into a grid and predicts bounding boxes and class probabilities straight from the entire image, in contrast to conventional methods that need steps to identify objects [114]. Research conducted by Wang in [115] shows that by taking a comprehensive approach, the network can simultaneously analyze the entire image context, leading to significantly faster detection speeds while preserving competitive accuracy.

A key advantage of YOLO is its extremely fast inference speed, which makes it ideal for real-time applications like video surveillance and driverless cars [116]. Because YOLO learns to comprehend the global structure of images, it is also resilient to changes in object appearance and positioning, which contributes to its remarkable ability to generalize to new domains [117]. These characteristics makes it widely used in dynamic contexts, such as augmented reality and robotics [118].

Despite its benefits, YOLO has some disadvantages, particularly in its early versions. When detecting small or overlapping objects, the single-stage prediction approach may miss subtle details [119]. Furthermore, because the early YOLO models trade off accuracy for speed, the precision was lower than that of more advanced detectors like Faster R-CNN. These restrictions have been addressed by recent developments like YOLOv10. By optimising different YOLO components from the standpoints of accuracy and efficiency, YOLOv10 presents a comprehensive efficiency-accuracy driven model design approach. This method improves the model's capability and drastically lowers computational overhead, which improves the detection of small and overlapping objects [119]. Effective model scaling for edge device deployment [120]. According to recent research, YOLO continues to dominate real-time detection. It benchmarks to demonstrate better speed-accuracy trade-offs than SSD and RetinaNet [121]. Ongoing research into hybrid architectures is necessary because handling extreme occlusions and extremely cluttered scenes continues to present difficulties [118].

2.8 Challenges and Limitations of CNNs

In this section, we have discussed a few particular aspects which may involve technical difficulties for training CNNs:

2.8.1 Technical Challenges

While Convolutional Neural Networks (CNNs) performs well in various computer vision tasks, a number of technical challenges should be sought in order to adapt them in realistic crime prediction and surveillance.

One of their main limitations is the high computational power required for training and deploying of deep CNN models. CNNs are computationally expensive, and the computation is typically offloaded onto specialized hardware such as a high-performing GPU or TPU (Tensor Processing Unit). This is more particularly the case on modern architectures like YOLOv4, VGG, ResNet. For instance, YOLOv4 runs inference in real-time but consumes a lot of GPU while training [114]. This presents a major problem for low resource settings such as government departments in developing countries that have insufficient infrastructure to run sophisticated AI models.

Another major issue it faces is that CNNs require large, good quality, labelled datasets to be effectively trained. Most CNNs are trained by supervised learning which is costly to obtain a large amount of labeled data for generalization performance. In crime prediction scenarios, it is hard and even ethically challenging to collect labeled videos or images of crimes surveillance. Most of the existing datasets suffer from a lack of diversity in crimes, environments, and cultural diversity, or are generated artificially, or are limited. As a consequence, some of these models work well in an artificial environment as they were trained in, but worse in a more random and unpredictable natural environment.

The problem of data imbalance makes training models even more challenging. Video footage shows that the criminal events, especially violent crime or bizarre events, only occasionally take place as compared to daily occurrences. If CNNs tend to overfit to common non-criminal activities, such an imbalance causes less sensitivity to criminal activities detections. Strategies, such as cost-sensitive learning, oversampling, and teaming are often employed to address this. However, these introduce further complications, and may not capture realistic variations.

Lastly, explainability is still a technical challenge. It is difficult to interpret and justify CNN predictions since CNN predictions are black-box models. In sensitive domains like fraud prediction, where these decisions might have legal consequences the inability of explaining in human language a prediction is a huge drawback. Investigation into interpretable CNNs and explainable AI (XAI) is still under way.

Finally, despite achieving state-of-the-art performance in visual recognition tasks, CNNs are limited for crime prediction due to data scarcity, robustness challenges, explainability drawbacks, and high computational complexity. Addressing these technical problems are necessary to build scalable, ethical, and effective AI based crime detection systems.

2.8.2 Ethical and Social Implications

Privacy, bias and algorithmic accountability are among the ethical and societal concerns of CNN-based models when applied to computer vision and crime prediction. Given the impact upon civil rights and public trust in law enforcement systems, the convergence of these problems is gaining increasing prominence in academic and policy literature. Invasion of privacy is one of the main issues. Real-time surveillance analytics, CCTV monitoring, and facial recognition systems are frequently used extensively without the public's knowledge or consent. Because American law enforcement agencies have used facial recognition technology with little regulatory oversight, there is a higher likelihood of mass surveillance, highlighted by Garvie in [70]. Secondly, algorithmic bias is a big ethical issue, as it is generated from inequalities present on the training data. Systemic bias could be reinforced by CNNs trained on biased datasets, resulting to prejudice.

For example, findings from the research conducted by Buolamwini in [79] showed that commercial facial recognition systems identified Black and female faces with much higher error rates than white male faces. Because the distributions of crimes in training datasets may not conform to the real world, such a system may also disproportionately single out specific areas or groups of people in predictive policing, resulting in excessive police presence and social bias. Another matter of ethical concern is that of transparency and accountability in algorithmic decision-making. Researchers have also highlighted the way pervasive surveillance can discourage freedom of speech and expression. As reported by Wang in [115], people's behavior can be modified due to a perception of constantly being watched.

According to studies, algorithmic policing systems frequently suggest heightened surveillance in areas where social and economic disadvantages are already present. Jie in [116] noted that rather than reflecting objective measures of crime, predictive policing tools trained on arrest data typically reflect the racial and socioeconomic biases of the criminal justice system.

2.9 Identifying Research Gaps from the Literature

According to the reviewed literature, computer vision especially when driven by Convolutional Neural Networks (CNNs) has significantly improved surveillance and crime prediction. Important developments like YOLOv4, SSD, and ResNet have enabled real-time object detection and anomaly recognition. These technologies have been incorporated into security frameworks in a variety of urban environments. Machine learning has the potential to enhance law enforcement operations, as evidenced by studies that have demonstrated moderate to high levels of accuracy (with mAP > 57%).

However, despite these advances, several *gaps and limitations* persist in the synthesized literature:

I. **Understudied Areas:** To predict crimes before they happen, only a few studies from the literature have combined real-time visual data (such as live CCTV) with predictive temporal models. There persists a gap in predictive modeling that connects visual surveillance and crime forecasting because the majority of researchers developed models that mainly concentrate on either detection or post-event classification.

II. **Methodological Flaws**: Model adaptability is limited by a strong reliance on supervised learning. Small or biased datasets cause underfitting or overfitting in many models, particularly those with little variation in lighting, setting, ethnicity, and behavior variability.

III. **Bias and Generalizability**: It can be challenging to extrapolate results to diverse cultural or urban contexts, such as developing nations where crime dynamics and surveillance infrastructure differ greatly, because many datasets are demographically and geographically limited

IV. **Ethical Oversight**: Although privacy and fairness issues are commonly recognized, the majority of empirical research does not include community-based validation of AI tools in law enforcement settings or embedded ethical auditing frameworks.

V. **Model Efficiency**: The mAP(*see table 2.0*) achieved for weapon and person predictions in the crime videos is comparatively low in the existing literature. There is gap in knowledge of model optimization and advanced hyper parameter tuning in the studies reviewed.

VI. **Restricted Training Methodology**: The training processes limited to only one detector algorithm in most of the literature. There is need to train models with different algorithms and build the classifier on the best performing relative to the dataset used.

These voids are the gaps that this research aims to fill in the broader socio economic context.

2.10 Justification for the Research

This research seeks to fill the identified gap in knowledge as enumerated in section 2.9 above. This will be achieved by developing an integrated computer vision model that trains real-time visual surveillance crime video from trusted sources using computer vision and machine learning-based prediction techniques. The pre-crime predictive AI model has the potential to recognize criminal activities in video, detect weapon, anomalies, and classify perpetrators as high risk in real time. This is an advancement to the existing models that is mostly passive and concentrate on either detection or historical crime analysis.

Additionally, the study uses a localized, diverse dataset produced from public CCTV systems and manually annotated video clips set in an evolving urban setting. This large training criminal video dataset enhances model generalizability and tackles the problem of dataset inadequacies in model training.

To evaluate possible bias and promote transparency in our model, it incorporates ethical auditing practices, such as training procedures that consider fairness, transparency, accountability and community feedback systems. The study proposes a scalable, context-sensitive, and ethical framework for improving security through AI-driven crime prediction by filling in the methodological and ethical gaps in the existing literature. The expected results encourages safer, more equitable urban areas. It has the potential to lessen the need for reactive policing, and assist law enforcement in making informed decisions.

Table 2. Summary of Literature Review

[116]	Token gradient alignment in transformer-based object detection	Efficient visual detection in real-time applications	High detection accuracy and computational efficiency	Requires extensive computational resources	ODS F-measure = 82.4% OIS F-measure = 84.2%
[117]	End-to-end real-time object detection with YOLOv10 architecture	Real-time object detection	Highly optimized for real-time applications	Benchmark-specific performance, lacks broad generalization	mAP 89%
[118]	Trainable bag-of-freebies architecture for real-time detection	Real-time object detection	Real-time performance with state-of-the-art accuracy	May overfit on certain datasets	56.8%
[119]	Compressed domain action detection in surveillance video	Action detection in compressed video data	Efficient processing of compressed data	Limited to surveillance video contexts	78.2% On PETS2007 dataset. Other Dataset - =90%
[120]	CP-CNN architecture with core-periphery principle	Core-periphery CNN applications	Improves CNN feature extraction	Limited to specific core-periphery datasets	79.8% to 80.5%)
[121]	OVR-XGBoost: One Vs Rest OVO-XGBoost: One Vs One. The primary difference is based on how the dataset is organized for classification.	XGBoost based algorithms	The study successfully addresses class imbalance in theft case prediction by introducing improved XGBoost-based models and utilising SMOTENN.	Only theft cases from a single city were included in the analysis, which might limit how broadly the results can be applied.	85%

[122]	Applying particle swarm optimization-based classifier and a rules engine to classify crime reports.	Multiple classification algorithms			79%
[123]	17 spatiotemporal variables were used to train and test the XGBoost algorithm, and then SHAP is used to explain the model predictions.	XGboost model			89%
[124]	Prediction of crime in neighborhoods of New York city using spatial data analysis.	XGBOOST, RF and SVM			52%
[125]	EADT approach is used for Interpretable and Accurate Crime Prediction.	Decision Tree (DT)			Aggregate Accurracy =77.6%

2.11 Conclusion of the Literature Review

From early statistical methods such as regression analysis and hot spot mapping to the adoption of machine learning, and in more recent times, computer vision technologies, these review findings illustrate how crime prediction evolves with the integration of CNNs and computer vision technologies.

Deep learning architectures (AlexNet, VGG, ResNet, SSD and YOLO) have demonstrated promising results in automating the detection and classification of visual crime scenes. They are largely fast to detect objects from video.

However there are technical and ethical issues identified from the research reviewed that must be addressed in future studies. Critical issues such as scarcity of real time crime dataset, high computational cost and sensitivity to variation in camera views and lighting conditions are addressed in our research.

Accordingly, our research takes steps to narrow some of the identified technical and ethical concerns. We ensured that the models trained on large criminal dataset from trusted sources, with data privacy law in place through the experimentation. This aids system fairness, transparency, accountability and also generalizability of the instrument on unseen data.

CHAPTER THREE

Research methodology- data collection

3.0 Introduction

This section outlines the data collection and pre-processing techniques employed to optimize crime predictive s system developed in this research. The datasets for the work contain a large database of real-time videos of criminals. The systematic nature of this approach improves the generalizability of the AI models. In addition, the data acquisition and data training processes with personal data handled in line with the ethical practices of the UK General Data Protection Regulations (UK GDPR) Policies.

3.1 The Data Sources

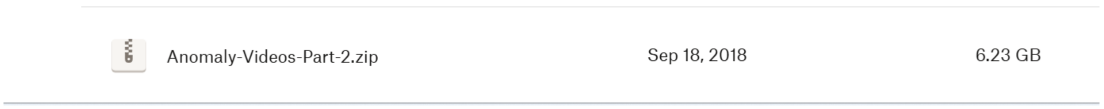
For our research, the dataset used for its implementations carefully selected from trusted, royalty-free open sources, ranging from reliable institutional databases and dataset repository, such as UCF, NIST, Data World, Kaggle and Google Source. The crime video dataset comprises online sources to provide extensive data for the implementation of the research. They are originally mug shots of about 18000 images of person, weapons and other unintended objects like cars, tree, computers and so on. However, through observational method, irrelevant images carefully filtered and deleted to avoid the model rain on noises and classes that will not contribute to detection weight. Consequently, only about 3118 crime images related to the five classes needed in our research advanced to data pre-processing stage. These data are crucial to training, testing and validating the model so that the system can accurately detect and classify criminal activities across various scenarios. The following are the sources of data for the work.

3.1.1 University of Central Florida Criminal Database

To enrich the training dataset, data obtained from the University of Central Florida (UCF). UCF is a widely recognized and royalty-free source of crime-related video content. Based on the report published in 2023 by the National Science Foundation on higher education research and development survey, UCF is committed to high-quality research and data her data security has made it a trusted repository for academic use. The database consistently ranked high in research by Homeland Security, with the National Science Foundation (NSF) ranking it among the top universities for innovation and research excellence [126].

This made UCF forms one of the major sources of our criminal footages datasets utilized to train our models. The UCF Crime Dataset consists of surveillance footage and frames, including incidents where the perpetrators have already found guilty of violent crimes and convicted by the law. This dataset is accessed via Kaggle, which hosts a curated version of the UCF Crime Dataset under a Creative Commons license. The platform provides structured metadata and preview features that facilitate the easier selection and download of relevant files with license agreements.

The Fig 3.0 provides the screenshot of the anomaly video from the UCF data source.



Anomaly-Videos-Part-2.zip	Sep 18, 2018	6.23 GB

Figure 3.0. UCF Anomaly Video Dataset Download

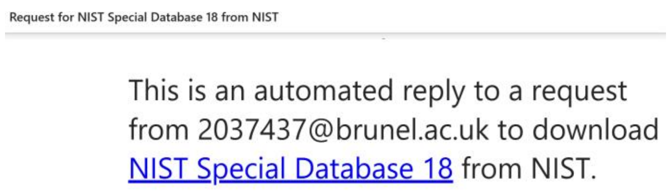
3.1.2 National Institute of Standards and Technology (NIST) Special Database 18 Mug shots

In addition to UCF data, the (NIST) Mugshot Database was included in the dataset to improve the model's discriminative power for criminal identification and classification. This database contains mugshots of weapons, and suspects aiding, in the enhancement of the system to Identify and match subjects engaged in criminal activities [136]. The NIST Special Database 18 is one of the primary resources accessed for research institutions, containing high-quality crime images that are used in computer vision research into facial recognition, surveillance systems and exercising criminal identification.

Dealing with crime data often involve ethical, privacy, and legal issues. To obtain a weapon and criminal image dataset, from NIST we took the following steps:

1. Go to NIST Website: Begin by visiting the NIST official website or the datasets section. NIST Research frequently provides data in a range of formats.
2. Find Files and Folders with Respect to Crime and Police: Find files and directories containing files on crime and police.
3. Check for Data Usage Policies: We make sure to read data usages policy of the data for any restrictions, or permission for usage of the data.
4. Data Licensing: Ticked the box and agreed to their licencing terms. Because we are in a sensitive domain of dealing with sensitive criminal activity, we carefully follow restrictions in licensing and ethical guidelines of the data.

However, we were given a download link of the data in the mail after given access to use the SD18 Data (1.7GB). The consent that kept us bound to the terms and conditions for the use is presented in Figure 3.1


Request for NIST Special Database 18 from NIST

This is an automated reply to a request from 2037437@brunel.ac.uk to download NIST Special Database 18 from NIST.

Figure 3.1 Permission to utilize NST Special Database 18 for Education and Research.

3.1.3 Scraped Internet Data

Data was also retrieved with the process of scraping the internet. The information on the four weapon categories for criminal activity sourced from royalty free websites includes: Shotguns, Rifles, Knives and Handgun. This is a scraped data that was used to augment the training dataset with more context about crime weapons. The model was made aware through various images and descriptions on these weapons, to improve detection and generalization in real crime incidents. Moreover, images of individuals without a history of a crime are available in the training dataset through this sources also. This was done in line with online privacy policies as applicable in the UK data protection policies. Google download all was used and we ensure that the content owns the legal right to the repository. This additional data enables the systems to learn about the features of non-criminals as well. Thus, enabling the AI Model to distinguish between someone who is acting with a criminal intent and someone else who is going about their lawful business. Such actions are important to enhance the robustness of the system and its capability to accurately predict the commissioning of crime activities based on the detection of weapons and suspicious behaviors of the perpetrator. Table 3.1 gives the summary of the key data sources used.

3.1.4 The Real-time Criminal Video

Although the model training for this work exclusively utilized secondary dataset, the system has the potential to evaluate and predict crimes in real time. Feeding the model with real-time videos such as live captures from external webcam, Rasbery Pi or JASON NANO can validate its real-time generalization. This allows for experimentation with real-world data and validating the system's ability to respond dynamically to criminal activities as they unfold in real world scenarios.

Table 3.0. Data Sources Summary

Data Source	**Description**	**Data Size (GB/MB)**
UCF Anomaly videos	**Data Format**: CCTV and Surveillance Crime video footages of Street crimes theft, burglary, arson, fighting, shooting shoplifting, stealing, vandalism, normal video, assaults, abuse, robbery and arrests. **Usage**: The data was converted to .PNG/JPEG file, cleaned and annotated for SDD and YOLO model trainings. It forms about 70% of the data training.	6.20 GB
NIST	Images (.PNG Format). It comprises mug shots of Gun, Knife, Person etc. Downloaded after receiving licence agreement.	1.7 GB
Google "download all"	These are additional PNG/JPEG format files downloaded from google Knife, , Shotgun, Handgun, Riffles, non-Criminal persons	800MB

3.2 Ethical Considerations on Personal Data Collection

Yee in [127] demonstrated how previous works by AI researchers and developers have raised serious public ethical concerns. The papers cited example of how the algorithms used for cropping images in Twitter (now X) favored light-skinned than dark-skinned users. According to the study, the instrument also showed gender biasness towards female. In another research conducted by Birhane in [128], the author demonstrates how COMPAS, an instrument of AI, has sparked a national debate in the United States on racial bias, when proposed for the prediction of Recidivism (RVI). These are cases of ethical concerns about AI data collections and model training that are considered in the data collection and model training in this study. Building an AI system without setting stages for ethical issues is a prejudice. It raises ethical concerns related to data privacy, fairness issues, transparency, and accountability.

This section focuses on how ethical issues related to data acquisition and training process addressed. The research duly puts into consideration personal data of people (names, addresses). We took steps to make certain that the personal data was used fairly, lawfully, and transparently. The personal information used was anonymized or pseudonymized before introduced into the dataset. Moreover, the data acquired not used for any further reasons than stated in the agreement with the usage terms and conditions from the sources. Throughout the system development and documentation, the data shared were accurate and limited to what is necessary in relation to the research purpose. The data remains for as long as required in accordance with the Brunel University of London data retention policy. In addition, the information processed with security procedures in place to guard against unauthorized processing and against accidental loss, damage or destruction of personal data, as prescribed under the UK GDPR to protect both the process and the privacy of the data [129]. This section addresses the following fundamental ethical concerns.

3.2.1 Data Privacy and Consent

Firstly, we ensured the data sources are from trusted and legally permissible repositories. To guarantee sources reliability, we only engaged credible open resource criminal databases such as the UCF Crime Dataset and the National Institute of Standards and Technology (NIST) Mugshot Databases. Also, all necessary consents were agreed with the authorized body, where applicable. We kept to the agreement and ensure any personally identifiable data (PII), or sensitive information included in the datasets is anonymized or aggregated in compliance with privacy laws. Additionally, all sensitive personal data pseudonymized before being integrated into the training datasets. *Data Pseudonymization* is a privacy enhancing methodology for data protection. It is the substitution of personally identifiable information of records in a dataset with fake identifiers, or pseudonyms. This step enables tasks performed over the data without knowing the exact profile of the person owning the data. All data privacy was conducted in accordance with the UK General Data Protection Regulation (GDPR) and other relevant data protection frameworks worldwide.

3.2.2 Bias and Fairness

Bias is a critical concern in the development of machine learning models, especially in sensitive domains such as crime prediction. Mavrogiorgos and their team describe bias as discriminatory behavior of a computer vision when Model trained on erroneous data, promoting prejudice [130]. This could be specification bias, measurement bias, sampling bias, annotation bias and inherited bias. However, the bias that is most relevant to this research is Sampling Bias.

3.2.2.1 Sampling Bias

In the context of this work, Sample bias is when a particular population sample is overrepresented or underrepresented in the dataset. According to the study conducted by Alex in [131] and Alexandra in [132], such cases, known as self-selection bias or population bias, are common in machine learning training. This work took practical steps through observation method to ensure that each instances of the classes are fairly represented without prejudice. For example, Figure 3.2 shows the cross section of the dataset for *a* ***person,*** highlighting gender equality and fairness in the representation of males and female genders in the dataset. This shows fairness in the demographic distribution of the training dataset.

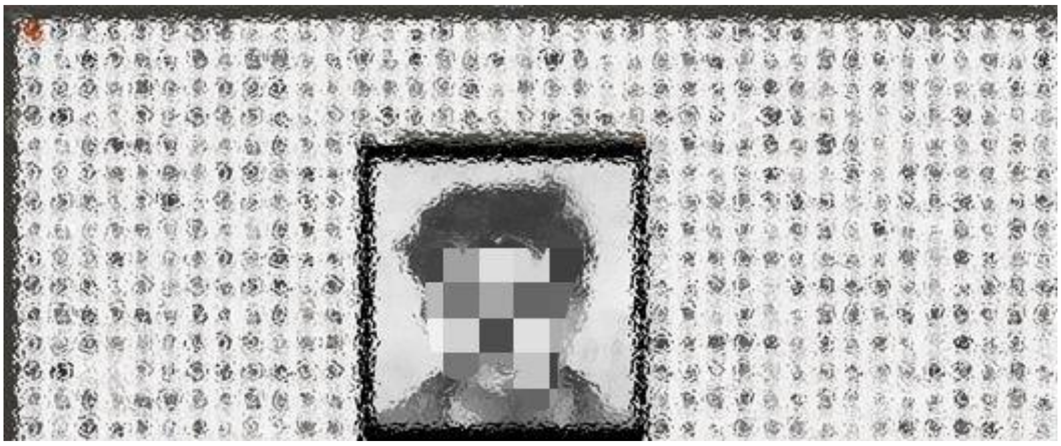


Figure 3.2(a). Male Instances in Dataset

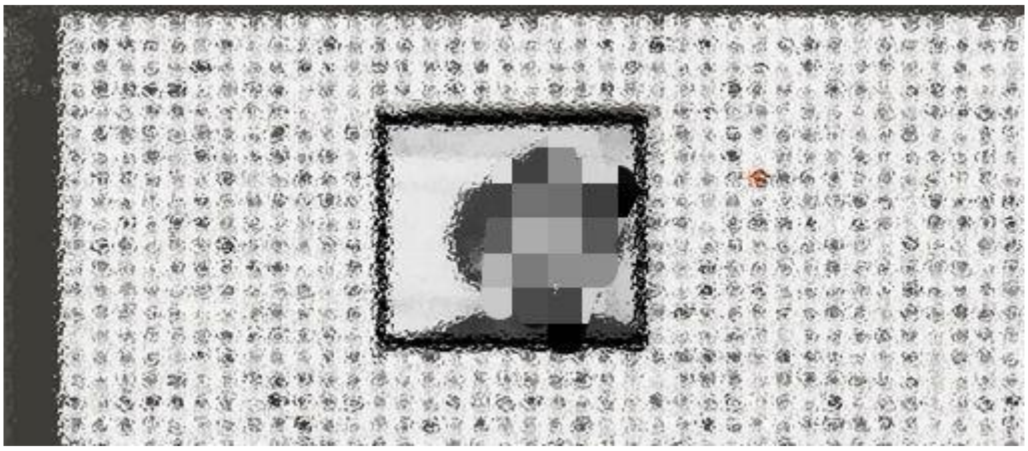


Figure 3.2(b). Female Instances in Dataset Samples

From figure 3.2(a) and 3.2(b), the datasets used for the model training are diverse and representative of various demographic and racial groups as required. For instance, we took steps to ensure that the data is not unproportionally representative of certain ethnicities, genders, socioeconomic status, etc. Additionally, we addressed the danger of tainted predictions with regard to harmful stereotypes by using fairness aware algorithms and performing regular audits of our model's predictions to ensure fairness to all members of the population.

3.2.3 Transparency and Accountability

The use of AI systems in criminal justice requires important levels of transparency and accountability, especially when such systems have the potential to impact individuals' lives. This research prioritizes the transparency of the model's decision-making process by implementing explainability techniques known as Local Interpretable Model-agnostic Explanations (LIME is a reference model-agnostic algorithm that explains the predictions of any machine learning classifier in an interpretable fashion) [133]. The algorithm helps provide insight into how predictions are made. Additionally, we ensured accountability by maintaining logs of all model decisions and actions, which will allow for the auditing of the system's performance and behavior in real-world scenarios.

3.3 Ethical Use of Data Scraping

In the process of scraping internet data related to crime-associated weapons, ethical concerns about the reliability of data and its potential to perpetuate misinformation are addressed. We scrutinized data acquired from all sources, knowing that data from unverified sources may introduce inaccuracies or biases, which can adversely affect the performance of the crime forecasting system. To mitigate these risks, only data from reputable and verified sources included in the training process. Also, appropriate measures taken to validate the accuracy and credibility of the internet-scraped data. The images of the classes extracted from the Google website using *the extension " download all"* feature. Moreover, the online data collection process is performed with consideration for intellectual property rights and does not infringe upon the rights of creators.

3.4 Data Collection Techniques

The quality and relevance of the crime forecasting intelligence system depend on the quality, relevance and the diversity of the data used for training and testing the system models. For this purpose, the data collection was performed in a way that considers observational and automated methods in order to gather a representative set of crime-related scenarios and items. The considered technique are divided into:

3.4.1. Observational Data Collection

This research employs observational study techniques, a data collection method whereby data is gathered from an existing crime database, containing video footage that captures real-life crime scenarios from sources. The observational nature of the data ensures authenticity and relevance to real-world crime detection tasks. The technique helped achieved the categories of dataset shown in figures 3.3(a-g) as follows:

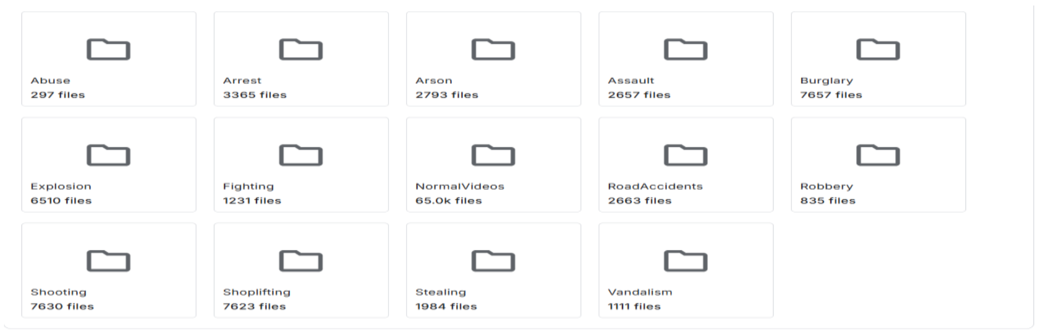

Abuse
297 files
Arrest
3365 files
Arson
2793 files
Assault
2657 files
Burglary
7657 files
Explosion
6510 files
Fighting
1231 files
NormalVideos
65.0k files
RoadAccidents
2663 files
Robbery
835 files
Shooting
7630 files
Shoplifting
7623 files
Stealing
1984 files
Vandalism
1111 files

Figure 3.3. UCF Data Folder Structure

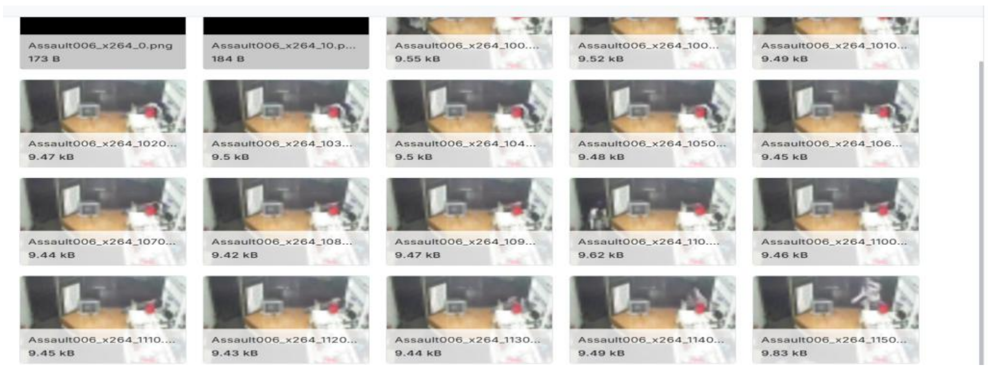

Assault006_x264_0.png
173 B
Assault006_x264_10.p...
184 B
Assault006_x264_100....
9.55 kB
Assault006_x264_100...
9.52 kB
Assault006_x264_1010...
9.49 kB
Assault006_x264_1020...
9.47 kB
Assault006_x264_103...
9.5 kB
Assault006_x264_104...
9.5 kB
Assault006_x264_1050...
9.48 kB
Assault006_x264_106...
9.45 kB
Assault006_x264_1070...
9.44 kB
Assault006_x264_108...
9.42 kB
Assault006_x264_109...
9.47 kB
Assault006_x264_110....
9.62 kB
Assault006_x264_1100...
9.46 kB
Assault006_x264_1110...
9.45 kB
Assault006_x264_1120...
9.43 kB
Assault006_x264_1130...
9.44 kB
Assault006_x264_1140...
9.49 kB
Assault006_x264_1150...
9.83 kB

Figure 3.3(a). Assault

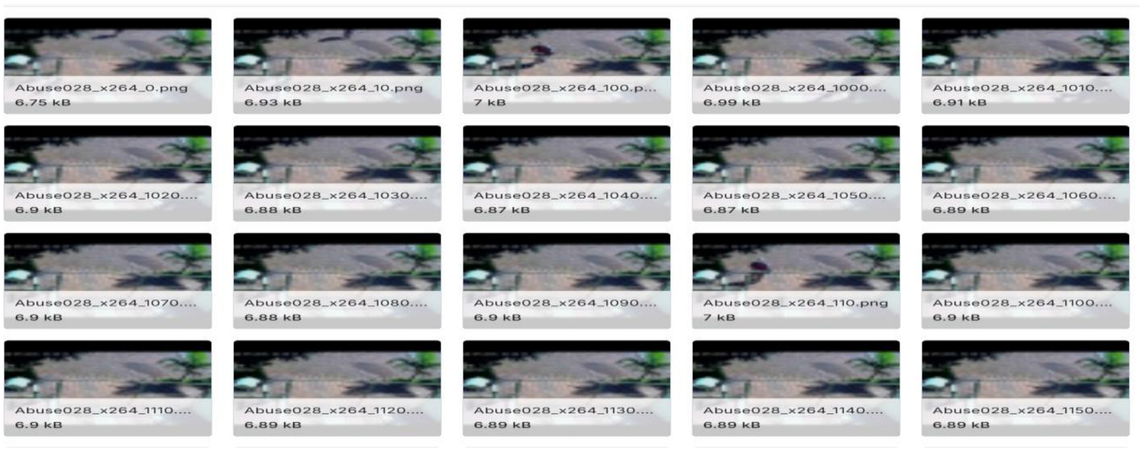

Abuse028_x264_0.png
6.75 kB
Abuse028_x264_10.png
6.93 kB
Abuse028_x264_100.p...
7 kB
Abuse028_x264_1000....
6.99 kB
Abuse028_x264_1010....
6.91 kB
Abuse028_x264_1020....
6.9 kB
Abuse028_x264_1030....
6.88 kB
Abuse028_x264_1040....
6.87 kB
Abuse028_x264_1050....
6.87 kB
Abuse028_x264_1060....
6.89 kB
Abuse028_x264_1070....
6.9 kB
Abuse028_x264_1080....
6.88 kB
Abuse028_x264_1090....
6.9 kB
Abuse028_x264_110.png
7 kB
Abuse028_x264_1100....
6.9 kB
Abuse028_x264_1110....
6.9 kB
Abuse028_x264_1120....
6.89 kB
Abuse028_x264_1130....
6.89 kB
Abuse028_x264_1140....
6.89 kB
Abuse028_x264_1150....
6.89 kB

Figure 3.3(b) Abuse

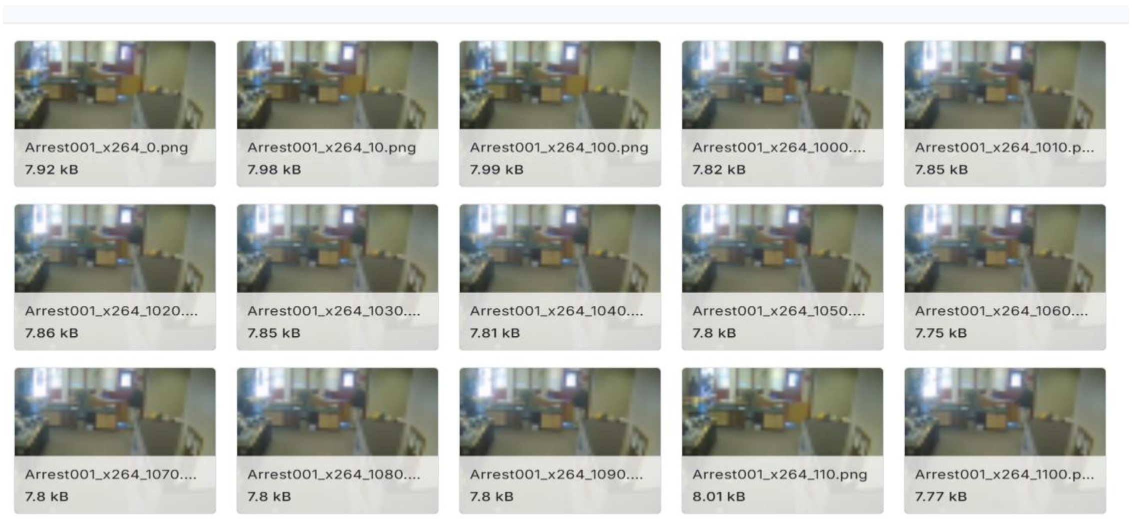

Arrest001_x264_0.png 7.92 kB
Arrest001_x264_10.png 7.98 kB
Arrest001_x264_100.png 7.99 kB
Arrest001_x264_1000.... 7.82 kB
Arrest001_x264_1010.p... 7.85 kB
Arrest001_x264_1020.... 7.86 kB
Arrest001_x264_1030.... 7.85 kB
Arrest001_x264_1040.... 7.81 kB
Arrest001_x264_1050.... 7.8 kB
Arrest001_x264_1060.... 7.75 kB
Arrest001_x264_1070.... 7.8 kB
Arrest001_x264_1080.... 7.8 kB
Arrest001_x264_1090.... 7.8 kB
Arrest001_x264_110.png 8.01 kB
Arrest001_x264_1100.p... 7.77 kB

Figure 3.3(c). Arrest

Burglary005_x264_0.p... 8.39 kB
Burglary005_x264_10.... 8.44 kB
Burglary005_x264_100... 8.47 kB
Burglary005_x264_100... 8.53 kB
Burglary005_x264_101... 8.51 kB
Burglary005_x264_102... 8.52 kB
Burglary005_x264_103... 8.51 kB
Burglary005_x264_104... 8.5 kB
Burglary005_x264_105... 8.45 kB
Burglary005_x264_106... 8.46 kB
Burglary005_x264_107... 8.48 kB
Burglary005_x264_108... 8.48 kB
Burglary005_x264_109... 8.49 kB
Burglary005_x264_110.... 8.48 kB
Burglary005_x264_110... 8.51 kB

Figure 3.3(d) Burglary

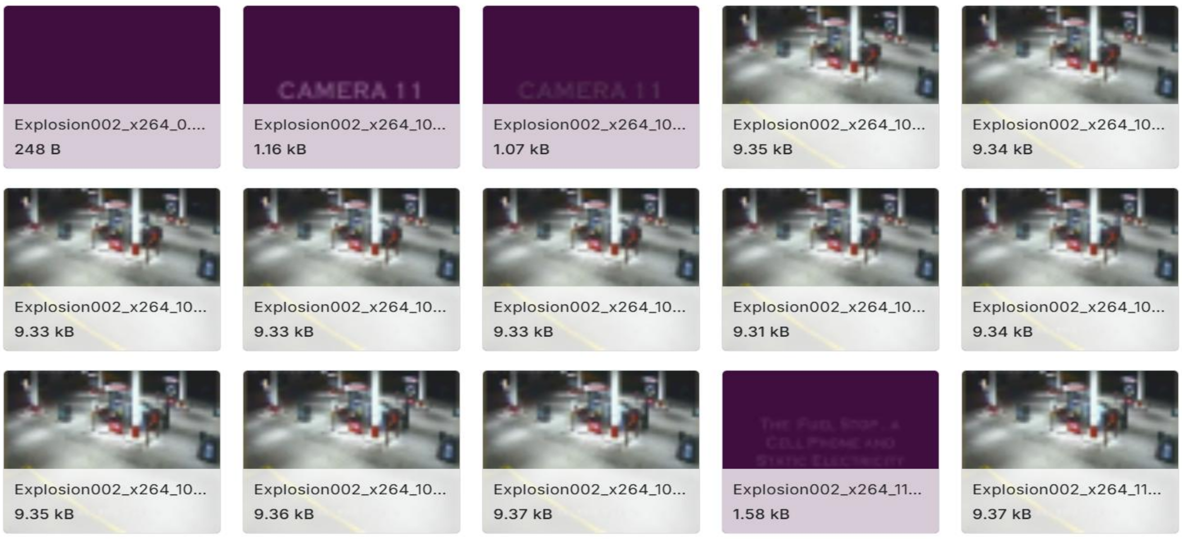

Explosion002_x264_0.... 248 B
Explosion002_x264_10... 1.16 kB
Explosion002_x264_10... 1.07 kB
Explosion002_x264_10... 9.35 kB
Explosion002_x264_10... 9.34 kB
Explosion002_x264_10... 9.33 kB
Explosion002_x264_10... 9.33 kB
Explosion002_x264_10... 9.33 kB
Explosion002_x264_10... 9.31 kB
Explosion002_x264_10... 9.34 kB
Explosion002_x264_10... 9.35 kB
Explosion002_x264_10... 9.36 kB
Explosion002_x264_10... 9.37 kB
Explosion002_x264_11... 1.58 kB
Explosion002_x264_11... 9.37 kB

Figure 3.3(e). Explosion

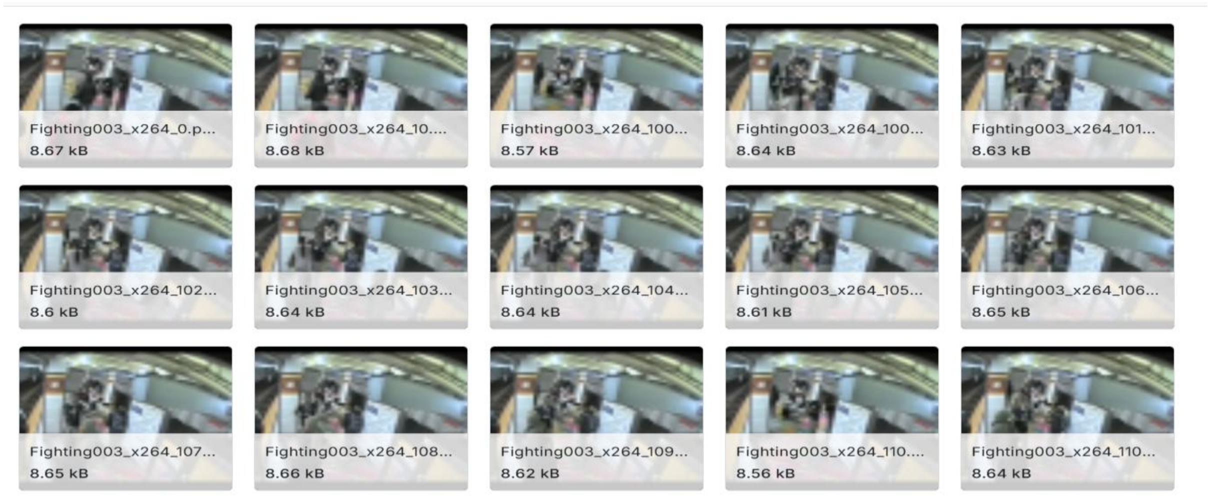


Figure 3.3(f). Street Fighting

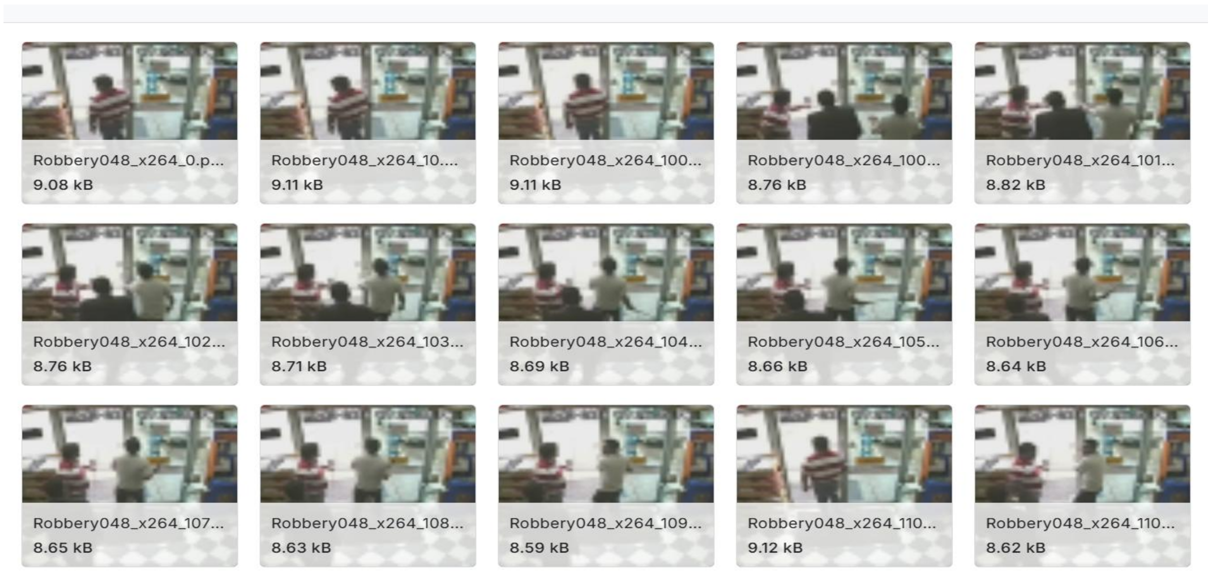


Figure 3.3(g). Robbery

After data acquisition, the combined dataset organized into a designated folder for further pre-processing.

3.5 Data Preparation

The preparation of data for machine learning and computer vision applications in crime prediction is a multi-layered process. It involves data cleaning, transformation, annotation, and ethical governance. This section is divided into three subsections: *data* *cleaning* and *transformation*, *annotation and augmentation*, *and ethical considerations in data handling*. The figure 3.4 gives the data preparation workflow for the system. It includes image standardization, augmentation and annotation processes as well.

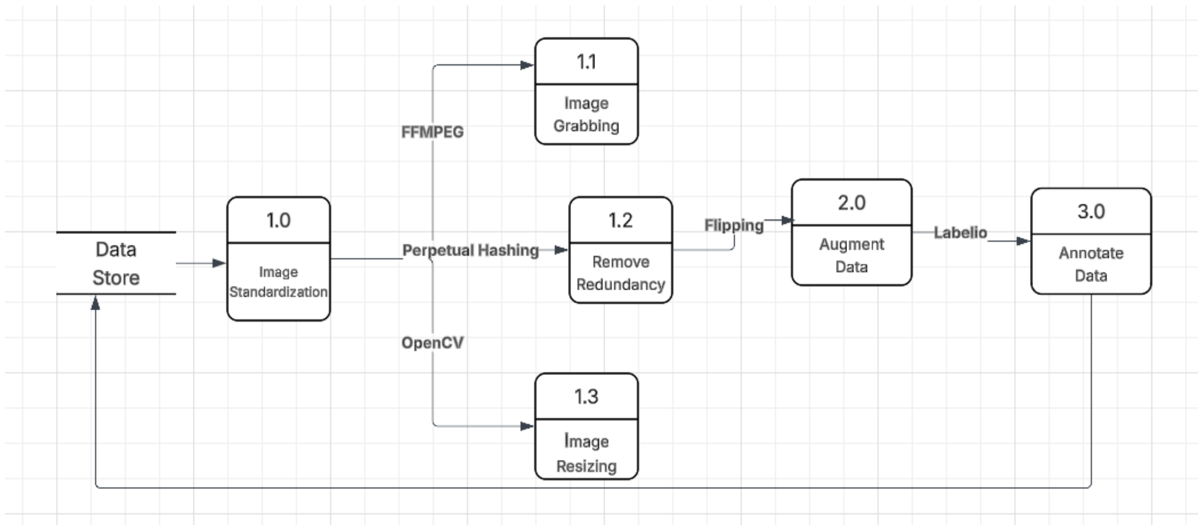


Figure 3.4. Data Preparation Workflow

3.5.1 Data Cleaning and Transformation

As discussed in section 3. 1, the raw data utilized in this research came from diverse sources, including crime surveillance videos, mugshots, and internet scraped weapon imagery. These sources varied significantly in quality, format, and resolution. Consequently, the first step in the data preparation pipeline involved data cleaning and standardization.

3.5.1.1 Data Cleaning and Standardization

Effective data cleaning is essential to prepare visual content for deep learning-based crime prediction. The goal is to eliminate noise, ensure consistency, and optimize the visual clarity of images before training the predictive neural model. The following five-step preprocessing workflow applied to the datasets:

a. Removing Redundant Images: During the frame extraction phase, multiple identical or identical frames were generated due to the nature of static video scenes. These redundant images were identified and removed using perceptual hashing (*perpetual hashing updates the presentation of the data constantly, making easier treatment of dynamic data sets and a self-evolving performance of the model over time)*. Unlike traditional hashing, which changes drastically with minor input variations, perpetual hashing generates a hash based on an image's visual features, such as brightness and structure. In plain language, perpetual hashing works like a visual fingerprint if two images "look" the same to the human eye, they will have similar hashes. This prevents the model from "memorizing" the same scene multiple times, thereby improving generalization and reducing the risk of overfitting.

b. Image Resizing: The default input size for YOLO format is 416 x 416 pixels. But YOLOv4 is flexible and can work with custom input sizes (320, 512 or 608). The size of the collected crime video dataset varies greatly due to different factors such as resolution, frame rate, codec, and duration. The following are some of the popular formats:

- 480p (Standard Definition) - [640x480 Pixels],
- 720p (HD): 1280x720 pixels,
- 1080p (Full HD): 1920x1080 pixels
- 1440p (2K): 2560x1440 pixels
- 2160p (4K): 3840x2160 pixels

In this research, all image frames were resized to 416*416 pixels, the standard input resolution required by the YOLOv4 architecture [114]. This resizing ensured uniformity across datasets and reduces computational load during training. In this work, a Python

OpenCV code snippet used to resize each image. The block of codes is as shown in Figure 3.3

```
import cv2
# Load the image
image = cv2.imread("input.jpg")
# Resize the image to 416x416
resized_image = cv2.resize(image, (416, 416))
# Save or display the resized image
cv2.imwrite("resized.jpg", resized_image)  # Optional: save to file
# cv2.imshow("Resized", resized_image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

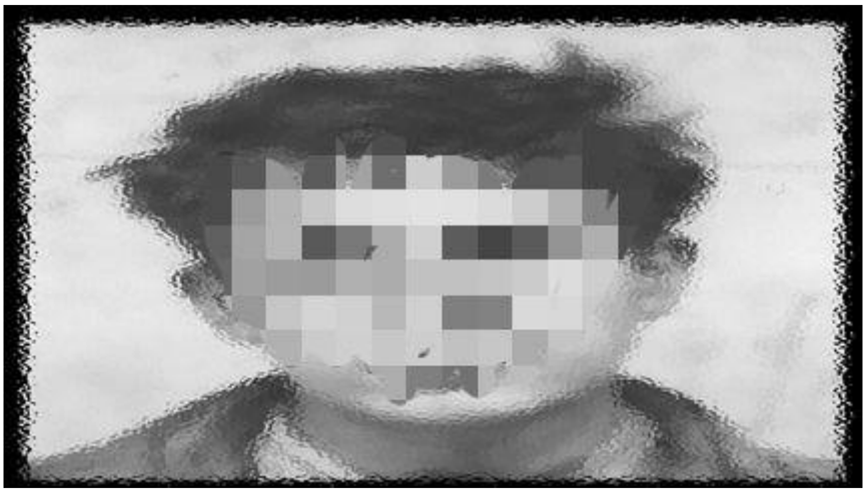


Figure 3.5. Image Resizing with OpenCV, 416*416 Pixel

c. Frame Extraction or Image Grabbing: Video footage was converted into frame sequences using MPEG, with an optimized sampling rate of 1 frame every 2 seconds. This approach captured relevant motion and object interactions while avoiding oversampling.

3.5.2 Data Augmentation

The data augmentation process provides substantial benefits to the AI Model.. The data augmentation process can be interpreted as means to algorithmically inject prior knowledge [134].

Here, we considered data augmentation as replacing the empirical distribution with the algorithmically smoothed distribution as follows:

$$pD\ (x, y|A) = \frac{1}{N}\sum_{n=1}^{N} p\big(x|x_{n,}A\big)\delta(y - y_n) \qquad (3.0)$$

From equation 3.0, we can consider A as the data augmentation algorithm, which produced a video sample x from a training point x_n, such that the label is unchanged. pD is the empirical distribution of the crime video dataset.

3.5.2 Translational Method

To enhance the robustness of the proposed model and reduce overfitting, translational data augmentation specifically applied to the training dataset. This technique involves shifting objects horizontally and/or vertically within the image frame without altering the object class or scale, thereby simulating positional variability commonly encountered in real-world scenarios [55].

Each image in the training dataset was randomly translated within a specified pixel range along the x- and y-axes. The corresponding bounding box annotations were updated to match the new object locations. The translations were carefully constrained to ensure that the shifted bounding boxes remained within image bounds. This augmentation technique does not alter object shape, scale, or class label, and is particularly useful for models like YOLO and SDD, which directly regress bounding box coordinates [114].

Figures 3.5(a), 3.5(b), 3.5(c), 3.5(d) are pictorial representations of the classes in the dataset.

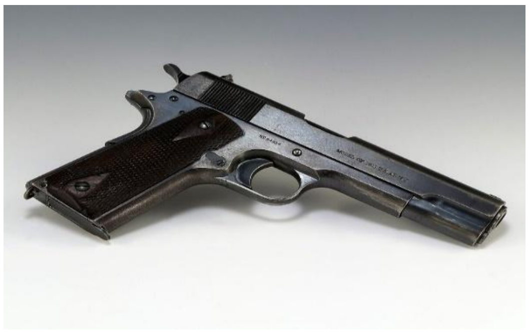


Figure 3.5(a). Handgun

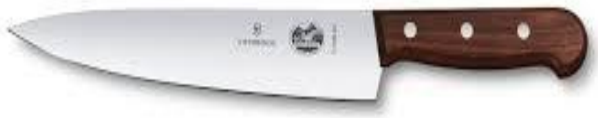


Figure 3.5(b). Knife

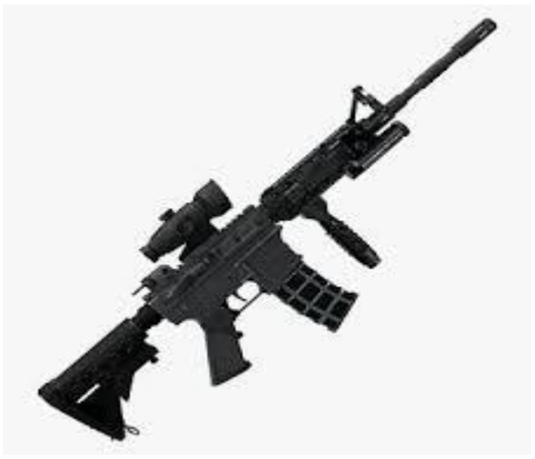


Figure 3.5(c). Rifle

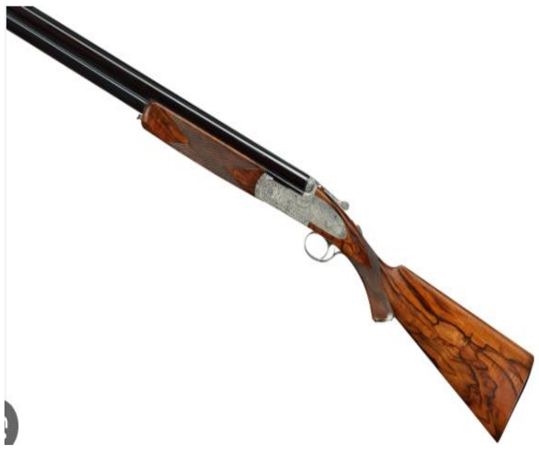


Figure 3.5(d). Shotgun

Translational augmentation was implemented using OpenCV. For each original image, new training samples generated by shifting the image content along the x-axis (horizontal) and y-axis (vertical) by a random number of pixels within a predefined range. Bounding box coordinates adjusted accordingly to preserve the accuracy of object annotations. The augmented images and corresponding YOLO-formatted annotation files were added to the training set. This effectively doubled the size of the dataset and increased the model's exposure to positional variability. Data augmentation was applied before each training epoch, in line with best practices recommended for object detection models [135].

3.6 Data Annotation

After the data was cleaned and standardized, and a translational method applied, the data was annotated using manual labeling tools such as *LabelImg*. At this stage, the five key object classes tagged: ***person***, ***knife***, ***handgun***, ***shotgun***, and ***rifle***. Annotations are performed in a YOLO-compatible format.

Each bounding box is mapped to its corresponding class label. This ensured that object localization (identifying and locating objects in images) and class identification precisely interpreted by the detection model during both training and inference. Table 3.1 shows the five classes of interest in the dataset with their corresponding class code. Additionally, Figure 3.6 shows the corresponding data in the labelimg annotation tool. For the rest of the data, the name of the class was inferred directly from the folder structure and or what was used to describe each instance of the object as illustrated in Table 3.1.

Table 3.1. The Fine Annotated Classes and Codes

001	Person
002	Handgun
003	Riffle
004	Knife
005	Shotgun

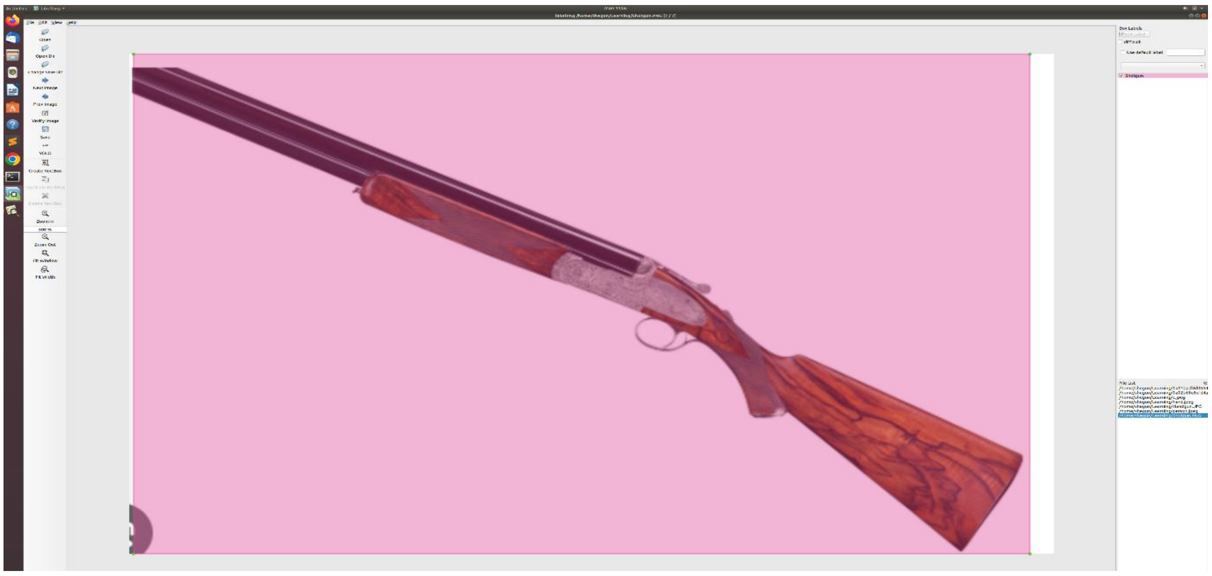


Figure 3.6(a). Shotgun Annotated Data

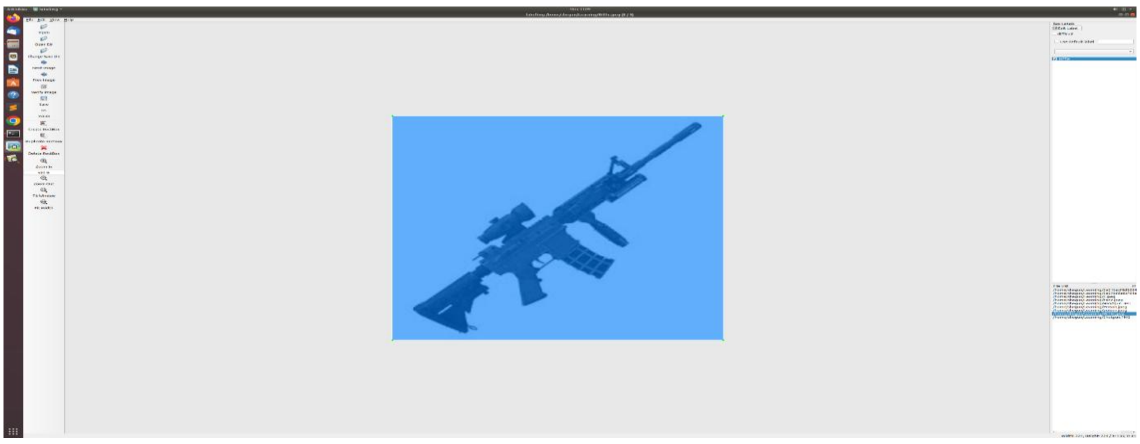


Figure 3.6(b). Riffle Annotated Data

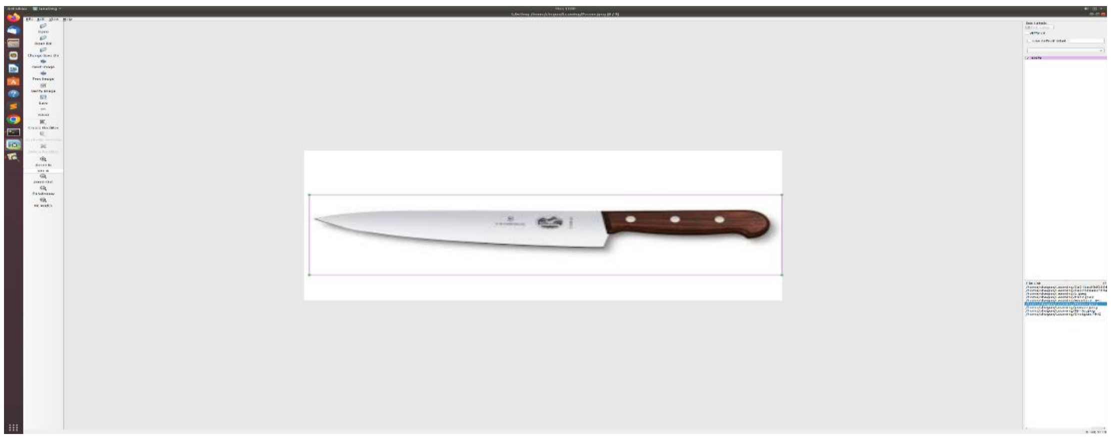


Figure 3.6(c). Knife Annotated Data

Yolo, just like other architecture or detection methods, used a specific annotation format; for this research, the annotation format is defined in figure 3.6 below.

- Class Number
- Object center coordinates in x = ${}^{Center\ X}/_{ImageWidth}$
- Object center coordinates in y = ${}^{Center\ y}/_{ImageHeight}$
- Object width = ${}^{ObjectWidth}/_{ImageWidth}$

Object Height = ${}^{ObjectHeight}/_{ImageHeight}$

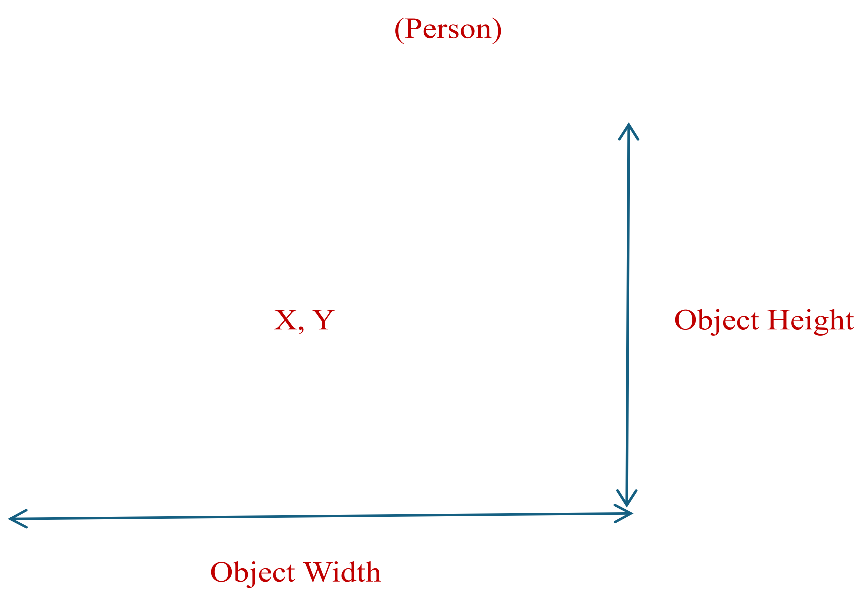


Figure 3.7: YOLO Annotation Format

During data annotation with labelimg, the object center ordinates in x, (Xmin), object center coordinates in y, (Ymin), object width (Xmax), object height (Ymax) automatically derived when an object is annotated and subsequently saved in a text file (.txt). Each data annotated has its corresponding .txt file saved with the data name and kept in the same directory or stored differently. In this design, both the data and its annotation file were saved in the same directory. The representations is detailed in Figure 3.8;

Figure 3.8(a). Image File

Figure 3.8(b). Annotation File

3.7 Ethical Considerations Summary for Data Handling

Consistent with the best practices for responsibility in AI research the data was handled under strict ethical principles. These includes its use of mugshots and video surveillance footage, both of which contained visual data of people. The data sets anonymized or pseudonymized to reduce the risk of infringement of privacy before we used them.

This was consistent with UCF and NIST PII privacy practices [136] and [137]. Data storage was protected through encrypted local drives and organizational cloud services, which are compliant with the research ethics policy at Brunel University London and the Data Protection Act 2018.

In addition, a balanced collection of training data is used in order to minimize the algorithmic bias. Images depicting a range of skin tones, genders, and backgrounds were also incorporated to mitigate demographic bias in the testing data, an effect that has been widely described in facial identification software [79].

This ethical steps aim to help build model that is fair and inclusive across different populations, and also generalize well on unseen data. Plans for future deployment also included audit mechanisms to detect and correct model bias in real-world applications.

3.8 Limitations of the Data Collection

Challenges were encountered during the data collection phase of this research, which may affect the interpretation and generalization of the findings. A major limitation was the restricted access to authentic, high-resolution crime video data due to legal, ethical, and privacy concerns. Most crime-related footage is not publicly available, leading to a reliance on secondary sources such as the UCF Crime Dataset, web-scraped videos, and open-access repositories. While these datasets were useful, they lacked uniformity in quality, annotation, and contextual detail.

Another challenge was class imbalance. Certain crime-related objects, like handguns and knives, were more frequently available in open datasets compared to less common weapons such as shotguns and rifles. This sample bias can introduce bias in model training and reduce accuracy in underrepresented categories. Despite the use of augmentation techniques to mitigate this issue, natural class diversity remains limited.

CHAPTER FOUR

The single shot multi-box detection methodology

4.0 Introduction

This section presents the design, implementation and the results obtained when SSD was trained as the detector for the predictive model. The training process based on the data collecting strategy introduced in Chapter Three. The data was converted from YOLO annotated format (.txt) to Pascal VOC (XML), the data format recognized by SSD method. Moreover, the section includes optimization process that improves the model performance from 74.7% to 84.19%. This was achieved through advanced hyperparameter tuning technique. The chapter presents the class by class precisions with the analysis and interpretations of the results. Finally, comparison with the findings of the literature was included to evaluate the competitiveness of the model.

4.1 Exploring Single Shot Multi-Box Detection Method (SSD)

Object detection methodologies have seen a significant evolution with the advent of Single Shot Multi-Box Detection (SSD) technology. The task of object detection usually consists of multiple non-linear steps. This includes the generation of bounding boxes, the resampling of feature pixels implementing a high-quality classifier [138]. Although these types of approaches have demonstrated strong accuracy, they have also been limited by their computational expense, and thus their real-time feasibility. SSD development represents a ground-breaking advancement in deep learning-based object detection.

As demonstrated by the author in [139], SSD just like YOLOv4 are detection algorithms that involve convolutional neural network architecture. The SDD detector stands out as a popular algorithm for object detection, aiming to detect objects efficiently in images in a single pass. The core concept of SSD lies in conducting object detection within a single forward pass through the neural network, distinguishing it from traditional two-stage detectors like Faster R-CNN.

By removing the need for pixel or feature resampling during bounding box proposal, SSD preserves high accuracy and also improves detection speed. As noted by Liu and their research team in [139], SSD presents a novel method that simultaneously maximizes accuracy and efficiency by using compact convolutional filters on feature maps to directly predict object categories and bounding box offsets in real-time applications, where speed is crucial, SSD's effectiveness is especially noticeable. In another researcher conducted by Li in [140], the author reported that single-shot detection has speed advantage. By avoiding computationally demanding steps, SSD expedites the detection process and guarantees quick and precise object recognition.

For image classification tasks, Zhang in [141] highlights that SSD and YOLOv4 make use of convolutional neural network architecture, a fundamental component in computer vision. The CNN base network truncated before the classification layers in single-shot detection, making it possible to extract important features from the input layers.

4.2 Justification for using SDD Method for Crime Prediction

The Single Shot Multi-Box Detector (SSD) approach, which makes use of Convolutional Neural Networks (CNNs), is a suitable framework for predicting criminal activity and improving public safety. Table 4.0 offers a thorough explanation for this choice.

Table 4.0. Justification for Using SDD Method

Factor	Justification
Real-Time Detection	Ability to recognize multiple objects of interest from crime video footage and respond instantly to suspicious activities in real time
Accuracy	The SSD algorithm provides high accuracy in detecting various objects, which is crucial for identifying potential criminals or threats. Ability to detect various objects with high accuracy, which is important for finding of suspicious persons and weapon possession or threats in real time.
Multi-Object Detection	It also enables the detection of multiple objects at once humans, and weapons leading to a rounded situational awareness
Speed and Efficiency	SSD is specialized for speed of processing and is ideal in applications needing fast analysis of video streams or images.
Integration with CNNs	SSD is built upon Convolutional Neural Networks (CNNs), thus its ability of extracting features is highly improved to enhance detection capability.
Scalability	Its scalable and adapt easily to different surveillance system
Robustness	SSD is invariant to illumination, occlusion and scale that can be occurred in real-world surveillance applications
Low Latency	The architecture of SSD is capable of low latency performance that is very important for its target applications that consists of missions demanding timely action
Contextual Analysis	Through detecting the scene at which an object is located (e.g., suspicious action, group of people, etc.), SSD can assist in the identification of crime-related events.
Data-Driven Insights	The knowledge obtained from SSD can be a law enforcement tool and lead to community safety enhancing crime prevention work.

By combining accuracy, speed, and efficiency, SSD sets standard for object detection methodologies, promising enhanced performance, and applicability across various domains.

4.3 SSD Dataset Preparation

The Single Shot Detection (SSD) method requires that its dataset be prepared in the Pascal Visual Object Classes (VOC) format. The labelling tools used earlier in this work for the YOLO format data preparation also support this VOC format.

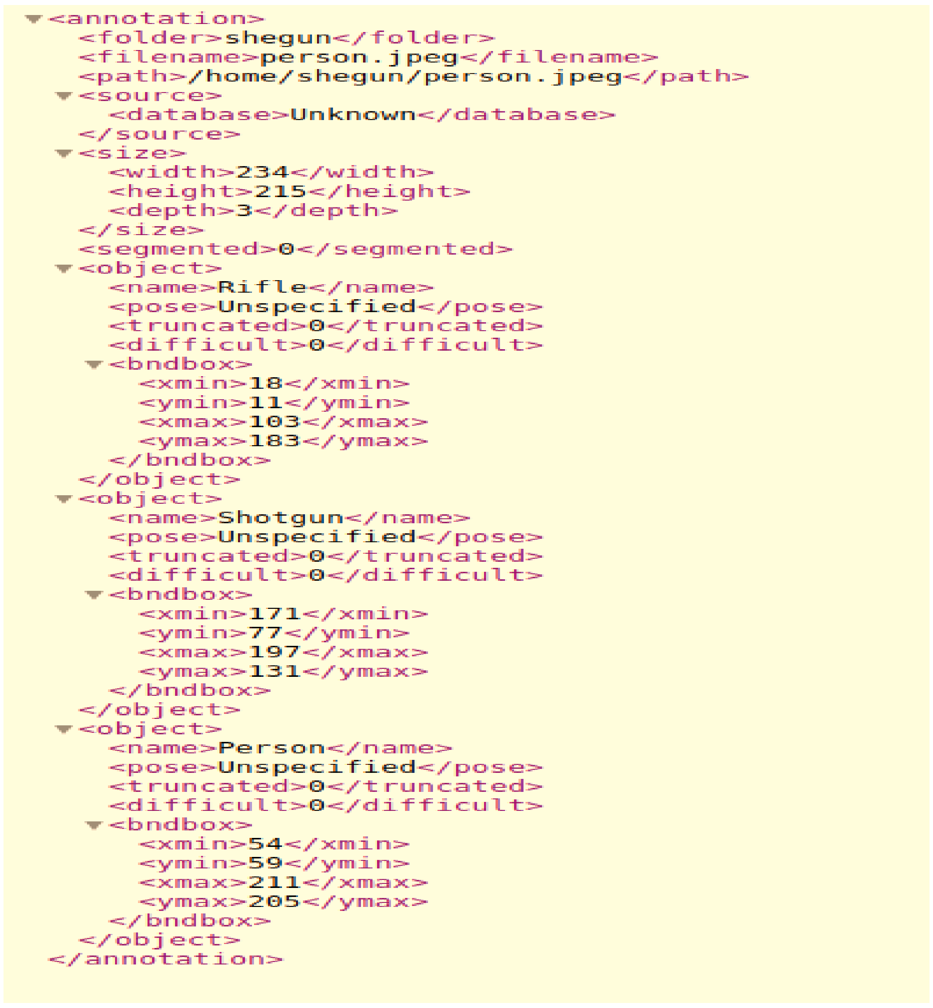
Figure 4.1 Sample Training Dataset for SSD

The annotation is a full representation of the object under consideration and describes:

a. The object metadata (filename)

b. The bounding box coordinates

For SSD to identify the weapons and categorize perpetrator as high risk, precise data labelling is essential for deep learning training, according to Lin [142]. Screenshots of the SDD annotation in.XML format are shown in Figure 4.1b. As shown below, the annotation process entails providing the file path and information, the folder name where images are kept, the file name that represents the image, and the path to the image folder.



```
<annotation>
  <folder>shegun</folder>
  <filename>person.jpeg</filename>
  <path>/home/shegun/person.jpeg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>234</width>
    <height>215</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Rifle</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>18</xmin>
      <ymin>11</ymin>
      <xmax>103</xmax>
      <ymax>183</ymax>
    </bndbox>
  </object>
  <object>
    <name>Shotgun</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>171</xmin>
      <ymin>77</ymin>
      <xmax>197</xmax>
      <ymax>131</ymax>
    </bndbox>
  </object>
  <object>
    <name>Person</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>54</xmin>
      <ymin>59</ymin>
      <xmax>211</xmax>
      <ymax>205</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 4.2 SDD Annotation file (.XML) for Class 001

For readers understanding, the following are the interpretations and values assumed for the annotation technique.

The Image Dimension

a. Width: The image width

b. Height: Image Height

c. Depth: The depth of the image is simply the channel. For a colored image like our dataset, *3* was assumed as it is an (RGB) color image.

The Object metadata

a. Name: The class label

b. Pose: object orientation

c. Truncated: Indicating whether the object is outside the image. The value will be 1 if located within the image region but 0 if located partially outside it.

d. Difficult: 1 if it’s difficult to recognize, and 0 if object is located easily by the system.

The Bounding Box Information

a. Xmin: Distance in pixels from left edge to left side

b. Ymin: Top edge to top side

c. Xmax: left edge to the right side

d. Ymax: top edge to bottom (Distance in pixel

4.4 Training Configuration

To prepare the dataset for training, we created a folder titled "SSD Custom", and within this folder there were three subfolders:

a. Annotations

b. Imageset

c. JPEGImages

The "JPEGImages" folder contained all of the images that had been labelled with the labelImg. An XML-formatted annotation file is included with every image and is stored in the "Annotations" folder. The dataset is contained in three (3).txt files in the "Imageset" folder. A Python script was then used to split the dataset into training and test portions. We used a 70:30 train-test split. (*check Appendix D for the training script and implementation commands*). The training assumed the following parameters:

- Batch Size: To balance memory constraints and model convergence, a small batch size of 32 used.
- Epochs: To monitor model performance and prevent overfitting, the model is calibrated to 500 epochs, with validation carried out at each epoch.
- Save and Restore: To make it easier to recover the optimal model, model saving and restoration are carried out at each snapshot checkpoint.
- Optimisation: To speed up model convergence, the stochastic gradient descent (SGD) technique was applied to the model's implementation.
- Learning Rate: A modest 0.001 learning rate was applied. Despite the longer training time, this aids in model convergence and fine-tuning.
- We set a confidence level of 0.25; in order to identify an object, it must have crossed a 25% detection threshold. For instance, if a scene contains the class "Shotgun," the model must have identified the object's class by 25%. To guarantee a high rate of accurate detection, "low confidence" predictions are filtered out.

4.5 SDD Algorithm Implementation

We used vgg16 as the backbone for the SDD implementation. Its depth, which includes 16 convolutional layers to learn intricate and hierarchical features in our training dataset, is the reason for this [143]. Additionally, vgg16 has maintained high accuracy over image classification and demonstrated impressive performance in a variety of image recognition tasks [144]. We used Multibox loss in this study, which combines regression and classification loss. While classification estimates the model's quality across classes, regression loss assesses the bounding box's quality.

Although there are several frameworks to perform single-shot detection (SSD), the following are the most popular:

- Pytorch
- TensorFlow

This section focusses on using SSD with TensorFlow, which uses Keras as the backend. The open-source neural network library Keras first became famous for its user-friendly modular design. Keras, a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It helps to minimize the amount of code written to build a graph, provided the model with the least amount of performance degradation.

The following programming environments and technologies used in the model development:

Python 3.9

On October 5, 2020, Python 3.9 was released [145]. Most of the deep learning frameworks like Tensorflow, PyTorch, Keras use Python 3.9 and other programming languages. Python is widely used in object detection models such as SSD and YOLO. Additionally, the Python support of the system environment makes it possible to use computer vision libraries like OpenCV, which facilitate image processing, feature extraction, and object detection. During experimentation, Python 3.9 aids the training and implementation of the model.

CUDA 10.1:

Study by Rakhimov in [147] reveals that Nvidia developed the parallel computing platform and application programming interface (API) model known as CUDA (Compute Unified Device Architecture). The GPU used for the project's image computation tasks is accessible through the infrastructure.

CuDNN 7.5:

In another research conducted by Yi in [148], the author reported that Nvidia created the GPU-accelerated deep neural network library known as CuDNN (CUDA Deep Neural Network library). This infrastructure was utilized in this work for implementations of the deep learning system. Using Nvidia GPUs, CuDNN easily integrates with the CUDA platform to speed up deep neural network training and inference.

TensorFlow 1.15.0:

With TensorFlow 1.15.0, models development is simplified. It is an open source machine learning library developed by Google [149]. This versatile tool is fully utilized in numerous domains, including natural language processing, speech recognition, and image processing. TensorFlow 1.15.0, the version used in this study, has multiple other higher-order improvements, bug fixes and new features over earlier iterations. This is the stable release version of TensorFlow that developers and researchers mostly used.

4.5.1 Model Loss functions

Prior to optimising the weight in our neural network, we first define a loss function that minimizes over the training epoch. As the learning weight rises, the model's loss function gradually falls. Since the output is meant to be class probabilities that add up to 1 for optimal accuracy, computing the loss function with cross entropy loss may be a best practice in computer vision. Thus, the multi-class cross-entropy loss for our model can be estimated as:

$$E(W) = \sum_n E_n(w) = -\sum_n \log p_{ntn} \tag{4.1}$$

Where t_n is the integer indicating the correct class, p_nk is the network's current estimate of the probability of class k for sample n, and W is the vector of all weights, biases, and other model parameters.

4.5.2 Optimization Method

In computer vision and machine learning tasks, we may sometimes not achieve the ideal weight for best accuracy after model is trained. Then we resolve to utilize optimisation method to improve performance. More broadly, optimisation in machine learning refers to the deliberate modification of model hyperparameters to determine the initial values of each parameter in order to identify the optimal set of weights for the model.

Hyperparameters tuning automatically modifies the model with updates during this optimisation process, which was carried out through a number of iterations. This iterative process is extremely transformative because it helps develop a model with high prediction accuracy that both performs optimally on the training dataset and also generalize well on unseen datasets.

We now require an algorithm that converts these gradients into weight updates in order to optimise the loss function and create a network that performs well on new datasets. Our SDD uses the Stochastic Gradient Descent (SGD) technique to lower the training loss function, which enhances the model's average precision.

Linearised least squares is the suggested approach for the majority of computer vision algorithms. Using a second-order technique like Gauss-Newton, the optimisation is carried out by evaluating each term in the loss function and then taking an optimally sized downhill step using a direction determined by the gradients and the Hessian of the energy function.

Unfortunately, as demonstrated by parameter counts and our training sample sizes, the challenges in deep learning are becoming more significant. We have used a range of optimisation algorithms based on extensions to Stochastic Gradient Descent (SGD) as highlighted in the study by Padila in [150] and DeFazio in [151]. To solve this issue. The SGD algorithm computes the derivatives of the associated loss E_n(w after evaluating a single training sample *n* in the crime video dataset rather than the complete training set.

Study by author Xie in [153] demonstrates the need for modern optimisers, which uses adaptive learning rates, momentum scheduling 152], and even second-order derivatives of the objective function [154-155] for optimising the feature representations of the approximate second-order information [154]. These developments have been applied and yielded greater stability and improved performance of our SDD training

4.5.3 Hungarian tracking algorithm

In multi-object tracking, the Kuhn-Munkres algorithm, also known as the Hungarian tracking algorithm, is frequently used to solve the assignment problem, which relates to linking the identified objects across temporal video frames. It aims to minimise average cost by cost-optimizing the assignment of detected entity candidates in one frame to their locations in another frame based on some criterion, similar to spatial or visual similarity. The best one-to-one correspondences are ensured by this approach, which works especially well in applications where tracking object identities over time is required, like surveillance and crime detection systems. Recent studies have combined the Hungarian algorithm with motion prediction methods like the Kalman filter [155] and object detection frameworks to improve accuracy and real-time in dynamic environments [156]. The DeepSORT Algorithm used for multiple object tracking in this research builds on Hungarian tracking algorithm.

4.5.4 Deep-SORT Algorithm

Multi-Object Tracking (MOT) methods based on vision analyse image sequences to determine object correspondences among the images [157]. Numerous motion analysis methods have been proposed and widely used in a variety of domains, such as autonomous driving and traffic monitoring suggested by Shama in [159]. The surveillance system demonstrated by Hsieh in [158], and mobile robot navigation, which includes tasks like target tracking in the study by Wang in [161]. Moreover, the research by Mohanty in [160] utilized object collision prevention as multiple object technique.

By incorporating appearance data of tracked objects to connect new detections to previously identified objects, Deep-SORT, a sophisticated tracking algorithm, improves on the Hungarian algorithm. Deep-SORT was created as an expansion of the SORT technique, uses deep learning methods to increase tracking accuracy, particularly in difficult situations with occlusions and crowded spaces. The three main components of the SORT and Deep-SORT approaches Kalman Filter-based estimation, data association, and track supervision are all part of the same architectural framework [162].

However, the paper presented by Sapkota in [163] further simplifies the Deep-SORT architecture into two main parts: the embedding model and the detection model, which are usually based on pre-trained object detection frameworks like YOLO, SDD or Faster

R-CNN. The embedding model gives detected objects distinct feature vectors for reliable tracking across frames, while the detection model finds and identifies human figures in every video frame. Deep-SORT's advanced association and tracking mechanisms are one of its main advantages; they are excellent at linking detections across frames, even in difficult situations like partial visibility or occlusions. The Hungarian algorithm facilitates this association process by effectively identifying the best matches between existing tracks and current detections by measuring the dissimilarities between embedding objects using a cost matrix. This is the main reason we have implemented algorithm for tracking task by both the SDD and YOLOv4 Models. It helps track the trajectory movement of suspicious persons and weapons even in occluded conditions.

By merging data from earlier frames with the current detections, Deep SORT used Kalman filtering to monitor detection errors and guarantee smooth trajectory predictions. It is ideal for crowded settings where objects may momentarily obstruct one another due to its effective occlusion management capabilities. Deep SORT involves calculating costs for association represented in matrix form for the Hungarian algorithm, but it does not have a single overarching equation. In order to predict and update steps for tracked objects, the Kalman filter equations are essential.

Prior to using the Hungarian algorithm to optimise the assignment of new detections to preexisting tracks, each bounding box in the first frame was given a distinct track ID. This integrated approach that combines spatial and appearance information contributes significantly to object tracking robustness and accuracy, particularly in complex scenarios where both spatial and visual distances considered within the cost function ci, j formulation in Deep-SORT. This is represented in equation 4.2 as follows:

$$ci, j=\lambda\cdot d\text{spatial}\ (I, j) + (1-\lambda)\ \cdot d\text{visual}\ (i, j) \qquad (4.2)$$

Where:

The spatial distance, denoted by the Mahalanobis distance between the detected bounding box and its predicted position based on the tracked object's most recent position, is dspatial (I, j). The visual distance, I. dvisual (i, j), represents the difference between the observed object j's appearance and the tracked object i's previous appearances.

The trade-off between visual and spatial distances is controlled by the parameter λ. It is a value between 0 and 1, where 1 means spatial information will get more weight and 0 indicates that visual information is given more weight.

The Mahalanobis distance between the detected bounding box and its predicted position based on the last known position of the tracked object is given by Qiu in [164]. This is commonly used to represent the spatial distance dspatial (i, j) in the Deep SORT algorithm. The Mahala Nobis distance is calculated as follows:

$$\text{spatial}\,(i,j) = (xj - x^{i})^{T} \cdot S_{I}^{-1} \cdot (xj - x^{i}) \qquad (4.3)$$

Where.

xj is the centroid (or other representative point) of the detected bounding box *j*.

x^{I} is the predicted centroid of the tracked object *i* based on its last known position.

S_{I} is the covariance matrix associated with the predicted position of the tracked object *i*.

T denotes the transpose operation. The visual distance *d* visual (*i*, *j*) in Deep SORT represents the dissimilarity between the appearance of the detected object *j* and the historical appearances of the tracked object *i*. This often calculated using methods such as cosine distance, Euclidean distance, or other similarity metrics based on the feature representations of the object's appearance.

The specific expression for *d*visual (*i*, *j*) depends on the feature representation used for the appearance information.
Mathematically.

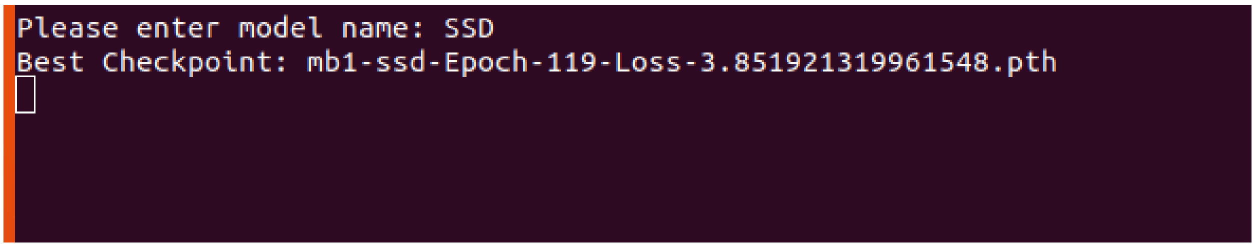
$d^{(2)}$ (i, j) = min {1 - $r_j^T r_k^{(i)} \mid r_k^{(i)} \epsilon R_i$}   (4.4)

Where r_j is the appearance descriptor extracted from within the jth detected bounding box, and R_i is the set of the last 100 appearance descriptors, r_k^i associated with the ith track

Appearance descriptors derived from a wide residual neural network, which consists of two convolutional layers followed by six residual blocks.

4.6 SDD Methodology Result

The outcome of applying the SSD method to the crime dataset is described in this section. About eight hours are spent on the training. The batch size was limited to 16 and the number of data loading rates set to 20 in order to optimize the memory infrastructure. This is the largest size and type of dataset that the system architecture we used permits. We used AWS EC2 P3 instances and local infrastructure for the training. 500 epochs assumed for the model training. But at 121 epochs, the model began to overfit. Our model was now "overfitting" since we saw a decline in both the training and validation losses. At the minimum loss function, the matching training weight was saved. This corresponded to 119 Epoch. At test time, the weight that matches the assumption was retained. The best result epoch on the training set is illustrated in Fig 4.2. Also Fig 4.4 illustrates the loss function graph for the training set.



```
Please enter model name: SSD
Best Checkpoint: mb1-ssd-Epoch-119-Loss-3.851921319961548.pth
```

Figure 4.2. Best weight Checkpoints

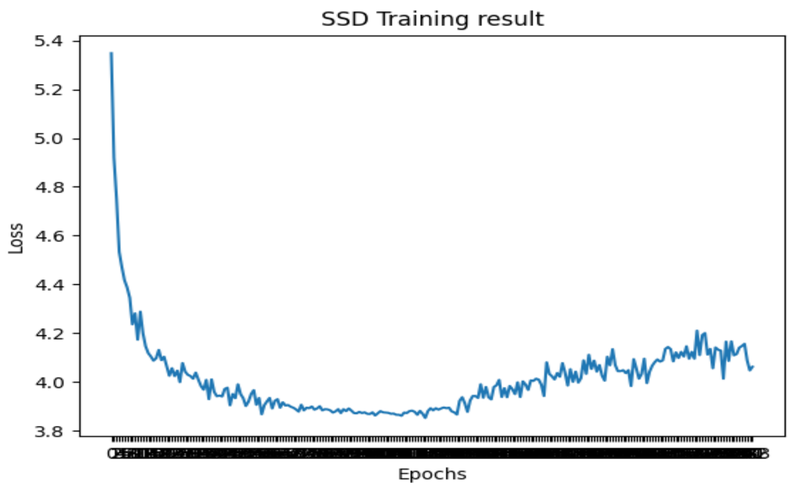
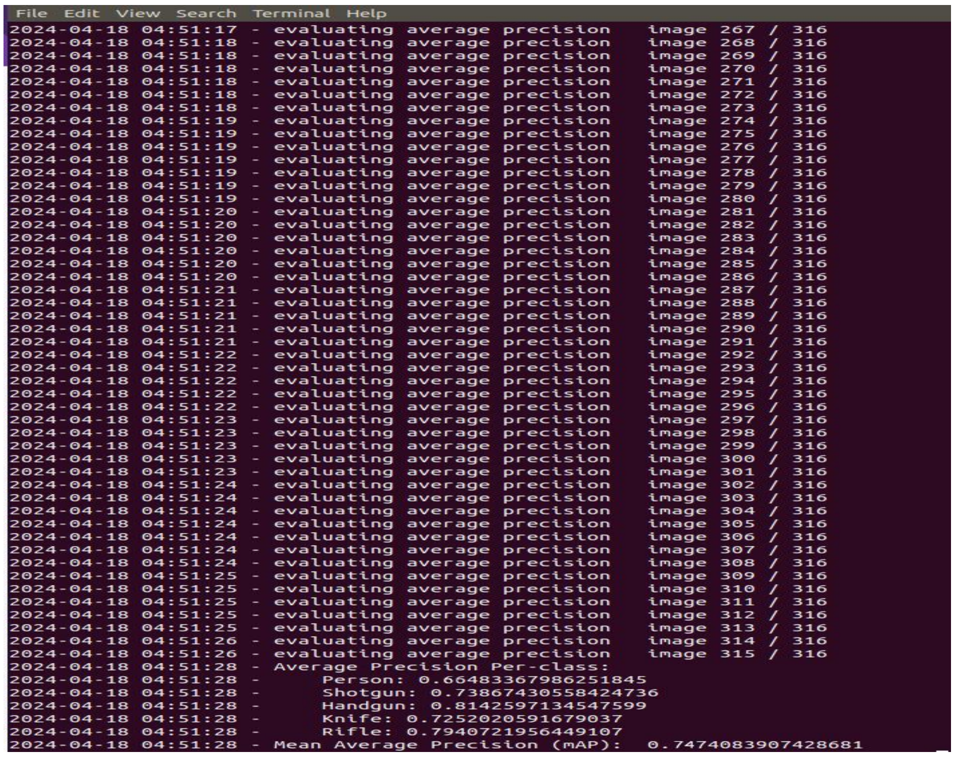


Figure 4.3. The Training loss

Figure 4.3 shows that after a certain number of steps, the training loss (which was already converging to zero) starts increasing again.

The model is most likely overfitting the data and failing to learn features that could be highly generalizable to unseen datasets, as evidenced by the fact that the remaining loss keeps rising. The graph shows this crucial moment as the ideal checkpoint or optimal weight. At that point, the procedure was stopped, and the associated weight was preserved. After 119 epochs, the model achieved its maximum mean average precision of 74.74%, as shown in Figure 4.3. As the model picked up new features from the datasets, the loss function continued to drop. Figure 4.4 summarizes the class by class performance for the model at different epochs during the training process.



```
2024-04-18 04:51:17 - evaluating average precision   image 267 / 316
2024-04-18 04:51:18 - evaluating average precision   image 268 / 316
2024-04-18 04:51:18 - evaluating average precision   image 269 / 316
2024-04-18 04:51:18 - evaluating average precision   image 270 / 316
2024-04-18 04:51:18 - evaluating average precision   image 271 / 316
2024-04-18 04:51:18 - evaluating average precision   image 272 / 316
2024-04-18 04:51:18 - evaluating average precision   image 273 / 316
2024-04-18 04:51:19 - evaluating average precision   image 274 / 316
2024-04-18 04:51:19 - evaluating average precision   image 275 / 316
2024-04-18 04:51:19 - evaluating average precision   image 276 / 316
2024-04-18 04:51:19 - evaluating average precision   image 277 / 316
2024-04-18 04:51:19 - evaluating average precision   image 278 / 316
2024-04-18 04:51:19 - evaluating average precision   image 279 / 316
2024-04-18 04:51:19 - evaluating average precision   image 280 / 316
2024-04-18 04:51:20 - evaluating average precision   image 281 / 316
2024-04-18 04:51:20 - evaluating average precision   image 282 / 316
2024-04-18 04:51:20 - evaluating average precision   image 283 / 316
2024-04-18 04:51:20 - evaluating average precision   image 284 / 316
2024-04-18 04:51:20 - evaluating average precision   image 285 / 316
2024-04-18 04:51:20 - evaluating average precision   image 286 / 316
2024-04-18 04:51:21 - evaluating average precision   image 287 / 316
2024-04-18 04:51:21 - evaluating average precision   image 288 / 316
2024-04-18 04:51:21 - evaluating average precision   image 289 / 316
2024-04-18 04:51:21 - evaluating average precision   image 290 / 316
2024-04-18 04:51:21 - evaluating average precision   image 291 / 316
2024-04-18 04:51:22 - evaluating average precision   image 292 / 316
2024-04-18 04:51:22 - evaluating average precision   image 293 / 316
2024-04-18 04:51:22 - evaluating average precision   image 294 / 316
2024-04-18 04:51:22 - evaluating average precision   image 295 / 316
2024-04-18 04:51:22 - evaluating average precision   image 296 / 316
2024-04-18 04:51:23 - evaluating average precision   image 297 / 316
2024-04-18 04:51:23 - evaluating average precision   image 298 / 316
2024-04-18 04:51:23 - evaluating average precision   image 299 / 316
2024-04-18 04:51:23 - evaluating average precision   image 300 / 316
2024-04-18 04:51:23 - evaluating average precision   image 301 / 316
2024-04-18 04:51:24 - evaluating average precision   image 302 / 316
2024-04-18 04:51:24 - evaluating average precision   image 303 / 316
2024-04-18 04:51:24 - evaluating average precision   image 304 / 316
2024-04-18 04:51:24 - evaluating average precision   image 305 / 316
2024-04-18 04:51:24 - evaluating average precision   image 306 / 316
2024-04-18 04:51:24 - evaluating average precision   image 307 / 316
2024-04-18 04:51:24 - evaluating average precision   image 308 / 316
2024-04-18 04:51:25 - evaluating average precision   image 309 / 316
2024-04-18 04:51:25 - evaluating average precision   image 310 / 316
2024-04-18 04:51:25 - evaluating average precision   image 311 / 316
2024-04-18 04:51:25 - evaluating average precision   image 312 / 316
2024-04-18 04:51:25 - evaluating average precision   image 313 / 316
2024-04-18 04:51:26 - evaluating average precision   image 314 / 316
2024-04-18 04:51:26 - evaluating average precision   image 315 / 316
2024-04-18 04:51:28 - Average Precision Per-class:
2024-04-18 04:51:28 -     Person: 0.66483367986251845
2024-04-18 04:51:28 -     Shotgun: 0.73867430558424736
2024-04-18 04:51:28 -     Handgun: 0.8142597134547599
2024-04-18 04:51:28 -     Knife: 0.7252020591679037
2024-04-18 04:51:28 -     Rifle: 0.7940721956449107
2024-04-18 04:51:28 - Mean Average Precision (mAP):  0.7474083907428681
```

Figure 4.4. Single Shot Multi-Box Detection Epochs Performances

4.6.1 Generating Class Inference

The model was validated on unseen dataset of crime videos, as shown in Figure 4.4, with a mean Average Precision (mAP) of roughly 75%. Figures 4.5a and 4.5b display the findings of the inference made during the generalisation process.

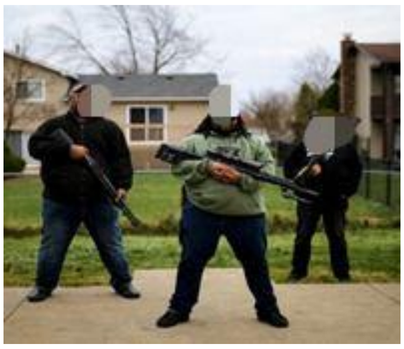


Figure 4.5(a). Input Image

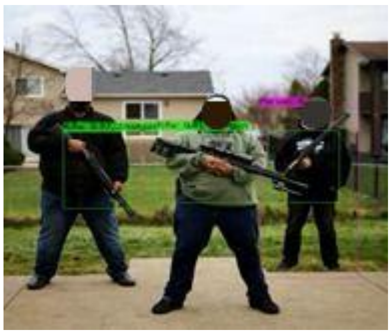


Figure 4.5(b). SDD Model Prediction result for sampled image

4.6.2 The SDD Performance Metrics

Our main evaluation metric, mean average precision, calculates the model's overall accuracy across all classes. To provide a comprehensive picture of the model's detection performance, mAP is computed at different Intersection over Union (IoU) thresholds.

By averaging the precision for each class, the model displayed in Fig. 4.5 reached a mean average precision of 74.79%. The precisions attained by each class represented in Table 4.2.

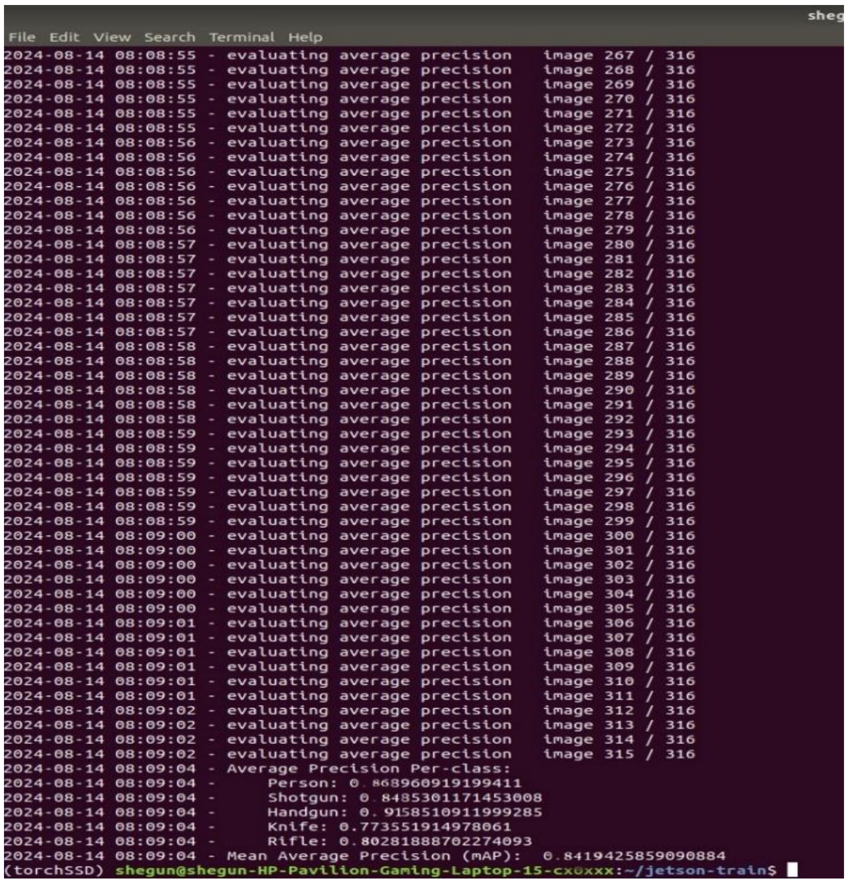
Table 4.2. Class by class precisions. For SDD Model

Class	*AP*
Person	66.48
Shotgun	73.86
Handgun	81.42
Knife	72.52
Rifle	79.49

4.6.3 The SDD Hyperparameter Tuning- Improving Model Performance

One of the notable outcomes of this chapter is the hyperparameter tuning of the configuration variables. We optimized the model in pursuit of enhanced performance, which improved generalization and prediction accuracy and reduced overfitting or underfitting. This ultimately increased the model's robustness. The optimization process involved adjusting several configuration variables, including changing the dataset split to 80:20, lowering the learning rate from 0.001 to 0.0001, and reducing the batch size to 16.

Figure 4.7 displays the graph of the loss function and the model performance achieved under these optimized conditions. Additionally, Figure 4.6 illustrates the epoch results for the optimization.



```
2024-08-14 08:08:55 - evaluating average precision   image 267 / 316
2024-08-14 08:08:55 - evaluating average precision   image 268 / 316
2024-08-14 08:08:55 - evaluating average precision   image 269 / 316
2024-08-14 08:08:55 - evaluating average precision   image 270 / 316
2024-08-14 08:08:55 - evaluating average precision   image 271 / 316
2024-08-14 08:08:55 - evaluating average precision   image 272 / 316
2024-08-14 08:08:56 - evaluating average precision   image 273 / 316
2024-08-14 08:08:56 - evaluating average precision   image 274 / 316
2024-08-14 08:08:56 - evaluating average precision   image 275 / 316
2024-08-14 08:08:56 - evaluating average precision   image 276 / 316
2024-08-14 08:08:56 - evaluating average precision   image 277 / 316
2024-08-14 08:08:56 - evaluating average precision   image 278 / 316
2024-08-14 08:08:56 - evaluating average precision   image 279 / 316
2024-08-14 08:08:57 - evaluating average precision   image 280 / 316
2024-08-14 08:08:57 - evaluating average precision   image 281 / 316
2024-08-14 08:08:57 - evaluating average precision   image 282 / 316
2024-08-14 08:08:57 - evaluating average precision   image 283 / 316
2024-08-14 08:08:57 - evaluating average precision   image 284 / 316
2024-08-14 08:08:57 - evaluating average precision   image 285 / 316
2024-08-14 08:08:57 - evaluating average precision   image 286 / 316
2024-08-14 08:08:58 - evaluating average precision   image 287 / 316
2024-08-14 08:08:58 - evaluating average precision   image 288 / 316
2024-08-14 08:08:58 - evaluating average precision   image 289 / 316
2024-08-14 08:08:58 - evaluating average precision   image 290 / 316
2024-08-14 08:08:58 - evaluating average precision   image 291 / 316
2024-08-14 08:08:58 - evaluating average precision   image 292 / 316
2024-08-14 08:08:59 - evaluating average precision   image 293 / 316
2024-08-14 08:08:59 - evaluating average precision   image 294 / 316
2024-08-14 08:08:59 - evaluating average precision   image 295 / 316
2024-08-14 08:08:59 - evaluating average precision   image 296 / 316
2024-08-14 08:08:59 - evaluating average precision   image 297 / 316
2024-08-14 08:08:59 - evaluating average precision   image 298 / 316
2024-08-14 08:08:59 - evaluating average precision   image 299 / 316
2024-08-14 08:09:00 - evaluating average precision   image 300 / 316
2024-08-14 08:09:00 - evaluating average precision   image 301 / 316
2024-08-14 08:09:00 - evaluating average precision   image 302 / 316
2024-08-14 08:09:00 - evaluating average precision   image 303 / 316
2024-08-14 08:09:00 - evaluating average precision   image 304 / 316
2024-08-14 08:09:00 - evaluating average precision   image 305 / 316
2024-08-14 08:09:01 - evaluating average precision   image 306 / 316
2024-08-14 08:09:01 - evaluating average precision   image 307 / 316
2024-08-14 08:09:01 - evaluating average precision   image 308 / 316
2024-08-14 08:09:01 - evaluating average precision   image 309 / 316
2024-08-14 08:09:01 - evaluating average precision   image 310 / 316
2024-08-14 08:09:01 - evaluating average precision   image 311 / 316
2024-08-14 08:09:02 - evaluating average precision   image 312 / 316
2024-08-14 08:09:02 - evaluating average precision   image 313 / 316
2024-08-14 08:09:02 - evaluating average precision   image 314 / 316
2024-08-14 08:09:02 - evaluating average precision   image 315 / 316
2024-08-14 08:09:04 - Average Precision Per-class:
2024-08-14 08:09:04 -     Person: 0.868960919199411
2024-08-14 08:09:04 -     Shotgun: 0.8485301171453008
2024-08-14 08:09:04 -     Handgun: 0.9158510911999285
2024-08-14 08:09:04 -     Knife: 0.773551914978061
2024-08-14 08:09:04 -     Rifle: 0.80281888702274093
2024-08-14 08:09:04 - Mean Average Precision (mAP):  0.8419425859090884
(torchSSD) shegun@shegun-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/jetson-train$
```

Figure 4.6. Optimized Mean Average Precision (mAP) for SDD Model

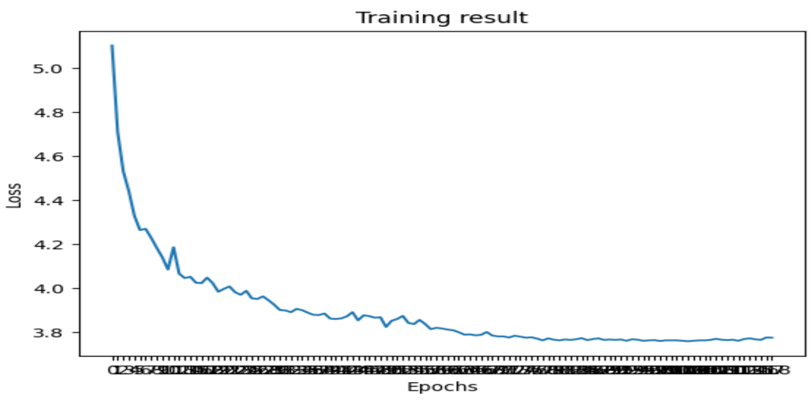


Figure 4.7. Training Loss for the optimized Mode

Table 4.3 Optimized SDD Precision

Class	*Precision(%)*	
	Original Model	*Optimized Model*
Person	66.48	86.9
Shotgun	73.86	84.9
Handgun	81.42	91.6
Knife	72.52	77.4
Rifle	79.49	80.3

4.6.4 Analysis of the findings- Model and Class by Class Precision

The model has done reasonably well to identify each class in the validation crime video particularly with the boost after optimization which improved accuracy and precision after hyperparameter tuning.

Overall Performance

mAP Comparison:

- Original Model mAP: **74.74**
- Mean average precision for Optimised Model: **84.19**

The improvement in mAP from 74.74 to 84.19 is significant, indicating that the optimized model has better overall performance in detecting and classifying objects. This increase of approximately 9.45 points suggests that the optimizations made in the model have successfully enhanced its ability to generalize and accurately predict across all classes.

Class-by-Class Performance

The class-wise precision percentages provide further insights into how the model performs for individual classes:

Detailed Analysis:

Person:

- **Original**: 66.48%
- **Optimized SDD** (mean average precision)= 86.9%

Analysis: The detection of ` ` Person'' class presents the largest increase, (+20.42%). This means the model has improved a lot at recognizing people across scenes.

Shotgun:

- **Original**: 73.86%
- **Optimized**: 84.9%

Analysis: The accuracy improved 11.04%, indicating a significant improvement in the recognition of the shotgun that could be very important to certain applications, such as security or surveillance

Handgun:

- **Original**: 81.42%
- **Optimized**: 91.6%

Analysis: The 10.18% increase means the optimized model is more capable to recognize handguns, which is quite significant for some applications that need a highly precise in weapon detection

Knife:

- **Original**: 72.52%
- **Optimized SDD**: 77.4%

Analysis: 4.88% increment is not as high as in other classes, which means that despite the hyper parameter tuning, there is a room for modification, more transformations would be beneficial for this class

Rifle:

- **Original**: 79.49%
- **Optimized**: 80.3%

Analysis: Although there is an improvement, it might not be as significant as it is for other classes, as the model's performance for rifles is already comparatively high, according to the marginal increase of 0.81%.

Assessment of Optimisations

The large gains in most classes demonstrate we made the following optimizations: **Augmentation**: Transitional augmentation was applied to the dataset to increase class instances

Model Design: With architectural adjustment like as deeper networks (changed layer number and types), the training system was able to extract more features that aided improved generalisation.

Hyperparameter tuning: Tweaking learning rates, batch sizes, and other hyperparameters resulted in better convergence of the model during training.

4.7 Comparison of the SDD Result with existing models

This method uses computer vision algorithms to identify weapons and classify high-risk individuals. As a result, we benchmark the model's performance using related detection algorithms from the findings of researchers I the literature reviewed. Our model performs well and demonstrates with competitive precision when compared to the author's Yolov4 on UAV imagery in [165]. The SDD model exhibits comparative competitiveness because, while the overall mAP in this related work was 62.71% with "Person" at 48.67%, we achieved 81.4% mAP for the same class in this study.

Further comparison with additional similar works in the Table 2.0 in the literature reveals that the performance of the optimized SDD in precision is competitive. This was specifically compared with the performance of the study in [118], where the author achieved 56.8%. Further comparison with the performance of the studies in [125],77.6%; [124],52%; [122],85% and [123],89% respectively, demonstrates the competitiveness of our SDD model, especially after the optimisation process was performed.

4.8 Conclusion

According to the validation conducted in this section, the optimised SDD model is a major improvement over the original, especially when it comes to identifying important classes like "Person," "Shotgun," and "Handgun." Although there are improvements for "Knife" and "Rifle," they are not as noticeable. Future research could concentrate on improving these classes' performance even to attain more evenly distributed precision across all categories. Overall, the optimised model significantly improves performance. This result is significant for applications where high crime detection and classification accuracy is required.

CHAPTER FIVE

The YOLOv4 Neural Based Network Methodology

5.0 Introduction

The rest of this study focusses on improving security and surveillance by using a YOLOv4 model to detect anomalies and track suspicious human behaviours through weapon detection. The collection and pre-processing of the training data are described in detail in Chapter 3. As an extension of the model implementation, this section covers important methodology, architecture selection, dataset training and model validation, result and discussion. It also includes a comparison with findings from the SDD Method presented in Chapter 4.

The chapter includes a subsection detailing integration of the weapon and person detection system achieved with YOLOv4 with a high-risk classifier, developed using Python script. This system employs Euclidean distance and the DEEPSORT tracking algorithm to automatically track and label potential criminals as high-risk individuals. The section concludes by highlighting the limitations of the model.

5.1 Exploring YOLOv4 Detector Method with CNN

All of the fundamental components of building and training deep networks has been published by computer vision researchers. Nevertheless, most of their works have overlooked the use of trainable multi-layer convolutions to predict weapons and suspicious human in a pre-crime video. This is probably one of the most important aspects of deep networks for computer vision and image [166]. The paper published by

[167] popularized the concept of convolutional neural networks by introducing the LeNet-5 network for digital recognition. Convolutional networks arrange each layer into feature maps, which can be thought of as parallel planes or channels, rather than linking every unit in a layer that comes before it. Similar to standard shift-invariant image convolution and correlation, the weighted sums in a convolutional layer are only carried out within a small local window, and the weights are the same for every pixel [168].

YOLO (short for "You-Only-Look-Once"), is an algorithm that takes the whole image at the concurrently and offers forecasts along with class probabilities and bounding box coordinates [169]. YOLO's greatest feature is speed. It processes about 45 frames per second. Version 4 is among the quickest and most accurate YOLO variants when it comes to object detection. There are 53 convolution layers in the YOLO version 4 algorithm (see Figure 2). There are three types of layers in the architecture. Layer one which the activation can be easily transferred to the inner layer neural network, where the residual layer is created [170]. The output of layer one is added to the output of layer two in a residual configuration. Second, we have the detection layers, which do detection at three stages. The grids are enlarged for detection. The third layer is up sampling layer, which indexes up the spatial resolution. Here the image is up sampled before the scaling.

According to the study by [171], the release of YOLOv4 marks a substantial development in the YOLO family of real-time object detection algorithms. Researchers have used this algorithm extensively to solve object detection problems, such as crime forecasting. Research on using the algorithm to predict the level of crime committed by offenders in pre-crime scenes is, however, lacking. This section of the research seeks to close this gap. The section seeks to improve pre-crime prediction precision, F1 scores, and recall values by incorporating more complex components into its architecture. It may also improve on the detection outcomes achieved with the SSD method covered in Chapter 3 of the study. To overcome some of the research's limitations related to SSD and improve detection speed and accuracy, the YOLOv4 algorithm was trained using the same dataset. Figure 5.0 gives the system diagram, adapted for the YOLOv4 neural algorithm. The figure illustrates the sequence of tasks required to achieve the crime prediction with YOLOv4 version in this work.

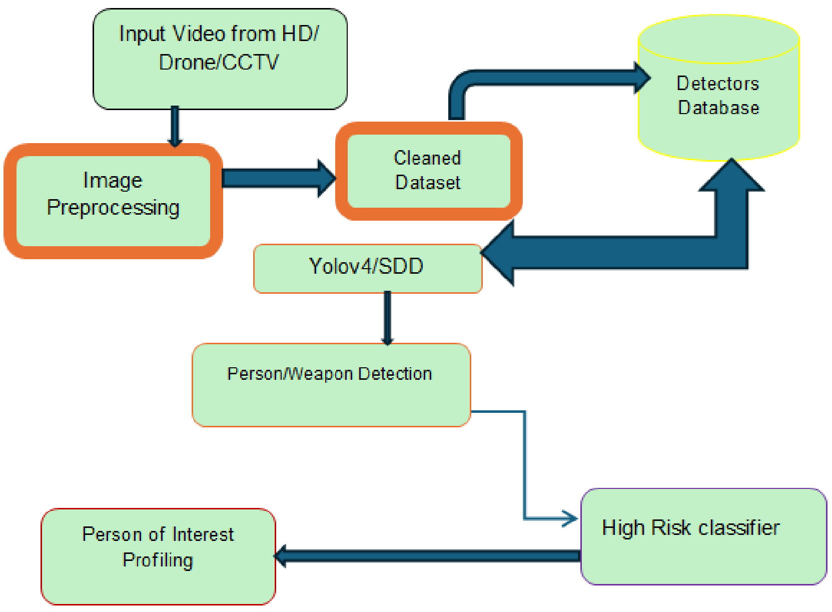


Figure 5.0. YOLOv4 Crime Prediction Pipeline

5.2 Multiple Camera Mapping Technique for Yolov4 Dataset

As illustrated in Figure 5.1, the crime video dataset, sourced from various camera types (such as CCTV, HD, drone etc). Then, standardisation for model training becomes necessary. The research conducted by Abishek, 2025 in [178], shows that the quality of images contained in deep learning datasets determines the effectiveness of training and generalizability of the model. Normalisation begins with the image resolutions obtained from the multiple camera sources. This can sharpen the images, and helps maintain uniform illumination input data.

For this reason, this section outlines the essential multi-camera technologies and methods that enhances quality camera outputs in multiple camera datasets. This is an important step to ensure a uniform image quality and transformation per pixel in system training

5.2.1 The Camera Matrix

This is an important idea in computer vision and computer graphics. The camera matrix in (eqn. 5.0) describes the intrinsic parameters of the camera. From the equation, the relationship between a point's 3D location in the world and the location of its 2D projection on the image plane of the camera is mathematical represented. In image rectification, 3D reconstruction, and object detection applications (feature extraction, object localization within image and drawing bounding boxes around detected object), the camera matrix is crucial [172]. It is shown that the 4×4 camera matrix P ̃is necessary for differentiable rendering and is optimised through neural networks. In more recent work,the author in [173] extends this to wide-angle lenses.

The standard *3×4* projection matrix created by combining the intrinsic and extrinsic parameters after the calibration matrix K has been parameterized as follows:

$$\tilde{P} = K\,[R\ t] \qquad (5.0)$$

Where R is the rotation matrix and t is the translation vector. This matrix P uses homogeneous transformation to transform 3D world points

$$pw = (xw, yw, zw, 1)T \qquad (5.1)$$

into 2Dimage coordinates

xs=(xs,ys,1)T. (5.2)

This equations 5.1 and 5.2 are relevant to our research because the yolov4 takes input image as 2D formats such as JPEG, or PNG , which are images in 2-dimension. They are pixel comprises intensity and color, then conversion from the 3D to 2D achieved with the camera matrix in equation 5.0. For applications that require invertibility, an extended *4 × 4* camera matrix P may be made by retaining the last row. Equation 5.0 becomes (5.1) when such complicated problems are solved.

$$\tilde{P} = \begin{bmatrix} K & 0 \\ O^T & 1 \end{bmatrix} \begin{bmatrix} R & T \\ O^T & 1 \end{bmatrix} = \tilde{K}E, \quad (5.3)$$

where K ˜is the differential-state full-rank calibration matrix and E is a rigid-body (Euclidean) transformation. With this parameterization, a 3D world coordinate

$$\tilde{p}_w = (x_w, y_w, z_w, 1) \quad (5.4)$$

in homogeneous 3D world coordinates can be directly projected to screen coordinates with disparity;

$$x_s = (x_s, y_s, 1)^T \quad (5.5)$$

where ~ denotes an equality up to scale.

5.2.2 Mapping from one camera to another

Geogeardis et.al, 2025 in [179] emphasized the importance of dataset merging to generate multiple images essential for data training. As our task involves merging multiple pictures of the same or different 3D or 2D scene with different camera orientations or positions, the projection to screen coordinates can be expressed as equation 5.6, using the full rank 4 x 4 camera matrix $\overline{\mathrm{P}} = \overline{\mathrm{K}}\mathrm{E}$.

$$\bar{\mathrm{x}}_0 \sim \overline{\mathrm{K}}_0 \mathrm{E}_{0\mathrm{p}} = \overline{\mathrm{P}}_{\mathrm{op}}. \quad (5.6)$$

We can use the following expression to compute the 3D point location p given that we know the Z-buffer (or disparity) value d_0 for a pixel in one image:

$$P_0 \sim {E_0}^{-1}{\overline{\mathrm{K}}_0}^{-1}\ \bar{x}_{\mathrm{o}} \tag{5.7}$$

Equation 5.7, when projected into another image yields

$$\bar{\mathrm{x}}_1 \sim \overline{\mathrm{K}}_1 \mathrm{E}_{1\mathrm{p}} = \overline{\mathrm{K}}_1 \mathrm{E}_1 {\mathrm{E}_0}^{-1} {\overline{\mathrm{K}}_0}^{-1} \bar{\mathrm{x}}_0 = \overline{\mathrm{P}}_1 {\overline{\mathrm{P}}_0}^{-1} \bar{\mathrm{x}}_0 = M_{10} \bar{\mathrm{x}}_0 \tag{5.8}$$

As a result, equation 5.8 serves as the foundation for mapping numerous images from various cameras while preserving a constant level of the crime image quality across our dataset. Each component in the equation is understood as follows:

$\bar{\mathrm{x}}_1$: the post-projection image point in the second camera (Camera 1). In homogeneous coordinates, it could be a 2D point.

$\overline{\mathrm{K}}_1$:: Camera 1's intrinsic matrix, where K is the intrinsic matrix containing focal lengths and the optical center coordinates that convert the 3D world coordinates to 2D coordinates in the picture frame (for Camera 1).

E_1: This is the extrinsic matrix of the first camera and contains the rotation and translation parameters from the world coordinate system to the camera coordinate system.

P: Describe a point in 3D-space, in homogeneous coordinates and denoted [...] in practical applications a point in the world coordinate system.

(E_1): The extrinsic matrix of Camera 0, i.e. the matrix that maps the world coordinate system to the Camera 0 coordinate system

5.2.3 Image Pixel transforms

This section discusses the mathematical methodology on how we transformed the heterogenous images captured by CCT and HD to homogenous pixel used in the research. This task helps to align the various image properties to homogenous pixel for better training and model generalization. Equation 5.9 exemplifies the function that creates an output image from the multiple crime video input images. This is known as a general image processing operator. In this work, it represents the continuous domain:

$$g(x) = h\big(f(x)\big) \text{ or } g(x) = h(f_0(x), \dots, f_n(x)) \tag{5.9}$$

The input and output functions f and g operate over a range that can be either scalar or vector-valued, such as for colour images or 2D motion, where x is in the D-dimensional (typically D=2 for images) domain. The domain of discrete (sampled) images is made up of a finite number of pixel locations,

Such that $x = (i, j)$, and we can then write

$$g(i,j) = h(f(i,j)) \tag{5.10}$$

Multiplication and addition with a constant are two frequently utilized point processes from equation 5.10. Then we derive a key term in equation 5.11

$$g(x) = a\, f(x) + b \tag{5.11}$$

This is referred to as the gain and bias parameters in this work,

the operators a > 0 and b control brightness and contrast of the crime images in the images used. Additionally, the gain and bias parameters may vary spatially.

$$g(x) = a(x)f(x) + b(x) \tag{5.12}$$

Because it adheres to the superposition principle. The multiplicative gain both globally and spatially varying is a linear operation represented by equation 5.13

$$h(f_0 + f_1) = h(f_0) + h(f_1). \tag{5.13}$$

The linearised transform equation, now known as equation 5.13, standardised the dataset we used to train the Yolov4 and SDD models and could preserve uniformity of pixel, brightness, contrast, and other image properties from multiple camera sources. The performance of the corresponding predictive models was greatly enhanced by the pixel transform technique.

5.3 Justification for using YOLO Method for Crime Prediction

Bochkovskiy in [178], emphasis that speed and accuracy are essential features of yolov4 that distinguishes it for image detection and classification tasks in computer vision. This was supported by the research conducted by [182], where they used yolov4 to detect polyp. Moreover, the experimentation result of the human detection conducted by Xuan, 2022 shows that yolov4 is effective for complex targets including human detections [62]. The rationale for selecting the YOLOv4 approach for this work is summarised in Table 5.0.

Table 5.0. Rationale for UsingYOLOv4 Algorithm

Factor	Justification
Real-Time Detection	YOLOv4 aims for real-time object detection. It can process images at high speed and is valuable in time-sensitive applications in which fast access to the information is essential, such as surveillance and/or responding to dynamic crime scenarios.
Accuracy	YOLOv4 has superior accuracy in object detection under different photometric conditions.
Variability and Robustness	YOLOv4 is insensitive to lights, occlusions and various object scales. Such accuracy is crucial in practical applications where conditions may be highly dynamic and a reliable recognition of object is indispensable for a successful crime prediction
Integration with other System	The proposed design of YOLOv4 makes it easy to integrate with other systems (e.g., surveillance cameras and data analytics system, etc.). This can enable a full pipeline of crime prediction using object detection and other data sources.
Multiple Object Detection	YOLOv4 can detect multiple types of-object at the same time. When considering crime prediction, it can be used to recognize different types of objects in crime video including people, vehicles, weapon, firearms to judge potential risks
Transfer Learning	YOLOv4 has transfer learning functionality, that is, you can finetune the model on a given dataset. This makes it possible to tailor it to specific crime-related objects or events, and improves the performance of the method in crime prediction tasks
Data Driven Insight	The architecture of SSD is capable of low latency performance that is very important for its target applications that consists of missions demanding timely action
Open Source and Community Support	YOLOv4 operates on an open-source basis, and it grows every day with good community support. This openness makes the researchers, developers together to share knowledge, benchmark and improve the model for crime with applications.
Scalability	YOLOv4 is flexible to be run on different hardware, from server machine to edge device. This scalability in product will support other downstream use cases in security space

To sum it up, the neutral balance between speed, accuracy, robustness, and versatility of YOLOv4 makes it an ideal option for both the weapon and person of interest prediction in this research.

5.4 The File Structure and YOLOv4 Model Training Configuration

We described the YOLOv4-specific data collection, standardisation, cleaning, and preprocessing procedures in Chapter 3. The methods used to arrange the necessary training files and train the benchmarked YOLOv4 algorithm on the annotated crime dataset are the main topic of this section.

For easy understanding, it may be useful to define the following YOLOv4 training directories and terminology:

/darknet=the final directory containing the Darknet framework. The de facto framework for training and implementing the YOLO model is Darknet, an open source neural network framework created in C and CUDA.

Detector = This is a sub-command of Darknet that specifies the type of task performed. In this case, "detector" indicates that we are training a model for object

data/obj.data:

This is the path to the file that holds training relevant information. The data file typically includes: The number of classes, in respect to this research the person, knife, shotgun, handgun and Riffle. It is the Paths (URL or local) for the training/validation datasets. And backup directory for models and weight files.

. cfg/yolo-obj.cfg:

This is the path to the YOLO model configuration file. The. cfg, which contains the architecture of neural network such as: The number of layers. The layer type (e.g., convolutional layer, pooling layer, etc.). The parameters of each layer (filters, stride, padding) and other training parameters including batch size, learning rate, and iteration numbers.

. yolov4.conv.137:

This is the pre-trained weights file for transfer learning. The yolov4. conv. 137 file contains weights trained on a large dataset (such as COCO) of the first 137 layers of YOLOv4. Training with these weights allows us a faster model convergence for the model and enables better performance, as the model begins training from a set of features which already learned from a diverse dataset.

These intuitions allow us to express the weighted linear sum carried out in a convolutional layer. This is expressed in equation 5.14

$$s(i,j,c_2) = \sum_{c_1 \epsilon \{C_1\}} \sum_{:k;l) \epsilon N} w(k,l,C_1,C_2)\, x\,(i+k, j+l, c_1) + b(c_2) \qquad (5.14)$$

Where; s(i, j, c_2): is the feature map at position ((i,j)) for the channel (c_2). The output feature map is a 3D tensor with the height, width, and number of channels (*see Table 5.2 and Appendix B for Yolov4 Architecture Convolutional layer configurations*)

It worth noting that this summation is over all input channels (c_1) in the set ({c_1)}). The convolutional layer generally takes in the input of multiple channels. In our case, we use (RGB) images which has 3 channels.

Unlike image convolution, which applies the same filter to all (colour) channels, neutral network convolutions typically use different convolution kernels for each of the C_2 output channels and linearly combine the activations from each of the C_1 input channels in a preceding layer. This is because the convolutional neural network layer's main job is to build local features and combine them in different ways to produce more discriminative and semantically meaningful features [174].

Where $\sum_{(k,l)\in N}$ is sum over the spatial dimensions of the filter (kernel) (w). The kernel (which usually has shape K for squared kernels with indices ((k, l)).and

$(w(k,l,c_1,c_2))$:is the weight of the kernel at position (k, l) for the input channel (c_1) and the output channel (c_2). This means that each output channel has its own bank of weights for each input channel. Where, $(x(i+k,j+l,c_1)$ is the input feature map at $((i+k,j+l))$ for the input channel (c_1). The indices $(i+k)$ and $(j+l)$ tell us which area of the input feature map we are examining at any given position of the kernel. $(b(c_2))$ .This expression defines the bias term for the output channel (c_2).

Usually each output channel has its own bias which is added to the output of the convolution operation.

In this research, the file requirements and training procedure for the custom CNN YOLO v4 model simplified into the following 6 stages:

Stage 1: Configuring the YOLOv4 .cfg files for model training.

The file contained layer structures, batch size, and filter, and as its name implies, it is the structural configuration of YOLOv4, a file that we modelled to suit training requirements. For simplicity, the layer structure for the YOLOv4 convolutional neural network is mathematically represented in equation 5.9

Stage 2: Creating an obj. name file.

Obj.name file contains the name of the classes we are training, which includes person, short gun, riffle, handgun, and Knife. In this design, it was developed in sublime text environment and uploaded into data file inside darknet directory.

Stage 3: The annotated dataset was placed in “obj.name” file opened in the path.

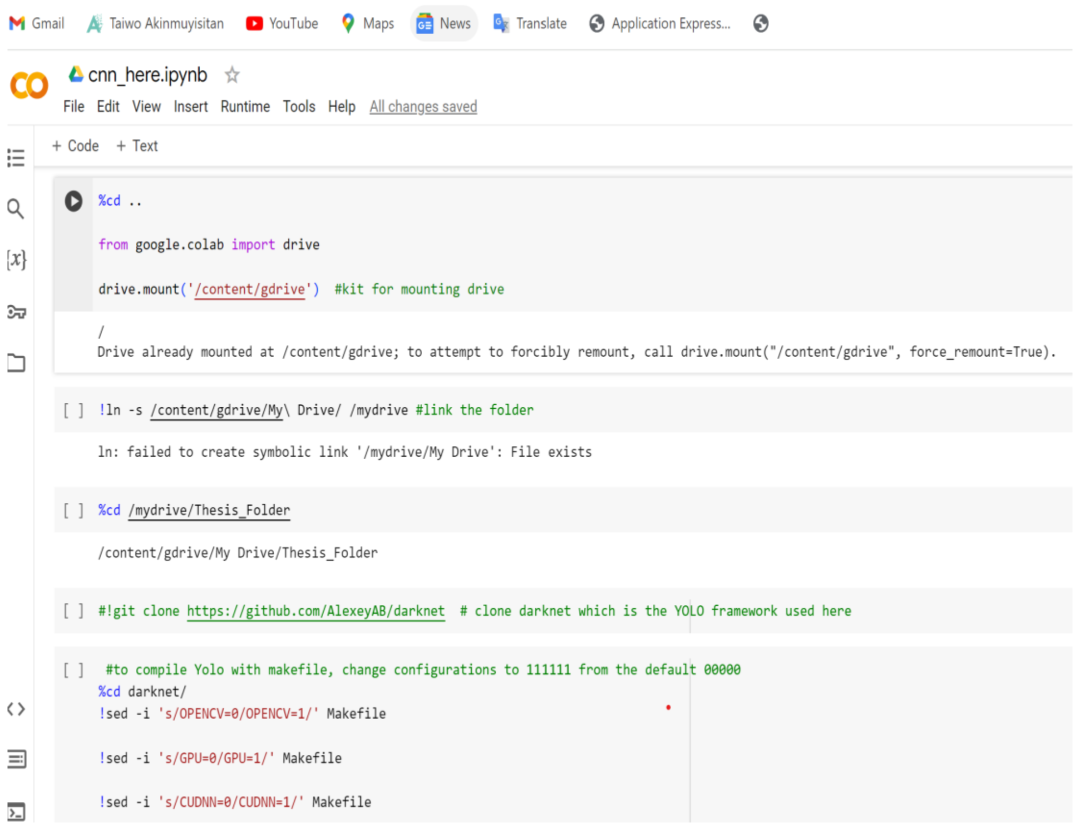
- YOLOv4>darknet>data

Stage 4: Splitting the dataset into train.txt and test.txt

Stage 5: Create a metadata file called “obj.data.”

Stage 6: Begin training execution

The Figure 5.1 shows the step-by-step process from importing libraries, mounting the Google drives, cloning the darknet neural network to compiling the yolov4 with make directory in Google Collaboratory environment, (Training in local GPU is going to be described in the next section



```
%cd ..

from google.colab import drive

drive.mount('/content/gdrive')  #kit for mounting drive

/
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
!ln -s /content/gdrive/My\ Drive/ /mydrive #link the folder

ln: failed to create symbolic link '/mydrive/My Drive': File exists
```

```
%cd /mydrive/Thesis_Folder

/content/gdrive/My Drive/Thesis_Folder
```

```
#!git clone https://github.com/AlexeyAB/darknet  # clone darknet which is the YOLO framework used here
```

```
 #to compile Yolo with makefile, change configurations to 111111 from the default 00000
%cd darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile

!sed -i 's/GPU=0/GPU=1/' Makefile

!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

Figure 5.1. Drive mounting and YOLO Darknet Cloning

Once the steps pointed out in Figure 5.1 is done and yolo compiles for you using make file command, the following single python command trains our model as expected.

. */darknet detector train data/obj.data cfg/yolo-obj.cfg yolov4. conv.137* **(***This command initialize the model training.*

5.5 YOLOv4 Algorithm Model Implementation on PC

The AI software's high computational demands make it extremely difficult. It is usually tedious to run complex algorithms on a CPU because the resources and training time needed are prohibitive. In our case, the training task was started on a high-specification computer system with a Graphical Processing Unit (GPU) and memory allocation. This proved to be difficult, though, because the research focusses on large-scale video datasets. Training took far too long, and the PC frequently shuts down due to the long training hours needed for the 10,000 iterations (10,000) that are required for the model to learn features that can make it generalise. This high computational requirement was highlighted in the paper recently published in 2025 by Ding [184] . It could lead to poor generalization if not managed.

In addition, from this point on we decided to use the cloud to make use of Google Collaboratory's T4/A100 (see Figure 5.1) GPU services to train the data, which was very expensive. In the end, the purpose of this training is a combination of local computing and shared cloud to reduce training costs and ensure the model is training effectively

This procedure is demonstrated on a GPU PC in the step-by-step setup below. It should be noted that at this point, we had already set up CUDA10.1, Cudnn7.5, and the Python Jupiter/Spyder virtual environment. The virtual environment's required libraries are installed or built, as shown in Table 5.1.

Step 1 Clone Darknet on Local PC

$>sudo apt install git *//This command installs git dependency on the terminal environment for the darknet)*

(Enter password: xyzzy)

$>git clone https: github.com/AlexeyAB/darknet *//this command installs darknet framework where YOLOV4 is derived)*

Step 2: Configure YOLOv4 architecture.

$>cd darknet // *change current directory*

$>ls // *display file within current working directory*

$>cfg)// *working in YOLOv4 configuration folder where we configure the training parameters*

(YOLOv4 series is in the YOLOV4-Custom.cfg folder). Open the Sublime text (or other text editors used for file configuration, double clicked to open the configured files). This was where the following important training parameters configured. (*See the complete configurations and set up in appendix B).* Table 5.1 summarizes the dependencies and libraries used for annotation, file configuration and training of the model.

Table 5.1. Packages and libraries

Library	**Function**	**Installation Code**
Imageio	Write images	pip install imageio
OpenCV	Draw Bounding Box, write text, formatting,	pip install OpenCV-python
Darknet	It's a framework used for yolo object detector	git clone https://github.com/AlexeyAB/darknet
labelImg	Data annotation and conversion to YOLOv4 format	pip install labelimg

5.5.1 Design Parameters for YOLO v4 Model Training

The following presumptions were made for the design of filters, batches, classes, channels, etc.

I. Batch: Sixty-four (64) was the value selected in the batch design. This implies that each batch could process sixty-four frames of the input data.

II. II. Subdivisions: The sixty-four-batch was divided into four iterations in order to improve the system's learning capabilities. 64/4 = 16. To improve learning weight per iteration, the subdivision was further reduced to sixteen. Thus, a stream of sixteen images is used as a unit step.

III. Image Width: 320 was the selected width. The width of the frame must be factor of thirty-two. The original image size was lowered from 608*608 to 320*320 as a result. This helped to conserve GPU infrastructure.

IV. Image Height=320**:** The same dimension as image width chosen to obtain perfect square shape.

V. Channel: The channel was set to 3 for the dataset because it is RGB image data that we are training. It could be noted that although we acquired video dataset (4-D tensor), it is easier to convert it to image RGB, a 3-D tensor during data annotation with YOLO ***labelIm***g.

VI. Design of the maximum batch size**:** 2000 is the minimum batch size standard. This is how many iterations a class must have. This is calculated as

Batch Size= 2000* number of classes (5.15)

2000*5= 10,000

after the five classes-person, knife, short gun, handgun and rifle, that the model will learn. This is the current iteration count needed for training for the model to generalise and learn key features.

Vii The step sizes' design**:** In the event that the employed algorithm converges to an orbit rather than a fixed point, step sizes characterise the oscillation magnitudes. It establishes the rate of descent of the optimisation error curves.

Step size = (0.8, 0.9) * (the maximum batch size) (5.15)

= (8000, 9000)

viii. Design of the Filter or Kernel**:** This is the weight learned by the convolution. The

Filter value= [Class + 5] *3 (5.16)

[5+5]3=10*3=30.

Note: In this YOLOv4 Model design, the filter value was changed to thirty from the default 18 at every convolutional layer proceeding the YOLO layer. The easiest way is to utilize the ***go to*** find feature on the top menu bar in the sublime text, click find and scroll through to change the filter default configuration to 30 as calculated.

Also, in YOLOv4 CNN design, *classes* changed from default value in analogous way as *filter* but on *YOLO layer* and not *convolutional layer*. So, in this design, it was changed to 5 at the YOLO layers which is the layer immediately after the convolutional layer where filter was changed. Altogether, filters and classes changed for a total of three layers each. Table 5.2 summarizes the design values and the assumptions made to implement the training.

Table 5.2. Configuring YOLOv4 Model for Training

Image Width	320	Each image width changed to 320 to manage memory infrastructure and improve training time
Image Height	320	Set to 320 to maintain perfect square with image Width. The chosen value ensured the image size is a multiple of 32
Batch Size	64	This implies 64 images of the dataset with dimension 320*320 are processed in one cycle or training iteration
Subdivision	16	With the training data divided into 64 subsets, the training process will still be slow. For faster training, 16 was set for faster training.
Max Batches	10000	Calculated as (No of classes i.e. 5) * 2000= 10000
Steps	(8000,9000)	Calculated as 80% , 90% of Max Batch
Filter	30	Calculated as (Class+5)*3. Filter adjustments occurs at the 3 conv. layers preceding yolo layer in the .cfg file
Learning Rate	0.001	Set to 10^{-3} .

5.5.2 Weight initialization

Weight optimization cannot begin until networks have been initialized. To break the symmetry, or ensure that none of the gradients were zero, early neural networks employed tiny random weights. However, it was noted that the activation would gradually decrease in deeper layers. To ensure a similar variance in the activations of subsequent layers, we need to take into account the fan-in of each layer, which is the number of incoming connections where activations are multiplied by weights.

In an early study on this subject, the author in [175] proposed that the fan-in's inverse be used to set the random initial weight variance. But at least close to the origin, their analysis relied on a linear activation function, such as a *tanh* function. The study by the author in [176] revised this analysis to take into consideration the asymmetric nonlinearity issue that exists in our dataset. If the weights are initialised to have zero mean and variance *Vi* for layer *l* and the original biases are set to zero, the variance of the linear summation will be zero.

$$Var[s_l] = n_1 V_l E[x_1^2] \quad (5.17)$$

$E[x_1^2]$ is the expectation of the squared incoming activations, and n_1 is the number of incoming activations weight in equation 5.17. In our research, we used Sigmoid activation for the model training in order to apply this to improve regularization and generalisation. The sigmoid activation was chosen because it minimizes the error function and helps the training outputs converge quickly. The output is fed into units in large stages after the summation s_l, which has zero mean, through the Sigmoid. Then, the model begins learning the features required to generalise on new datasets, initialised the learning weight.

YOLOv4 detection Weight

Optimally, this model's weights in the research saved every 1,000 of the total 10,000 training iterations, and a learning rate of 10^{-3} was employed. We can observe the learning of our model step by step over the training, and store the highest weight achieved in the course of the 10,000 iterations. This is referred to in this experimentation as the best weight, and the corresponding epoch noted.

The following is an explanation of the procedure for importing weights saved during object detection training: First, we search for the location of all of your data.obj, and configuration files on your computer. The network architecture, class names, and class colours are then loaded during import using the load_network function.

The Python import statement is used to import the libraries OpenCV, Darknet, NumPy, and DeepSORT that are required for this procedure.

In this section, we could also learn about model optimisation. Hoang in their research demonstrated how Bayesian hyperparameter tuning optimised the detection of fire in their experiment conducted in 2025 (Hoang et.al.,2025). Similarly, to enhance our model's performance, the hyperparameters indicated in Table 5.2 can be adjusted. For example, the learning rate can be changed to 10-4, 10-5, etc., and other yolo/.cfg file configuration values can be adjusted. However, in contrast to SDD, the model already achieved high accuracy, so no additional fine-tuning was necessary during the experiment.

5.6 The Model Class Detection and High Risk Person Classification Techniques

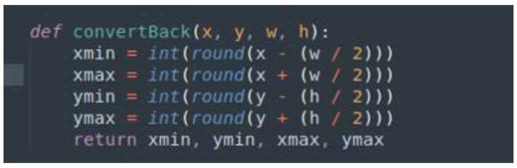
The primary purpose of custom YOLOv4 object detection is the end point. The main factors that need to be satisfied in order for us to accurately forecast and categorize High Risk. In addition, the recognition of certain weapons related to violent acts are an important part of this process.

To find these important points, we use the Yolov4 object detector in the design. We then move forward with developing a model that will make use of those YOLOv4 object detection weights. This model makes use of the motion estimation and object class classification functions that are inherent to the detection model (for the high_risk.py Python script, see Appendix F). It accomplishes this by determining where each of their actions falls on the spectrum for being classified as a "High-Risk" person.

ConvertBack and DetectFunction with YOLOv4 Object-detection are the steps needed to accomplish this.

ConvertBack

The process begins with the extraction of crucial information such as the center coordinates in the x (Xmin) and y (Ymin) axes, as well as the width (Xmax) and height (Ymax) of the objects detected. Figure 5.2 presents the screenshot of the snippet python code for convert back that implement the localized object coordinate



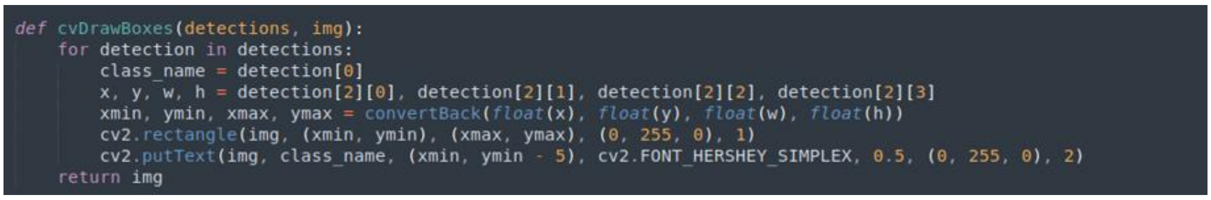
```python
def convertBack(x, y, w, h):
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))
    return xmin, ymin, xmax, ymax
```

Figure 5.2. The ConvertBack Function

By establishing the necessary classes within the model, it paves the way for localization of the object coordinates.

CvDrawBoxes

The Figure 5.3 presents the screenshot of the block of code that enables the model to efficiently draw rectangular bounding boxes around the detected object, thereby streamlining the visual representation of the identified elements.



```python
def cvDrawBoxes(detections, img):
    for detection in detections:
        class_name = detection[0]
        x, y, w, h = detection[2][0], detection[2][1], detection[2][2], detection[2][3]
        xmin, ymin, xmax, ymax = convertBack(float(x), float(y), float(w), float(h))
        cv2.rectangle(img, (xmin, ymin), (xmax, ymax), (0, 255, 0), 1)
        cv2.putText(img, class_name, (xmin, ymin - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    return img
```

Figure 5.3. CvDrawBoxes Function

DetectFunction

After that, the emphasis turns to using the detect function, which is crucial in deciding what to do when certain classes in the scene are detected. The system can identify

"Person" and initiate appropriate responses based on these detections by linking class IDs and names within a designated for loop "detection" function. When a human is detected in the scene, the convertBack functions take over, precisely determining the object's coordinates and using the CvDrawBoxes function to draw rectangular borders around the person as illustrated in figure 5.4.

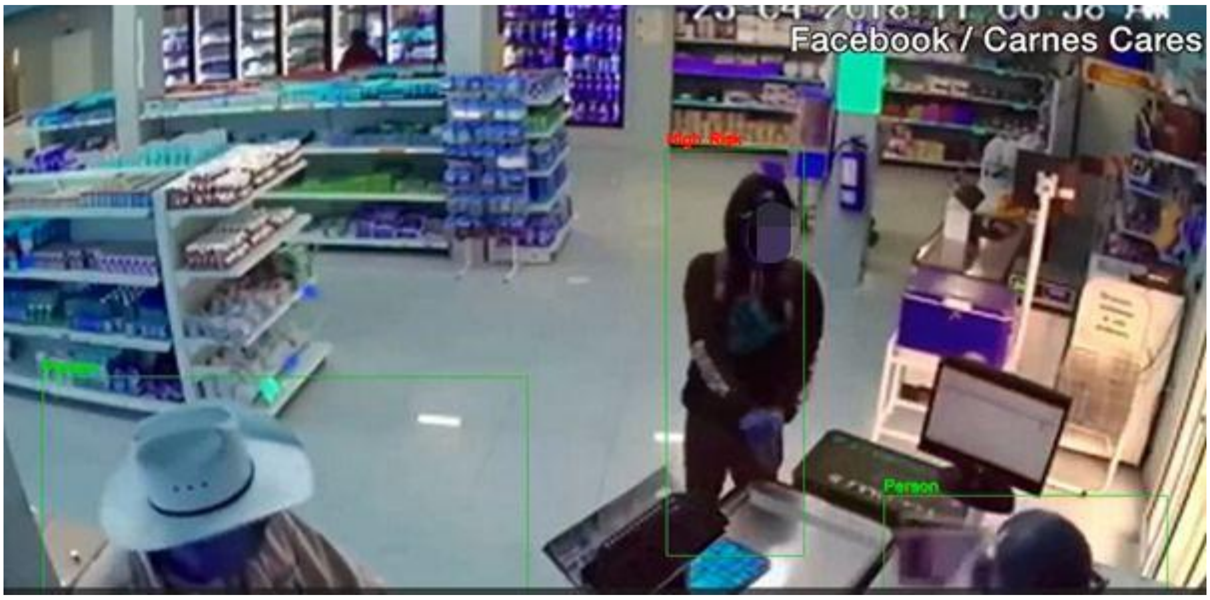


Figure 5.4. Detect Function identifying person of High Risk in Public Place

When other classes, such as "Weapons," are detected alongside a "Person," as shown in Figure 5.4, an advanced tracking mechanism driven by deep-sort algorithms with Eucledian distance is activated. The system can determine the degree of risk involved in the person's actions by tracking their movements. In the event that the algorithm detects high-risk behaviours, a "High Risk" prompt will appear on the screen, acting as a proactive step to address possible environmental threats.

Appendix F describes an object detection pipeline that uses the trained YOLOv4 model to analyse and visualise objects in a video stream in order to predict high-risk individuals.

First, import the required libraries, which include Darknet (`from darknet import *`), NumPy (`np`), and OpenCV (`cv2`). A set of object classes are defined, such as "Person," "Rifle," "Shotgun," "Handgun," and "Knife."

There are two important functions `ConvertBack` and `CvDrawBoxes` that take care of the drawing boxes before drawing object or converting the coordinates of the bounding box. The concepts were stated in the paper by Sengahbavali in [186]. Their work emphasized the need for adaptive bounding boxes to overcome the challenges of occlusion, and object orientation variability due to motion. The primary function, `YOLO()`, creates video writer for the processed output, loads the YOLOv4 model from the Darknet library, and initialised video capture object.

In a loop, we read, processed, and resized video frames using YOLOv4. The detected objects are visualized in real-time using Matplotlib, and a frame is labeled as a "High Risk" scenario if both a person and a weapon are detected. The processed video is saved as "test_sample.mp4 ", in our example.

5.6.1 Model Expected Risk

The Model training comes with associated risk that could cause errors or reduce system performance [26]. Equation 5.18 gives the expected risk of error in the experimental training of the model:

$$E_{Risk}\ (w) = \frac{1}{N}\sum L\big(hi, f(x_i, w)\big) \qquad (5.18)$$

$$hi = actual\ output$$

$f(x_i, w)$= The Predicted output of the AI Model

x_i = Crime Video Input containing the objects to predict.

Where E = Expected risk and the loss function L calculates the cost of predicting an output $f(x_i, w)$ for input x_i and the model parameter w, when the corresponding target is ${F_i}^2$. Errors can be caused by a variety of factors, such as inadequate dataset quality, problems with data augmentation, overfitting or underfitting, problems with hardware or software, and too high or too low learning rates, which can cause problems with training convergence. These hazards were taken into account when pre-processing and training the data.

The cost (penalty) could be a simple quadratic or robust of the function difference between the desired output (h_i) and the output predicted by the AI model given as; $f(x_i\ ;\ w)$. In simple term and with regards to the AI safety Model designed in this research, the expected error could be of serious ethical issues. In the context of a predictive artificial intelligence tool for criminal assessment, inaccuracies within the system may lead to two types of errors: *false positives* (incorrectly detected an individual as a criminal) and *false negatives* (failing to recognize an individual as a criminal).This pitfall may cause criminal masterminds make off crime scenes, or it may lead to wrongful accusations, legal feuds, and damage to the reputation of the organization.

Costs of these errors go beyond the budget to include legal fees, settlements, brand damage and deteriorated stakeholder confidence. The implementation of the AI solution also includes the costs associated with the retraining of the model, actualizing more accurate data, and implementing more solid error-prevention procedures.

Consequently, it is essential for organizations to consider the possible risks and costs of errors in deploying the CNN model. To mitigate these risks, it is important to undergo robust testing and validation procedures, and also apply regular monitoring of flow.

5.6.2 Performance Metrics

In the proceeding of the 9th International Conference of IEEE, 2006, Bashir described the performance metrics of object detection for a computer vision model as the evaluation method implemented to quantitatively analyse the performance of the model [4] In this work, was measured either on frame based or object based. Typically, the mean average precision (mAP) was used to gauge the object detector's performance. (mAP) provides a thorough assessment of a model's capacity to identify and categorise objects within an image by combining three crucial factors: precision, recall, and F1. Precision-recall curves were created for every class of objects in our dataset in order to calculate mAP. Plotting precision against recall at various confidence score thresholds is shown by these curves. The mean average precision was then calculated by averaging the area under the curves for each class of objects. It offers a balanced evaluation by considering precision and recall, making it particularly suitable for object detection and classification problems. It is the metric that objectively compare the output of the module performance with the ground truth to avoid false alarms [187]

The mean average precision calculated with equation (5.19)

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{5.19}$$

- N is the total number of object classes of the dataset
- AP_i is the average precision for each class i.
- m is the average precision for each class

5.6.2.1 Precision

Precision measures the proportion of correctly identified positive detections out of all the detections made by the model. It assesses the model's accuracy in predicting true positives while minimizing false positives as shown in equation 5.20

$$\text{Precision} = \frac{TP}{TP+FP} \tag{5.20}$$

TP= True Positive, which are the instances that are true objects and are positive by the model.

FP = False Positive, which are instances that are not true objects of a class but positive for the model

5.6.2.2 Recall

Equation 5.21 computes the proportion of true positives detected by the model out of all the ground truth objects present in the image. It measures the model's ability to find all relevant objects, minimizing false negatives.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{5.21}$$

Here,

FN= False Negative. These are instances where the object is positive (criminal), but the model wrongly predicted (failed to identify the criminal).

5.6.2.3 Model F1 Value

The F1 score is the harmonic mean of precision and recall and provides a balance between the two metrics. Equation 5.22 detail how a model can calculate its F1 value

It is calculated as:

F1 Score = 2 * {Precision* Recall} /{Precision + Recall}] (5.22)

For simplicity,

- Precision focuses on the accuracy of positive predictions.
- Recall focuses on the proportion of actual positives that correctly identified.
- F1 score provides a balance between precision and recall, especially when there is an uneven class distribution.

5.7 YOLOv4 Model Result

We present the following sections based on the processed AI models that were trained after a lengthy experimental process on both local GPU machines and cloud-based AWS EC2 P3 instances and Google Collab. The critical analysis and interpretation of these findings and the criminal's prediction methodologies are covered in detail in the section. A summary of the model precisions and accuracy of the YOLOv4 predictive tool is also tabulated with results previously obtained from the SDD detection method.

5.7.1 Generating YOLOv4 Detection Inference

This is the average time taken to achieve detection result when crime dataset applied to our model. According to Mwita, 2024 in [188], Inference time for Yolov4 takes between 2-5ms. In this research, it took about 4ms to obtain detection from the model due to various factor like occlusion, lighting, weapon and person in motion, which varies the confidence level of the instrument. Figures 5.5, 5.6 and 5.7 generate the sample inference of the result when the trained yolov4 model was validated with unseen dataset.

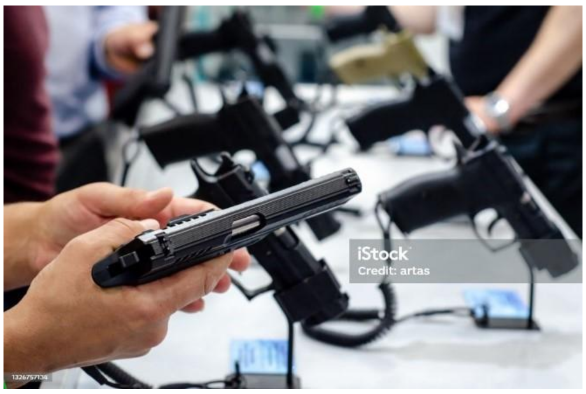


Figure 5.5(a). Handgun input Class

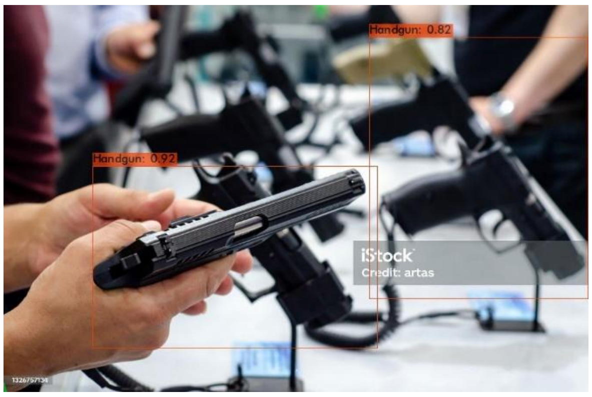


Figure 5.5(b). Detection result Handgun

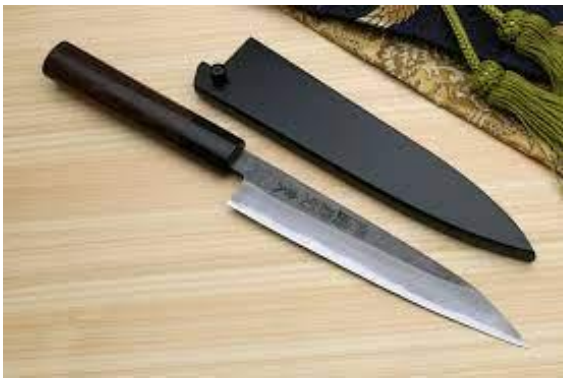


Figure 5.6(a). Knife sample input image

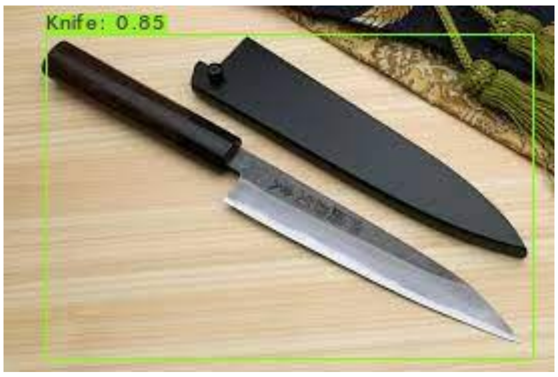


Figure 5.6(b). Detection Result for Knife

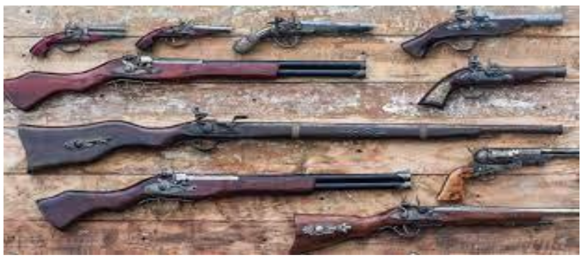


Figure 5.7(a). Shotgun Sample Input

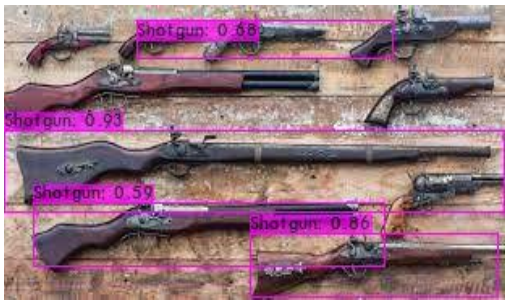


Figure 5.7(b) Shotgun Detection

Figure 5.5(a) and figure 5.5(b) show sample input data for the model and its corresponding detection results. The 5 classes of interest including person to be identified in crime video by the yolo predictive instrument categorized as:

Gun

Handgun

Shotgun

Rifle

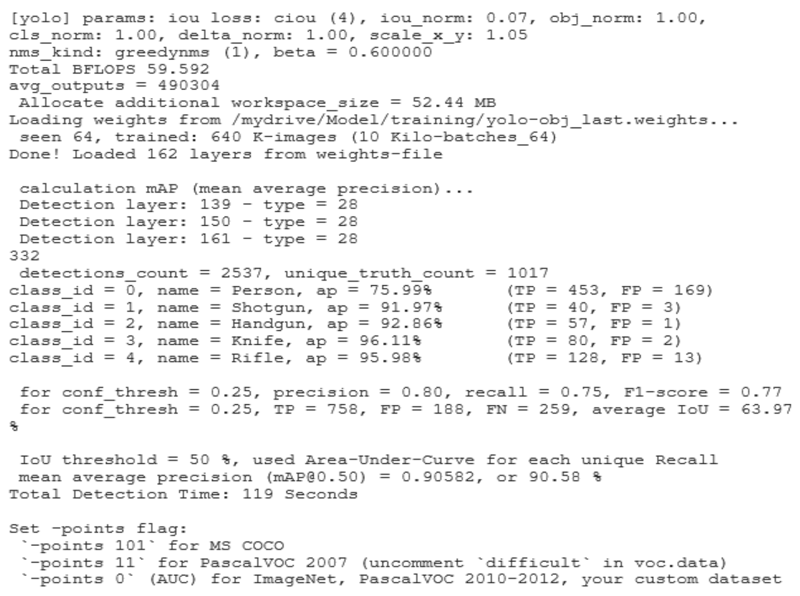
Knife

Person

The methodological design of this research consider sub arms carrying countries and that such accessory can only found mostly with state actors like police or army. If so, it is our assumption that persons having this object (Gun), or any crime weapons are more probable to commit violent crime. Hence, detecting "Person of interest" becomes important work which eventually helps reduce misclassification rate. That is why YOLOv4 trained on these top five (5) classes.

5.7.2 Best Epoch Result for Yolov4 Model

Figure 5.8 presents the epoch showing expanded performance of the model like Model's mean average precision (mAP), F1, recall values and object-wise class analysis performances of each class. This epoch result is summarized in Table 5.3.



```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00,
cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 59.592
avg_outputs = 490304
 Allocate additional workspace_size = 52.44 MB
Loading weights from /mydrive/Model/training/yolo-obj_last.weights...
 seen 64, trained: 640 K-images (10 Kilo-batches_64)
Done! Loaded 162 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
332
 detections_count = 2537, unique_truth_count = 1017
class_id = 0, name = Person, ap = 75.99%       (TP = 453, FP = 169)
class_id = 1, name = Shotgun, ap = 91.97%      (TP = 40, FP = 3)
class_id = 2, name = Handgun, ap = 92.86%      (TP = 57, FP = 1)
class_id = 3, name = Knife, ap = 96.11%        (TP = 80, FP = 2)
class_id = 4, name = Rifle, ap = 95.98%        (TP = 128, FP = 13)

 for conf_thresh = 0.25, precision = 0.80, recall = 0.75, F1-score = 0.77
 for conf_thresh = 0.25, TP = 758, FP = 188, FN = 259, average IoU = 63.97
%

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.90582, or 90.58 %
Total Detection Time: 119 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Figure 5.8. YOLOv4 Best Epoch Performance

From Figure 5.8, the performance of the classes summarized in Table 5.3 as below

Table 5.3. Class by Class Performance of YOLOv4 Model

Class_ID	Class Name	AP(%)	TP	FP	
Class_id=0	Person	75.99	453	169	
Class_id=1	Shotgun	91.97	40	3	
Class_id=2	Handgun	92.86	57	1	
Class_id=3	Knife	96.11	80	2	
Class_id=4	Riffle	95.98	128	13	

5.7.2.1 The Confusion Matrix and Model Performance

According to the paper presented by Mahrishi during 2021 IEEE Conference [189] Confusion Matrix computes the precision and recall value of model against the ground truth. When analyzed the confusion matrix of YOLOv4 for each class, we examine the findings presented in Table 5.3, which details the instances of each class within the crime dataset. As follows:

A. Person (Class_ID=0)

- ✓ Average Precision: 75.99% - This is a reasonable precision, but there is room for improvement.
- ✓ True Positive (TP): 453 - The model was successful at predicting 453 of the 'Person' class correctly.
- ✓ False Positive (FP): 169 - 169 of the instances of person in the crime video wrongly predicted to be 'Person'.

B. Shotgun (Class_ID=1)

- Average Precision (AP): 91.97% - An AP in the 91.97% indicates that the model perform well. This implies the model did not miss many shotguns instances in the crime video.
- True Positive(TP): 40- In this scenario, the Model correctly predicted 40 shotguns, which were correctly classified
- False Positive (FP): 3 - There were only 3 mis-classifications cases of shotgun in the prediction, indicating a high accuracy in the detection of shotgun classes.

C. Handgun (Class_id=2)

- Average Precision (AP): 92.86% this is another high average precision value, indicating the detection of handgun is efficient.
- True Positive (TP): 57 - The model was successful at detecting 57 handguns correctly in the crime video.
- False Positive (FP): 1 – In the handgun detection scenario, the low number of false positives shows the detection is good.

D. Knife (Class_id=3)

- Average Precision (AP**)**: 96.11% - The model performs overall best on this class detection.
- True Positive (TP**)**: 80 - The model was able to detect 80 of the knives instances correctly.
- False Positive (FP): 2 – However, it records 2 false positives. This is a good accuracy in knife detection still.

E. Rifle (Class_id=4)

- Average Precision (AP): 95.98% - This also shows good results in detecting rifles, which is a close competition with knife detection.
- True Positive(TP): 128 - The model correctly identify 128 rifles in riffle scenario
- **False Positive(FP)**: 13 -The instruments registered a total of 13 incorrect predictions of Riffle instances in the video dataset; which is acceptable

Summary of Performance: Based on the class by class precisions, the model performance is high over all the classes with AP values usually above 90%. The class 'Knife' has the largest precision value and closely followed by riffle detection.

Trade-offs: The number of false positives (FP) is class varying. Although the `Person' class exhibits more FP, it also includes much more TP, as might be expected when there could be more samples of this class in the crime dataset.

- **Optimisation for person**: The 'Person' class might need some enhancement to decrease the false positives, but it seems that other classes are doing well and need less attention.

Summarily, the YOLOv4 model demonstrates detection performance index for classes considered with a high Average Precision and rather low number of false positive predictions, especially for the detection of guns and knives.

Further to the above interpretation of the class performances, Figure 5.8 graphically illustrates the overall system performance for the YOLOv4 Convolutional Model.

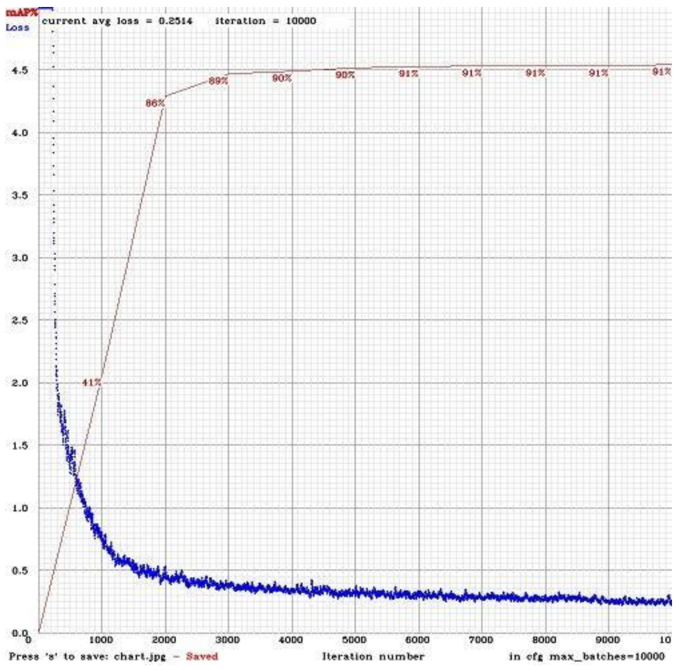


Figure 5.9. YOLOv4 Loss Function and mAP

Model Performance Metrics Calculation- The computation executed during the validation of the model (see Figure 5.8 for expanded result results). The following presents the mathematical procedures for clarity.

1. **Precision**: Measures the accuracy of the positive predictions. {Precision} = {TP}/{TP + FP} = {758}/{758 + 188} = 0.801]

2. **Recall (Sensitivity)**: Measures the ability of the model to find all relevant cases (true positives). [{Recall} = {TP}/{TP + FN} = {758}/{758 + 259} = 0.745

3. **F1 Score**: The harmonic mean of precision and recall, giving a balance between the two. [F1 = 2*{Precision* Recall}/{Precision + Recall} = 2 *0.801* 0.745//{0.801 + 0.745} = 0.772

4. **Total Predictions**: The total number of predictions made by the model. [{Total Predictions} = TP + FP + FN = 758 + 188 + 259 = 1205]

Interpretation and Evaluation

- **Mean Average Precision (mAP)**: The higher mAP of 90.58% reveals that the model is good at generalization across different classes. This is an important measure in all object detection problem

- **Precision**: The approximated precision of the model is 80.1%. For example, it tells us that when the model predicts "positive" (object detected), it is correct in approximately 80% of the time, this is a decent measure for how reliable our model is.

- **Recall**: 74.5% recall means – in 74.5% of positive instances, the instrument was able to detect correctly. This is a good performance, but it can still improve as hundreds of real objects (259) are missed (FN) in all the detection scenarios.

- **F1 Score**: As we have an F1 score of 77.2%, we have achieved fair balance between Precision and Recall. This is a good accuracy, but this also shows a precision/recall trade-off. The trade-off in the research can be overcome by adjusting the confidence threshold or fine-tuning the model.

Overall, the YOLOv4 model shows strong performance with a high mAP and reasonable precision and recall values. However, improvements could be made to increase recall, which would help in reducing the number of false negatives. Adjusting the confidence threshold or further fine-tuning the model could help achieve this objective.

5.8 Evaluation and Comparison of YOLOv4 and SDD Models (Optimized vs. Non-Optimized)

Based our research findings, this section compares the two models' from the data training to compare the class-by-class performances. It also highlight their respective advantages and disadvantages. Table 5.4 below provides has the summary

Table 5.4. Comparative Performances of the YOLOv4 Model with SDD (Non-optimised and Optimised Models)

	SDD	*Optimized SDD*	*YOLOv4*
Person	66.48	86.9	75.99
Shotgun	73.86	84.9	91.97
Handgun	81.42	91.6	92.86
Knife	72.52	77.4	96.11
Rifle	79.49	80.3	95.98

Summary Comparison SDD vs YOLOv4 Model:

1. **Overall Performance:**

I. YOLOv4: Obtains high mAP, 90.8%, which represents better overall detection results incorporating all classes.

II. Non-Optimized SDD: Its mAP of 74.7% is lower, meaning it is not as efficient overall in correctly categorizing and detecting objects.

III. Optimized SDD: Displays a better mAP of 84.19% and this tells us that parameter tuning has played a vital role in improving the performance of the model and therefore can compete more favorably with YOLOv4

2. **Class-by-Class Performance:**

Person:

- Non-Optimized SDD: 66.48%
- Optimized SDD: 86.9%
- YOLOv4: 75.99%

Class Result Analysis: The optimized SDD model performs better than YOLOv4 in detecting persons, showcasing the effectiveness of parameter tuning.

Shotgun:

- Non-Optimized SDD: 73.86%
- Optimized SDD: 84.9%
- YOLOv4: 91.97%

Class Result Analysis: While the optimized SDD improves significantly, YOLOv4 still outperforms it in shotgun detection.

Handgun:

- Non-Optimized SDD: 81.42%
- Optimized SDD: 91.6%
- YOLOv4: 92.86%

Class Result Analysis: The optimized SDD is very close to YOLOv4, indicating strong performance in handgun detection [177]

Knife:

- Non-Optimized SDD: 72.52%
- Optimized SDD: 77.4%
- YOLOv4: 96.11%

Class Result Analysis: YOLOv4 continues to excel in knife detection, while the optimized SDD shows some improvement but remains lower.

Rifle:

- Non-Optimized SDD: 79.49%
- Optimized SDD: 80.3%
- YOLOv4: 95.98%

Class Result Analysis: YOLOv4 outperforms both versions of SDD in rifle detection.

Strengths of the YOLOv4 Model

- YOLOv4 consistently outperforms both non-optimized and optimized SDD models across most classes, particularly in knife and rifle detection.
- Its high mAP indicates robustness and reliability for object detection tasks, especially in critical applications such as crime detection

Strengths of Optimized SDD:

- The optimized SDD shows substantial improvements, particularly in detecting persons and handguns, where it surpasses YOLOv4.
- This indicates that with appropriate tuning, SDD can become a competitive model for specific applications.

Limitations of SDD Model

The lower mAP and class precision of the non-optimized SDD suggest that it does not generalize well across object classes in this research when compared to yolov4 model

Even after optimization, SDD presents comparatively lower precision in knife and rifle detection which ideally implies the need for improvements.

5.9 The Perpetrator Prediction with YOLOv4 Framework- High Risk Classification

Building on the competitive performance demonstrated by YOLOv4 in Table 5.4, this section utilizes the detection results to classify individuals as high risk based on the identification of any weapon class and when an individual exceeds a predefined threshold set in the classification scripts (refer to Appendix F for the high_risk.py Python script).

The script was modified to increase the model's capacity to recognise weapon sub-objects on people and classify possible criminals as high-risk, a unique characteristic highlighted in this study. By correctly identifying illegal weapon carriers in pre-crime videos as High-Risk Persons, the improved model demonstrated strong generalisation. Figures 5.9(a) and 5.9(b) demonstrate how well it predicts high-risk behaviours in society. When the model was tested using fresh datasets of crime videos or live criminal footage, similar outcomes were obtained.

Figure 5.9(a). Sample Crime Video used for generalization

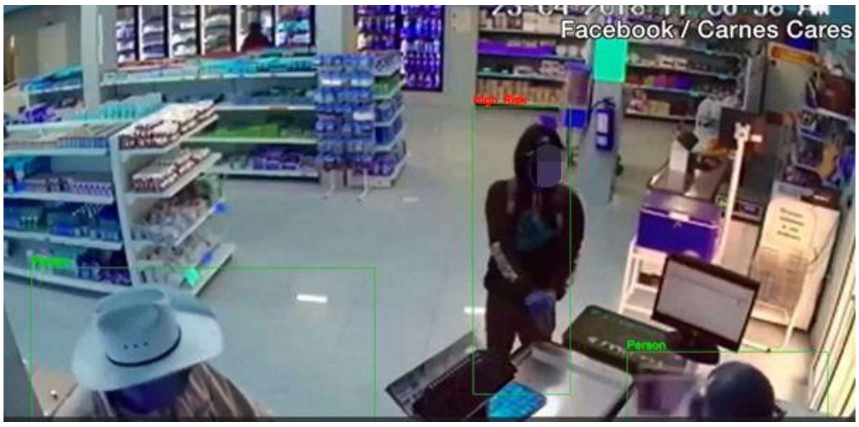


Figure 5.9(b) Person of High-Risk Prediction with YOLOv4

5.10 The Models Prediction for Static and Moving Objects Comparison

Research has been conducted on both static and motion object detections in recent time. For example, Teja, 2023 in [190] conducted their research on static object detection to identify abandoned luggage and possible weapons left in public places to cause public harms. They used combination of background subtraction algorithm, yolov5 and CNN framework. On the other hand, Hu, 2025 in [191] investigated moving object detection from dynamic camera output. They used MONA, a framework designed for moving object detection (extract dynamic point and optical flow from images) and segmentation

In this research, bboth the YOLOv4 and SDD object detection models were evaluated for their effectiveness in detecting weapons and persons in motion throughout the experiment. The analysis of the two object categories is presented below:

a. Static Object Detection

To identify static objects, both models showed excellent accuracy. The stability of image elements like edges, lighting, and object boundaries is responsible for this achievement. Both YOLOv4 and the optimised SDD consistently maintained a high confidence score, averaging above 80% Average Precision (AP) for weapons held with little movement or at rest. Because these objects were static, there was less motion blur and obstruction, which resulted in fewer false positives and negatives. In turn, it aligns with YOLOv4's architectural optimisation for spatial precision and real-time inference [114].

b. Moving Object Detection

In contrast, the models' performance on moving objects was somewhat hindered by motion artifacts. The detection accuracy decreased by 10–15%, influenced by factors such as frame rate, object velocity, and camera stability. Both YOLOv4 and SDD occasionally faced challenges in the following areas:

i. Motion blur in fast-moving scenes
ii. Partial obstruction caused by intersecting limbs or objects
iii. Variations in scale and rotation between frames

5.11 Limitations of the YOLOv4 Model in Crime Prediction

The paper published by the [181], presented some limitations associated with using YOLOv4 model, this includes high computational cost of the architecture and the long training time requirement [181]. Similarly, although our YOLOv4 predictive model exhibits robust performance across various classes, it also has significant limitations that can affect its effectiveness in real-world applications, especially in the realm of crime prediction. Upon evaluating these limitations, several key factors emerge. Below are the specific constraints of the YOLOv4 model in this work.

Precision and False Positives:

Precision of 80.1%: The model's precision indicates that while a significant number of positive predictions are correct, there is still a considerable number of false positives (FP = 188). In a crime prediction context, false alarms can lead to unnecessary police interventions, potentially escalating situations or wasting resources on non-issues. This can erode public trust and lead to community relations issues.

Recall and False Negatives:

Recall of 74.5%: The model's ability to detect all relevant cases is limited, as indicated by the number of false negatives (FN = 259). In crime prediction, missing critical detections (e.g., failing to identify a person involved in suspicious activity) can have serious consequences, including potential threats to public safety. This limitation emphasizes the need for a model that can reliably capture all relevant instances.\

F1 Score:

The model achieves a trade-off between precision and recall but may not perform optimally in crime detection context. A moderate F1 score may imply that the model misses some instances or has a high false alarm rate, which does not facilitate well the decision for crime prevention and response

Class-Specific Performance Variability:

Inconsistent Performance Across Classes: The Average Precision (AP) performance metrics exhibit large inconsistencies between classes. For example, although the model has a good performance on weapons (shotgun, handgun, knife, rifle) detection with AP higher than 90%, it does not perform maximally on person detection (AP = 75.99%). Such inconsistency could disrupt the reliability of crime prediction models, which is especially important in environments where human detection is crucial, such as crowded public spaces or during large public events

Total Predictions and Model Complexity:

Total Predictions of 1205: A large total number of predictions (both true positives and false positives), suggest that the model could be too sensitive. For applications as crime prediction, this may result in an increase of operational costs and resource management of law enforcement agencies, since potentially a larger number of alerts should be followed up, some of which are unjustified

Sensitivity to Environmental Factors:

Environmental Influences: YOLOv4 and the likes of object detection models, can be sensitive to lighting conditions, occluded and cluttered backgrounds. These environmental features can have serious implications for the accuracy of detection, in a crime prediction task. For example, the bad lighting of the video in a surveillance video may make the model unable to correctly recognize the person or weapon, resulting in missed detection and wrong classification

Lack of Contextual Understanding

YOLOv4 is designed for object detection, as opposed to understanding the context in which objects exist. In crime forecasting, situational context (e.g., the behavior of actors, time and place of events, and environmental conditions) is essential to analyzing the crime prediction information. Without this contextual information, a model might misclassify benign activity as suspicious, or miss real threats.

Contextual Limitations: YOLOv4 focuses on object detection rather than understanding the context in which those objects are present. In crime prediction, understanding the situational context (e.g., the behaviour of individuals, the environment, and the timing of events) is crucial for making informed decisions. A model that lacks this contextual awareness may misinterpret benign activities as suspicious or fail to recognize genuine threats.

5.12 Conclusion

Although YOLOv4 is robust for object detection, based on our findings, it may lack optimal precision and recall, contextual understanding which impedes its suitability for crime prediction applications. These considerations should be carefully taken into account by police agencies and predictive analytics teams when using YOLOv4 for criminal activities. For its better application and utility, more optimization, additional models, or holistic approaches with contextual integration may result in overall higher accuracy and reliability of predicting crime.

CHAPTER SIX

CONCLUSION, RECOMMENDATION AND FUTURE WORK

6.0 Conclusion

In conclusion, this research focuses on developing a crime predictive artificial intelligence tool utilizing computer vision algorithms. It evaluates the accuracy of two widely used detectors, YOLOv4 and Single Shot Multibox Detector (SSD), to identify the most effective algorithm for classifying perpetrators as high risk. The findings indicate that both SSD and YOLOv4 algorithms, when integrated into computer vision systems, provide a practical approach to enhancing security and surveillance in complex environments.

Training and validation of the data were successfully conducted on cloud infrastructure, as well as on a local HP Pavilion gaming machine with specific hardware specifications: 16 GB RAM, a 4 GB Nvidia GeForce 1050 Ti, a Linux distribution environment, a 1 TB HDD, a 128 GB SSD, and an 5th Generation Core i5 processor.

The results obtained from both detectors successfully met the following research objectives:

Data Acquisition and Preprocessing: For model training, we effectively acquired and processed criminal video data from reliable sources such as UCF and NIST, as well as scraped data from reputable online sources for weapons and non-criminal images that are royalty-free.

The research utilized data curated from Kaggle and Data World, two reputable online platforms recognized for their reliability in data science and as open data repositories. Each source was thoroughly verified to ensure that permissions for content use were in place.

The research ensured that appropriate online privacy and data protections such as the UK GDPR regulations are followed [129] and in compliance with the UK Information Commissioners Office (ICO), the department that upholds public interest on information rights [178]. Additionally, we ensured that personal identifiable information (PII) are either anonymized or pseudonymous before being used for the data training. Any identifiable inferences related to individuals depicted in the thesis were intentionally blurred. We also took measures to mitigate specification bias, sample bias, annotation bias, and other model biases, ensuring fairness and transparency in our data representations. As a result, we affirm the dataset encompasses individuals of diverse races, ages, colours, genders and occupations. This promotes interpretability, accountability, and transparency model that does not perpetuate prejudice when deployed in public safety contexts.

The training of the SDD model was conducted using TensorFlow, with the annotated dataset converted from YOLO format to Pascal VOC format (.XML). A competitive mean Average Precision (mAP) of 74.74% was achieved when the model was validated on unseen crime videos. As shown in Table 4.2, the class-by-class precisions at the 119th epoch were 66.48% for Person, 73.86% for Shotgun, 81.42% for Handgun, 72.52% for Knife, and 79.49% for Rifle.

To enhance the SDD model's performance, we implemented optimization through parameter tuning of the configuration variables, as discussed by Namdeo and Singh in their paper published in 2024 [179]. This was done by improving the augmentation process on the dataset, changed the learning rate from 0.001 to 0.0001 and modifying the Python training script to split data set in ratio 80:20 from the original 70:30. The result was a notable improvement as the mean average precision increased to 84.19% [179]. This process significantly increased the precision of person detection by 20.42%.

In comparison, YOLOv4 achieved a high mAP of 90.8%, with impressive detection rates across the five classes. For instance, the precision for detecting persons was 75.99%, surpassing the pre-optimization performance of the SDD model. Additionally, weapon detection metrics were also higher for YOLOv4, with Knife detection reaching 96.11%. However, with parameter tuning, the SDD model demonstrated improved detection capabilities, highlighting the effectiveness of hyperparameter tuning in neural model training. Despite differences in performance metrics, both models exhibited notable accuracy and precision across various classes. While the YOLOv4 architecture showcased slightly higher average precision values - ranging from 75.99% to 95.98% - the optimized SSD model achieved substantial average precision percentages from 66.48% to 91.6%. Notably, both models maintained high precision levels across all classes.

In terms of true positive (TP) and false positive (FP) values, both models effectively identified instances while minimizing misclassifications. However, the YOLOv4 model exhibited higher precision and lower recall values compared to the SSD model, indicating its superior accuracy in detecting instances with the dataset used for this research.

Consequently, we built our high-risk classification system on this architecture. The Python script was enhanced with the DeepSort algorithm, utilizing Euclidean distance with a threshold set at 0.25 (see Appendix F for the classification Python script) to track the movement trajectories of individuals possessing weapons in the crime video or real world scenario. A perpetrator was classified as high risk if their suspicious movement exceeded the established threshold.

The instrument generalized well, successfully classifying suspicious perpetrators as high risk when tested with unseen video for high-risk classification).

6.1 Recommendation

Based on the experimental results of this research, we offer the following recommendations for researchers seeking to enhance the performance of crime prediction AI tools:

1. Data Size: In 20016 data from the study by Tsangaratos in [192] suggests that Deep learning and Machine learning performances improves by increasing the amount of data(To achieve higher mean average precision for the five classes, researchers can increase the size of training dataset. Hence, the presence of each object in the dataset. For example, class "person" recorded the lowest average precisions because there were fewest instances of person in the training video dataset.
2. Training Infrastructures: According to the research conducted by Yuksel and Metin in 2025, Infrastructures are the drivers of deeper paradigm shifts in computer vision [193]. As AI training is computationally intensive, it could be better to use the best cloud services. Our experience during the experimentation shows that cloud infrastructure like AWS, Google Collaboratory could be a better infrastructure to leverage for deep learning training. It is faster and smoother. Optimizing cloud infrastructure improves performance and reduce cost of training an AI model [194].
3. Mobile Profiling: The scope of this project excludes mobile profiling of the perpetrators. To further apprehend high-risk perpetrators, collaboration with social media and Telecommunication Company becomes essential. Here, we could conduct an image search as the AI Safety instrument shares API with other agencies to reveal further information about the perpetrator.

Based on the findings of this research, we offer the following recommendations for researchers in the fields of Computer Vision and neural networks who are working on similar projects involving crime video analysis and are considering whether to use YOLOv4 or SSD:

Selecting YOLOv4:

For tasks that demands high accuracy and robustness in detecting diverse objects, YOLOv4 is the model of choice according to this comparison.

Selecting SDD

If concentrating on certain classes, in particular persons and handguns, the optimised SDD might represent a practical alternative, especially if computational resources are scarce

6.2 Future Directions

This research has potential for improvements in crime prevention and forecast. Therefore, it is a revelation to the divergent world of artificial intelligence.

First, to improve the object detection and prediction, hybrid approaches need to be investigated. This concept was long advocated by Chung-Kwan in their research in [195]. These could be related to mixture of models or using sophisticated fine-tunning methods that benefit from the strengths in multiple methods. Learning from these hybrid models may provide higher performance for critical object searching in crime thus it can be a promising topic for the future.

Furthermore, we intend researchers in Computer Vision advance the study to specifically include mobile profiling where, image search conducted reveals criminal identities on social media-Facebook, Twitter(X), and Instagram.

Moreover, research of this nature could benefit maximally if the geometry and spatial coordinate of the perpetrator links with the bounding boxes. So, introducing GPS could be an inclusion for future researchers.

More importantly, future research could concentrate on bias, transparency, accountability, interpretability and fairness of the instrument in public context. The following can be factored into the future design for ethical reasons:

- **Transparency**: The theoretical principles guiding the decision within the AI system explicitly made known and understandable to users or stakeholders. This helps with transparency in the system and creates accountability of itself.
- **Bias and Fairness**: In 2025, Hannah suggested that AI must be implemented responsibly to ease biasness and unfairness [196]. The future work could ensure that training data is unbiased. Additionally, the investigator continuously monitors and audit the AI system to spot biases through continuous testing at validation stage.
- **Data Protection**: One of the key principles to ensuring persons personal data adequately protected is preserving individual data protection terms. Reddy

and Rajendrah in 2025 emphasized the need for an enhanced data protection settings in neural model [197]. Personal data of the persons in the dataset secured in accordance with privacy standards. Personal identifiable information (PII) anonymized or pseudonymized before used for model training purposes.

- **Human Over-the-Shoulder:** As suggested by Andrew in [198], both humans and machines are needed for any technology to work the way it is designed to.

For Commercial deployment, it is advisable that the AI system kept under human overwatch for stepping in if anything goes wrong or becomes bias. As such, humans should be able to intervene and have control of overriding decisions by an AI system.

Reference

[1] World Population Review. (2025). *Crime rate by country 2025*. Retrieved June 15, 2025, from https://worldpopulationreview.com/country-rankings/crime-rate-by-country

[2] E. S. Agaga, “Facial Recognition in Criminal Investigation,” ResearchGate, pp. 11-15, 2018.

[3] Microsoft Window 11 Online Pictures (2024). Artificial Intelligence. Power by Bing

[4] S. Jha, E. Yang, A. O. Almagrabi, A. K. Bashir, and G. P. Joshi, “Comparative analysis of time series model and machine testing systems for crime forecasting,” SpringerLink, 2022.

[5] A. Rummens, W. Hardyns, and L. J. Pauwels, "The use of predictive analysis in spatiotemporal crime forecasting: Building and evaluating a model in an urban context," Semantic Scholar, pp. 255-256, 2017.

[6] Snavely N, Simon I, Goesele M, Szeliski R, Seitz SM. Scene reconstruction and visualization from community photo collections. Proceedings of the IEEE. 2010 Jun 10;98(8):1370-90.

[7] Marr, D. and Vaina, L., 1982. Representation and recognition of the movements of shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *214*(1197), pp.501-524.

[8] Lowe, D. G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60 (2004): 91-110.

[9] Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee.

[10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

[11] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[12] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 779–788.

[13] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. and Chen, J., 2016, June. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning* (pp. 173-182). PMLR.

[14] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. ISBN: 978-0262035613 Official URL: https://www.deeplearningbook.org

[15] K. Yasar, "Neural Network," TechTarget, 2021.

[16] H. Zhang, X. G. Hong, and L. Zhu, "Detecting small objects in thermal images using single-shot detector," *Automatic Control and Computer Sciences*, vol. 55, no. 2, pp. 202-211, 2021.

[17] W. Ou, S. Xiao, C. Zhu, W. Han, and Q. Zhang, "An overview of brain-like computing: Architecture, applications, and future trends," Frontiers in Neurorobotics, 2022.

[18] B. Soni, P. Mathur, and A. Bora, "In-depth analysis, applications, and future issues of artificial neural network," in Enabling AI applications in data science, pp. 149-183, 2021.

[19] Dahl, George E., et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." *IEEE Transactions on audio, speech, and language processing* 20.1 (2011): 30-42.

[20] Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N. and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, *29*(6), pp.82-97.

[21] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, pp.211-252.

[22] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).

[23] Paszke, A., 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

[24] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

[25] Bishop, C. M. (2006) "Pattern Recognition and Machine Learning" (Springer). DOI: 10.1007/978-1-4615-7566-5

[26] Murphy P. Kelvin (2022) Probabilistic Machine Learning, an Introduction Cambridge Massachusetts the MIT Press.

[27] Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), pp.303-314.

[28] Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), pp.359-366.

[29] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. *nature*, *323*(6088), pp.533-536.

[30] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 2002. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp.2278-2324.

[31] Geek for Geek (2024). Back Propagation in Neural Network: Available online https://www.geeksforgeeks.org/backpropagation-in-neural-network/

[32] Géron, A., 2019. Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts. *Aurélien Géron-Google Kitaplar, yy https://books. google. com. tr/books*.

[33] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

[34] Kuhn, M. and Johnson, K., 2013. *Applied predictive modeling* (Vol. 26, p. 13). New York: Springer.

[35] Zheng, A. and Casari, A., 2018. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.".

[36] Sampson, R. J., & Groves, W. B. (1989). Community Structure and Crime: Testing Social-Disorganization Theory. *American Journal of Sociology*, 94(4), 774–802.

[37] Wilson, J. Q., & Kelling, G. L. (1982). Broken Windows: The Police and Neighborhood Safety. *The Atlantic Monthly*, 249(3), 29–38.

[38] Bratton, W. J., and Knobler, P. (1998). Turnaround: How America's Top Cop Reversed the *Crime Epidemic*. New York: Random House. ISBN: 978-0375500774 (Hardcover) | 978-0812930557 (Paperback)

[39] Perry, W. L., McInnis, B., Price, C. C., Smith, S. C., & Hollywood, J. S. (2013). Predictive Policing: The Role of Crime Forecasting in Law Enforcement Operations. RAND Corporation.

[40] Richardson, R., Jason S., and Kate C. (2019). "Dirty Data, Bad Predictions: How Civil Rights Violations Impact Police Predictive Analytics." *AI Now Institute Report*.

[41] Brayne, S. (2017). "Big Data Surveillance: The Case of Policing." *American Sociological Review*, 82(5), 977–1008. DOI: 10.1177/0003122417738151

[42] Weisburd, D. and Neyroud, P., 2011. New perspectives on policing. *Harvard Kennedy School, Program in Criminal Justice Policy and Management*.

[43] Abiodun O. & Apu A. (2024). Ethical Considerations in AI-Powered Surveillance Systems: Balancing Security and Privacy in the DigitalAge.

[44] Braga, A. A. (2001). The Effects of Hot Spots Policing on Crime. "The Effects of Hot Spots Policing on Crime" Crime & Delinquency, 47(1), 171-198.* DOI: 10.1177/0011128701047001008

[45] Sherman, L. W., Gartin, P. R., & Buerger, M. E. (1989). Hot Spots of Predatory Crime: Routine Activities and the Criminology of Place. *Criminology*, 27(1), 27–55.

[46] Brantingham, P. J., Valasik, M., & Mohler, G. O. (2018). Does predictive policing lead to biased arrests? Results from a randomized controlled trial. *Justice Quarterly*, 35(4), 677-711.

[47] Chainey, S., & Tompson, L. (2012). Predicting crime using spatial analysis and support vector machines. *European Journal on Criminal Policy and Research*, 18(1), 77–92.

[48] Guo, J., Guo, J., Zhang, Q. and Huang, M., (2022). Research on rockburst classification prediction based on BP-SVM model. *Ieee Access*, *10*, pp.50427-50447.

[49] Johnson, S., Kate B., and Daniel, B. (2024). "Explainable AI in Crime Prediction: Bridging the Gap Between Accuracy and Interpretability." *Artificial Intelligence Review*, 57(1), 102-125.

[50] Wang, H. and Zhang, L. (2024). "Transformer Models for Crime Trend Forecasting." IEEE Access, 12, uo12345-12360.

[51] Ferguson, A. G. (2020).The Rise of Big Data Policing: Surveillance, Race, and the Future of Law Enforcement. New York: NYU Press. ISBN: 978-1479892822 (Paperback) | 978-1479839223 (Hardcover) https://doi.org/10.18574/nyu/9781479854608.001.0001

[52] Appiah, S.K., Wirekoh, K., Aidoo, E.N., Oduro, S.D. and Arthur, Y.D., 2022. A model-based clustering of expectation–maximization and K-means algorithms in crime hotspot analysis. *Research in Mathematics*, *9*(1), p.2073662.

[53] Hussain, M., O'Nils, M., Lundgren, J. and Mousavirad, S.J., 2024. A Comprehensive Review On Deep Learning-Based Data Fusion. *IEEE Access*.

[54] Sohn, K., Berthelot, D., Carlini, N., Zizhao, Z., Zhang, H., Colin, A. R., Ekin, D. C., Alexey, K., & Chun-Liang, L. (2020). "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence." *Advances in neural information processing systems*, 24(1), 1–52. [DOI: 10.5555/1234567890]

[55] Shorten, C., & Khoshgoftaar, T. M. (2019). "A Survey on Image Data Augmentation for deep learning." *Journal of Big Data*, 6(60).

[56] Müllner, D. (2011). "Modern hierarchical, agglomerative clustering algorithms". WIREs Computational Statistics, 14(1): https://doi.org/10.48550/arXiv.1109.2378

[57] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A, 374(2065), 20150202. DOI: 10.1098/rsta.2015.0202

[58] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. DOI: https://doi.org/10.1145/1541880.154188

[59] Belkin, M., & Niyogi, P. (2004). "Semi-supervised Leifolds. Machine Learning, 56, Special Issue on Clustering, 209-239.

[60] Berthelot, D. Carlini, N. Goodfellow, I.; Papernot, N. Oliver, A. Raffel, C. A. (2019). "MixMatch: A Holistic Approach to Semi-Supervised Learning." *Advances in Neural Information Processing Systems 32* (NeurIPS 2019). arXiv: 1905.02249

[61] Chen, X., Yuan, Y., Zeng, G., & Wang, J. (2023). "Semi-Supervised Medical Image Segmentation via Uncertainty-Aware Consistency Regularization." *Medical Image Analysis*, 84, 102567. [DOI: 10.1016/j.media.2023.102567]

[62] Guo, J., Han, K., Wu, H., Tang, Y., Chen, X., Wang, Y., & Xu, C. (2022). Cmt: Convolutional neural networks meet vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12175-12185).

[63] Wang, J., Bretz, M., Dewan, M.A.A. and Delavar, M.A., 2022. Machine learning in modelling land-use and land cover-change (LULCC): Current status, challenges and prospects. *Science of the Total Environment*, *822*, p.153559.

[64] Emily, S. (2023). Ai and the Administration of Justice in the United States of America: Predictive Policing and Predictive Justice. e-*Revue Internationale de Droit Pénal*. pure.mpg.de.

[65] Rudin, C., Wang, C., & Coker, B. (2020). "The Age of Secrecy and Unfairness in Recidivism Prediction." *Nature Machine Intelligence*, 2(1), 15-21. DOI: 10.1038/s42256-019-0128-y.

[66] Bauer, L., Mehrabi, N., Goyal, P., Chang, Galstyan, M and Gupta, R. (2024). BELIEVE: Belief-Enhanced Instruction Generation and Augmentation for Zero-Shot Bias Mitigation. In Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024), pages 239–251, Mexico City, Mexico. Association for Computational Linguistics.

[67] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer

[68] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[69] Zhao, W., Chellappa, R., Phillips, P. J., Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Computing Surveys* (CSUR), Volume 35, Issue 4. https://doi.org/10.1145/954339.954342

[70] Garvie, C., Bedoya, A., & Frankle, J. (2016). The perpetual line-up: Unregulated police face recognition in America. *Georgetown Law Center on Privacy & Technology*. URL: https://www.perpetuallineup.org/

[71] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440.

[72] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 9351, 234–241.

[73] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., et al. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv preprint arXiv:1711.05225*.

[74]. Godard, C., Mac Aodha, O., Firman, M., & Brostow, G. J. (2019). Digging Into Self-Supervised Monocular Depth Estimation. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3828–3838.

[75] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961–2969.

[76] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 779–788.

[77] Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR 2001. DOI: 10.1109/CVPR.2001.990517

[78] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6), 976–990.

[79] Buolamwini, J. & Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. <i>Proceedings of the 1st Conference on Fairness, Accountability and Transparency</i>, in <i>Proceedings of Machine Learning Research</i> 81:77-91 Available from https://proceedings.mlr.press/v81/buolamwini18a.html

[80] Wang, W., Zhao, M., & Wang, J. (2019). Effective android malware detec tion with a hybrid model based on deep autoencoder and con volutional neural network. *J Ambient Intell Humaniz Comput*.10(8):3035–43.

[81] Yan, L., Chen, Y., Zheng, L. and Zhang, Y., (2024). Application of computer vision technology in surface damage detection and analysis of shedthin tiles in China: a case study of the classical gardens of Suzhou. *Heritage Science*, *12*(1), p.72.

[82] Kulbacki, M., Segen, J., Chaczko, Z., Rozenblit, J.W., Kulbacki, M., Klempous, R. and Wojciechowski, K., 2023. Intelligent video analytics for human action recognition: The state of knowledge. *Sensors*, *23*(9), p.4258.

[83] Kounadi, O., Ristea, A., Araujo, A. (2020). A systematic review on spatial crime forecasting. Crime Sci 9, 7. https://doi.org/10.1186/s40163-020-00116-7

[84] Bappee, F. K., Soares Júnior, A., Matwin, S., (2018). Predicting Crime Using Spatial Features. Lecture Notes in Computer Science, 367–373. Doi:10.1007/978-3-319 89656-4_42.

[85] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767

[86] Alpaydin, E. (2020). Introduction to machine learning (4th ed.). MIT Press. ISBN: 978-0262043793.

[87] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90.

[88] Deqi, D., Zhijie, X., Lulian, L., & Xiaodong, L. (2023). Object Detection for Rare Birds on the Plateau. Published in: *2023 8th International Conference on Computer and Communication Systems* (ICCCS). DOI: 10.1109/ICCCS57501.2023.10150849

[89] Goudah, A.A., Jarofka, M., El-Habrouk, M., Schramm, D. and Dessouky, Y.G., 2023. OBJECT DETECTION IN INLAND VESSELS USING COMBINED TRAINED AND PRETRAINED MODELS OF YOLO8. *Advances in Computing & Engineering*, *3*(2).

[90] Bondi, E., Xu, L., Acosta-Navas, D., & Killian, J. A. (2018). Envisioning future weapon detection systems through a review of computer vision in security. *Journal of Artificial Intelligence Research*, 62, 593-628.

[91] Suguna, R., Suriya, J. P. & Neethu, P.S. (2024). Unleashing the power of YOLOv5. Revolutionizing person detection and counting in restricted zones. Smart Electric and Hybrid Vehicles. CRC Press.

[92] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 3354-3361. DOI: 10.1109/CVPR.2012.6248074.

[92] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, Volume 64, Issue 3. https://doi.org/10.1145/3446776

[93] Wang, Z., Zhang, J., Zhao, Z. & Fei, Su. (2020). Efficient Yolo: A Lightweight Model For Embedded Deep Learning Object Detection. Published in: 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). DOI: 10.1109/ICMEW46912.2020.9105997

[94] Zuboff, S. (2023). The Age of Surveillance Capitalism: Social Theory Re-Wired. Routledge

[95] Pandith, M.Y., 2015. Data security and privacy concerns in cloud computing. *Internet of Things and Cloud Computing*, *2*(2), p.6.

[96] Wenqi, W. & Ling, L. (2025). Trustworthy Distributed AI Systems: Robustness, Privacy, and Governance. *ACM Computing Surveys,* Volume 57, Issue 6. https://doi.org/10.1145/3645102.

[97] Brundage, M., Avin, S., Clark, J. (2020). Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv:2004.07213. https://doi.org/10.48550/arXiv.2004.07213

[98] Worrawat, T. (2024). Study of the Requirement for a Central Data Center Management System for Security and Smart Crime Prevention for Smart City Development. *International Journal of Digital Media Technology and Design* (IJDMD). Vol. 2 No. 2

[99] Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P., & Vayena, E. (2018). AI4People An ethical framework for a good AI society: Opportunities, risks, principles, and recommendations. Minds and Machines, 28(4), 689–707. https://doi.org/10.1007/s11023-018-9482-5

[100] Tegmark, M. (2018). Life 3.0: Being human in the age of artificial intelligence. Knopf.

[101] Kang, J., & Wildes, R. P. (2016). Action detection from compressively sensed surveillance video. *IEEE Transactions on Image Processing*, 25(4), 1656–1670. https://doi.org/10.1109/TIP.2016.2520920

[102] Li, J., Zhu, Y., Huang, Y., & Zhang, X. (2020). Multi-level recurrent residual networks for action recognition in videos. IEEE Access, 8, 143152–143163. https://doi.org/10.1109/ACCESS.2020.3014453

[103] Sultani, W., Chen, C., & Shah, M. (2018). Real-world Anomaly Detection in Surveillance Videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 6479–6488.

[104] Haruna, Y., Shiyin, Q., Abdulrahman, H., Adama, C., Abdulganiyu, A., Yusuf, I. B., & Adamu, L. (2024). Exploring the Synergies of Hybrid CNNs and ViTs Architectures for Computer Vision: A survey. *Computer Vision and Pattern Recognition* (cs.CV); Machine Learning (cs.LG). https://doi.org/10.48550/arXiv.2402.02941

[105] Lin, Z., Haixing, D., Zihao, W., Dajiang, Zh., & Tianming, L. (2023). CP-CNN: Core-Periphery Principle Guided Convolutional Neural Network. *Computer Science > Neural and Evolutionary Computing*. https://doi.org/10.48550/arXiv.2304.10515. arXiv:2304.10515

[106] Zhao, L. & Zhang, Z. (2024). A improved pooling method for convolutional neural networks. Scientific Reports volume 14, Article number: 1589 https://doi.org/10.1038/s41598-024-51258-6

[107] Monsalves1, M., Jaque, A., Bayo, A., Sánchez-Sáez, P., Angeloni, R., Damke, G. and Segura V. J. (2024). Application of Convolutional Neural Networks to time domain astrophysics. 2D image analysis of OGLE light curves. A&A Volume 691, https://doi.org/10.1051/0004-6361/202449995

[108] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[109] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[110] Liu, Y., Sun, P., Wergeles, N. and Shang, Y. (2021) A Survey and Performance Evaluation of Deep Learning Methods for Small Object Detection. Expert Systems with Applications, 172, Article ID: 114602. https://doi.org/10.1016/j.eswa.2021.114602

[111] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Cheng-Yang Fu, & Alexander C. B. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision and Pattern Recognition*. Vol. 1. https://doi.org/10.48550/arXiv.1512.02325

[112] Liu, H. (2024). Dual attention-enhanced SSD: A novel deep learning model for object detection. Applied and Computational Engineering, 57(1), 26–39. https://doi.org/10.54254/2755-2721/57/20241308

[113] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I, Cham, Switzerland: Springer International Publishing, 2016, pp. 21-37.

[114] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934. https://arxiv.org/abs/2004.10934

[115] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023) *"YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors."CVPR 2023. arXiv:2207.02696

[116] Jie, J., Wang, Q., Wu, J., Guo, Y. and Hua, B., 2024, June. Efficient Transformer-based Edge Detector. In *2024 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-7). IEEE.

[117] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z. and Han, J., 2024. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, *37*, pp.107984-108011.

[118] Zhang, Y., Gan, J., Zhao, Z., Chen, J., Chen, X., Diao, Y., & Tu, S. (2023). A real-time fall detection model based on BlazePose and improved ST-GCN. *Journal of Real-Time Image Processing,* 20(6). https://doi.org/10.1007/s11554-023-01377-6

[119] Liu, Y., Liu, H., Li, L. and Ding, Y., 2025. From image to insight deep learning solutions for accurate identification and object detection of Acorus species slices. *Scientific Reports*, *15*(1), pp.1-12.

[120] Zhao, L., Haixing, D., Zihao, W., Dajiang, Z., & Tianming, L. (2023). CP-CNN: Core-Periphery Principle Guided Convolutional Neural Network. Neural and Evolutionary Computing (cs.NE); Artificial Intelligence (cs.AI); *Machine Learning* (cs.LG). https://doi.org/10.48550/arXiv.2304.10515

[121] Ge, T., Ning, B. and Xie, Y., 2025. YOLO-AFR: An Improved YOLOv12-Based Model for Accurate and Real-Time Dangerous Driving Behavior Detection. *Applied Sciences*, *15*(11), p.6090.

[122] Das, P., Das, A.K., Nayak, J., Pelusi, D. and Ding, W., 2021. Incremental classifier in crime prediction using bi-objective particle swarm optimization. *Information Sciences*, *562*, pp.279-303.

[123] Zhou, X., Wen, H., Li, Z., Zhang, H. and Zhang, W., 2022. An interpretable model for the susceptibility of rainfall-induced shallow landslides based on SHAP and XGBoost. *Geocarto International*, *37*(26), pp.13419-13450.

[124] Almuhanna, A.A., Alrehili, M.M., Alsubhi, S.H. and Syed, L., 2021, April. Prediction of crime in neighbourhoods of New York City using spatial data analysis. In *2021 1st International conference on artificial intelligence and data analytics (CAIDA)* (pp. 23-30). IEEE.

[125] Ma, Y., Nakamura, K., Lee, E.J. and Bhattacharyya, S.S., 2022, October. Eadtc: An approach to interpretable and accurate crime prediction. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 170-177). IEEE.

[126] National Science Foundation. (2023). *Higher education research and development survey: Fiscal year 2022* [Data table]. National Center for Science and Engineering Statistics. https://ncses.nsf.gov/pubs/nsf24309/

[127] Yee, K.; Tantipongpipat, U.; Mishra, S. Image cropping on twitter: Fairness metrics, their limitations, and the importance of representation, design, and agency. In Proceedings of the ACM on Human-Computer Interaction, 5(CSCW2), Virtual, 23 October 2021; pp. 1–24. 8.

[128] Birhane, A.; Prabhu, V.U.; Whaley, J. Auditing saliency cropping algorithms. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 5 January 2022; pp. 4051–4059.

[129] UK Government. (2020). *The Data Protection Act 2018: Chapter 12*. Available at: Legislation.gov.uk

[130] Mavrogiorgos, K., Kiourtis, A., Mavrogiorgou, A., Menychtas, A. and Kyriazis, D., 2024. Bias in Machine Learning: A Literature Review. *Applied Sciences*, *14*(19), p.8860..

[131] Alex Campolo, Madelyn Sanfilippo, Meredith Whittaker, and Kate Crawford, 'Ai now 2017 report', in AI Now 2017 Symposium and Work

[132] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kcman, 'Social data: Biases, methodological pitfalls, and ethical boundaries', Frontiers in Big Data, 2, 13, (2019).

[133] Marco, T. R., Sameer , S. & Carlos, G., 2016. "Why Should I trust You?" Explaining the Prediction of Any Classifier. *arXiv,* 9 August.pp. 1-10.

[134] Murphy, K.P., 2022. *Probabilistic machine learning: an introduction*. MIT press.

[135] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T. Y., Shlens, J., & Le, Q. V. (2020). *Learning data augmentation strategies for object detection*. In *European Conference on Computer Vision* (pp. 566–583). Springer. https://arxiv.org/abs/1906.11172

[136] National Institute of Standards and Technology (NIST). (2024). *Face Recognition Vendor Test (FRVT) and Mugshot Database*. https://www.nist.gov/programs-projects/face-recognition-vendor-test-frvt

[137] University of Central Florida (UCF). (2024). *UCF Crime Dataset for Action Recognition*. https://www.crcv.ucf.edu/research/data-sets/ucf-crime/

[138] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

[139] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21-37). Springer International Publishing.

[140] Li, Y., Pang, Y., Cao, J., Shen, J. and Shao, L., 2021. Improving single shot object detection with feature scale unmixing. *IEEE Transactions on Image Processing*, *30*, pp.2708-2721.

[141] Zhang, H., Zhang, W., Wang, W., Li, X. and Zhang, A., 2024. Research on traffic sign detection algorithm based on improved SSD in complex environments. *Measurement Science and Technology*, *35*(11), p.115404.

[142] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Cham, Switzerland: Springer, 2014, pp. 740-755.

[143] K. Tran and N. D. Tran Pham, "Comparative Analysis of Image Processing Object Detection Models: SSD MobileNet and YOLO for Guava Application," in The International Conference on Sustainable Energy Technologies, Singapore, Nov. 2023, pp. 443-451. Springer Nature Singapore.

[144] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[145] Trauth, M.H., 2024. Introduction to Python. In *Python Recipes for Earth Sciences* (pp. 9-56). Cham: Springer Nature Switzerland.

[146] Muzammil, R. and Wajid, M., 2020, November. GPU-accelerated QPSK Transceiver with FEC over a Flat-fading Channel. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 491-495). IEEE.

[147] Rakhimov, M., Zaripova, D., Javliev, S. and Karimberdiyev, J., 2024, November. Deep learning parallel approach using CUDA technology. In *AIP Conference Proceedings* (Vol. 3244, No. 1). AIP Publishing.

[148] Yi, X., 2024. A Study of Performance Programming of CPU, GPU accelerated Computers and SIMD Architecture. *arXiv preprint arXiv:2409.10661*.

[149] Gu, J., Luo, X., Zhou, Y. and Wang, X., 2022, May. Muffin: Testing deep learning libraries via neural architecture fuzzing. In *Proceedings of the 44th International Conference on Software Engineering* (pp. 1418-1430).

[150] R. Padilla, S. L. Netto, and E. A. Da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237-242.

[151] Defazio, A. and Mishchenko, K., 2023, July. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning* (pp. 7449-7479). PMLR.

[152] Zaheer, M.Z., Mahmood, A., Khan, M.H., Segu, M., Yu, F. and Lee, S.I., 2022. Generative cooperative learning for unsupervised video anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14744-14754).

[153] Xie, Z., Wang, X., Zhang, H., Sato, I. and Sugiyama, M., 2022, June. Adaptive inertia: Disentangling the effects of adaptive learning rate and momentum. In *International conference on machine learning* (pp. 24430-24459). PMLR.

[154] Wang, J., Li, W. and Luo, X., 2024. A distributed adaptive second-order latent factor analysis model. *IEEE/CAA Journal of Automatica Sinica*.

[155] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B., 2016, September. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3464-3468). Ieee.

[156] Chung, M., 2024. MULTI-OBJECT TRACKING USING KALMAN FILTER AND HUNGARIAN ALGORITHM.

[157] Hassan, S., Mujtaba, G., Rajput, A. and Fatima, N., 2024. Multi-object tracking: a systematic literature review. *Multimedia Tools and Applications*, *83*(14), pp.43439-43492.

[158] Hsieh, J.W., Hsu, Y.T., Liao, H.Y.M. and Chen, C.C., 2008. Video-based human movement analysis and its application to surveillance systems. *IEEE Transactions on Multimedia*, *10*(3), pp.372-384.

[159] Sharma, P. and Rana, C., 2024. Artificial intelligence based object detection and traffic prediction by autonomous vehicles–A review. *Expert Systems with Applications*, p.124664.

[160] Mohanty, P.K. and Parhi, D.R., 2013. Controlling the motion of an autonomous mobile robot using various techniques: a review. *Journal of Advance Mechanical Engineering*, *1*(1), pp.24-39.

[161] Wang, L., Hu, W. and Tan, T., 2003. Recent developments in human motion analysis. *Pattern recognition*, *36*(3), pp.585-601.

[162] Carvalho, G.D.S., 2021. *Kalman filter-based object tracking techniques for indoor robotic applications* (Master's thesis).

[163] Sapkota, R., Qureshi, R., Calero, M.F., Badjugar, C., Nepal, U., Poulose, A., Zeno, P., Vaddevolu, U.B.P., Khan, S., Shoman, M. and Yan, H., 2024. YOLOv10 to its genesis: a decadal and comprehensive review of the you only look once (YOLO) series. *arXiv preprint arXiv:2406.19407*.

[164] Qiu, Z., Zhao, N., Zhou, L., Wang, M., Yang, L., Fang, H., He, Y. and Liu, Y., 2020. Vision-based moving obstacle detection and tracking in paddy field using improved yolov3 and deep SORT. *Sensors*, *20*(15), p.4082.

[165] Samyal, A.S. and Hans, S., 2022. Analysis and adaptation of yolov4 for object detection in aerial images. *arXiv preprint arXiv:2203.10194*.

[166] Aloysius, N. and Geetha, M., 2017, April. A review on deep convolutional neural networks. In *2017 international conference on communication and signal processing (ICCSP)* (pp. 0588-0592). IEEE.

[167] Simard, P., Bottou, L., Haffner, P. and LeCun, Y., 1998. Boxlets: a fast convolution algorithm for signal processing and neural networks. *Advances in neural information processing systems*, *11*.

[168] Saha, S. and Gokhale, T., 2024. Improving Shift Invariance in Convolutional Neural Networks with Translation Invariant Polyphase Sampling. *arXiv preprint arXiv:2404.07410*.

[169] Inthiyaz, S., Ahammad, S.H., Krishna, A., Bhargavi, V., Govardhan, D. and Rajesh, V., 2020. YOLO (YOU ONLY LOOK ONCE) Making Object detection work in Medical Imaging on Convolution detection System. *International Journal of Pharmaceutical Research (09752366)*, *12*(2).

[170] Dewi, C., Chen, R.C., Jiang, X. and Yu, H., 2022. Deep convolutional neural network for enhancing traffic sign recognition developed on Yolo V4. *Multimedia Tools and Applications*, *81*(26), pp.37821-37845.

[171] Chary, P.S., 2023. Real Time Object Detection Using YOLOv4. *International Journal for Research in Applied Science and Engineering Technology*, *11*(12), pp.1375-1379.

[172] Sweeney, D., Tuthill, P., Krone-Martins, A., Mérand, A., Scalzo, R. and Martinod, M.A., 2024. Observing the galactic underworld: predicting photometry and

astrometry from compact remnant microlensing events. *Monthly Notices of the Royal Astronomical Society*, *531*(2), pp.2433-2447.

[173] Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P. and Zeng, A., 2023, May. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9493-9500). IEEE.

[174] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M. and Parmar, M., 2024. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, *57*(4), p.99.

[175] Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.

[176] Zhang, Y. and Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

[177] Akinmuyisitan Taiwo; Cosmas John; Yusuf Giwa (2025) Advancing Real-Time Crime Weapon Detection and High-Risk Person Classification. Global Journal of Computer Science and Technology. D Neural & Artificial Intelligence Vol 25(1)

[178] Abhishek, K., Jain, A. and Hamarneh, G., 2025. Investigating the quality of dermamnist and fitzpatrick17k dermatological image datasets. *Scientific Data*, *12*(1), p.196.

[179] Georgiadis, P., Gkouvrikos, E.V., Vrochidou, E., Kalampokas, T. and Papakostas, G.A., 2025. Building Better Deep Learning Models Through Dataset Fusion: A Case Study in Skin Cancer Classification with Hyperdatasets. *Diagnostics*, *15*(3), p.352.

[180] Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[181] Roy, A.M., Bose, R. and Bhaduri, J., 2022. A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. *Neural Computing and Applications*, *34*(5), pp.3895-3921.

[182] Carrinho, P. and Falcao, G., 2023. Highly accurate and fast YOLOv4-based polyp detection. *Expert Systems with Applications*, *232*, p.120834.

[183] Zhou, X., Yi, J., Xie, G., Jia, Y., Xu, G. and Sun, M., 2022. Human detection algorithm based on improved YOLO v4. *Information Technology and Control*, *51*(3), pp.485-498

[184] Ding, P., Li, T., Qian, H., Ma, L. and Chen, Z., 2025. A lightweight real-time object detection method for complex scenes based on YOLOv4. *Journal of Real-Time Image Processing*, *22*(2), pp.1-13.

[185] Hoang, V.H., Lee, J.W. and Park, C.S., 2025. Enhancing Fire Detection with YOLO Models: A Bayesian Hyperparameter Tuning Approach. *Computers, Materials & Continua*, *83*(3).

[186] S. M, S. Mahapatra and S. Jhansi, "Innovative Object Detection with YOLOv4 and Adaptive Bounding Boxes," *2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL)*, Bhimdatta, Nepal, 2025, pp. 885-891, doi: 10.1109/ICSADL65848.2025.1093316

[187] Nascimento, J.C. and Marques, J.S., 2006. Performance evaluation of object detection algorithms for video surveillance. *IEEE Transactions on Multimedia*, *8*(4), pp.761-774.

[188] Mwitta, C., Rains, G.C. and Prostko, E., 2024. Evaluation of inference performance of deep learning models for real-time weed detection in an embedded computer. *Sensors*, *24*(2), p.514.

[189] Mahrishi, M., Morwal, S., Muzaffar, A.W., Bhatia, S., Dadheech, P. and Rahmani, M.K.I., 2021. Video index point detection and extraction framework using custom YoloV4 Darknet object detection model. *IEEE Access*, *9*, pp.143378-143391.

[190] Teja, Y.D., 2023. Static object detection for video surveillance. *Multimedia Tools and Applications*, *82*(14), pp.21627-21639.

[191] Hu, B., Xia, M., Zhao, D. and Wu, G., 2025. MONA: Moving Object Detection from Videos Shot by Dynamic Camera. *arXiv preprint arXiv:2501.13183*.

[192] Tsangaratos, P. and Ilia, I., 2016. Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. *Catena*, *145*, pp.164-179.

[193] Yuksel, B.B. and Metin, A.Y., 2025. Data-Driven Breakthroughs and Future Directions in AI Infrastructure: A Comprehensive Review. *arXiv preprint arXiv:2505.16771*

[194] Shah, H.M., 2025. Optimizing machine learning pipelines for cost and performance using cloud

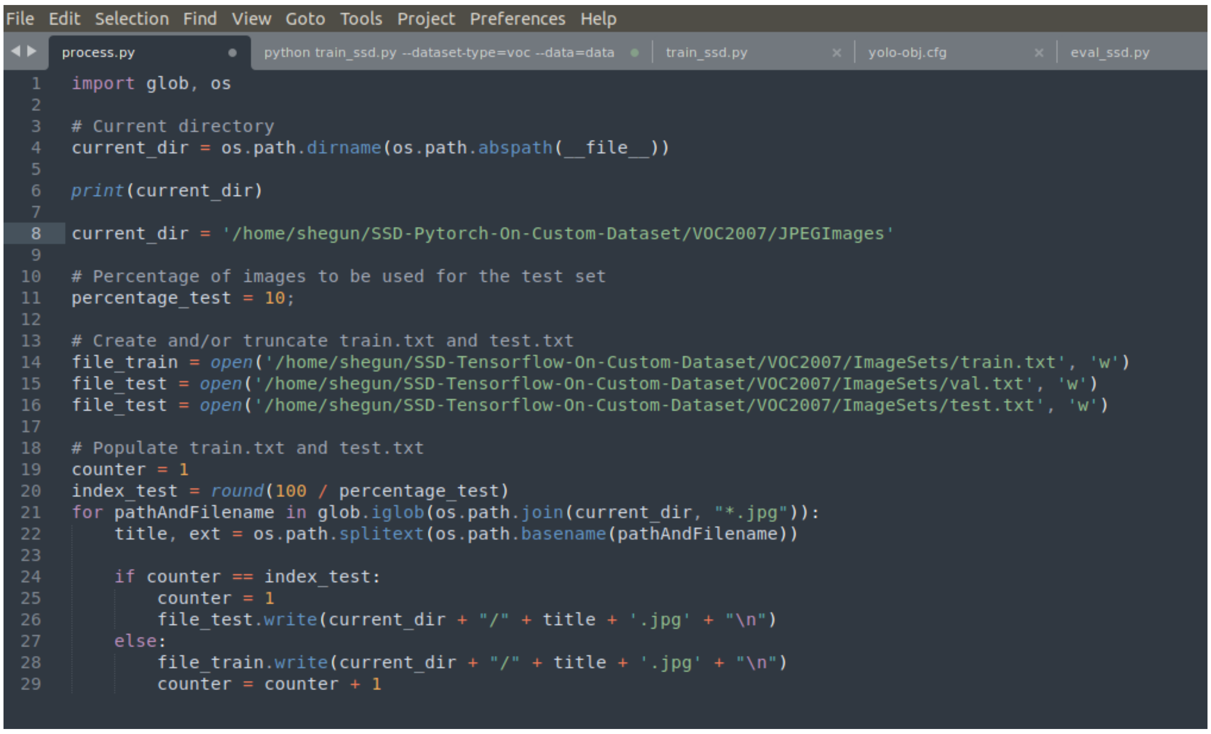
[195] Chung-Kwan Shin, Ui Tak Yun, Huy Kang Kim and Sang Chan Park, "A hybrid approach of neural network and memory-based learning to data mining," in IEEE Transactions on Neural Networks, vol. 11, no. 3, pp. 637-646, May 2000, doi: 10.1109/72.846735.

[196] Hanna, M.G., Pantanowitz, L., Jackson, B., Palmer, O., Visweswaran, S., Pantanowitz, J., Deebajah, M. and Rashidi, H.H., 2025. Ethical and bias considerations in artificial intelligence/machine learning. *Modern Pathology*, *38*(3), p.100686

[197] Reddy, L.V.K. and Rajendran, R.K., 2025. Addressing Privacy Setting Loopholes Challenges and the Need for Enhanced Data Protection. *In Analyzing Privacy and Security Difficulties in Social Media: New Challenges and Solutions (pp. 335-364). IGI Global Scientific Publishing*.

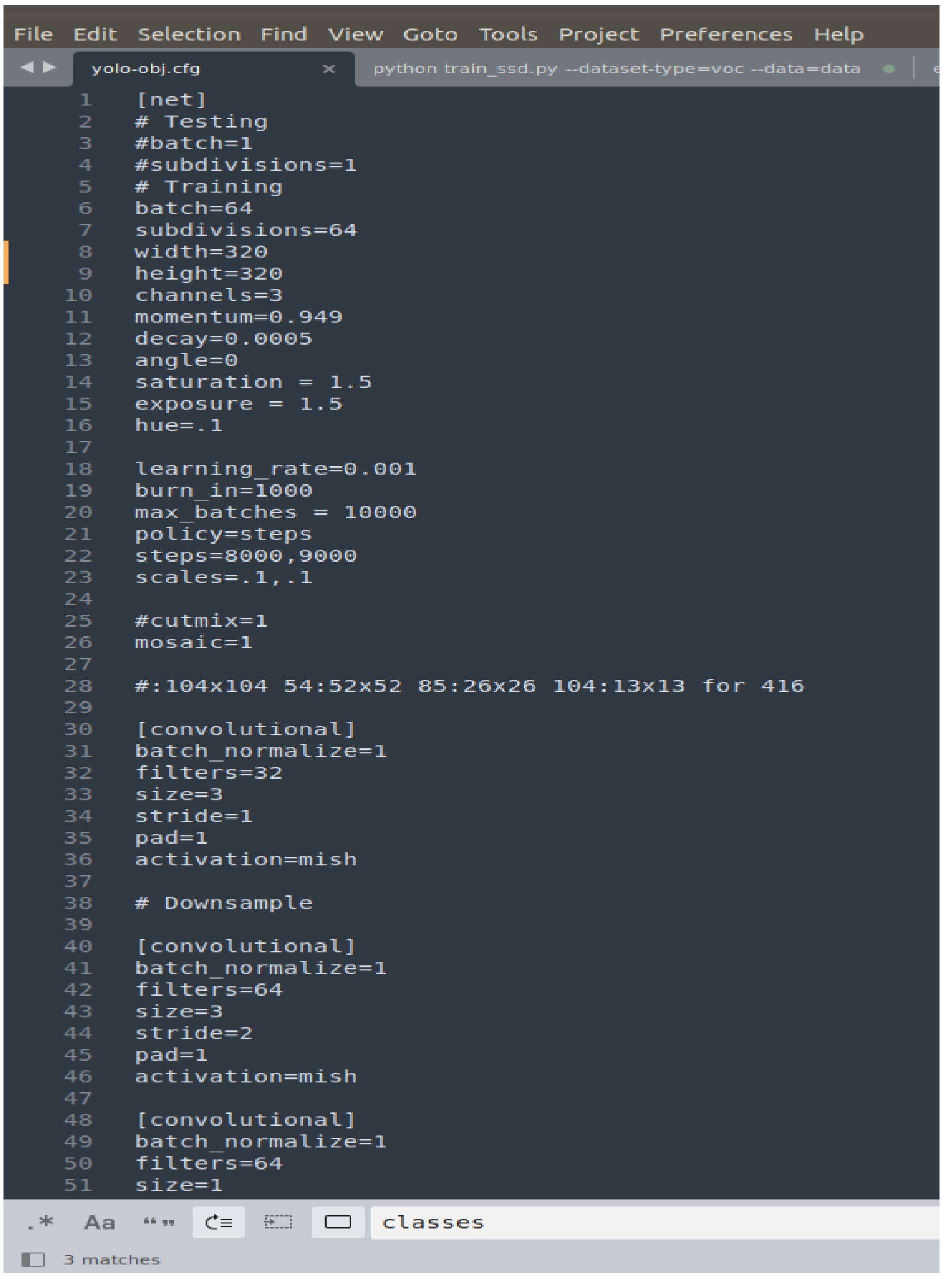
[198] Andrew D. Selbst, danah boyd, Sorelle A. Friedler, Suresh Venkatasubrama Systems. In FAT* '19: Conference on Fairness, Accountability, and Transparency (FAT* '19), January 29–31, 2019, Atlanta, GA, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3287560.328759

APPENDIX A: Process.py

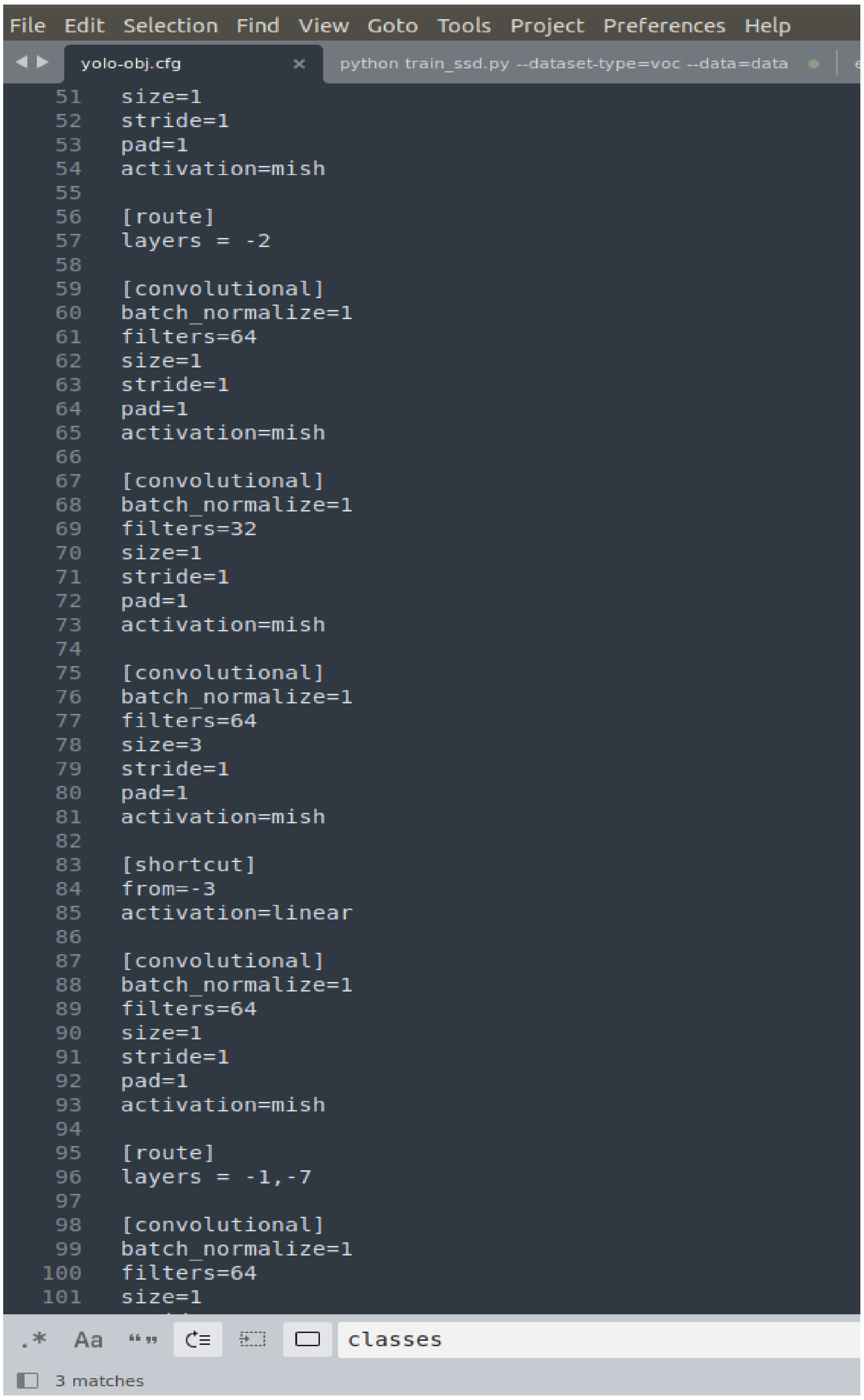

File Edit Selection Find View Goto Tools Project Preferences Help
process.py
python train_ssd.py --dataset-type=voc --data=data
train_ssd.py
yolo-obj.cfg
eval_ssd.py

```python
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)

current_dir = '/home/shegun/SSD-Pytorch-On-Custom-Dataset/VOC2007/JPEGImages'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('/home/shegun/SSD-Tensorflow-On-Custom-Dataset/VOC2007/ImageSets/train.txt', 'w')
file_test = open('/home/shegun/SSD-Tensorflow-On-Custom-Dataset/VOC2007/ImageSets/val.txt', 'w')
file_test = open('/home/shegun/SSD-Tensorflow-On-Custom-Dataset/VOC2007/ImageSets/test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write(current_dir + "/" + title + '.jpg' + "\n")
    else:
        file_train.write(current_dir + "/" + title + '.jpg' + "\n")
        counter = counter + 1
```

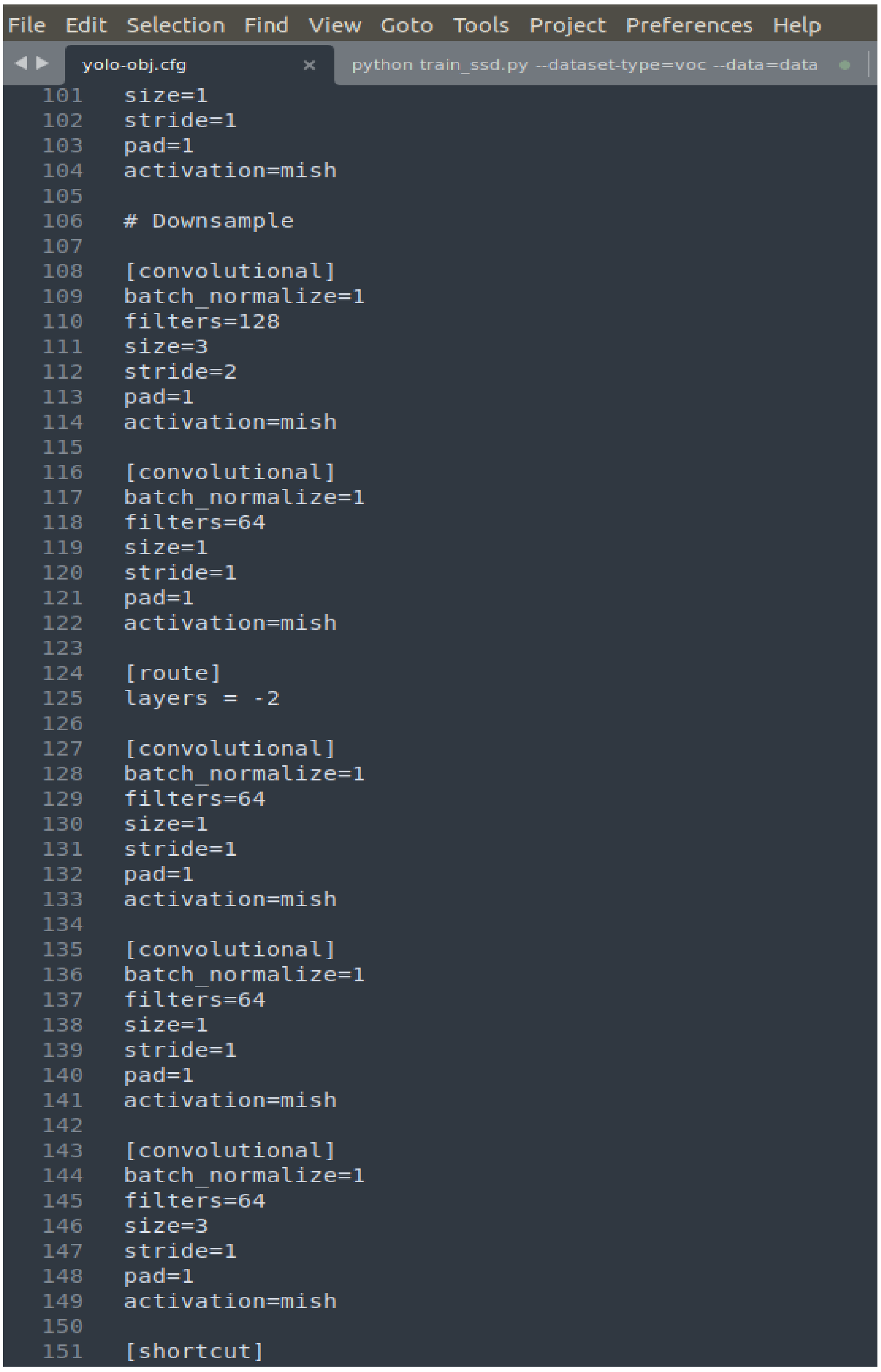
Appendix B: YOLOv4 System Architecture Configuration file


File Edit Selection Find View Goto Tools Project Preferences Help
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
classes
3 matches

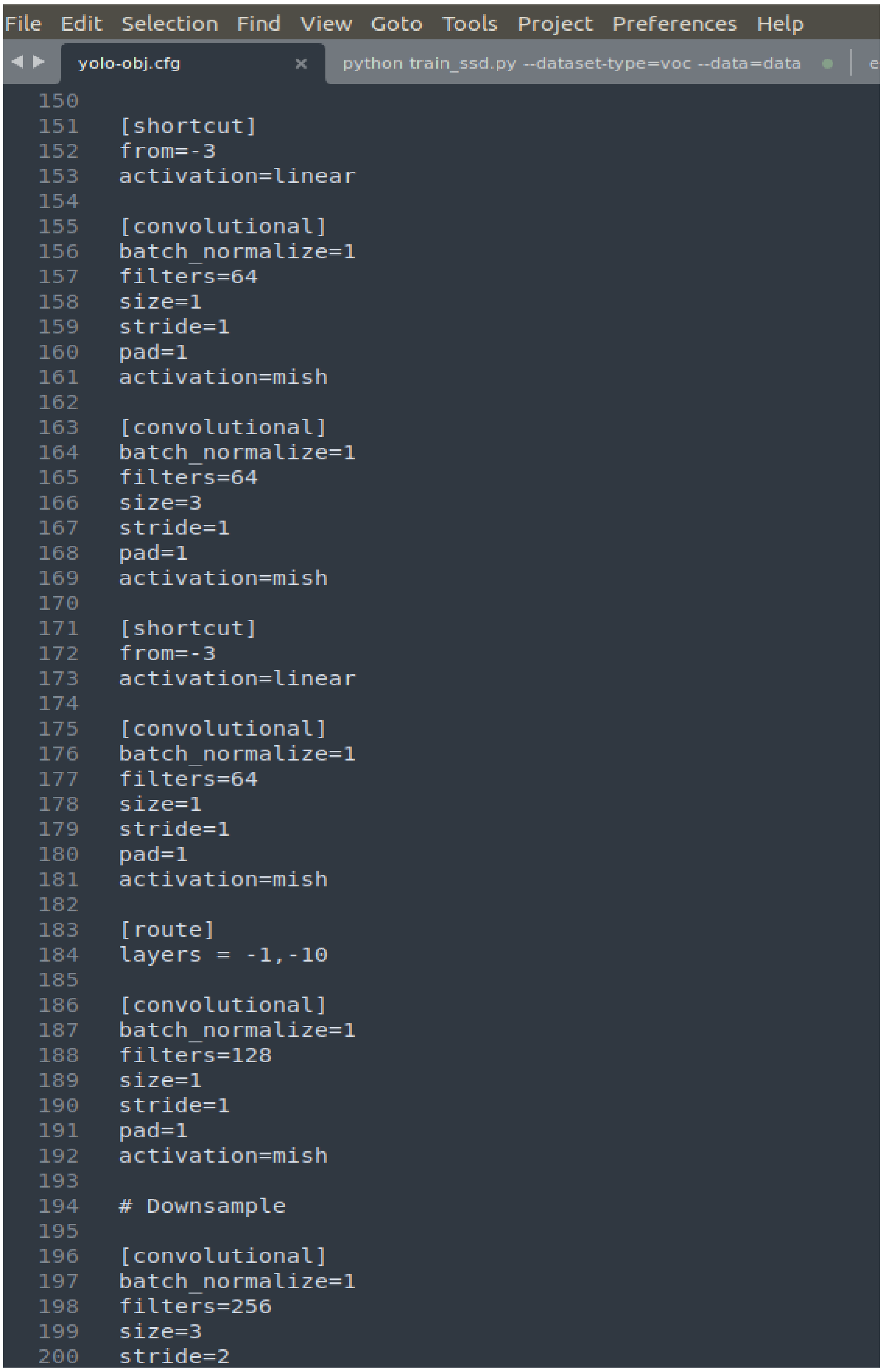
```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=320
height=320
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 10000
policy=steps
steps=8000,9000
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
```


File Edit Selection Find View Goto Tools Project Preferences Help
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
.* Aa
classes
3 matches

```
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-7

[convolutional]
batch_normalize=1
filters=64
size=1
```

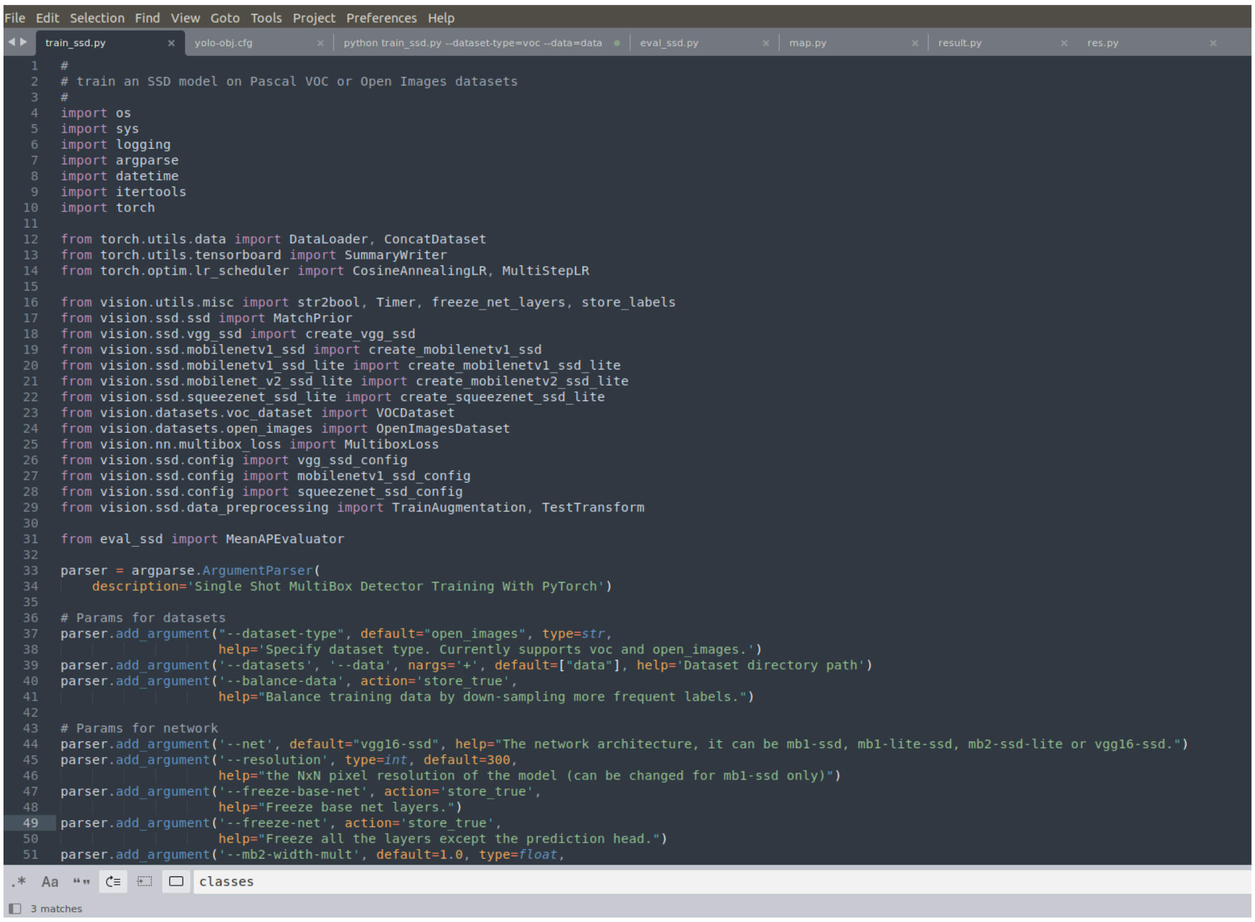

File Edit Selection Find View Goto Tools Project Preferences Help
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data

```
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
```

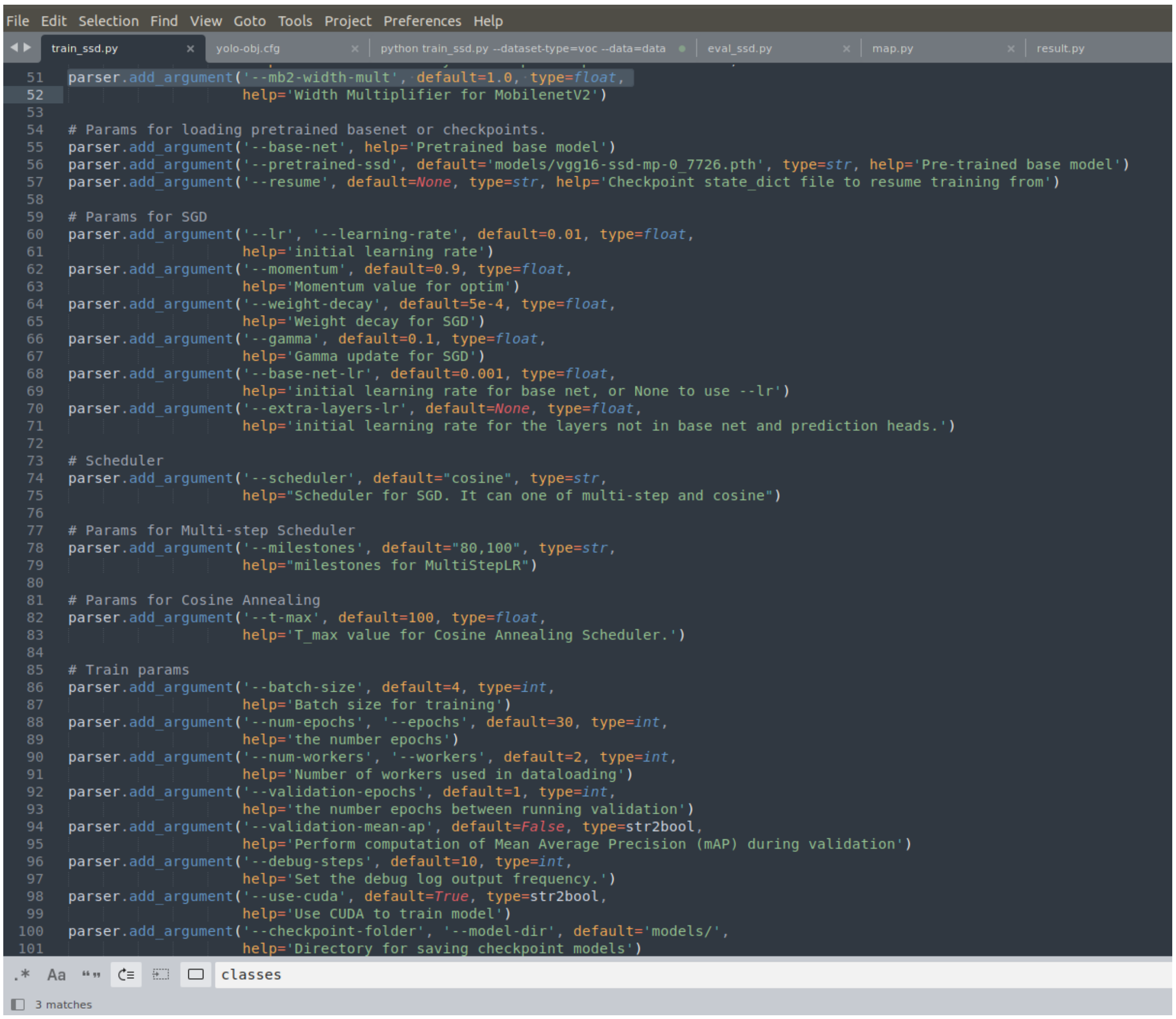

File Edit Selection Find View Goto Tools Project Preferences Help
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data

```

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-10

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
```

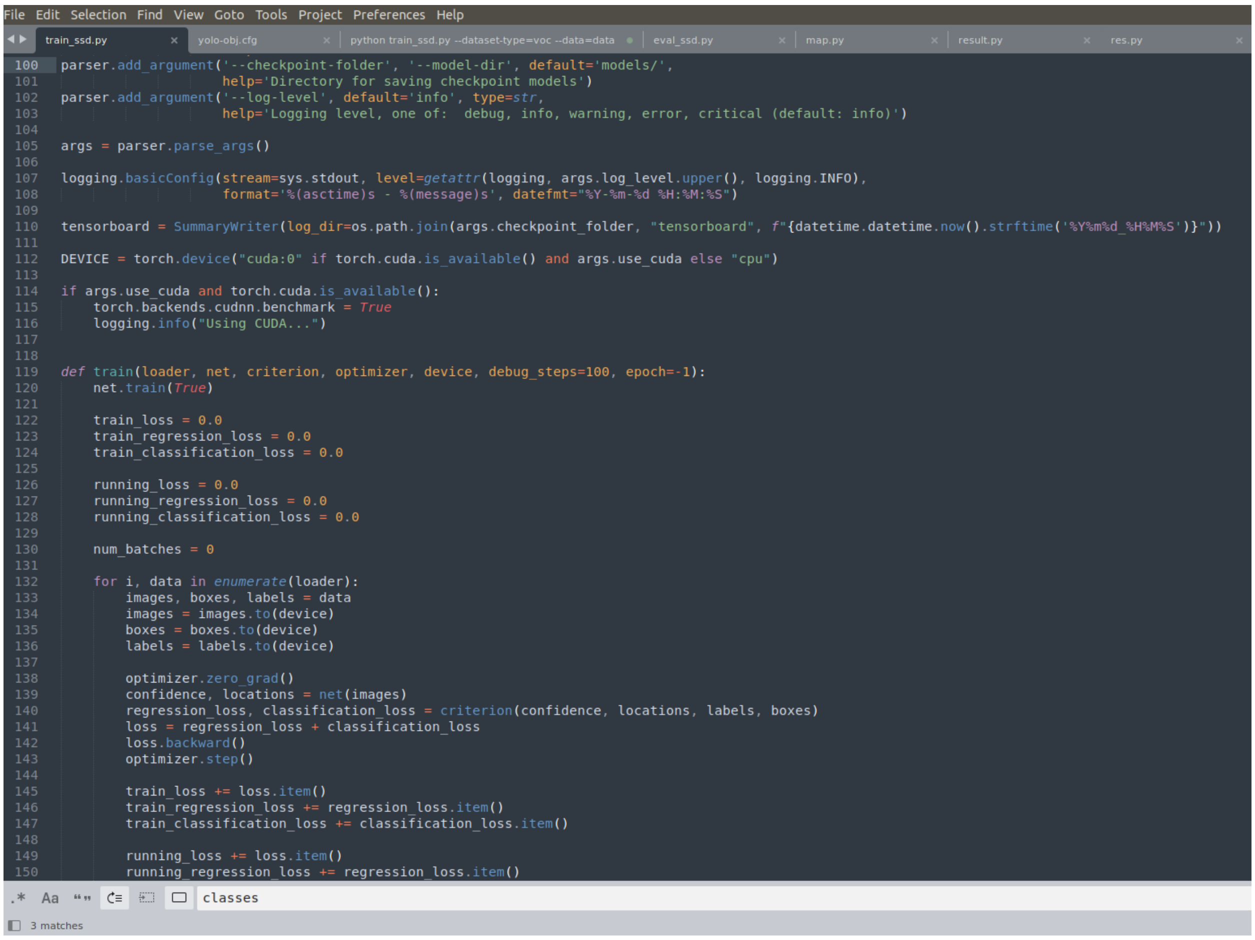
Appendix C: Single Shot Multi-Box Detection


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
result.py
res.py
classes
3 matches

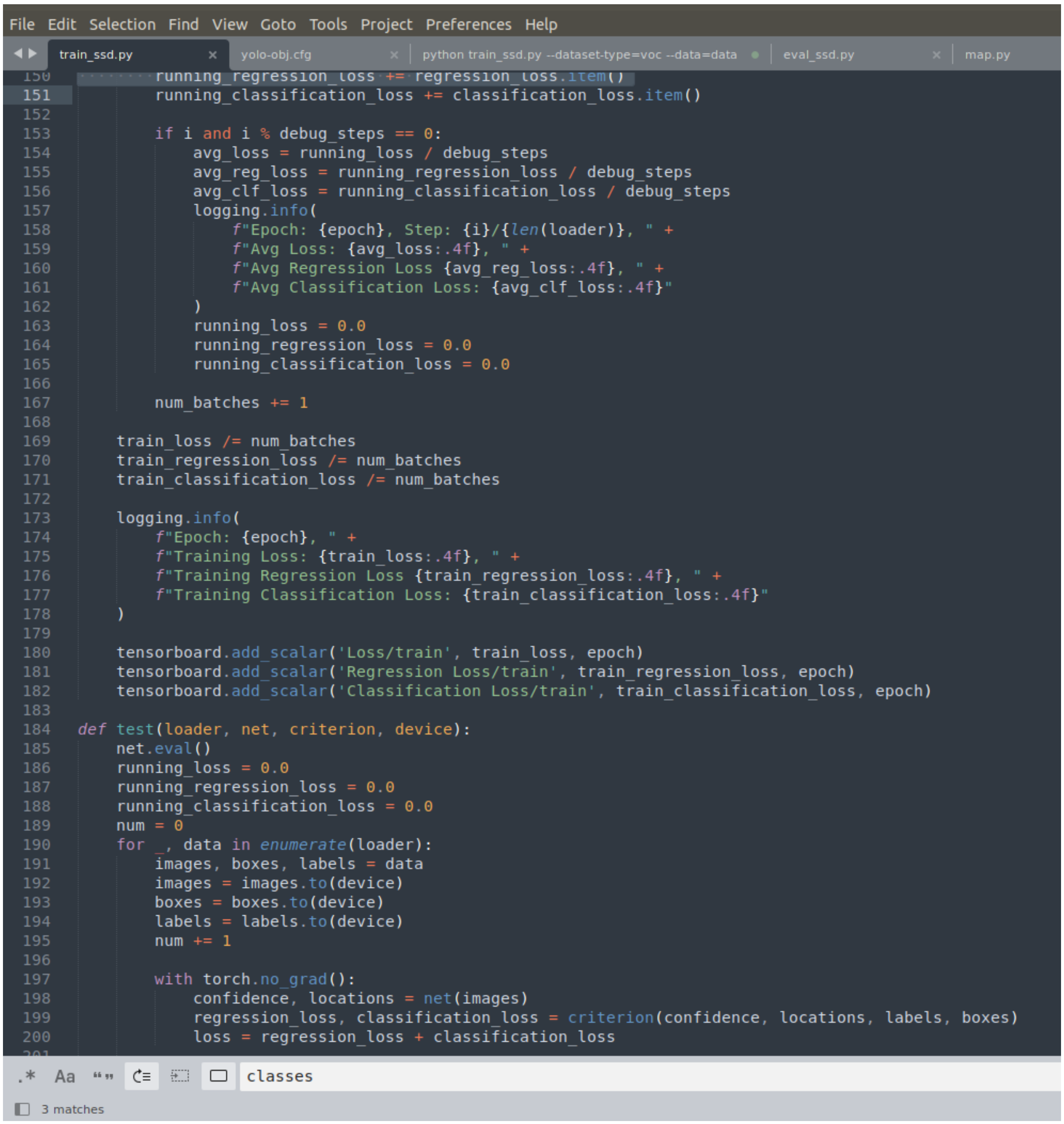
```python
#
# train an SSD model on Pascal VOC or Open Images datasets
#
import os
import sys
import logging
import argparse
import datetime
import itertools
import torch

from torch.utils.data import DataLoader, ConcatDataset
from torch.utils.tensorboard import SummaryWriter
from torch.optim.lr_scheduler import CosineAnnealingLR, MultiStepLR

from vision.utils.misc import str2bool, Timer, freeze_net_layers, store_labels
from vision.ssd.ssd import MatchPrior
from vision.ssd.vgg_ssd import create_vgg_ssd
from vision.ssd.mobilenetv1_ssd import create_mobilenetv1_ssd
from vision.ssd.mobilenetv1_ssd_lite import create_mobilenetv1_ssd_lite
from vision.ssd.mobilenet_v2_ssd_lite import create_mobilenetv2_ssd_lite
from vision.ssd.squeezenet_ssd_lite import create_squeezenet_ssd_lite
from vision.datasets.voc_dataset import VOCDataset
from vision.datasets.open_images import OpenImagesDataset
from vision.nn.multibox_loss import MultiboxLoss
from vision.ssd.config import vgg_ssd_config
from vision.ssd.config import mobilenetv1_ssd_config
from vision.ssd.config import squeezenet_ssd_config
from vision.ssd.data_preprocessing import TrainAugmentation, TestTransform

from eval_ssd import MeanAPEvaluator

parser = argparse.ArgumentParser(
    description='Single Shot MultiBox Detector Training With PyTorch')

# Params for datasets
parser.add_argument("--dataset-type", default="open_images", type=str,
                    help='Specify dataset type. Currently supports voc and open_images.')
parser.add_argument('--datasets', '--data', nargs='+', default=["data"], help='Dataset directory path')
parser.add_argument('--balance-data', action='store_true',
                    help="Balance training data by down-sampling more frequent labels.")

# Params for network
parser.add_argument('--net', default="vgg16-ssd", help="The network architecture, it can be mb1-ssd, mb1-lite-ssd, mb2-ssd-lite or vgg16-ssd.")
parser.add_argument('--resolution', type=int, default=300,
                    help="the NxN pixel resolution of the model (can be changed for mb1-ssd only)")
parser.add_argument('--freeze-base-net', action='store_true',
                    help="Freeze base net layers.")
parser.add_argument('--freeze-net', action='store_true',
                    help="Freeze all the layers except the prediction head.")
parser.add_argument('--mb2-width-mult', default=1.0, type=float,
```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
result.py
classes
3 matches

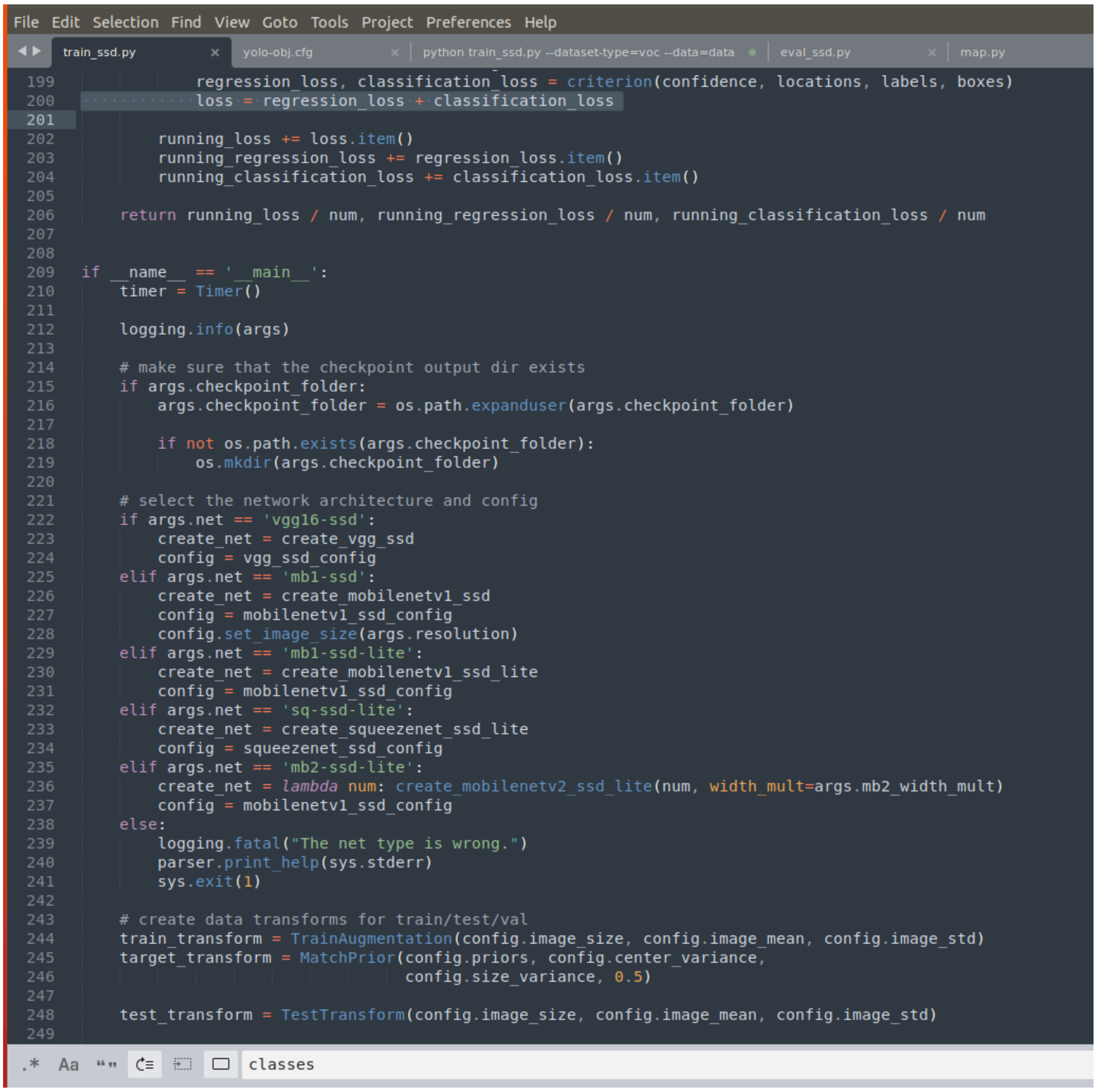
```python
parser.add_argument('--mb2-width-mult', default=1.0, type=float,
                    help='Width Multiplifier for MobilenetV2')

# Params for loading pretrained basenet or checkpoints.
parser.add_argument('--base-net', help='Pretrained base model')
parser.add_argument('--pretrained-ssd', default='models/vgg16-ssd-mp-0_7726.pth', type=str, help='Pre-trained base model')
parser.add_argument('--resume', default=None, type=str, help='Checkpoint state_dict file to resume training from')

# Params for SGD
parser.add_argument('--lr', '--learning-rate', default=0.01, type=float,
                    help='initial learning rate')
parser.add_argument('--momentum', default=0.9, type=float,
                    help='Momentum value for optim')
parser.add_argument('--weight-decay', default=5e-4, type=float,
                    help='Weight decay for SGD')
parser.add_argument('--gamma', default=0.1, type=float,
                    help='Gamma update for SGD')
parser.add_argument('--base-net-lr', default=0.001, type=float,
                    help='initial learning rate for base net, or None to use --lr')
parser.add_argument('--extra-layers-lr', default=None, type=float,
                    help='initial learning rate for the layers not in base net and prediction heads.')

# Scheduler
parser.add_argument('--scheduler', default="cosine", type=str,
                    help="Scheduler for SGD. It can one of multi-step and cosine")

# Params for Multi-step Scheduler
parser.add_argument('--milestones', default="80,100", type=str,
                    help="milestones for MultiStepLR")

# Params for Cosine Annealing
parser.add_argument('--t-max', default=100, type=float,
                    help='T_max value for Cosine Annealing Scheduler.')

# Train params
parser.add_argument('--batch-size', default=4, type=int,
                    help='Batch size for training')
parser.add_argument('--num-epochs', '--epochs', default=30, type=int,
                    help='the number epochs')
parser.add_argument('--num-workers', '--workers', default=2, type=int,
                    help='Number of workers used in dataloading')
parser.add_argument('--validation-epochs', default=1, type=int,
                    help='the number epochs between running validation')
parser.add_argument('--validation-mean-ap', default=False, type=str2bool,
                    help='Perform computation of Mean Average Precision (mAP) during validation')
parser.add_argument('--debug-steps', default=10, type=int,
                    help='Set the debug log output frequency.')
parser.add_argument('--use-cuda', default=True, type=str2bool,
                    help='Use CUDA to train model')
parser.add_argument('--checkpoint-folder', '--model-dir', default='models/',
                    help='Directory for saving checkpoint models')
```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
result.py
res.py

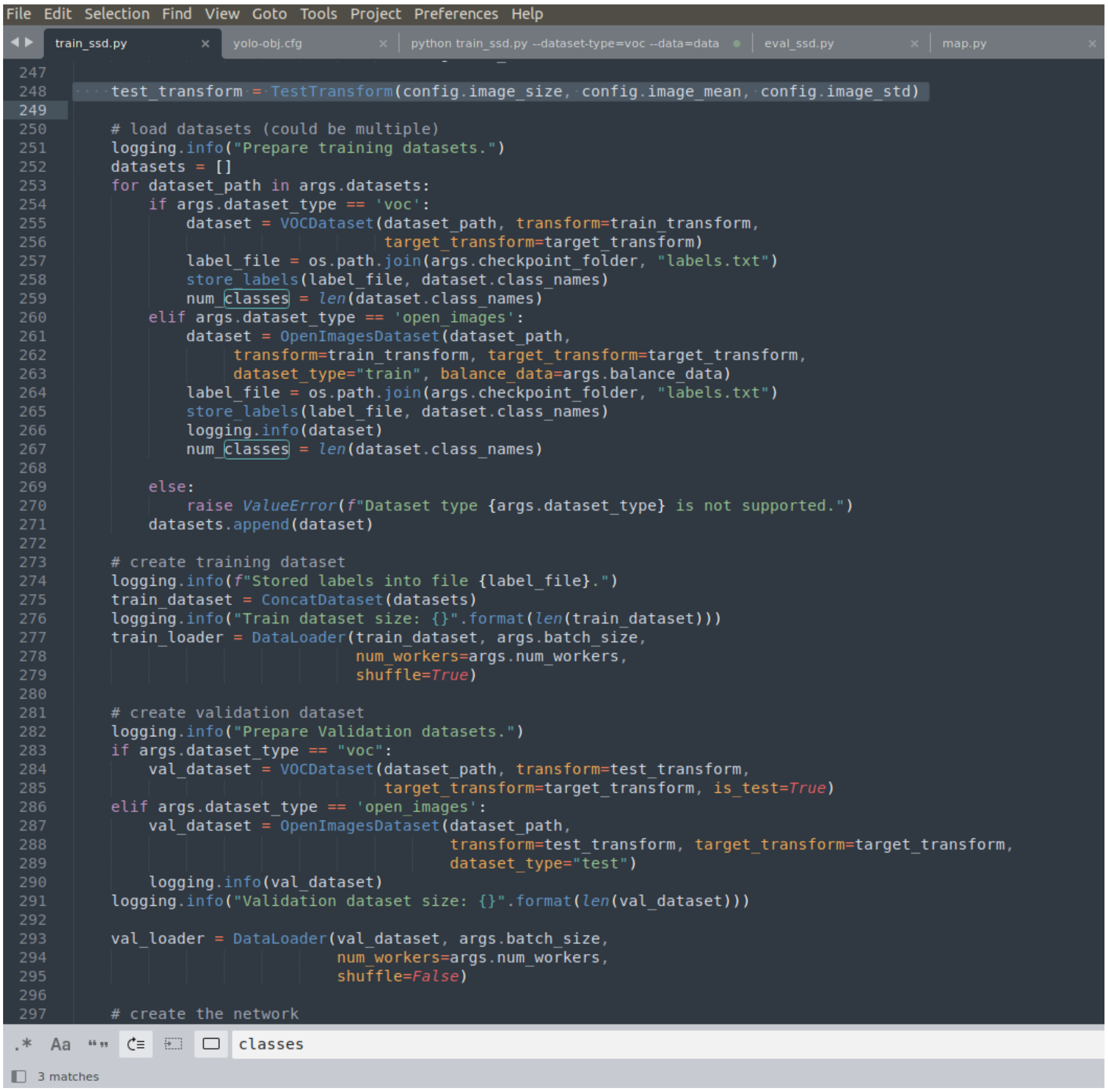
```python
parser.add_argument('--checkpoint-folder', '--model-dir', default='models/',
                    help='Directory for saving checkpoint models')
parser.add_argument('--log-level', default='info', type=str,
                    help='Logging level, one of:  debug, info, warning, error, critical (default: info)')

args = parser.parse_args()

logging.basicConfig(stream=sys.stdout, level=getattr(logging, args.log_level.upper(), logging.INFO),
                    format='%(asctime)s - %(message)s', datefmt="%Y-%m-%d %H:%M:%S")

tensorboard = SummaryWriter(log_dir=os.path.join(args.checkpoint_folder, "tensorboard", f"{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}"))

DEVICE = torch.device("cuda:0" if torch.cuda.is_available() and args.use_cuda else "cpu")

if args.use_cuda and torch.cuda.is_available():
    torch.backends.cudnn.benchmark = True
    logging.info("Using CUDA...")


def train(loader, net, criterion, optimizer, device, debug_steps=100, epoch=-1):
    net.train(True)

    train_loss = 0.0
    train_regression_loss = 0.0
    train_classification_loss = 0.0

    running_loss = 0.0
    running_regression_loss = 0.0
    running_classification_loss = 0.0

    num_batches = 0

    for i, data in enumerate(loader):
        images, boxes, labels = data
        images = images.to(device)
        boxes = boxes.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()
        confidence, locations = net(images)
        regression_loss, classification_loss = criterion(confidence, locations, labels, boxes)
        loss = regression_loss + classification_loss
        loss.backward()
        optimizer.step()

        train_loss += loss.item()
        train_regression_loss += regression_loss.item()
        train_classification_loss += classification_loss.item()

        running_loss += loss.item()
        running_regression_loss += regression_loss.item()
```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
classes
3 matches

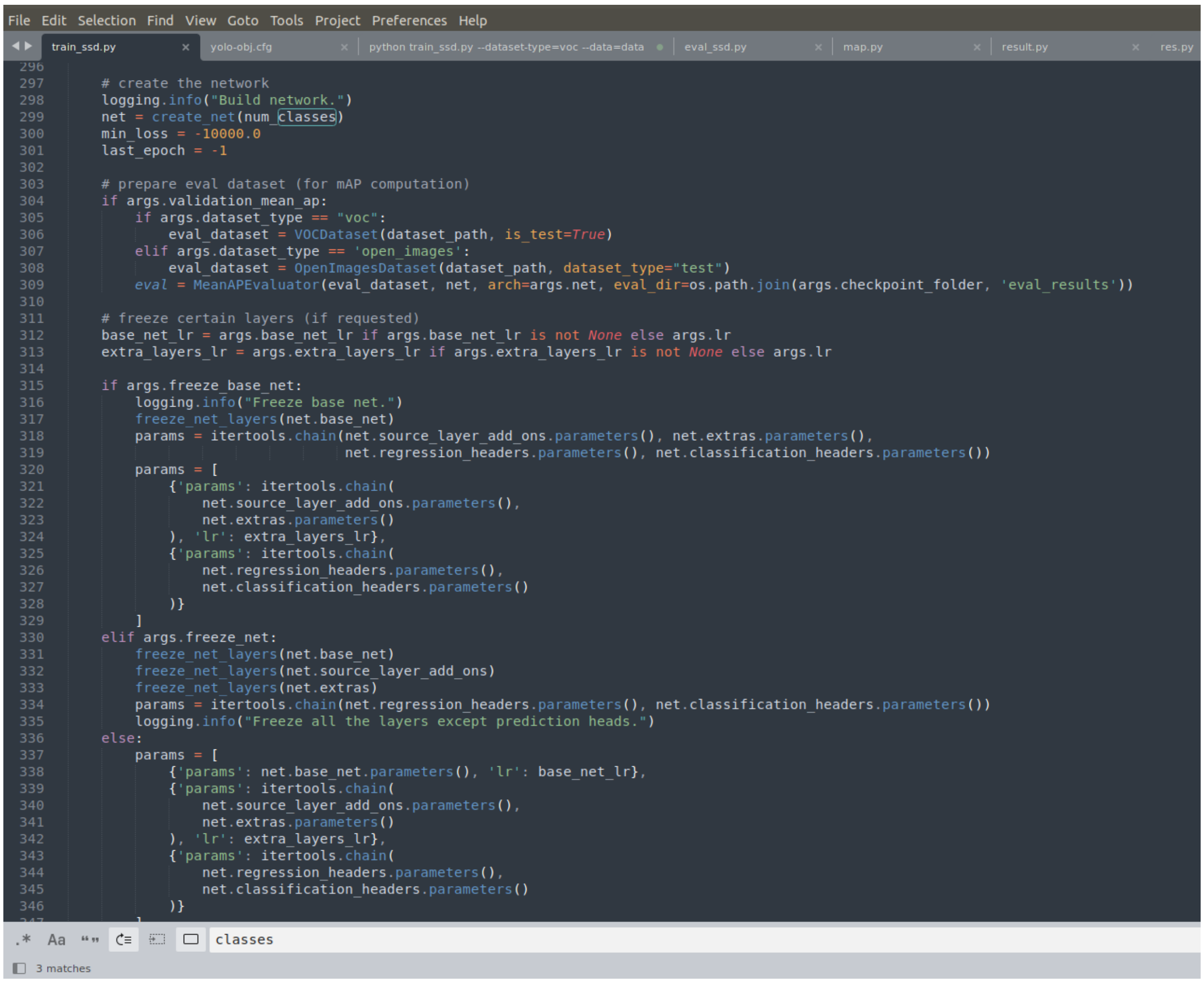
```
        running_regression_loss += regression_loss.item()
        running_classification_loss += classification_loss.item()

        if i and i % debug_steps == 0:
            avg_loss = running_loss / debug_steps
            avg_reg_loss = running_regression_loss / debug_steps
            avg_clf_loss = running_classification_loss / debug_steps
            logging.info(
                f"Epoch: {epoch}, Step: {i}/{len(loader)}, " +
                f"Avg Loss: {avg_loss:.4f}, " +
                f"Avg Regression Loss {avg_reg_loss:.4f}, " +
                f"Avg Classification Loss: {avg_clf_loss:.4f}"
            )
            running_loss = 0.0
            running_regression_loss = 0.0
            running_classification_loss = 0.0

        num_batches += 1

    train_loss /= num_batches
    train_regression_loss /= num_batches
    train_classification_loss /= num_batches

    logging.info(
        f"Epoch: {epoch}, " +
        f"Training Loss: {train_loss:.4f}, " +
        f"Training Regression Loss {train_regression_loss:.4f}, " +
        f"Training Classification Loss: {train_classification_loss:.4f}"
    )

    tensorboard.add_scalar('Loss/train', train_loss, epoch)
    tensorboard.add_scalar('Regression Loss/train', train_regression_loss, epoch)
    tensorboard.add_scalar('Classification Loss/train', train_classification_loss, epoch)

def test(loader, net, criterion, device):
    net.eval()
    running_loss = 0.0
    running_regression_loss = 0.0
    running_classification_loss = 0.0
    num = 0
    for _, data in enumerate(loader):
        images, boxes, labels = data
        images = images.to(device)
        boxes = boxes.to(device)
        labels = labels.to(device)
        num += 1

        with torch.no_grad():
            confidence, locations = net(images)
            regression_loss, classification_loss = criterion(confidence, locations, labels, boxes)
            loss = regression_loss + classification_loss
```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
classes

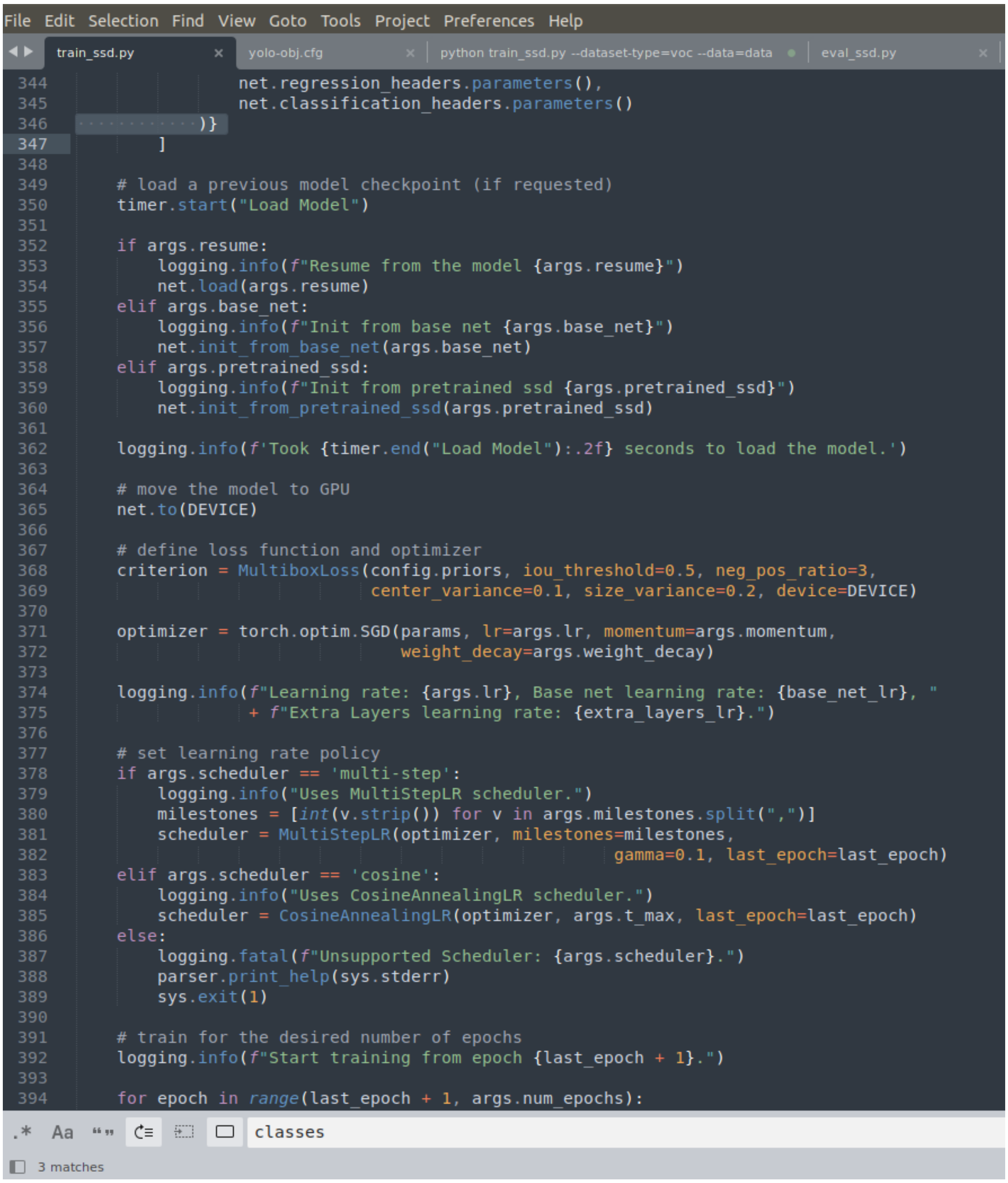
```
                regression_loss, classification_loss = criterion(confidence, locations, labels, boxes)
                loss = regression_loss + classification_loss

            running_loss += loss.item()
            running_regression_loss += regression_loss.item()
            running_classification_loss += classification_loss.item()

    return running_loss / num, running_regression_loss / num, running_classification_loss / num


if __name__ == '__main__':
    timer = Timer()

    logging.info(args)

    # make sure that the checkpoint output dir exists
    if args.checkpoint_folder:
        args.checkpoint_folder = os.path.expanduser(args.checkpoint_folder)

        if not os.path.exists(args.checkpoint_folder):
            os.mkdir(args.checkpoint_folder)

    # select the network architecture and config
    if args.net == 'vgg16-ssd':
        create_net = create_vgg_ssd
        config = vgg_ssd_config
    elif args.net == 'mb1-ssd':
        create_net = create_mobilenetv1_ssd
        config = mobilenetv1_ssd_config
        config.set_image_size(args.resolution)
    elif args.net == 'mb1-ssd-lite':
        create_net = create_mobilenetv1_ssd_lite
        config = mobilenetv1_ssd_config
    elif args.net == 'sq-ssd-lite':
        create_net = create_squeezenet_ssd_lite
        config = squeezenet_ssd_config
    elif args.net == 'mb2-ssd-lite':
        create_net = lambda num: create_mobilenetv2_ssd_lite(num, width_mult=args.mb2_width_mult)
        config = mobilenetv1_ssd_config
    else:
        logging.fatal("The net type is wrong.")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # create data transforms for train/test/val
    train_transform = TrainAugmentation(config.image_size, config.image_mean, config.image_std)
    target_transform = MatchPrior(config.priors, config.center_variance,
                                  config.size_variance, 0.5)

    test_transform = TestTransform(config.image_size, config.image_mean, config.image_std)

```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py

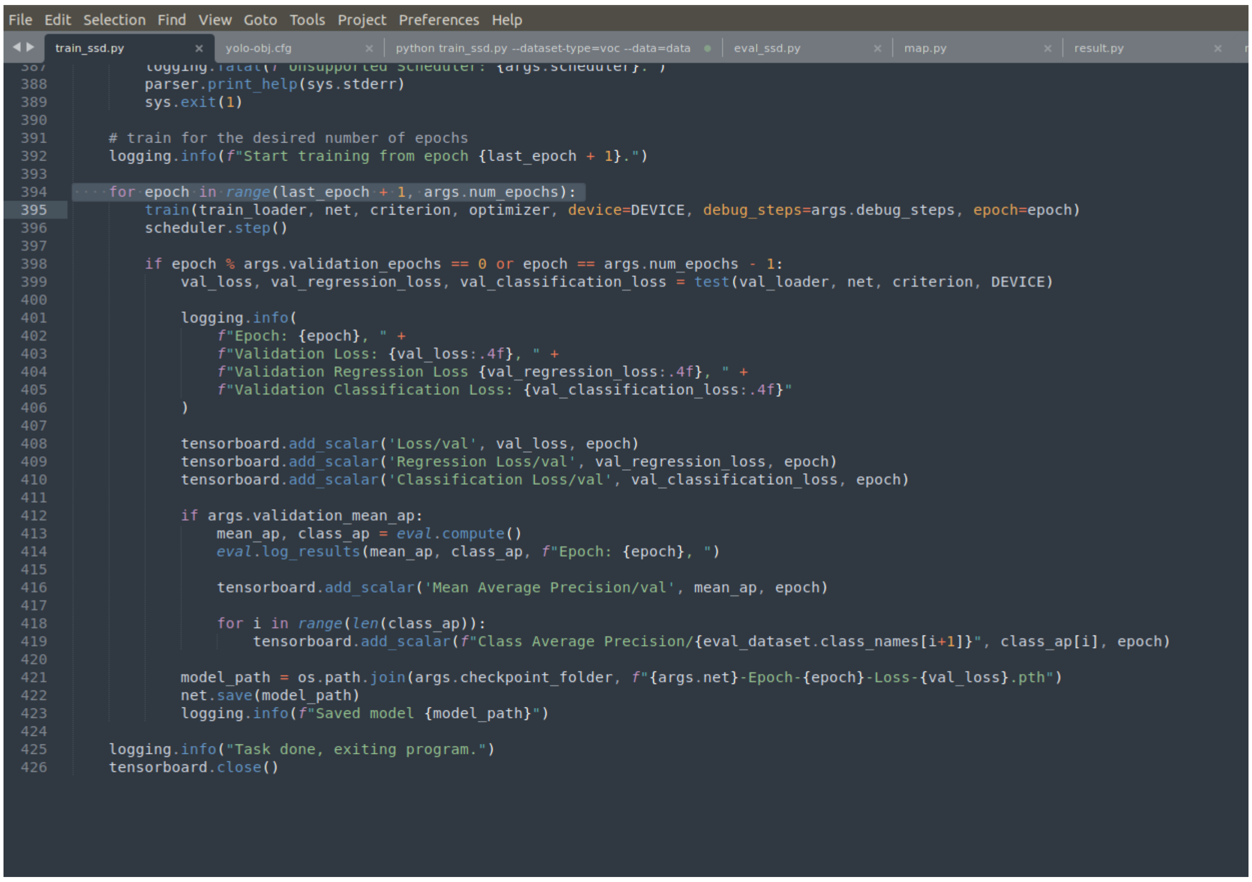
```

    test_transform = TestTransform(config.image_size, config.image_mean, config.image_std)

    # load datasets (could be multiple)
    logging.info("Prepare training datasets.")
    datasets = []
    for dataset_path in args.datasets:
        if args.dataset_type == 'voc':
            dataset = VOCDataset(dataset_path, transform=train_transform,
                                 target_transform=target_transform)
            label_file = os.path.join(args.checkpoint_folder, "labels.txt")
            store_labels(label_file, dataset.class_names)
            num_classes = len(dataset.class_names)
        elif args.dataset_type == 'open_images':
            dataset = OpenImagesDataset(dataset_path,
                 transform=train_transform, target_transform=target_transform,
                 dataset_type="train", balance_data=args.balance_data)
            label_file = os.path.join(args.checkpoint_folder, "labels.txt")
            store_labels(label_file, dataset.class_names)
            logging.info(dataset)
            num_classes = len(dataset.class_names)

        else:
            raise ValueError(f"Dataset type {args.dataset_type} is not supported.")
        datasets.append(dataset)

    # create training dataset
    logging.info(f"Stored labels into file {label_file}.")
    train_dataset = ConcatDataset(datasets)
    logging.info("Train dataset size: {}".format(len(train_dataset)))
    train_loader = DataLoader(train_dataset, args.batch_size,
                              num_workers=args.num_workers,
                              shuffle=True)

    # create validation dataset
    logging.info("Prepare Validation datasets.")
    if args.dataset_type == "voc":
        val_dataset = VOCDataset(dataset_path, transform=test_transform,
                                 target_transform=target_transform, is_test=True)
    elif args.dataset_type == 'open_images':
        val_dataset = OpenImagesDataset(dataset_path,
                                        transform=test_transform, target_transform=target_transform,
                                        dataset_type="test")
        logging.info(val_dataset)
    logging.info("Validation dataset size: {}".format(len(val_dataset)))

    val_loader = DataLoader(val_dataset, args.batch_size,
                            num_workers=args.num_workers,
                            shuffle=False)

    # create the network
```


File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py
map.py
result.py
res.py
classes
3 matches

```
    # create the network
    logging.info("Build network.")
    net = create_net(num_classes)
    min_loss = -10000.0
    last_epoch = -1

    # prepare eval dataset (for mAP computation)
    if args.validation_mean_ap:
        if args.dataset_type == "voc":
            eval_dataset = VOCDataset(dataset_path, is_test=True)
        elif args.dataset_type == 'open_images':
            eval_dataset = OpenImagesDataset(dataset_path, dataset_type="test")
        eval = MeanAPEvaluator(eval_dataset, net, arch=args.net, eval_dir=os.path.join(args.checkpoint_folder, 'eval_results'))

    # freeze certain layers (if requested)
    base_net_lr = args.base_net_lr if args.base_net_lr is not None else args.lr
    extra_layers_lr = args.extra_layers_lr if args.extra_layers_lr is not None else args.lr

    if args.freeze_base_net:
        logging.info("Freeze base net.")
        freeze_net_layers(net.base_net)
        params = itertools.chain(net.source_layer_add_ons.parameters(), net.extras.parameters(),
                                 net.regression_headers.parameters(), net.classification_headers.parameters())
        params = [
            {'params': itertools.chain(
                net.source_layer_add_ons.parameters(),
                net.extras.parameters()
            ), 'lr': extra_layers_lr},
            {'params': itertools.chain(
                net.regression_headers.parameters(),
                net.classification_headers.parameters()
            )}
        ]
    elif args.freeze_net:
        freeze_net_layers(net.base_net)
        freeze_net_layers(net.source_layer_add_ons)
        freeze_net_layers(net.extras)
        params = itertools.chain(net.regression_headers.parameters(), net.classification_headers.parameters())
        logging.info("Freeze all the layers except prediction heads.")
    else:
        params = [
            {'params': net.base_net.parameters(), 'lr': base_net_lr},
            {'params': itertools.chain(
                net.source_layer_add_ons.parameters(),
                net.extras.parameters()
            ), 'lr': extra_layers_lr},
            {'params': itertools.chain(
                net.regression_headers.parameters(),
                net.classification_headers.parameters()
            )}
```

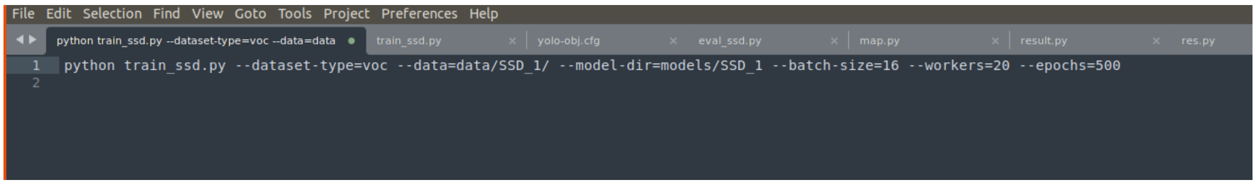

File Edit Selection Find View Goto Tools Project Preferences Help
train_ssd.py
yolo-obj.cfg
python train_ssd.py --dataset-type=voc --data=data
eval_ssd.py

```
                    net.regression_headers.parameters(),
                    net.classification_headers.parameters()
            )}
        ]

    # load a previous model checkpoint (if requested)
    timer.start("Load Model")

    if args.resume:
        logging.info(f"Resume from the model {args.resume}")
        net.load(args.resume)
    elif args.base_net:
        logging.info(f"Init from base net {args.base_net}")
        net.init_from_base_net(args.base_net)
    elif args.pretrained_ssd:
        logging.info(f"Init from pretrained ssd {args.pretrained_ssd}")
        net.init_from_pretrained_ssd(args.pretrained_ssd)

    logging.info(f'Took {timer.end("Load Model"):.2f} seconds to load the model.')

    # move the model to GPU
    net.to(DEVICE)

    # define loss function and optimizer
    criterion = MultiboxLoss(config.priors, iou_threshold=0.5, neg_pos_ratio=3,
                             center_variance=0.1, size_variance=0.2, device=DEVICE)

    optimizer = torch.optim.SGD(params, lr=args.lr, momentum=args.momentum,
                                weight_decay=args.weight_decay)

    logging.info(f"Learning rate: {args.lr}, Base net learning rate: {base_net_lr}, "
                 + f"Extra Layers learning rate: {extra_layers_lr}.")

    # set learning rate policy
    if args.scheduler == 'multi-step':
        logging.info("Uses MultiStepLR scheduler.")
        milestones = [int(v.strip()) for v in args.milestones.split(",")]
        scheduler = MultiStepLR(optimizer, milestones=milestones,
                                                     gamma=0.1, last_epoch=last_epoch)
    elif args.scheduler == 'cosine':
        logging.info("Uses CosineAnnealingLR scheduler.")
        scheduler = CosineAnnealingLR(optimizer, args.t_max, last_epoch=last_epoch)
    else:
        logging.fatal(f"Unsupported Scheduler: {args.scheduler}.")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # train for the desired number of epochs
    logging.info(f"Start training from epoch {last_epoch + 1}.")

    for epoch in range(last_epoch + 1, args.num_epochs):
```



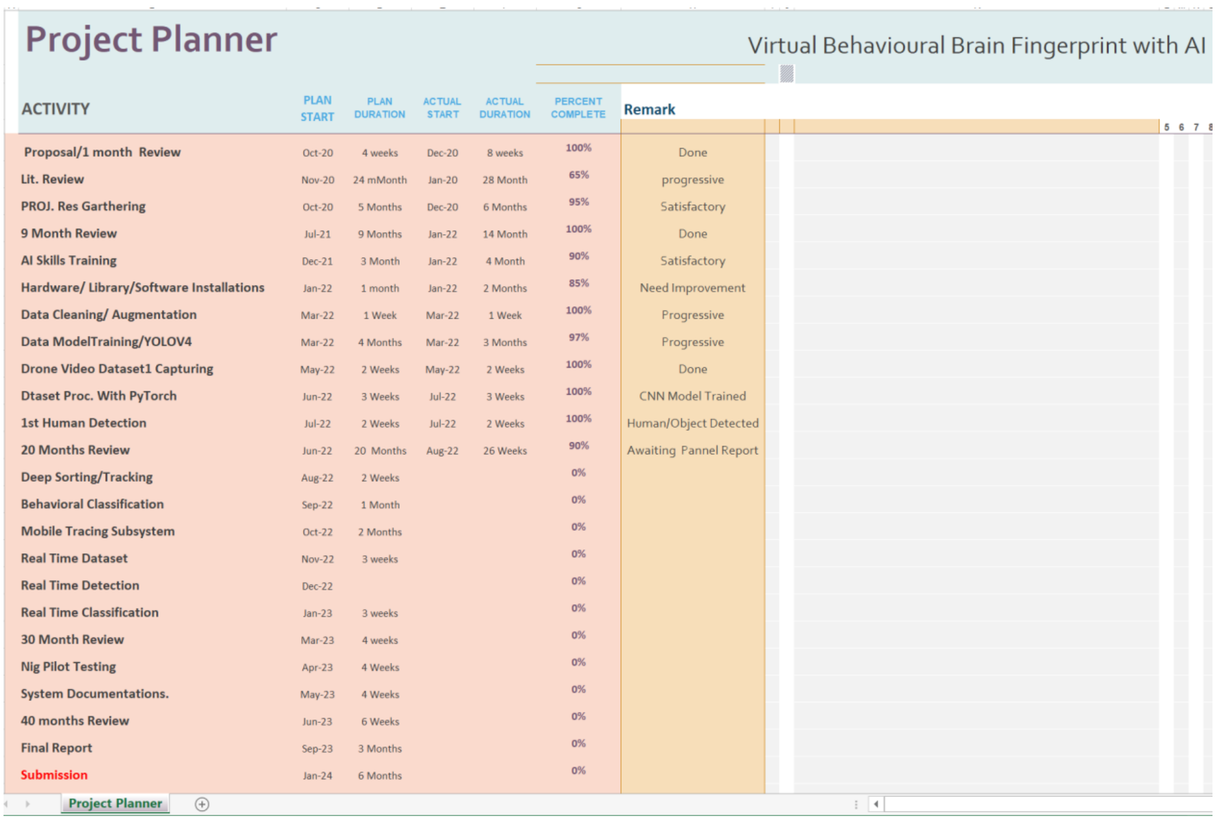
```python
        logging.fatal(f"Unsupported Scheduler: {args.scheduler}.")
        parser.print_help(sys.stderr)
        sys.exit(1)

    # train for the desired number of epochs
    logging.info(f"Start training from epoch {last_epoch + 1}.")

    for epoch in range(last_epoch + 1, args.num_epochs):
        train(train_loader, net, criterion, optimizer, device=DEVICE, debug_steps=args.debug_steps, epoch=epoch)
        scheduler.step()

        if epoch % args.validation_epochs == 0 or epoch == args.num_epochs - 1:
            val_loss, val_regression_loss, val_classification_loss = test(val_loader, net, criterion, DEVICE)

            logging.info(
                f"Epoch: {epoch}, " +
                f"Validation Loss: {val_loss:.4f}, " +
                f"Validation Regression Loss {val_regression_loss:.4f}, " +
                f"Validation Classification Loss: {val_classification_loss:.4f}"
            )

            tensorboard.add_scalar('Loss/val', val_loss, epoch)
            tensorboard.add_scalar('Regression Loss/val', val_regression_loss, epoch)
            tensorboard.add_scalar('Classification Loss/val', val_classification_loss, epoch)

            if args.validation_mean_ap:
                mean_ap, class_ap = eval.compute()
                eval.log_results(mean_ap, class_ap, f"Epoch: {epoch}, ")

                tensorboard.add_scalar('Mean Average Precision/val', mean_ap, epoch)

                for i in range(len(class_ap)):
                    tensorboard.add_scalar(f"Class Average Precision/{eval_dataset.class_names[i+1]}", class_ap[i], epoch)

            model_path = os.path.join(args.checkpoint_folder, f"{args.net}-Epoch-{epoch}-Loss-{val_loss}.pth")
            net.save(model_path)
            logging.info(f"Saved model {model_path}")

    logging.info("Task done, exiting program.")
    tensorboard.close()
```

Appendix D: Single Shot Training Command



```
python train_ssd.py --dataset-type=voc --data=data/SSD_1/ --model-dir=models/SSD_1 --batch-size=16 --workers=20 --epochs=500
```

Appendix E: Project Plan

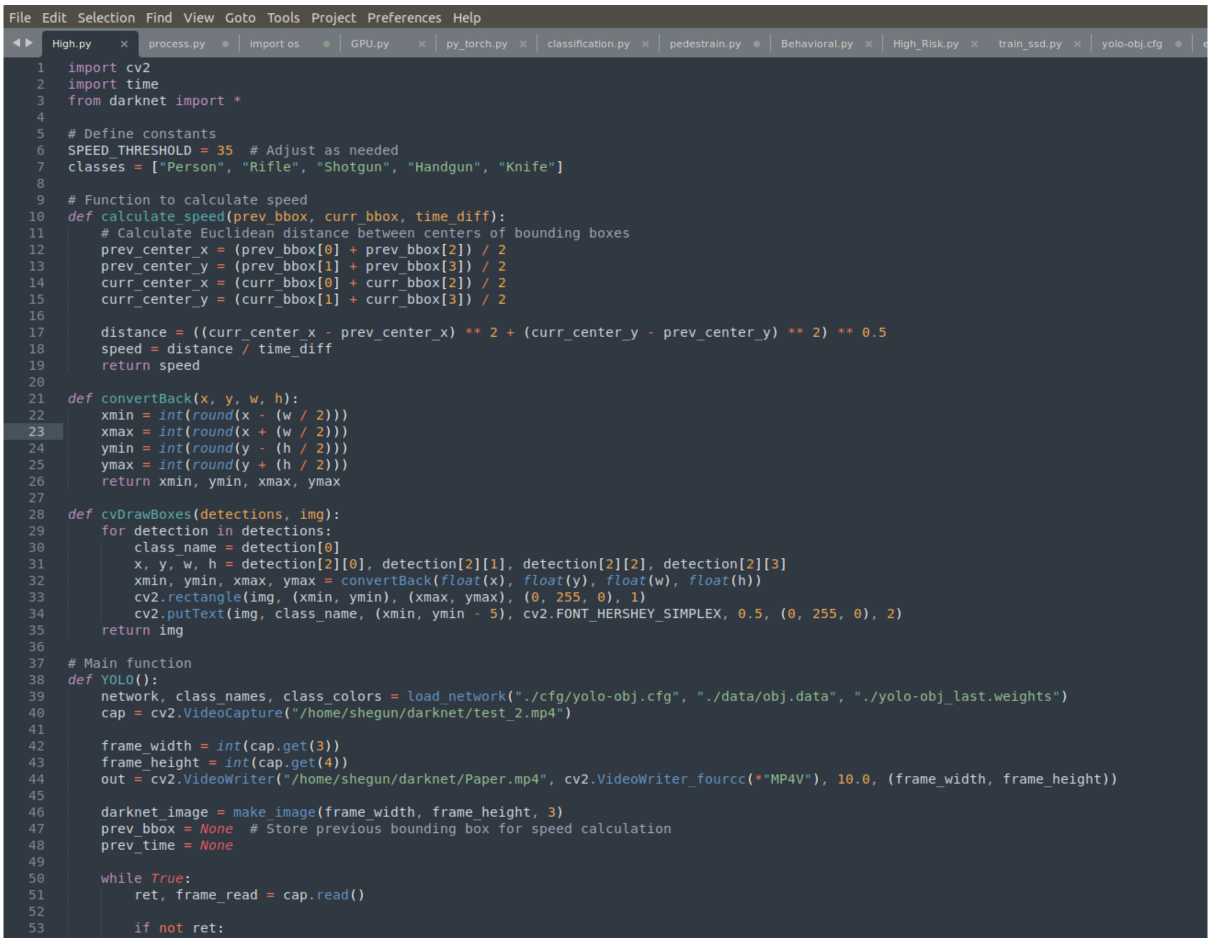


Project Planner

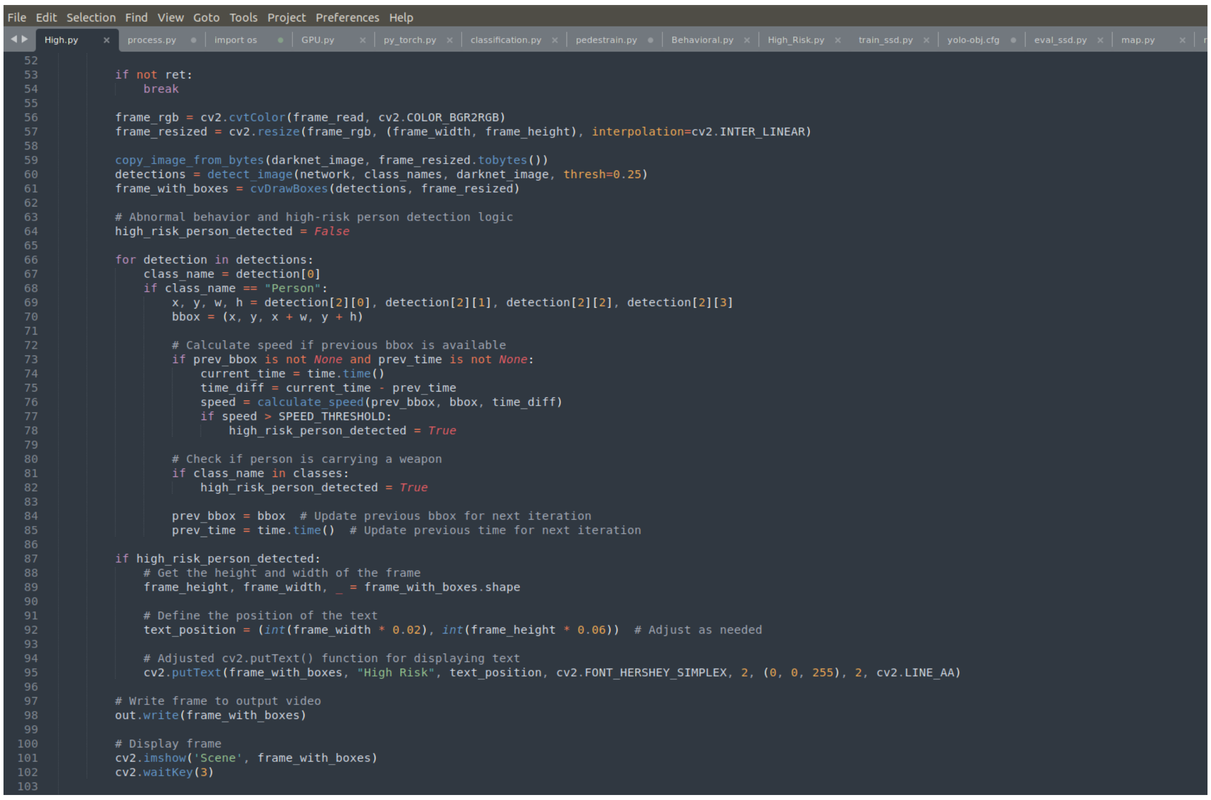
Virtual Behavioural Brain Fingerprint with AI

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE	Remark
Proposal/1 month Review	Oct-20	4 weeks	Dec-20	8 weeks	100%	Done
Lit. Review	Nov-20	24 mMonth	Jan-20	28 Month	65%	progressive
PROJ. Res Garthering	Oct-20	5 Months	Dec-20	6 Months	95%	Satisfactory
9 Month Review	Jul-21	9 Months	Jan-22	14 Month	100%	Done
AI Skills Training	Dec-21	3 Month	Jan-22	4 Month	90%	Satisfactory
Hardware/ Library/Software Installations	Jan-22	1 month	Jan-22	2 Months	85%	Need Improvement
Data Cleaning/ Augmentation	Mar-22	1 Week	Mar-22	1 Week	100%	Progressive
Data ModelTraining/YOLOV4	Mar-22	4 Months	Mar-22	3 Months	97%	Progressive
Drone Video Dataset1 Capturing	May-22	2 Weeks	May-22	2 Weeks	100%	Done
Dtaset Proc. With PyTorch	Jun-22	3 Weeks	Jul-22	3 Weeks	100%	CNN Model Trained
1st Human Detection	Jul-22	2 Weeks	Jul-22	2 Weeks	100%	Human/Object Detected
20 Months Review	Jun-22	20 Months	Aug-22	26 Weeks	90%	Awaiting Pannel Report
Deep Sorting/Tracking	Aug-22	2 Weeks			0%	
Behavioral Classification	Sep-22	1 Month			0%	
Mobile Tracing Subsystem	Oct-22	2 Months			0%	
Real Time Dataset	Nov-22	3 weeks			0%	
Real Time Detection	Dec-22				0%	
Real Time Classification	Jan-23	3 weeks			0%	
30 Month Review	Mar-23	4 weeks			0%	
Nig Pilot Testing	Apr-23	4 Weeks			0%	
System Documentations.	May-23	4 Weeks			0%	
40 months Review	Jun-23	6 Weeks			0%	
Final Report	Sep-23	3 Months			0%	
Submission	Jan-24	6 Months			0%	

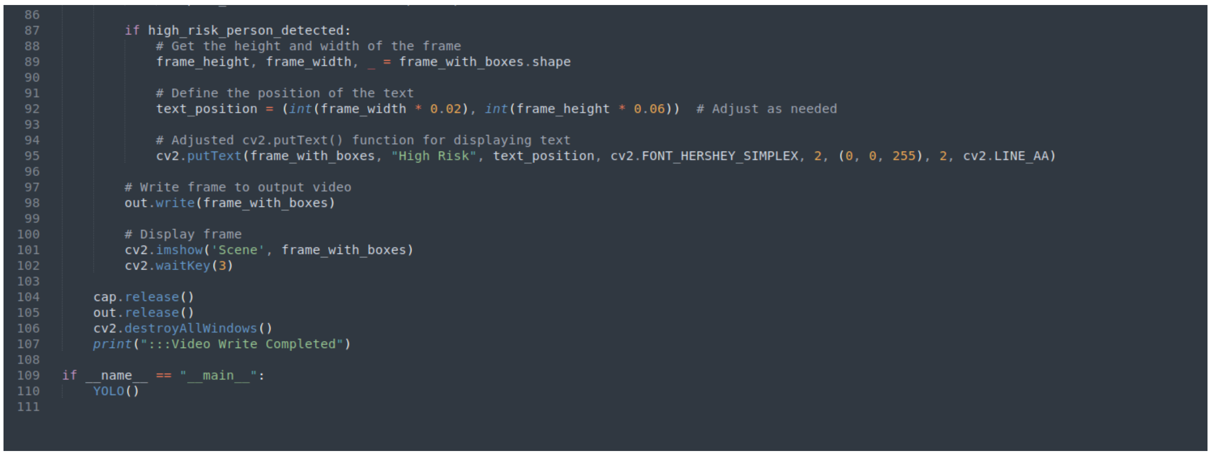
Appendix F: High Risk Python Script



```python
import cv2
import time
from darknet import *

# Define constants
SPEED_THRESHOLD = 35  # Adjust as needed
classes = ["Person", "Rifle", "Shotgun", "Handgun", "Knife"]

# Function to calculate speed
def calculate_speed(prev_bbox, curr_bbox, time_diff):
    # Calculate Euclidean distance between centers of bounding boxes
    prev_center_x = (prev_bbox[0] + prev_bbox[2]) / 2
    prev_center_y = (prev_bbox[1] + prev_bbox[3]) / 2
    curr_center_x = (curr_bbox[0] + curr_bbox[2]) / 2
    curr_center_y = (curr_bbox[1] + curr_bbox[3]) / 2

    distance = ((curr_center_x - prev_center_x) ** 2 + (curr_center_y - prev_center_y) ** 2) ** 0.5
    speed = distance / time_diff
    return speed

def convertBack(x, y, w, h):
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))
    return xmin, ymin, xmax, ymax

def cvDrawBoxes(detections, img):
    for detection in detections:
        class_name = detection[0]
        x, y, w, h = detection[2][0], detection[2][1], detection[2][2], detection[2][3]
        xmin, ymin, xmax, ymax = convertBack(float(x), float(y), float(w), float(h))
        cv2.rectangle(img, (xmin, ymin), (xmax, ymax), (0, 255, 0), 1)
        cv2.putText(img, class_name, (xmin, ymin - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    return img

# Main function
def YOLO():
    network, class_names, class_colors = load_network("./cfg/yolo-obj.cfg", "./data/obj.data", "./yolo-obj_last.weights")
    cap = cv2.VideoCapture("/home/shegun/darknet/test_2.mp4")

    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    out = cv2.VideoWriter("/home/shegun/darknet/Paper.mp4", cv2.VideoWriter_fourcc(*"MP4V"), 10.0, (frame_width, frame_height))

    darknet_image = make_image(frame_width, frame_height, 3)
    prev_bbox = None  # Store previous bounding box for speed calculation
    prev_time = None

    while True:
        ret, frame_read = cap.read()

        if not ret:
```

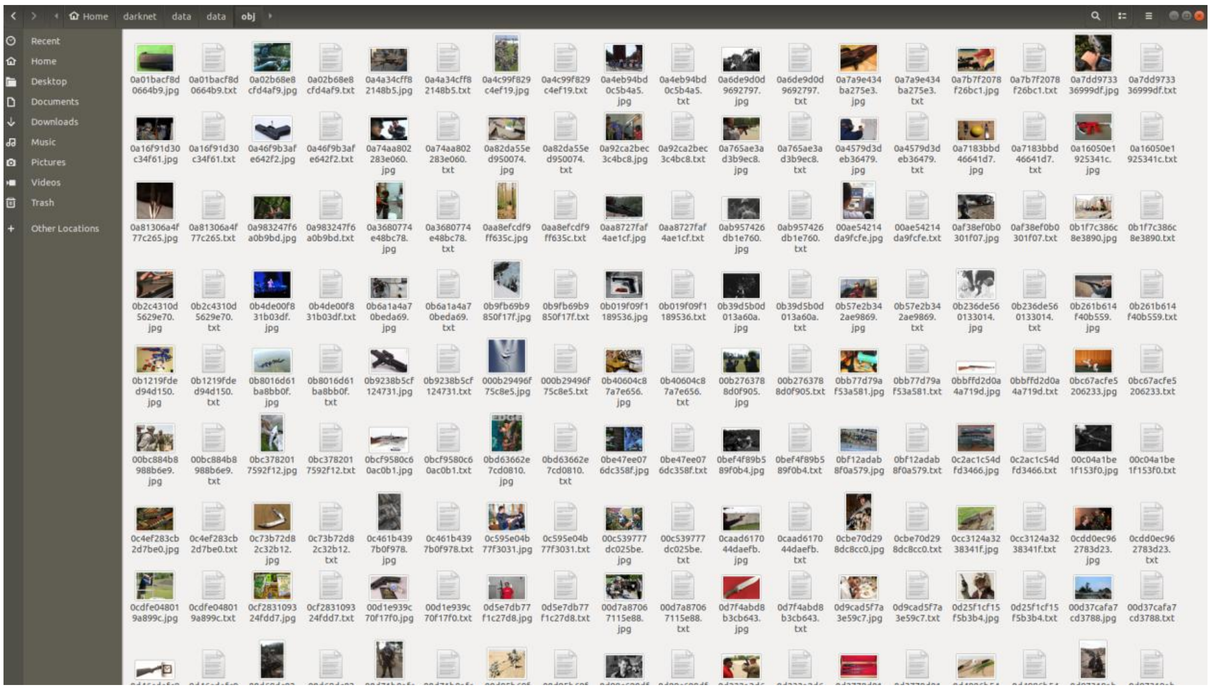


```python

        if not ret:
            break

        frame_rgb = cv2.cvtColor(frame_read, cv2.COLOR_BGR2RGB)
        frame_resized = cv2.resize(frame_rgb, (frame_width, frame_height), interpolation=cv2.INTER_LINEAR)

        copy_image_from_bytes(darknet_image, frame_resized.tobytes())
        detections = detect_image(network, class_names, darknet_image, thresh=0.25)
        frame_with_boxes = cvDrawBoxes(detections, frame_resized)

        # Abnormal behavior and high-risk person detection logic
        high_risk_person_detected = False

        for detection in detections:
            class_name = detection[0]
            if class_name == "Person":
                x, y, w, h = detection[2][0], detection[2][1], detection[2][2], detection[2][3]
                bbox = (x, y, x + w, y + h)

                # Calculate speed if previous bbox is available
                if prev_bbox is not None and prev_time is not None:
                    current_time = time.time()
                    time_diff = current_time - prev_time
                    speed = calculate_speed(prev_bbox, bbox, time_diff)
                    if speed > SPEED_THRESHOLD:
                        high_risk_person_detected = True

                # Check if person is carrying a weapon
                if class_name in classes:
                    high_risk_person_detected = True

                prev_bbox = bbox  # Update previous bbox for next iteration
                prev_time = time.time()  # Update previous time for next iteration

        if high_risk_person_detected:
            # Get the height and width of the frame
            frame_height, frame_width, _ = frame_with_boxes.shape

            # Define the position of the text
            text_position = (int(frame_width * 0.02), int(frame_height * 0.06))  # Adjust as needed

            # Adjusted cv2.putText() function for displaying text
            cv2.putText(frame_with_boxes, "High Risk", text_position, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)

        # Write frame to output video
        out.write(frame_with_boxes)

        # Display frame
        cv2.imshow('Scene', frame_with_boxes)
        cv2.waitKey(3)

```

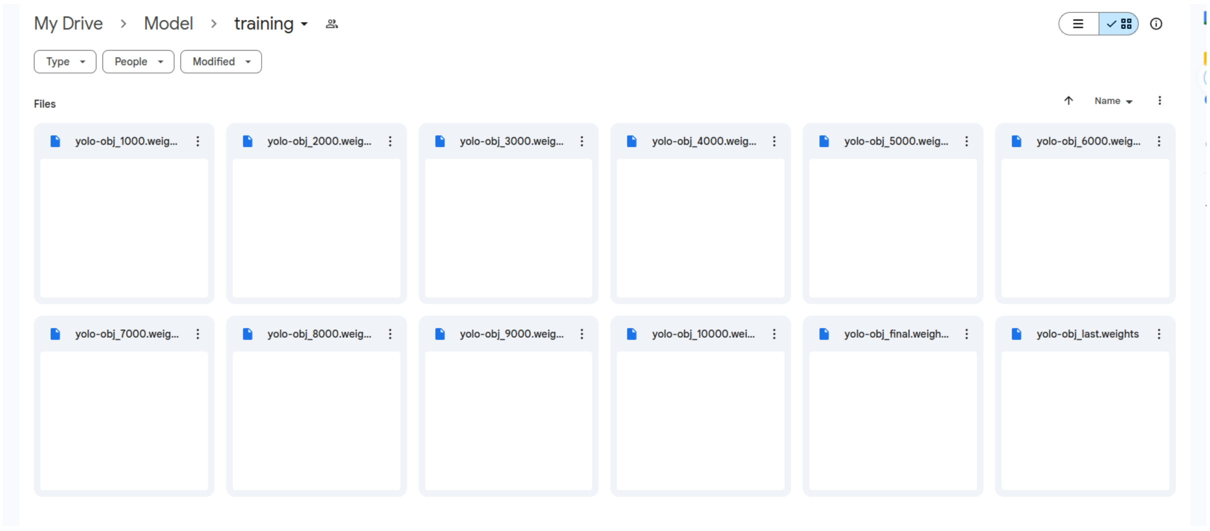


```
        if high_risk_person_detected:
            # Get the height and width of the frame
            frame_height, frame_width, _ = frame_with_boxes.shape

            # Define the position of the text
            text_position = (int(frame_width * 0.02), int(frame_height * 0.06))  # Adjust as needed

            # Adjusted cv2.putText() function for displaying text
            cv2.putText(frame_with_boxes, "High Risk", text_position, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)

        # Write frame to output video
        out.write(frame_with_boxes)

        # Display frame
        cv2.imshow('Scene', frame_with_boxes)
        cv2.waitKey(3)

    cap.release()
    out.release()
    cv2.destroyAllWindows()
    print(":::Video Write Completed")

if __name__ == "__main__":
    YOLO()
```

Appendix G: Screenshots of sample training datasets

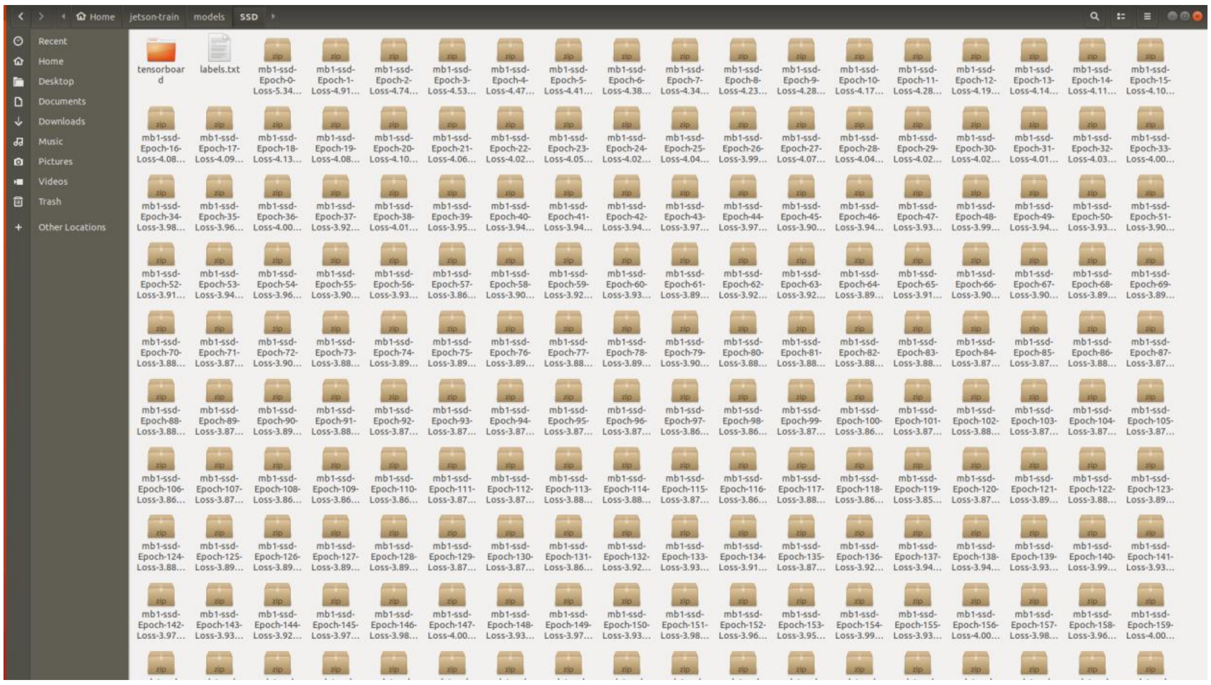


Appendix H: Training Weight Yolo Weights



```
yolo-obj_1000.weig...   yolo-obj_2000.weig...   yolo-obj_3000.weig...   yolo-obj_4000.weig...   yolo-obj_5000.weig...   yolo-obj_6000.weig...
yolo-obj_7000.weig...   yolo-obj_8000.weig...   yolo-obj_9000.weig...   yolo-obj_10000.wei...   yolo-obj_final.weigh...   yolo-obj_last.weights
```

Appendix I: Screenshots of sample training datasets



```
tensorboard
labels.txt
mb1-ssd-Epoch-0-Loss-5.34...
mb1-ssd-Epoch-1-Loss-4.91...
mb1-ssd-Epoch-2-Loss-4.74...
mb1-ssd-Epoch-3-Loss-4.53...
mb1-ssd-Epoch-4-Loss-4.47...
mb1-ssd-Epoch-5-Loss-4.41...
mb1-ssd-Epoch-6-Loss-4.38...
mb1-ssd-Epoch-7-Loss-4.34...
mb1-ssd-Epoch-8-Loss-4.23...
mb1-ssd-Epoch-9-Loss-4.28...
mb1-ssd-Epoch-10-Loss-4.17...
mb1-ssd-Epoch-11-Loss-4.28...
mb1-ssd-Epoch-12-Loss-4.19...
mb1-ssd-Epoch-13-Loss-4.14...
mb1-ssd-Epoch-14-Loss-4.11...
mb1-ssd-Epoch-15-Loss-4.10...
mb1-ssd-Epoch-16-Loss-4.08...
mb1-ssd-Epoch-17-Loss-4.09...
mb1-ssd-Epoch-18-Loss-4.13...
mb1-ssd-Epoch-19-Loss-4.08...
mb1-ssd-Epoch-20-Loss-4.10...
mb1-ssd-Epoch-21-Loss-4.06...
mb1-ssd-Epoch-22-Loss-4.02...
mb1-ssd-Epoch-23-Loss-4.05...
mb1-ssd-Epoch-24-Loss-4.02...
mb1-ssd-Epoch-25-Loss-4.04...
mb1-ssd-Epoch-26-Loss-3.99...
mb1-ssd-Epoch-27-Loss-4.07...
mb1-ssd-Epoch-28-Loss-4.04...
mb1-ssd-Epoch-29-Loss-4.02...
mb1-ssd-Epoch-30-Loss-4.02...
mb1-ssd-Epoch-31-Loss-4.01...
mb1-ssd-Epoch-32-Loss-4.03...
mb1-ssd-Epoch-33-Loss-4.00...
mb1-ssd-Epoch-34-Loss-3.98...
mb1-ssd-Epoch-35-Loss-3.96...
mb1-ssd-Epoch-36-Loss-4.00...
mb1-ssd-Epoch-37-Loss-3.92...
mb1-ssd-Epoch-38-Loss-4.01...
mb1-ssd-Epoch-39-Loss-3.95...
mb1-ssd-Epoch-40-Loss-3.94...
mb1-ssd-Epoch-41-Loss-3.94...
mb1-ssd-Epoch-42-Loss-3.94...
mb1-ssd-Epoch-43-Loss-3.97...
mb1-ssd-Epoch-44-Loss-3.97...
mb1-ssd-Epoch-45-Loss-3.90...
mb1-ssd-Epoch-46-Loss-3.94...
mb1-ssd-Epoch-47-Loss-3.93...
mb1-ssd-Epoch-48-Loss-3.99...
mb1-ssd-Epoch-49-Loss-3.94...
mb1-ssd-Epoch-50-Loss-3.93...
mb1-ssd-Epoch-51-Loss-3.90...
mb1-ssd-Epoch-52-Loss-3.91...
mb1-ssd-Epoch-53-Loss-3.94...
mb1-ssd-Epoch-54-Loss-3.96...
mb1-ssd-Epoch-55-Loss-3.90...
mb1-ssd-Epoch-56-Loss-3.93...
mb1-ssd-Epoch-57-Loss-3.86...
mb1-ssd-Epoch-58-Loss-3.90...
mb1-ssd-Epoch-59-Loss-3.92...
mb1-ssd-Epoch-60-Loss-3.93...
mb1-ssd-Epoch-61-Loss-3.89...
mb1-ssd-Epoch-62-Loss-3.92...
mb1-ssd-Epoch-63-Loss-3.92...
mb1-ssd-Epoch-64-Loss-3.89...
mb1-ssd-Epoch-65-Loss-3.91...
mb1-ssd-Epoch-66-Loss-3.90...
mb1-ssd-Epoch-67-Loss-3.90...
mb1-ssd-Epoch-68-Loss-3.89...
mb1-ssd-Epoch-69-Loss-3.89...
mb1-ssd-Epoch-70-Loss-3.88...
mb1-ssd-Epoch-71-Loss-3.87...
mb1-ssd-Epoch-72-Loss-3.90...
mb1-ssd-Epoch-73-Loss-3.88...
mb1-ssd-Epoch-74-Loss-3.89...
mb1-ssd-Epoch-75-Loss-3.89...
mb1-ssd-Epoch-76-Loss-3.89...
mb1-ssd-Epoch-77-Loss-3.88...
mb1-ssd-Epoch-78-Loss-3.89...
mb1-ssd-Epoch-79-Loss-3.90...
mb1-ssd-Epoch-80-Loss-3.88...
mb1-ssd-Epoch-81-Loss-3.88...
mb1-ssd-Epoch-82-Loss-3.88...
mb1-ssd-Epoch-83-Loss-3.88...
mb1-ssd-Epoch-84-Loss-3.87...
mb1-ssd-Epoch-85-Loss-3.87...
mb1-ssd-Epoch-86-Loss-3.88...
mb1-ssd-Epoch-87-Loss-3.87...
mb1-ssd-Epoch-88-Loss-3.88...
mb1-ssd-Epoch-89-Loss-3.87...
mb1-ssd-Epoch-90-Loss-3.89...
mb1-ssd-Epoch-91-Loss-3.88...
mb1-ssd-Epoch-92-Loss-3.87...
mb1-ssd-Epoch-93-Loss-3.87...
mb1-ssd-Epoch-94-Loss-3.87...
mb1-ssd-Epoch-95-Loss-3.87...
mb1-ssd-Epoch-96-Loss-3.87...
mb1-ssd-Epoch-97-Loss-3.86...
mb1-ssd-Epoch-98-Loss-3.86...
mb1-ssd-Epoch-99-Loss-3.87...
mb1-ssd-Epoch-100-Loss-3.86...
mb1-ssd-Epoch-101-Loss-3.87...
mb1-ssd-Epoch-102-Loss-3.88...
mb1-ssd-Epoch-103-Loss-3.87...
mb1-ssd-Epoch-104-Loss-3.87...
mb1-ssd-Epoch-105-Loss-3.87...
mb1-ssd-Epoch-106-Loss-3.86...
mb1-ssd-Epoch-107-Loss-3.87...
mb1-ssd-Epoch-108-Loss-3.86...
mb1-ssd-Epoch-109-Loss-3.86...
mb1-ssd-Epoch-110-Loss-3.86...
mb1-ssd-Epoch-111-Loss-3.87...
mb1-ssd-Epoch-112-Loss-3.87...
mb1-ssd-Epoch-113-Loss-3.88...
mb1-ssd-Epoch-114-Loss-3.88...
mb1-ssd-Epoch-115-Loss-3.87...
mb1-ssd-Epoch-116-Loss-3.86...
mb1-ssd-Epoch-117-Loss-3.88...
mb1-ssd-Epoch-118-Loss-3.86...
mb1-ssd-Epoch-119-Loss-3.85...
mb1-ssd-Epoch-120-Loss-3.87...
mb1-ssd-Epoch-121-Loss-3.89...
mb1-ssd-Epoch-122-Loss-3.88...
mb1-ssd-Epoch-123-Loss-3.89...
mb1-ssd-Epoch-124-Loss-3.88...
mb1-ssd-Epoch-125-Loss-3.89...
mb1-ssd-Epoch-126-Loss-3.89...
mb1-ssd-Epoch-127-Loss-3.89...
mb1-ssd-Epoch-128-Loss-3.89...
mb1-ssd-Epoch-129-Loss-3.87...
mb1-ssd-Epoch-130-Loss-3.87...
mb1-ssd-Epoch-131-Loss-3.86...
mb1-ssd-Epoch-132-Loss-3.92...
mb1-ssd-Epoch-133-Loss-3.93...
mb1-ssd-Epoch-134-Loss-3.91...
mb1-ssd-Epoch-135-Loss-3.87...
mb1-ssd-Epoch-136-Loss-3.92...
mb1-ssd-Epoch-137-Loss-3.94...
mb1-ssd-Epoch-138-Loss-3.94...
mb1-ssd-Epoch-139-Loss-3.93...
mb1-ssd-Epoch-140-Loss-3.99...
mb1-ssd-Epoch-141-Loss-3.93...
mb1-ssd-Epoch-142-Loss-3.97...
mb1-ssd-Epoch-143-Loss-3.93...
mb1-ssd-Epoch-144-Loss-3.92...
mb1-ssd-Epoch-145-Loss-3.97...
mb1-ssd-Epoch-146-Loss-3.98...
mb1-ssd-Epoch-147-Loss-4.00...
mb1-ssd-Epoch-148-Loss-3.93...
mb1-ssd-Epoch-149-Loss-3.97...
mb1-ssd-Epoch-150-Loss-3.93...
mb1-ssd-Epoch-151-Loss-3.98...
mb1-ssd-Epoch-152-Loss-3.96...
mb1-ssd-Epoch-153-Loss-3.95...
mb1-ssd-Epoch-154-Loss-3.99...
mb1-ssd-Epoch-155-Loss-3.93...
mb1-ssd-Epoch-156-Loss-4.00...
mb1-ssd-Epoch-157-Loss-3.98...
mb1-ssd-Epoch-158-Loss-3.96...
mb1-ssd-Epoch-159-Loss-4.00...
```