ANOMALY DETECTION IN SURVEILLANCE VIDEOS

FOO JIA QI

UNIVERSITI TUNKU ABDUL RAHMAN

ANOMALY DETECTION IN SURVEILLANCE VIDEOS

FOO JIA QI

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2025

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Name : Foo Jia Qi

ID No. : 2003619

Date : 05/06/2025

COPYRIGHT STATEMENT

This final year project report is submitted in partial fulfilment of the requirements for the degree of Electrical and Electronic Engineering with Honours at Universiti Tunku Abdul Rahman (UTAR). This final year project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this final year project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to thank everyone who contributed to the successful completion of this project. I would like to express my deepest gratitude to my research supervisor, Ir. Ts. Dr. Tham Mau Luen, for his invaluable advice, guidance and immense patience throughout the development of the research. His expertise and insights were essential in shaping the direction of my work and ensuring its quality.

I am also grateful to my moderator, Ir. Ts. Dr. Chua Sing Yee, for her insightful feedback and suggestions, which greatly improved my thesis.

In addition, I would like to express my appreciation to my loving parents and friends for their encouragement, motivation, and unwavering support during the course of this project..

ABSTRACT

In the present society, video surveillance systems are rapidly evolving with intelligent video analytics to improve public safety. With the increasing installation of surveillance cameras in both public and private spaces, there is a growing reliance on continuous monitoring to ensure public safety. However, human-based monitoring is labour-intensive and inefficient. Video anomaly detection (VAD) plays a vital role in modern surveillance systems by automatically identifying unusual events in video streams. This study focuses on developing a lightweight and efficient VAD framework that supports both binary and multiclass detection. The proposed system, AnomLite combines MobileNetV2, a lightweight Convolutional Neural Network (CNN) for spatial feature extraction, and Long Short-Term Memory (LSTM) for temporal modelling. By leveraging the strengths of MobileNetV2 in extracting efficient spatial features and LSTM in capturing temporal dependencies in video sequences, the model detects anomalous events across various classes. The system trains on two datasets: UCF-Crime, which contains real-world CCTV footage, and XD-Violence, which includes video content from movies and YouTube. Preprocessing steps are employed to ensure the model performs well under varying data conditions. The evaluation of the proposed model shows strong performance on the first dataset, achieving an ROC AUC of 0.99 and an average precision of 0.99 on UCF-Crime. The model demonstrates strong performance on another well-known dataset in video anomaly detection, achieving an ROC AUC of 0.98 and an average precision of 0.97 on XD-Violence. The model also achieves high accuracy of 94% on UCF-Crime and 93% on XD-Violence, with strong F1 scores across both datasets (F1-Micro 0.93 on UCF-Crime, 0.89 on XD-Violence). The model achieves high per-class accuracy across the UCF-Crime dataset, with 10 out of 14 classes exceeding 0.95 accuracy and several classes, such as *Arson, Explosion, Fighting, Shooting,* and *Vandalism,* reaching a perfect accuracy of 1.00, demonstrating the model's strong and consistent performance in detecting diverse types of anomalies. Moreover, the model performs well on the XD-Violence dataset, with accuracies ranging from 0.79 to 0.95. It shows highest accuracy on Car

Accidents (0.95) and strong performance across other classes like Abuse, Riot, and Fighting, indicating its effectiveness in handling diverse anomalies. Additionally, the model is optimized for inference through quantization. With a reduction of around 70% in model size through model compression techniques such as quantization, the flexibility of the model is further improved, particularly for low-end devices. These results highlight how deep learning techniques, such as SMOTE, data augmentation, and advanced loss functions like cross-entropy loss, contribute to high accuracy and effective performance in automating surveillance tasks, even when dealing with highly imbalanced datasets. Data augmentation techniques that simulate real-world conditions enhance the efficiency of anomaly detection systems in practical applications.

Keywords: Video anomaly detection, deep learning, edge computing, artificial intelligence, neural network

Subject Area: TK7885-7895 Computer engineering. Computer hardware

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

μ	mean vector
AP	Average Precision
AUC	Area Under the Curve
BLS	Batch-level Selection
BN	Batch Normalization
CNN	Convolutional neural networks
CLT	Central Limit Theorem
DFM	Divergence from mean
FP	False Positive
FPR	False Positive Rate
FN	False Negative
FPS	Frame Per Second
GCN	Graph convolutional networks
GAN	Generative Adversarial Networks
GL-MHSA	Global and Local Multi-Head Self-Attention
I3D	Inflated 3D
LSTM	Long Short-Term Memory
MHSA	Multi-Head Self-Attention
MIL	Multi-Instance Learning
MIST	Multiple Instance Self-Training
MPP	Mean-based Pull-Push
NLP	Natural language processing
OCC	One-class classification
PR	Precision-Recall
R	Recall
ROC	Receiver Operating Characteristics
SBS	Sample-Batch Selection
SLS	Sample-level Selection
TP	True Positive
TPR	True Positive Rate
TN	True Negative
UVAD	Unsupervised Video Anomaly Detection

VAD	Video Anomaly Detection
ViTs	Vision transformers
WVAD	Weakly Supervised Video Anomaly Detection

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Crime remains a significant issue in society, despite the widespread presence of surveillance camera networks. Recently, surveillance cameras have been extensively used in both public and private areas. However, the monitoring of surveillance is typically carried out by humans, which is a laborious and time-consuming process.

Video anomaly detection (VAD) refers to the task of identifying unusual or suspicious activities in video footage, typically used in surveillance, security, and monitoring systems. With the rapid increase of surveillance cameras, effectively and efficiently monitoring numerous surveillance cameras with human intervention has become challenging, prompting the need for automating surveillance monitoring. The goal of VAD is to automatically identify unusual events that diverge from the typical patterns within a scene, such as theft, accidents, or violent behaviour. This field has gained attention with the growth of deep learning and computer vision technologies.

Anomalous events can be categorized into two types, namely global and local anomalies. Global anomalies refer to events that deviate from the overall scene or context, such as a car driving in the wrong direction in traffic, while local anomalies refer to specific objects or individuals acting out of the ordinary, like a person running in a typically walking area. There are several approaches to VAD, including supervised learning, unsupervised learning, semi-supervised learning, and weakly–supervised learning. However, note that researchers in the field have not focused on supervised learning techniques due to the lack of frame-level annotations for large-scale real-world VAD datasets (Sertis, 2023). Thus, based on the learning method exploited, VAD methods can be separated into two primary categories, including one-class classification approaches and weakly-supervised learning approaches.

1.2 Importance of the Study

To detect and locate abnormal events in videos, VAD has become an essential task in analyzing activities within unedited videos. Even with years of research in Video Anomaly Detection (VAD), developing a model that effectively identifies anomalies in videos is still challenging, as it must differentiate between normal and abnormal events, particularly since anomalies are rare and can vary significantly (Feng, Hong and Zheng, 2021).

Gathering a large-scale dataset with precise temporal annotations of abnormal events is both labor-intensive and time-consuming, posing a challenge to the advancement of VAD. Throughout the years, unsupervised video anomaly detection (UVAD) has attracted significant interest because it can detect anomalies without needing extra annotations. However, these methods are trained solely on normal videos, limiting their ability to understand anomaly data. As a result, they often produce high false alarm rates for new, unseen normal events.

In response to this, the study utilized a more realistic approach – weakly supervised anomaly detection (WVAD) as the first solution. It overcomes the issue of incorrect anomaly detection in videos within an unsupervised framework and strikes a more effective trade-off between detection accuracy and effort required for manual annotations compared to unsupervised method.

1.3 Problem Statement

In such a weakly supervised framework, existing methods for determining abnormality often depend on certain assumptions or opaque models, resulting in less dependable pseudo-temporal annotations. For example, the commonly used feature magnitude approach assumes that abnormal snippets will have a greater feature magnitude compared to normal ones (Zhou et al., 2024). However, simply focusing on large feature magnitudes does not always ensure effective differentiation of abnormal snippets.

The second challenge arises from the constraints of the previous sample-level selection method. The abnormality ratio refers to the proportion of abnormal snippets within each video. Previous approaches tend to identify the top-k potential abnormal snippets within each video without accounting for the varying abnormality ratios across different videos. By uniformly selecting

potential abnormal snippets, these methods might overlook significant abnormalities in videos with higher abnormality ratios, thereby missing valuable guidance for accurate anomaly detection (Zhou et al., 2024).

In WVAD, errors in selecting abnormal snippets are unavoidable, which introduces noise into the pseudo-temporal labels. Although Multi-Instance Learning (MIL) methods are used, the anomaly classifier continues to be affected by this label noise, struggling with the challenge of distinguishing genuinely abnormal snippets from those that are incorrectly labelled.

Moreover, most models only handle binary classification instead of distinguishing specific anomaly types, struggle to perform well on both binary and multiclass tasks, suffer from class imbalance due to the rarity of anomalies, and lack scalability for processing large volumes of video data efficiently. To address these challenges, there is a need for a model with improved criteria for abnormality and a more effective selection strategy.

1.4 Aim and Objectives

The main objectives of this study are as follows:

1. To develop an AI model that detects abnormal events in video streams.
2. To implement the developed AI model in a real-world scenario.
3. To compare the performance of the developed model with conventional approach.

1.5 Scope and Limitation of the Study

This study focuses on Video Anomaly Detection (VAD), specifically targeting the identification of unusual or abnormal events in video sequences. It involves training and evaluating models using selected datasets, such as UCF-Crime or XD-Violence, and employing specific methodologies and performance metrics like Average Precision (AP) and Area Under the Curve (AUC). The study is geared toward applications in surveillance, security, and monitoring by leveraging particular video data and experimental setups to assess the effectiveness of VAD techniques (Sertis, 2023).

1.6 Contribution of the Study

This research contributes to the field of video-based anomaly detection by proposing several notable advancements. Firstly, it presents a deep learning

framework that effectively handles both binary anomaly detection and multiclass classification of violent events. Secondly, the model is trained and evaluated on two diverse benchmark datasets, which are UCF-Crime and XD-Violence, demonstrating improved generalization and robustness across different video domains. Lastly, the study applies model optimization techniques, including quantization, to significantly reduce computational overhead, enabling real-time inference suitable for deployment on edge devices. These efforts collectively enhance both the accuracy and practicality of automated video anomaly detection systems.

1.7 Outline of the Report

The report covers 5 chapters, in which it first covers the introduction. This chapter provides an overview of the research project in video anomaly detection in surveillance videos, which includes an introduction, a problem statement, an aim, objectives, and the scope and limitations of the research. Chapter 2 covers the literature review of several papers related to the project, mainly on human action recognition and abnormal activity detection systems. The research approach and methodology are presented in Chapter 3, which outlines the system implementation process, the criteria considered, and the timeline for conducting the study. Chapter 4 covers the results and discussions, which provide a comprehensive report of the findings, incorporating both qualitative and quantitative results, along with in-depth analysis and interpretation. Finally, Chapter 5 covers the conclusion and recommendations, which offer a summary of the overall study and propose suggestions for future enhancements.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Although the conventional methods of VAD have been widely investigated, the quick development of deep learning has introduced new possibilities for more efficient anomaly detection. Several methods, such as convolutional neural networks (CNNs) and vision transformers (ViTs), have been shown to be highly proficient at identifying complex data relationships in large datasets. These developments have significantly improved VAD performance, making anomaly detection in video streams more accurate and consistent.

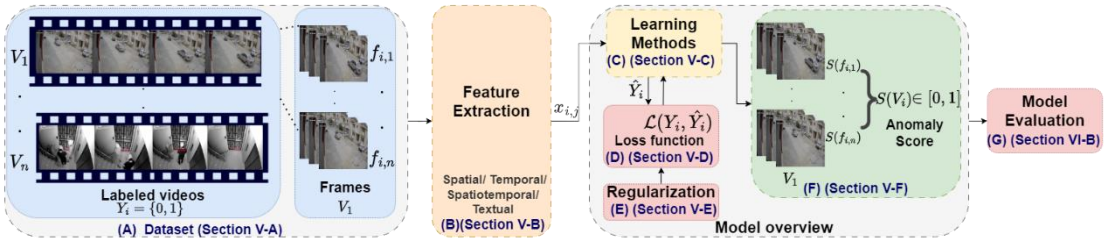


Figure 2.1: Video Anomaly Detection paradigm (Zhou et al., 2024)

Figure 2.1 shows the typical video anomaly detection paradigm by Zhou et al. 2024, which can be categorized into as following:

(A) Development and selection of state-of-the-art datasets

(B) Extraction of spatial, temporal, spatio-temporal, and textual deep features

(C) Deep learning and supervision approaches

(D) Choice of loss functions

(E) Incorporation of regularization techniques within loss functions

(F) Calculation of anomaly scores

(G) Techniques for model evaluation

This literature review explores the process, evolution of VAD systems, and the potential for enhancing the performance and addressing the limitations of existing VAD models.

2.2 Benchmarking Datasets

Large-scale benchmarking datasets are essential for developing effective deep-learning models. In the realm of VAD, several public datasets are available, typically offering video-level labels (normal or anomalous) for training. These datasets support weakly supervised learning. Frame-level annotations, however, are usually reserved for test sets to allow for detailed model evaluation. This section highlights key public datasets and evaluation metrics used in VAD.

2.2.1 Shanghai Tech

As shown in Figure 2.2, the ShanghaiTech dataset, as introduced by Luo et al. (2017), was gathered under complex lighting conditions and varying camera angles. It comprises 13 real-world scenes, each with several videos. It introduces anomalies resulting from abrupt movements, such as chasing and fighting. Initially created for one-class classification, the dataset includes 270,000 frames of normal videos for training and features 130 anomaly events with pixel-level annotations for testing. In a subsequent update, Zhong et al. (2019) revised the dataset's protocol by dividing it into 238 training videos (175 normal and 63 anomalous) and 199 test videos (155 normal and 44 anomalous).

Figure 2.2: Shanghai Tech Datasets (Luo, Liu and Gao, 2017)

	Training Set	Test Set	Total
Normal Videos	175	155	330
Anomaly Videos	63	44	107
Total	238	199	437

Figure 2.3: Reorganization of Shanghai Tech (Zhong et al., 2019)

2.2.2 UCF-Crime

The UCF-Crime dataset, developed by Sultani et al. (2018), is a significant early dataset for video anomaly detection (VAD) that captures a variety of realistic anomalies. It features 13 distinct anomaly types, such as 'abuse', 'arrest', and 'robbery ', among others. The dataset includes 1,900 untrimmed surveillance videos, with a total of 128 hours of footage, with an average of 7,247 frames per video. It is split into a training set with 1,610 videos (800 normal and 810 anomalous) and a test set with 290 videos (150 normal and 140 anomalous). The training set is annotated with video-level labels, while the test set provides frame-level annotations.

Figure 2.4: UCF-Crime Datasets (Sultani, Chen and Shah, 2018)

2.2.3 XD-Violence

Unlike other datasets, XD-Violence (Wu et al., 2020) offers a substantial collection of 4,754 untrimmed videos, complete with audio, enabling models to utilize multimodal data for detecting anomalies. Figure 2.5 shows the six types of physical violence featured in the datasets, such as abuse, car accident, and others, spanning a total of 217 hours. It is split into a training set of 3,954 videos (2,049 normal and 1,905 anomalous) with video-level annotations, and a test set of 800 videos (300 normal and 500 anomalous) with frame-level annotations. Each anomalous video includes 1 to 3 instances of abnormal events.

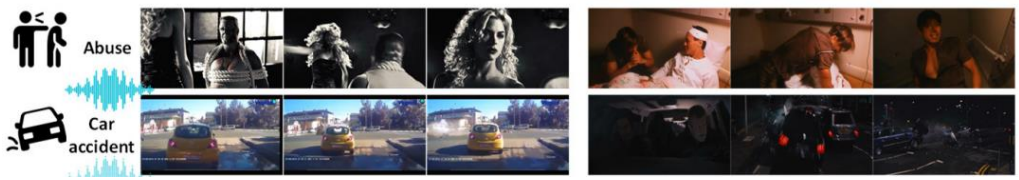


Figure 2.5: Sample videos from the XD-Violence dataset (Wu et al., 2020)

2.3 Deep Feature Extractors

Different feature extractors have been utilized by previous researchers, including convolutional neural networks (CNNs), autoencoders, generative adversarial Networks (GANs), and others.

2.3.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized type of deep learning model widely used for image recognition tasks. They come in various forms: one-dimensional (1D), two-dimensional (2D), and three-dimensional (3D). Among these, 2D CNNs are the most applied for image recognition (Raziyeh Pourdarbani et al., 2023). 1D CNNs are primarily used for analyzing text and sequential signals. In 2D CNNs, a convolutional filter moves across the input both vertically and horizontally to perform the convolution process, where the filter weights are applied to the input data, and a bias is added. On the other hand, 3D CNNs use three-dimensional filters and are suited for processing 3D data, such as MRI and CT scans, as well as hyperspectral images (HSIs), which have two spatial dimensions and one spectral dimension. Figure 2.6 shows an overview of CNN.

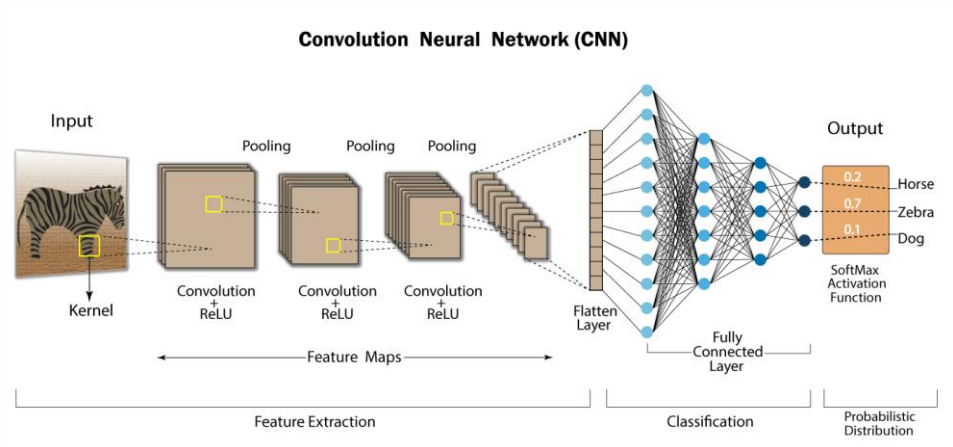


Figure 2.6: An Overview of Convolutional Neural Network (Kalita, 2022)

2.3.2 2D Convolutional Neural Networks (2D CNNs)

CNNs have transformed the processing of spatial features, allowing for in-depth analysis of scene structures. According to Mansour et al. (2021), Faster R-CNN, a CNN architecture, is highlighted for its precision and ability to perform both object classification and bounding box regression simultaneously. This dual function enables accurate object detection and classification within video frames, making it essential for identifying and locating anomalies.

2.3.3 3D Convolutional Neural Networks (3D CNNs)

These networks extend traditional CNNs by integrating temporal analysis, enhancing the assessment of spatiotemporal features in video data. Models like C3D and I3D have significantly boosted performance in cutting-edge systems. Numerous studies utilize 3D CNNs as core components, showcasing their exceptional capability in spatiotemporal feature extraction (Zhou et al., 2024).

2.3.4 Comparison between 2D and 3D CNNs

In summary, 2D CNNs are designed to handle two-dimensional inputs, which makes them ideal for applications such as image classification and object detection, but they lack the ability to capture temporal relationships. In contrast, 3D CNNs handle spatiotemporal data by processing both spatial and temporal dimensions, making them ideal for video analysis and motion detection. While 3D CNNs offer better performance in tasks requiring time-based analysis, they are more computationally intensive compared to 2D CNNs, which are faster and more efficient for simpler spatial tasks. Each model excels in its respective domain, depending on the complexity of the data.

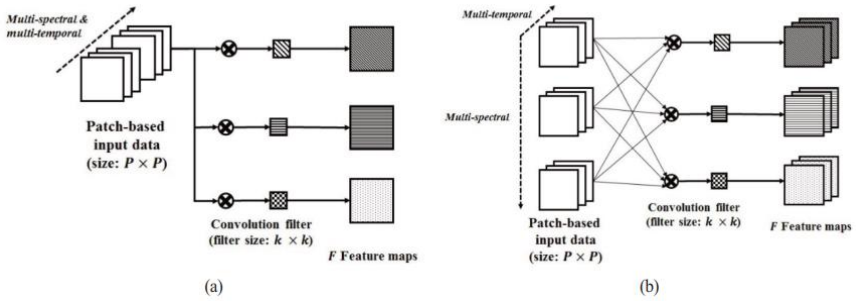


Figure 2.7: Architecture of CNN model. (a) 2D-CNN and (b) 3D-CNN (Kim et al., 2018)

2.3.5 Transformer

Transformers are a strong model architecture mostly utilized in natural language processing (NLP). Figure 2.8 shows the model architecture of a Transformer. The attention mechanism, which allows the model to focus on various parts of the input sequence while producing output, is the core of transformers. This helps the model to determine how important each component of the input is in relation to the others, which is especially useful for understanding long-range dependencies. Transformers are known for their self-attention function, which aids the model in understanding the connections between various words in a sentence. For instance, self-attention enables the model to make the connection between "cat" and "mat," even if they are not contiguous, in the sentence "The cat sat on the mat." This process is essential for understanding meaning and context within sequences (Allard, 2020).

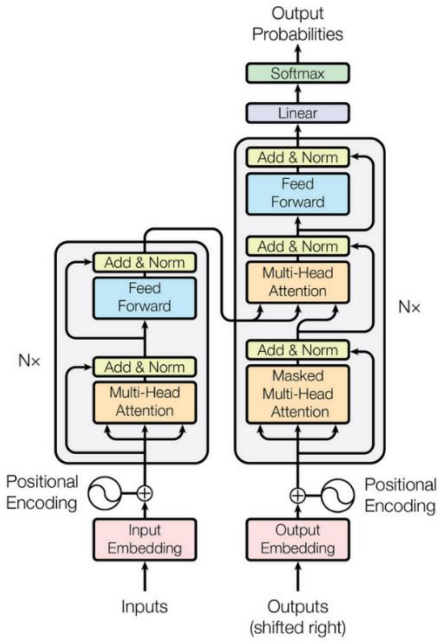


Figure 2.8: Model Architecture of a Transformer (Vaswani et al., 2017)

Transformers use an encoder-decoder design to process and generate sequences. The encoder creates representations from the input, which the decoder then uses to produce the output. They utilize multiple layers of feedforward neural networks and self-attention to recognize complex patterns. Multi-head attention improves performance by focusing on different aspects of the sequence simultaneously, while positional encoding helps the model understand the order of tokens. Transformers are scalable and efficient, enabling

parallel training and demonstrating strong capabilities in areas like machine translation and generating coherent text due to their strong contextual understanding (Allard, 2020).

2.3.6 Graph Convolutional Networks (GCN)

Graph Convolutional Networks (GCNs) extend the concept of CNNs to graph-structured data. Unlike images, which have a grid-like structure, graphs have nodes (points) connected by edges (lines), and their relationships are less regular. GCNs work by aggregating information from a node's neighbours to update the node's representation, allowing the model to learn from the graph's structure (Kwok, 2022). As shown in Figure 2.9, just like in a CNN for images, where a filter slides over each pixel and combines the values of neighbouring pixels to generate the next layer's output, a GCN operates similarly. Instead of pixels, a filter in a GCN moves across each node in a graph, aggregating the values of neighbouring nodes to produce the output for the next layer, as shown in Figure 2.10. This way, GCNs can handle data where relationships are complex and irregular, making them useful for tasks like social network analysis or molecular chemistry.

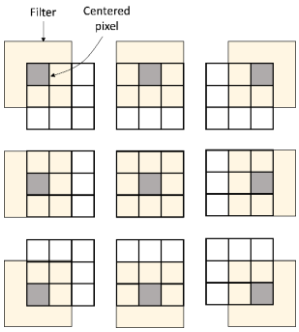


Figure 2.9: Filter passing over each pixel in CNN (Matthew N. Bernstein, 2023)

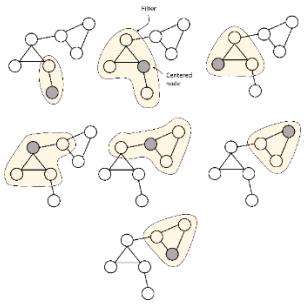


Figure 2.10: Filter passing over each node in GCN (Matthew N. Bernstein, 2023)

2.4 Overview of Current Approaches on Video Anomaly Detection

While most deep learning – based VAD systems have traditionally relied on supervised learning models, recent advancements in video anomaly detection (VAD) have shifted towards exploring weakly supervised, self-supervised, and unsupervised methods as alternatives to traditional supervised approaches. These methods address challenges like the need for fully annotated datasets and capturing complex patterns.

2.4.1 Self-Supervised Approach

Self-supervised approaches generate supervision from the input data, eliminating the need for human-labelled data, which is valuable for anomaly detection where labelled anomalies are rare. In one study, Georgescu et al. (2021) introduced a self-supervised method using multi-task learning at the object level was introduced. The model trains a 3D CNN on tasks such as predicting object movement direction, detecting motion irregularities, and reconstructing object appearances from adjacent frames. By learning normal object behavior from video data, the model becomes capable of identifying anomalies based on deviations from this learned behavior, even without explicit labels (Georgescu et al., 2021).

2.4.2 Unsupervised Approach

Early VAD methods often relied on one–class classification (OCCs, also known as unsupervised anomaly detection), where models were trained solely on normal video data. These models aimed to capture normal feature patterns using either hand-crafted features or deep autoencoder models. Once trained, the models could reconstruct normal input videos with minimal error. During testing, if the reconstruction error exceeded a certain threshold, the input was flagged as anomalous, as it likely differed significantly from the normal training data. However, these methods often failed to generalize well to test datasets because the models were never trained with anomalous examples (Sertis, 2023).

2.4.3 Weakly - Supervised Approach

Weakly supervised learning refers to a set of techniques in machine learning designed to develop predictive models using limited or imprecise supervision. It involves incorporating domain-specific knowledge and applying functions to

generate labels from imperfect or automatically derived training data (Kanjilal, 2022).

These methods become particularly useful when working with data that does not fully align with the model's expected input format or structure. In practice, much of the data available is unstructured or poorly labelled, which makes traditional supervised learning less feasible (Kanjilal, 2022). Weak supervision offers a practical solution by enabling the use of such data for training purposes, even when the annotations are unreliable or incomplete.

Weakly supervised learning enables model training from datasets that are labelled through indirect or noisy processes rather than manual annotation. It spans various strategies that rely on approximate, partial, or less accurate information, allowing for large-scale data utilization with significantly reduced labeling effort.

While one-class classification-based VAD trains only on normal videos, weakly-supervised VAD uses both normal and anomalous videos, but without frame-level labels. Instead, video-level labels indicate if a video contains anomalies, without specifying when they occur. This requires methods to leverage these broader labels to detect anomalies at the frame level. Since video-level labels are easier to obtain, they enable the creation of large-scale datasets for weakly-supervised VAD.

2.4.4 Overview of Current Approaches

Table 2.1: Comparison of Different Supervision Approaches

Aspect	Self-Supervised	Unsupervised	Weakly Supervised
Label Need	No manual labels, supervision from the data itself	Uses only normal data, no anomaly labels	Needs video-level labels (anomalous or normal)
Training Data	Learns patterns via pretext tasks on video data	Learns from normal videos only	Uses both normal and anomalous videos without exact timing info

Detection Basis	Detects deviation from learned normal behaviour	Anomalies flagged by high reconstruction error	Learns to detect anomalies at the frame level from video-level tags
Methods Used	Motion prediction, appearance reconstruction	Autoencoders, one-class classification	Multiple Instance Learning, score regression
Pros	No labelling cost, learns detailed features	Easier setup, no anomaly data needed	Easier labelling, scales to large datasets
Cons	Relies on task design, may miss subtle anomalies	Poor generalization to unseen anomalies	Needs smart methods to localize frame-level anomalies

2.5 Previous Approaches on VAD

2.5.1 Weakly Supervised Anomaly Detection with Multiple Instance Learning (MIL) Frameworks

Sultani, Chen, and Shah (2018) introduced an innovative multiple instance learning (MIL) model, marking the first use of weakly labelled training videos. In this approach, normal videos are treated as negative bags, while anomalous ones are treated as positive bags, with video segments acting as instances within the MIL framework. These bags are processed through feature extractors to capture spatiotemporal features, which are then passed through a fully connected network to generate the final output. The anomaly score, ranging from 0 to 1, is optimized to increase for abnormal segments and decrease for normal ones.

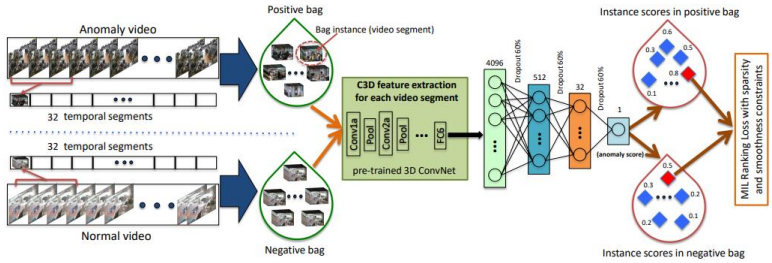


Figure 2.11: Flow Diagram of Weakly Supervised Anomaly Detection with MIL Frameworks (Sultani, Chen and Shah, 2018)

2.5.2 MIST: Multiple Instance Self-Training Framework for Video Anomaly Detection

Feng, Hong and Zheng (2021) presented "MIST: Multiple Instance Self-Training Framework for Video Anomaly Detection," a novel WSVAD approach. Unlike conventional MIL methods, MIST employs a pseudo-label generator combined with a sparse continuous sampling strategy to improve the accuracy of clip-level pseudo labels. It also features a self-guided, attention-enhanced encoder designed to focus on anomalous regions within video frames (Zhou et al., 2024).

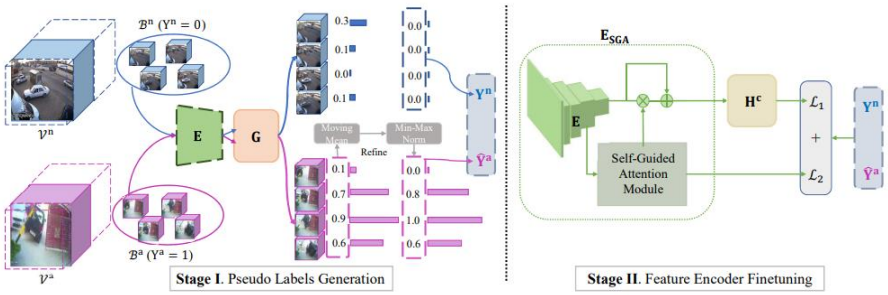


Figure 2.12: Flow Diagram of MIST for VAD (Feng, Hong and Zheng, 2021)

2.5.3 Graph Convolutional-based Label Noise Cleaner

In contrast to the conventional MIL methodology, the authors proposed a new technique for weakly supervised anomaly detection, considering it as a supervised learning task with noisy labels. The noisy labels were cleaned up using a Graph Convolutional Network (GCN), which enhanced the training procedure and the effectiveness of fully supervised action classifiers in identifying anomalies.

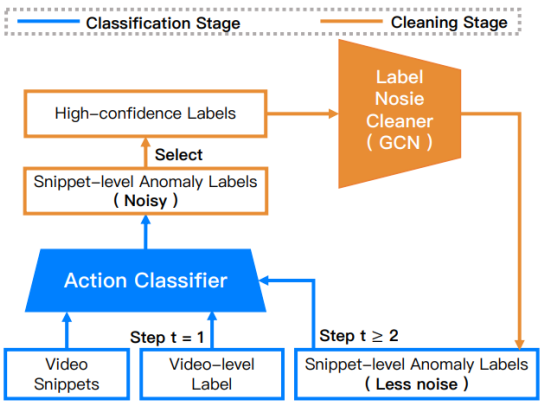


Figure 2.13: Graph Convolutional Label Noise Cleaner (Zhong et al., 2019)

2.5.4 BN-WVAD

BN-WVAD is a framework specifically designed to detect anomalies in videos using only video-level labels (weakly-supervised), avoiding the need for detailed frame-level annotations and utilizing batch normalization. Figure 2.14 shows the overall framework of the BN-WVAD model. The model is designed for weakly supervised learning. The datasets UCF-Crime and XD-Violence are chosen to match the weak supervision approach. In this model, the deep feature extraction is done by using 3D CNNs. The model utilizes an I3D network (Inflated 3D ConvNet), which is widely used to extract spatial and temporal features from video sequences.

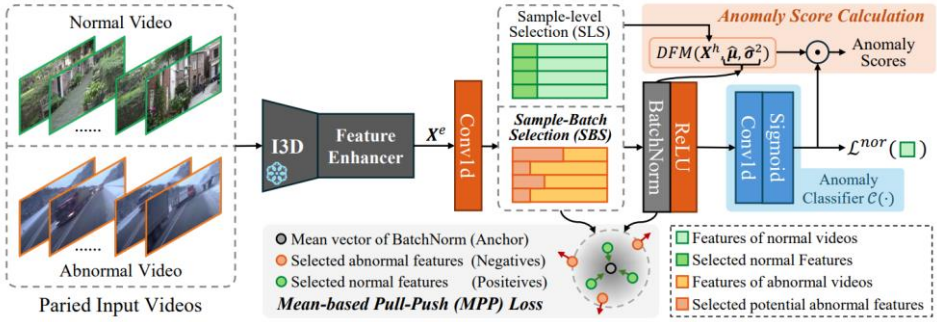


Figure 2.14: Overall Framework of BN-WVAD model (Zhou et al., 2024)

The I3D model utilised in this model, or Two-Stream Inflated 3D ConvNet, extends 2D ConvNets into 3D by adding a temporal dimension to filters and pooling layers, enabling it to capture both spatial and temporal features. It reuses 2D filters from pretrained models like ImageNet and employs a two-stream setup: one stream processes RGB frames and the other handles optical flow for motion. The outputs are fused at the prediction stage to enhance understanding of video content (Carreira & Zisserman, 2017).

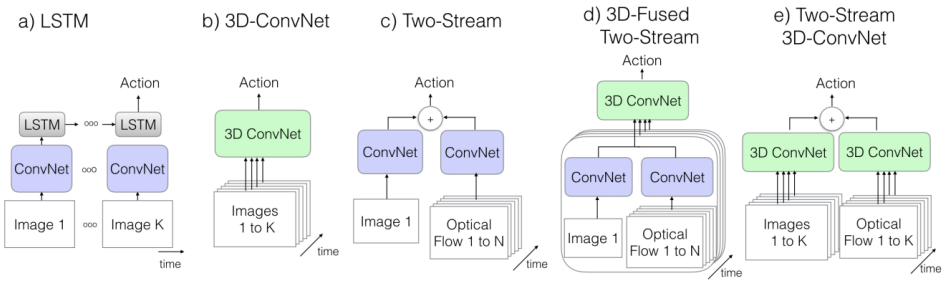


Figure 2.15: Five types of architecture evaluated (Carreira and Zisserman, 2017)

In addition, the extracted features are refined using a Global and Local Multi-Head Self-Attention (GL-MHSA) module, designed to capture both long-range dependencies across the entire video and short-range temporal patterns. This module builds upon the standard Multi-Head Self-Attention (MHSA) mechanism by incorporating an additional encoder layer equipped with a temporal mask, allowing it to more effectively learn local temporal features.

As illustrated in Figure 2.16, it uses a temporal mask to balance the influence of different time points, enhancing the model's capability to capture both long-range and short-range dependencies effectively (Zhou, Yu and Yang, 2023). The approach utilizes the transformer's self-attention mechanism while adding a layer of complexity to enhance the comprehension and modeling of video sequence dynamics for anomaly detection. By learning the spatial and temporal features in combination, the model ensures robust feature extraction for identifying anomalies.

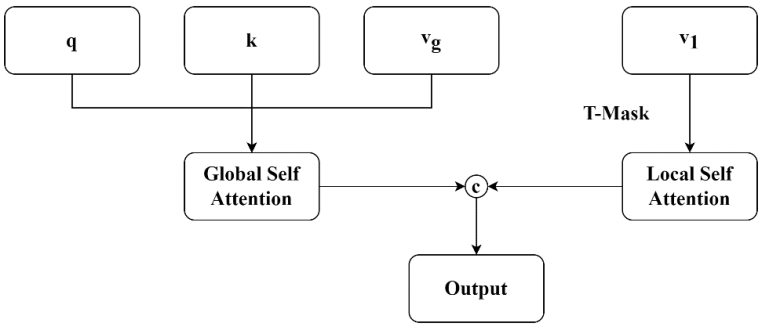


Figure 2.16: GL-MHSA module (Zhou, Yu and Yang, 2023)

Thus, it can be summarized that the backbone of the model is a combination of I3D for feature extraction and GL-MHSA for enhancing the learning of global and local temporal relationships.

2.6 Model Optimization Techniques

Model optimization in deep learning refers to a process designed to refine a neural network to boost its performance and efficiency of machine learning models. The process includes techniques that minimize the use of computational resources, such as memory and processing time, by refining the structure and functionality of the model, without compromising the model's accuracy and overall effectiveness. Additionally, as deep learning models are deployed in web applications, mobile devices, and edge devices, it is crucial to compress these

models without compromising the quality and performance of the original models (VK, 2024).

Optimizing models can lead to a decrease in their size, which offers several benefits. These include reduced storage requirements, smaller download sizes, and lower memory consumption. Techniques like quantization can decrease the model's size in all of these areas, although there might be a slight trade-off in accuracy. Additionally, pruning and clustering methods can make models more compressible, which helps in reducing download sizes.

Latency refers to the time it takes for a model to make a prediction. Certain optimization techniques can reduce the amount of computation needed to perform inference, thereby decreasing latency. This also helps to reduce power consumption. Currently, quantization is a widely used method to reduce latency by simplifying the operations performed during inference, though it may result in a slight loss of accuracy.

2.6.1 Quantization

Quantization operates by lowering the precision of the numbers that represent a model's parameters, which are typically from 32-bit floating point values to 16-bit or 8-bit. This reduction in precision leads to a smaller model size and quicker computation, enhancing both memory efficiency and processing speed. Figure 2.17 shows an example of quantization from 32-bit to 8-bit.

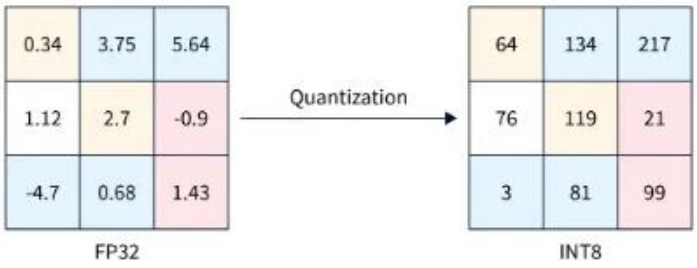


Figure 2.17: Quantization Technique (VK, 2024)

Quantization-Aware Training involves incorporating quantization into the training process. During QAT, the model is trained with simulated lower precision weights and activations, allowing the network to learn to compensate for the reduced precision (Ray, 2024). This method typically leads to better performance compared to PTQ since the model has been explicitly optimized for quantization.

On the other hand, Post-Training Quantization is applied after the model has already been trained. It involves quantizing the weights and activations of a pre-trained model without requiring additional training. PTQ is generally faster and easier to implement than QAT, but it may result in some performance degradation due to the lack of optimization for quantization during training (Ray, 2024). PTQ is useful when computational resources or time constraints limit the ability to retrain the model.

2.6.2 Pruning

Model pruning is a technique used to reduce the size of a model by eliminating unnecessary weights and parameters, thereby improving efficiency. In computer vision, particularly with deep neural networks, the presence of a vast number of parameters, such as weights and activations, which are the intermediate outputs that assist in producing the final result, can significantly increase both the complexity and computational requirements (Vina, 2024). Pruning addresses this by identifying and removing parameters that have little impact on the performance of the model, thus resulting in a more lightweight and efficient model. While pruned models maintain the same size on disk and exhibit the same runtime latency, they become more compressible. This makes pruning an effective technique for reducing the model's download size.

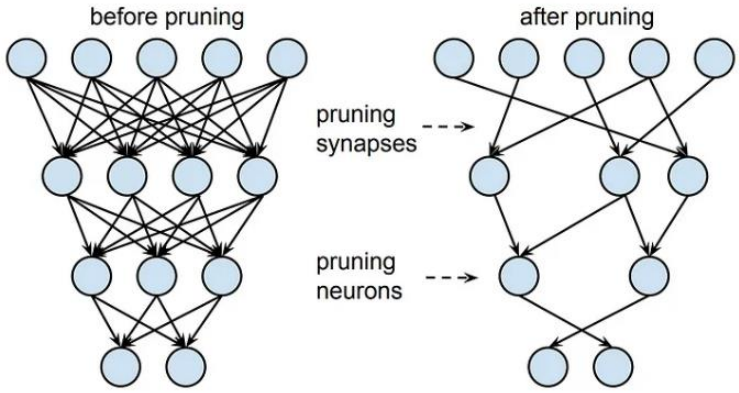


Figure 2.18: Pruning (VK, 2024)

2.6.3 Knowledge Distillation

Knowledge distillation is an optimization method where knowledge is transferred from a larger, more complex model (referred to as the "teacher") to a smaller, more computationally efficient model (the "student"). The underlying

idea is that, while a large and intricate model may be necessary to understand complex patterns in data, a smaller model can learn and capture the same relationships, achieving similar performance levels in tasks with lower computational demand. This technique is commonly applied to classification models (both binary and multi-class) with a softmax activation in the output layer (Lamberti, 2024).

The core of knowledge distillation is based on two key principles: the teacher-student framework and distillation loss. In this setup, the teacher model is a high-capacity network that performs well on the task, while the smaller student model is more compact and optimized for efficiency. The student is trained to replicate the teacher's predictions but also to match the output distributions produced by the teacher. This allows the student model to grasp the relationships between the data samples and their corresponding labels, especially in classification tasks where it learns to approximate the decision boundaries defined by the teacher model (Lamberti, 2024).

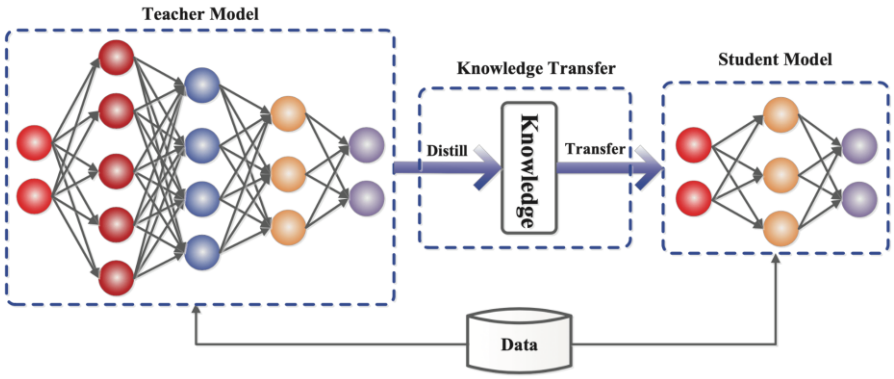


Figure 2.19: Knowledge Distillation (Teki, 2022)

2.6.4 OpenVINO

One of the most effective optimization tools available is the OpenVINO optimization method. In many real-world applications, deep learning AI models need to be optimized to make better use of computational resources, ensuring they deliver faster performance. OpenVINO provides a range of tools that support this goal, including the Model Optimizer, Post-training Optimization Tool (POT), and the Neural Network Compression Framework (NNCF), all of which are designed to improve model efficiency and reduce memory usage.

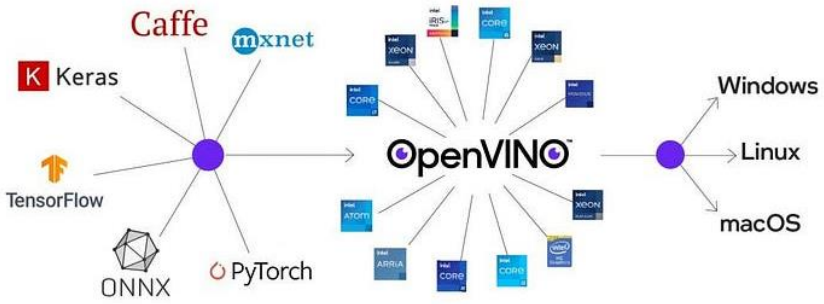


Figure 2.20: OpenVINO Optimization Tool

2.7 Inference

During the inference phase, the model is tasked with identifying unusual actions or events within videos by analysing the anomaly scores it produces. However, during inference, videos often have different durations, leading to a variety of frames. There are two primary categories to handle this.

2.7.1 Variable-Length Sequence Handling (Dynamic)

In this approach, all sampled frames from a video are passed through the model. The model uses pack_padded_sequence in Pytorch to process sequences of different lengths efficiently. To process a batch, all sequences are padded to match the length of the longest sequence, as shown in Figure 2.21. This is suitable when the model was trained with variable-length sequences, ensuring consistency between training and efficiency. However, this might waste processing power to perform unnecessary padding while ensuring uniformity in sequence lengths by adding zeros to shorter sequences (GeeksforGeeks, 2024).

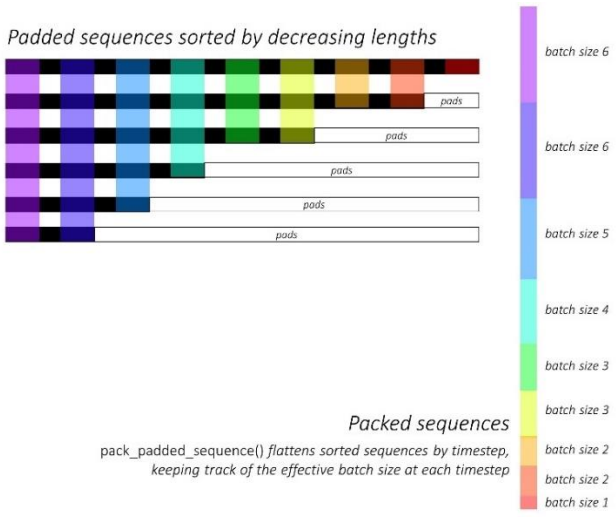


Figure 2.21: Overview of the working of pack_padded_sequence

2.7.2 Fixed-Length Sequences

In this approach, all video inputs are transformed into a fixed sequence length. There are two sub-strategies for this, which include padding and truncation, and the sliding window approach.

2.7.2.1 Padding and Truncation

In this method, all video sequences are standardized to a fixed length, with shorter sequences being padded with zeros or another neutral value, ensuring uniform input size for the model. Conversely, sequences longer than the target length are truncated to fit. This approach is straightforward and efficient, particularly when the average or median sequence length from the training data is known and representative of most samples. It ensures consistent input dimensions, which is especially helpful when using models that expect fixed-size inputs.

2.7.2.2 Sliding Window approach

The sliding window technique breaks long videos into smaller, fixed-length segments, which are then processed individually. These windows can either overlap or remain distinct depending on how much temporal coverage is desired (Overload, 2022). This method is especially useful for analysing lengthy videos, where anomalies might occur at any point in time. By examining each segment in isolation, the model can effectively localize abnormal events without needing to process the entire video at once.

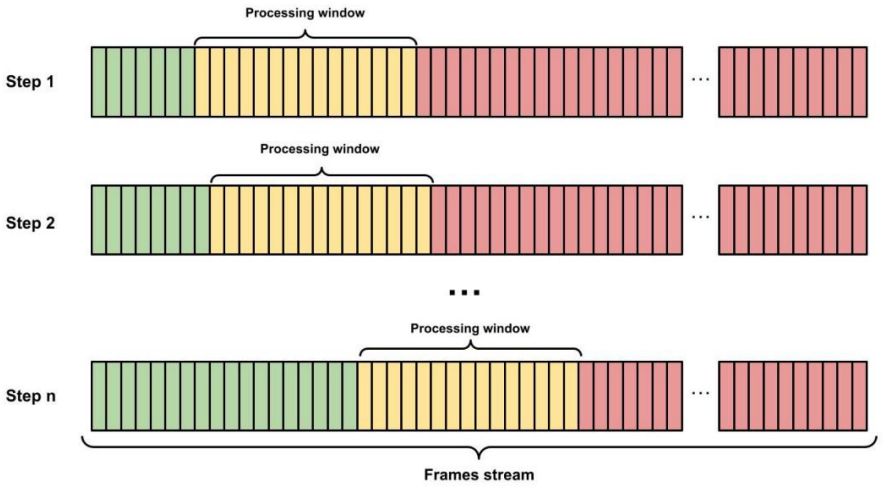


Figure 2.22: Sliding Window Approach (Jaime-Rodrigo González-Rodríguez et al., 2024)

2.8 Summary

This chapter provides an overview of recent advancements in Video Anomaly Detection (VAD), highlighting the impact of deep learning techniques. Traditional methods are increasingly being outperformed by models using Convolutional Neural Networks (CNNs), Transformers, and other architectures that effectively capture complex spatio-temporal features for more accurate anomaly detection in surveillance and other video streams.

Benchmark datasets like ShanghaiTech, UCF-Crime, and XD-Violence are discussed, each presenting unique challenges such as annotation levels, anomaly types, and input modalities. These datasets are critical for evaluating and comparing VAD methods, especially in supervised and weakly supervised learning settings.

The chapter also reviews key feature extractors. While 2D CNNs handle spatial features well, 3D CNNs capture temporal dynamics across frames. Modern architectures like Transformers and Graph Convolutional Networks (GCNs) are also explored, offering strong sequence modelling and relational reasoning capabilities, respectively.

Finally, the chapter outlines the current VAD approaches, focusing on the shift from fully supervised to self-supervised, weakly supervised, and unsupervised learning techniques.

Additionally, model optimization techniques, including quantization, pruning, and knowledge distillation, are crucial for deploying VAD models on resource-limited devices. These techniques help maintain performance while reducing model size and latency. Inference strategies like dynamic sequence handling and sliding windows ensure consistent performance across videos of varying lengths, enabling real-time and practical deployment.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The essential elements for creating a video anomaly detection system are covered in this chapter, with an emphasis on the XD-Violence and UCF-Crime datasets, which provide a variety of real-world scenarios. The model architecture is presented, integrating temporal analysis and feature extraction with deep learning techniques. In addition, methods for enhancing anomaly detection are covered, as well as the performance measures for the model, AUC and AP, which are utilized for evaluation.

3.2 Experimental Setup

3.2.1 Hardware

Table 3.1: Experimental Platform Configuration

Name	**Configuration**
Operating System	Ubuntu 22.04.4 LTS (Jammy)
CPU Model	12th Gen Intel(R) Core(TM) i5-12450H @ 2.50 GHz
RAM	16.0 GB
GPU Model	NVIDIA GeForce RTX 3050
GPU Memory	4 GB
NVIDIA Driver Version	535.183.01
CUDA Version	12.2

3.2.2 Software

Throughout this project, multiple frameworks, platforms, and tools, including Python, Pytorch, Kaggle, Google Colab, OpenCV, and VLC, are utilized for multi-task model development and experimentation.

3.2.2.1 Google Colaboratory

Google Colab is a cloud-based Jupyter notebook service that requires no installation or setup, enabling users to run computationally intensive tasks-such as machine learning training, directly in a browser. It provides free access to high-performance GPUs and TPUs, significantly accelerating model development. In this project, the majority of the training and testing procedures were conducted using the Google Colab platform, leveraging its high-performance A100 GPU to accelerate deep learning computations, as shown in Table 3.2. The platform's deep integration with Google Drive and streamlined sharing capabilities make it an ideal environment for collaborative development. Table 3.2 shows the computational resources mainly used in this project for training.

Figure 3.1: Google Colaboratory logo

Table 3.2: Computational Resources

Graphics Processor	Video Memory	Memory Capacity
A100 GPU	40 GB VRAM	83.5 GB RAM

3.2.2.2 Kaggle

In this project, the datasets after preprocessing steps are uploaded to Kaggle for further usage. Kaggle is a platform and online community for data scientists and machine learning practitioners, hosted by Google. Besides, Kaggle provides a vast library of public datasets for users to explore, analyze, and build models on. It supports both private and public dataset sharing, along with robust storage capabilities, allowing up to 200 GB per dataset. Particularly relevant for this project's needs, which involve processing large volumes of video data, Kaggle's generous storage allocation and computational resources make it perfectly suited for the requirements. Once published, these datasets can be programmatically accessed by third parties through Kaggle's API, enabling seamless integration with external tools and workflows.

Figure 3.2: Kaggle logo

3.2.2.3 Python

Python, a high-level programming language with dynamic semantics, is the choice of programming language for this deep learning project. Although deep learning can be implemented in multiple programming languages such as C++, Java, and others, Python continues to be the preferred language for most developers. Leading open-source frameworks like TensorFlow and PyTorch offer intuitive Python APIs, making it easier for developers to design and train neural networks efficiently (Williamson, 2021).

Figure 3.3: Python logo

3.2.2.4 VLC Media Player

In this project, VLC media player is commonly utilized for video-related operations, including frame-by-frame navigation and extraction. These functionalities are particularly useful for tasks such as verifying annotations, inspecting temporal boundaries of anomalous events, and ensuring the accuracy of ground truth labels.

Figure 3.4: Icon of VLC media player

3.2.2.5 Pytorch

Pytorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It provides a flexible, Pythonic interface for building deep learning models, and it is widely used in both research and production. Pytorch provides some key features such as strong GPU acceleration support, easy debugging and customization, and a rich ecosystem including torchvision, torchaudio, and more (NVIDIA, n.d.). It enables efficient model design, training, and evaluation in this anomaly detection project.

Figure 3.5: Pytorch Icon

3.2.2.6 OpenCV

OpenCV is a free and open-source software library for computer vision and machine learning. It is widely used for image and video processing. In this VAD (Video Anomaly Detection) project, OpenCV plays a key role in handling video input, frame extraction, and display of detection results. It supports both video inference and real-time inference, enabling frame-by-frame processing and visualization of anomalies (Kulhary, 2019). Its speed, Python compatibility, and strong community support make it ideal for real-time detection tasks.

Figure 3.6: Icon of OpenCV

3.3 Work Plan

Figure 3.7 shows the flowchart of the entire workflow, summarizing all the steps.

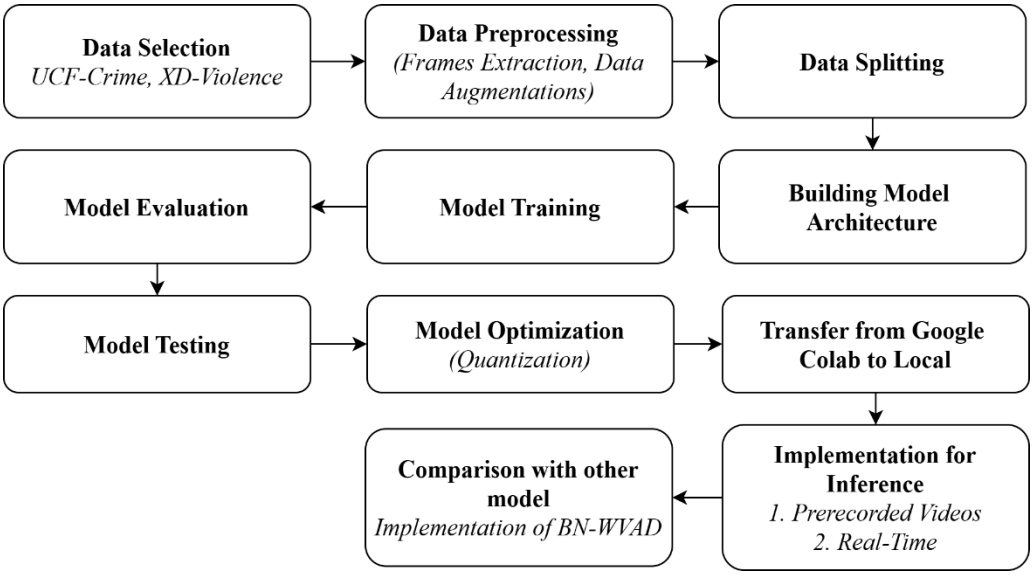


Figure 3.7: Flowchart of Entire Workflow

3.3.1 Dataset Selection

In this study, two different datasets were used, which are UCF-Crime and XD-Violence.

3.3.1.1 XD-Violence

The first datasets used are XD-Violence obtained from Wu et al. (2020). The dataset is selected for several reasons. It covers six distinct categories of physical violence, including Abuse, Car Accident, Explosion, Fighting, Riot, and Shooting. This diverse range of violence types provides a comprehensive representation of different violent events, making it a valuable resource for training VAD models. By including a variety of violence categories, the dataset ensures that the VAD model can generalize well across different types of anomalies.

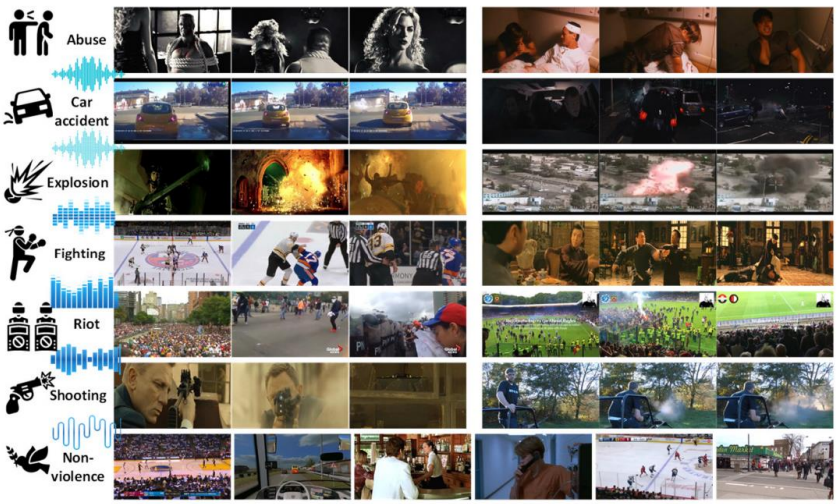


Figure 3.8: Sample Videos from XD-Violence Datasets

Besides, the XD-Violence dataset is compiled from both movies and YouTube videos. This blend of sources enriches the dataset with a wide range of scenarios, from scripted and staged violence in movies to real-world, in-the-wild scenes from YouTube. This variety helps the VAD model to learn from both controlled environments and more unpredictable, real-world situations, enhancing its robustness.

The XD-Violence dataset is particularly suitable for VAD models due to its precise frame-level annotations, which detail the start and end frames of violent events, ensuring accurate temporal information. With a substantial collection of 4,754 videos, including a diverse mix from movies and YouTube, the dataset offers a broad range of scenarios and violence types. The high-quality annotations, derived from multiple annotators and averaged for precision, contribute to the dataset's reliability. Additionally, the inclusion of various temporal locations within the videos helps the VAD model recognize anomalies across different time frames, enhancing its overall robustness and generalization.

3.3.1.2 UCF-Crime

The UCF-Crime dataset's thorough and accurate depiction of abnormalities in actual surveillance film makes it an excellent option for use in VAD systems. It offers a comprehensive and diverse collection of 1,900 untrimmed movies, showcasing 13 different types of anomalies that can occur in real-world scenarios, such as stealing, abuse, fighting, and others, as shown in Figure 3.9. This contrasts with other datasets provided, which frequently involve

constrained scenarios or unrealistic anomalies. These anomalies are chosen due to their significant impact on public safety (Sultani, Chen, and Shah, 2018). Because of this diversity, VAD models trained on this dataset are exposed to a greater variety of scenarios, which enhances their capacity to generalize across various violent and abnormal event types.

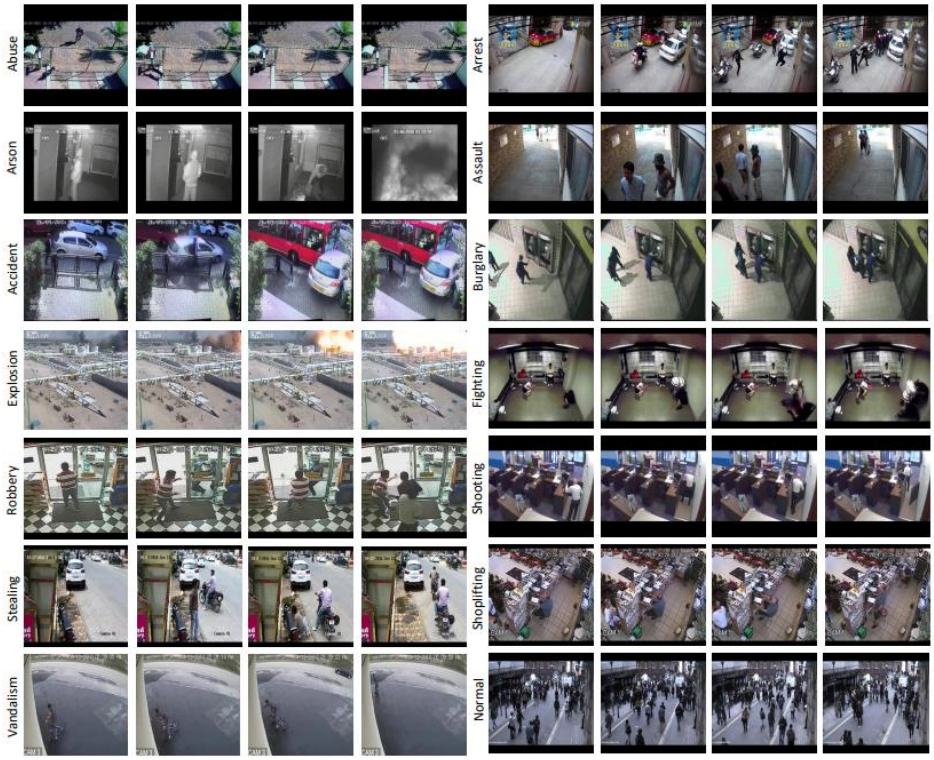


Figure 3.9: Datasets from UCF-Crime

With 128 hours of footage total and an average of 7,247 frames per movie, the dataset's large scale provides a significant amount of data for training and testing VAD models. Detailed annotations offer precise temporal data regarding the onset and duration of anomalous events, featuring frame-level labels for testing and video-level labels for training. This degree of detail improves the model's high-accuracy detection and differentiation between normal and aberrant activity.

In addition, strict guidelines for video selection and annotation were followed during the collection of the UCF-Crime dataset. To verify the accuracy of the anomalies, videos that were taken from actual surveillance scenarios were carefully reviewed to remove modified, prank, or non-CCTV material. The comprehensive methodology used for data gathering and annotation enhances the validity and significance of the dataset.

3.3.1.3 Overview of Datasets Used

Table 3.3 shows the summary of the two datasets used in this research.

Table 3.3: Comparison of the two datasets used

Feature	XD-Violence	UCF-Crime
Source of videos	Movies and YouTube (mixed realism)	Real-world CCTV surveillance footage
Number of Video Clips	4,754	1,900 (untrimmed)
Violence videos	2405	950
Non – violence videos	2349	950
Anomaly Types	6 violence categories	13 real-world categories
Annotation	• Video-level labels for training • Frame-level labels for testing	

3.3.2 Data Preprocessing

Before training deep learning algorithms, the video data must undergo several preprocessing steps to ensure proper preparation. In this study, the UCF-Crime dataset is readily available at the frame level on Kaggle, as provided by other authors, allowing it to be used directly for further preprocessing. However, the XD-Violence dataset is only accessible at the video level, requiring additional preprocessing steps to extract frame-level data. Figure 3.10 shows the overview of the dataset preprocessing steps applied to the datasets used.

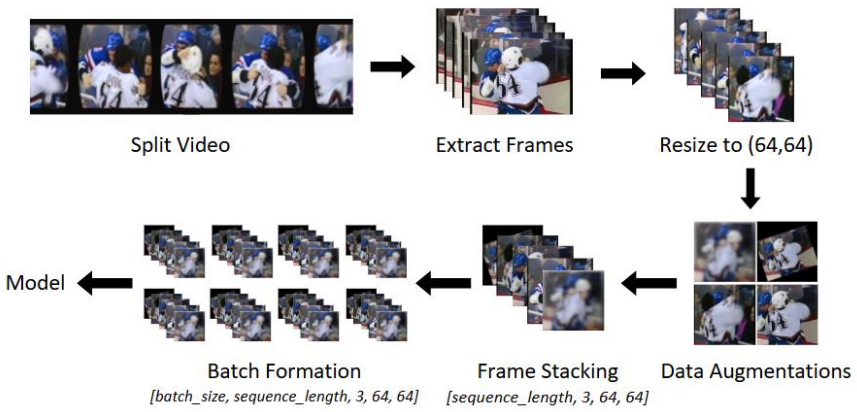


Figure 3.10: Overview of Data Preprocessing Steps

3.3.2.1 UCF-Crime

3.3.2.1.1 Reconstruction of the Dataset

The UCF-Crime dataset utilised in this study comprises 1610 video sequences covering 14 distinct classes. As shown in Figure 3.11, the dataset exhibits significant class imbalances, with 'Normal Videos' representing the majority class, posing challenges for model training for multi-classification. The dataset consists of individual frames in small sizes rather than pre-arranged video sequences. The frames were then reconstructed into video sequences by sorting them based on their filenames. This involved sequentially ordering the frames and labelling them according to their corresponding classes (Zvereva, Kaprielova, and Andrey Grabovoy, 2025). The following steps are taken:

1. Frame Grouping: Frame sequences were reconstructed by identifying and grouping related frames through filename pattern analysis. Each frame's unique video identifier was extracted to ensure proper temporal grouping.
2. Class Annotation: Each video was categorized based on its folder hierarchy, with labels corresponding to the particular anomaly type or indicating normal behavior.
3. Analysis of Data Distribution: Analysis of the video lengths indicated that the shortest sequence comprised 11 frames, while the longest reached 97,651 frames. On average, sequences contained approximately 786.5 frames, with a median length of 222.5 frames.

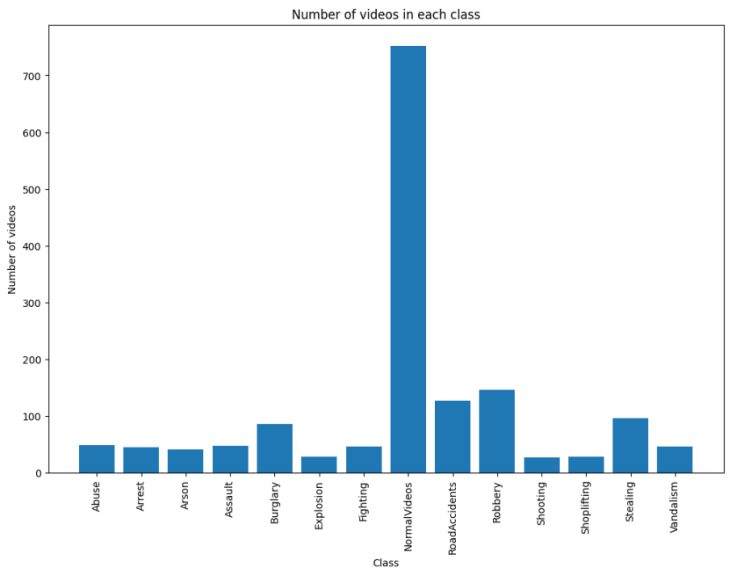


Figure 3.11: Overview of Original Class Distribution in UCF-Crime

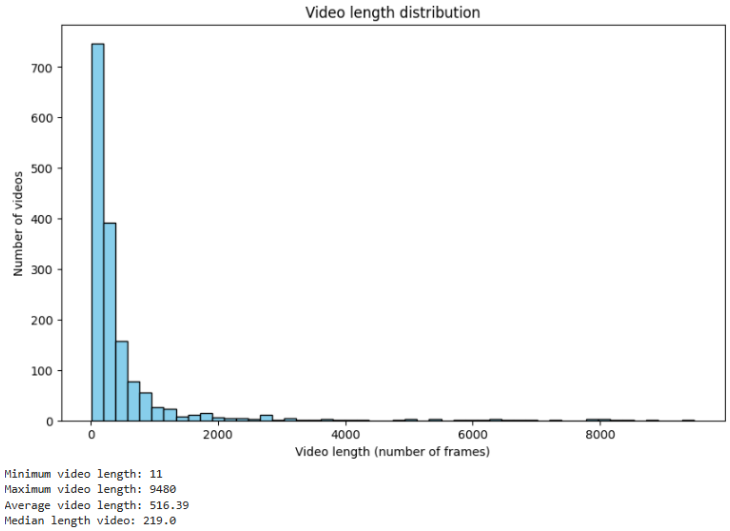


Figure 3.12: Distribution of Video Duration Under 10,000 Frames

Table 3.4: Count and Proportion of Videos Exceeding 2,500 Frames

Class	Total Videos	Percentage (%)
Abuse	1	1.61 %
Arrest	1	1.61 %
Burglary	2	3.23 %
Fighting	2	3.23 %
Shoplifting	2	3.23 %
Normal Videos	54	87.10 %

Figure 3.12 demonstrates the variation in video sequence lengths, with a noticeable occurrence of clips having a duration of up to 2,500 frames. To enhance dataset balance, sequences exceeding 2,500 frames were removed, as they were predominantly from the 'Normal' class. This step effectively reduced the imbalance between classes, producing a final dataset with 1,528 video sequences.

3.3.2.2 XD-Violence

XD-Violence dataset is only accessible at the video level, thus requiring additional preprocessing steps to extract frame-level data. Moreover, the XD-Violence dataset has different structures compared to UCF-Crime, thus requiring unique handling. In the XD-Violence dataset, each video filename encodes a class label, the labels are shown in Table 3.5.

Table 3.5: Label Explanation

Label	Class
B1	Fighting
B2	Shooting
B4	Riot
B5	Abuse
B6	Car accident
G	Explosion

For example: “Black.Hawk.Down.2001__#01-13-59_01-14-49_label_B2-0-0” indicates there are shootings in the video ‘Black.Hawk.Down.2001__#01-13-59_01-14-49’.

3.3.2.2.1 Class Grouping

Based on the pattern shown in Figure 3.11 and 3.12, the videos are first grouped according to their classes based on the label name in the video name, in order to facilitate further frame extraction.

3.3.2.2.2 Frames Extraction

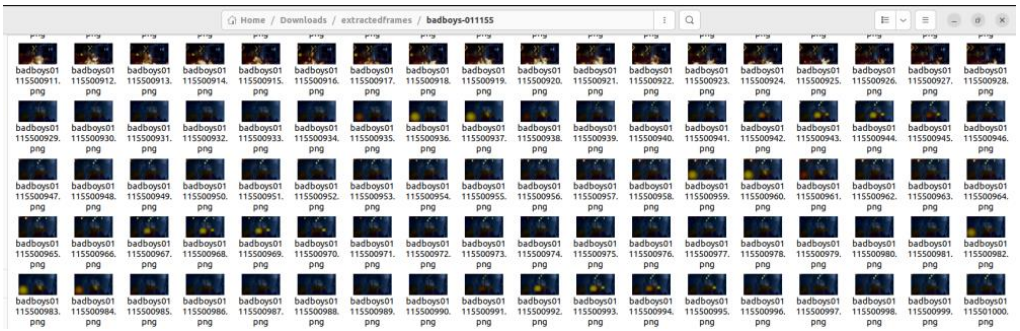


Figure 3.13: Frames extracted

The frames are then extracted from the video files of the XD-Violence dataset to produce a series of frames in which the sequence forms a single video sample, as shown in Figure 3.13. These frames capture the temporal changes in scenes and serve as the primary input for subsequent analysis. Every 10^{th} frame is extracted from each full-length video. After extraction, all the frames are resized to standardized dimensions of 64 x 64 pixels to maintain consistency across all samples. This spatial normalization ensures uniform preprocessing while

preventing potential artifacts like distortion or blurring that could affect the neural network performance, while also improving the computational efficiency.

3.3.2.2.3 Dataset Uploaded to Kaggle

After frames are extracted, the frame-level dataset is then uploaded to Kaggle, so that it is further accessible through Kaggle API. Figure 3.14 shows the frame-level dataset uploaded to Kaggle.

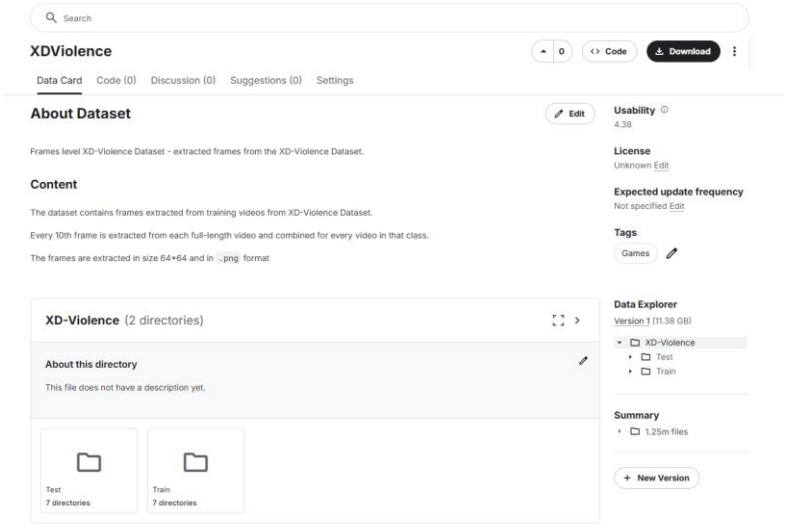


Figure 3.14: Dataset Published on Kaggle

3.3.2.2.4 Reconstruction of XD-Violence Dataset

XD-Violence dataset utilised in this study comprises of 2460 video sequences covering 6 distinct classes. Similar to UCF-Crime, the frames are then grouped into video sequences by parsing filenames, and each video sequence is categorized according to its folder structure, which determines its label.

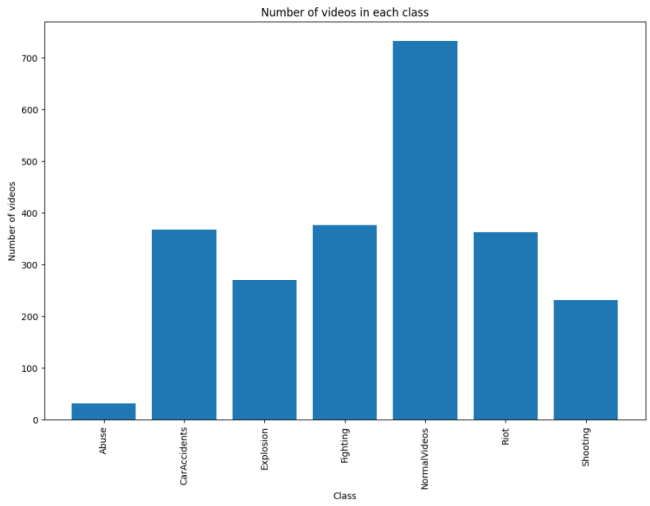


Figure 3.15: Overview of Original Class Distribution in XD-Violence dataset

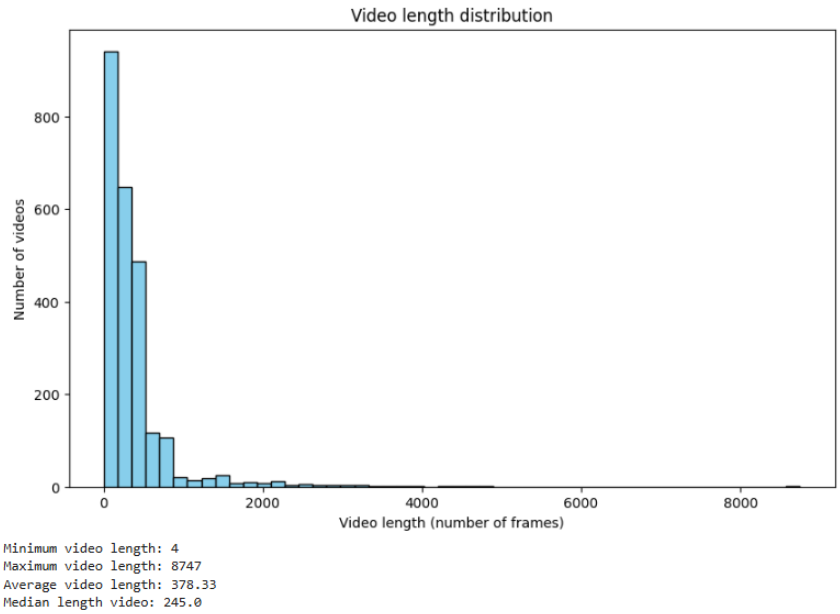


Figure 3.16: Video lengths in XD-Violence when limited to 10,000 frames

Table 3.6: Count and Proportion of Videos Exceeding 2,500 Frames

Class	Total Videos	Percentage (%)
Explosion	1	3.33 %
Riot	4	13.33 %
Normal Videos	25	83.33 %

Figure 3.15 shows the overview of original class distribution in XD-Violence dataset. Figure 3.16 illustrates the variation in video sequence lengths, with a noticeable occurrence of clips having a duration of up to 2,500 frames. To enhance dataset balance, sequences exceeding 2,500 frames were removed, as they were predominantly from the 'Normal' class (83.33%). This step effectively reduced the imbalance between classes, producing a final dataset with 2,423 video sequences.

3.3.3 Data Augmentations

Figure 3.17: Data augmentations on UCF-Crime

Figure 3.18: Data augmentations on XD-Violence

Similar to the data augmentations applied for UCF-Crime, a range of augmentations was also applied to the XD-Violence dataset to enhance the model's robustness and adaptability to practical conditions. Table 3.7 shows the data augmentations applied, whereas Figures 3.17 and 3.18 show 5 examples of augmented UCF-Crime and XD-Violence datasets, respectively.

Table 3.7: Data Augmentations Parameter applied

Augmentation type	Parameters	Probability	Purpose
Gaussian Blur	Kernel size: 5	30%	Imitate out-of-focus frames
Gaussian Noise	Mean: 0.0, Std Dev: 1.0	30%	Simulate noise from sensors
Color Jittering	Brightness and Contrast variation: ±30%	30%	Reflect lighting fluctuations
Random Rotation	Rotation angle: ±15%	20%	Simulate camera angle variations

3.3.4 Normalization

Input normalization was carried out using the typical RGB mean and standard deviation values, as shown in Table 3.8, ensuring consistency in the input distribution and aligning with established preprocessing norms in deep learning.

Table 3.8: RGB Mean and Standard Deviation Values

Color Channel	**Mean**	**Standard Deviation**
Red (R)	0.485	0.229
Green (G)	0.456	0.224
Blue (B)	0.406	0.225

3.3.5 Data Splitting

As part of the data splitting process, the dataset is segmented into training, validation, and testing sets. Initially, video frame sequences from the training directory are grouped by class and split into training and validation sets using a stratified sampling approach, ensuring balanced class distribution. Specifically, 70% of the data is allocated for training, while the remaining 30% is reserved for validation. This division improves model training and allows for evaluation of generalization during the training process. Separately, an independent test set is loaded from a designated test directory to assess the model's final performance. This structured splitting strategy ensures reliable model training, validation, and performance evaluation.

3.3.6 Model Architecture

In MIL-based techniques, most of the VAD models consist of at least two modules, which include the prediction head and backbone for video processing (Sertis, 2023). In this research, the AnomLite model is a lightweight, yet powerful architecture designed for video anomaly detection, combining spatial feature extraction (using MobileNetV2) with temporal modeling (via LSTM).

The input videos are processed in combined batches, containing either normal behaviours or abnormal events, as shown in Figure 3.19. These batches are then passed through the AnomLite model. The model consists of three main components: a spatial feature extractor, followed by a temporal modeling unit, and then a fully connected layer. Using the processed video data, the model produces a prediction that identifies whether the video shows normal activity or contains any of 13 distinct anomaly types, including actions like fighting, arrest, or other uncommon behaviors.

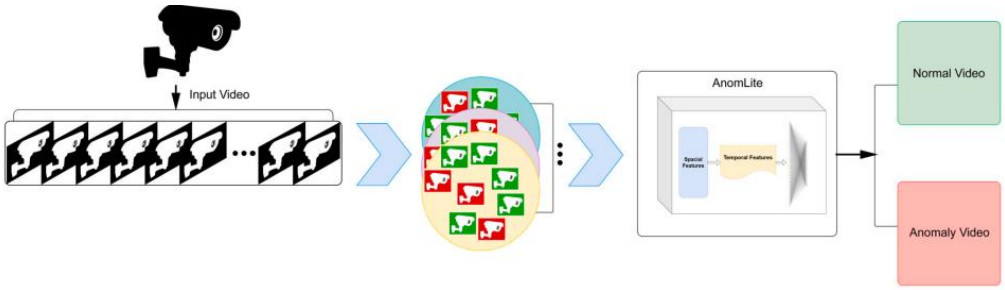


Figure 3.19: Spatio-temporal features from video batches processed by the model (Zvereva, Kaprielova, and Grabovoy, 2025)

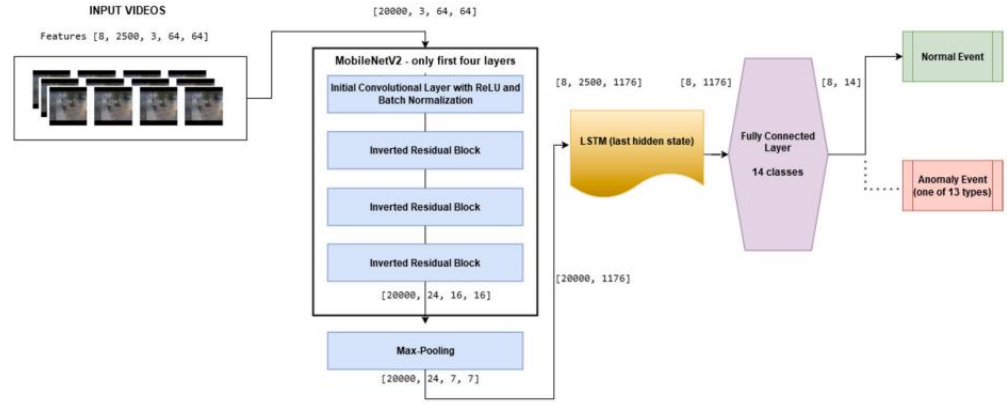


Figure 3.20: Overview of AnomLite Model Structure (Zvereva, Kaprielova and Andrey Grabovoy, 2025)

The model architecture begins with an input layer that takes a 5D tensor of shape [batch_size, sequence_length, 3, 64, 64], representing batches of video sequences with RGB frames resized to 64×64 pixels (Zvereva, Kaprielova, and Andrey Grabovoy, 2025), as shown in Figure 3.20. Spatial feature extraction is performed using only the first four layers of MobileNetV2, a lightweight convolutional backbone optimized for efficiency. These layers consist of an initial convolutional block followed by three InvertedResidual blocks, which gradually refine the input with depthwise separable convolutions (Zvereva, Kaprielova and Andrey Grabovoy, 2025).

Once the data passes through the MobileNetV2 layers, a max-pooling operation is performed to decrease the spatial size of the feature maps. These resulting features are then flattened to set them up for temporal analysis.

An LSTM model is used to capture temporal dependencies across frames, processing input sequences of shape [8, 2500, 1176], which is the batch size of 8, sequence length of 2500, and feature size of 1176. The model uses

only the final hidden state, producing a summary representation of shape [8, 1176].

Finally, a fully connected layer maps this representation to an output tensor of shape [8,14], corresponding to 13 anomaly classes and 1 normal class (Zvereva, Kaprielova, and Andrey Grabovoy, 2025).

3.3.7 Model Training

The model training process for the AnomLite architecture is designed to handle video anomaly classification by leveraging both spatial and temporal features. The model combines MobileNetV2, a lightweight convolutional neural network pretrained on ImageNet, with an LSTM (Long Short-Term Memory) layer to capture temporal dynamics across frames (Zvereva, Kaprielova and Andrey Grabovoy, 2025). During training, only the first few convolutional layers from MobileNetV2 are used for feature extraction to maintain computational efficiency. Initially, these convolutional layers are frozen to retain their pretrained weights but are later unfrozen after the second epoch to allow fine-tuning based on the target task.

In each training epoch, the model processes video clips represented as sequences of image frames. The spatial features are extracted by MobileNetV2, and then a MaxPool2d operation is used to reduce their spatial resolution. These features are then flattened and passed through the LSTM, which models the sequence information across frames. The final hidden state of the LSTM is passed through a dropout layer and batch normalization, before being classified by a fully connected layer (Zvereva, Kaprielova and Andrey Grabovoy, 2025).

Training uses a Cross Entropy loss function with class weights to handle class imbalance, and Adam optimizer with an initial learning rate of 0.0002. Among all configurations, a learning rate of 0.0002 yielded the most consistent and superior performance, with higher ROC AUC, AP, and weighted F1-score results. In contrast, lower rates like 0.0005 led to slower training and significantly lower accuracy. Higher learning rates, on the other hand, introduced training stability, establishing 0.0002 as the ideal compromise between rapid convergence and model stability. Additionally, SMOTE has been used to further alleviate class imbalance by oversampling minority classes. Figure 3.21 shows a comparison of the dataset with and without the application

of SMOTE. An early stopping mechanism is employed to terminate the training process if the validation loss fails to improve over five consecutive epochs, thereby mitigating the risk of overfitting. Throughout training, performance is tracked using both micro and macro F1-scores, providing a robust evaluation across both balanced and imbalanced datasets. Model checkpoints are saved whenever a new best validation loss is achieved (Zvereva, Kaprielova, and Andrey Grabovoy, 2025).

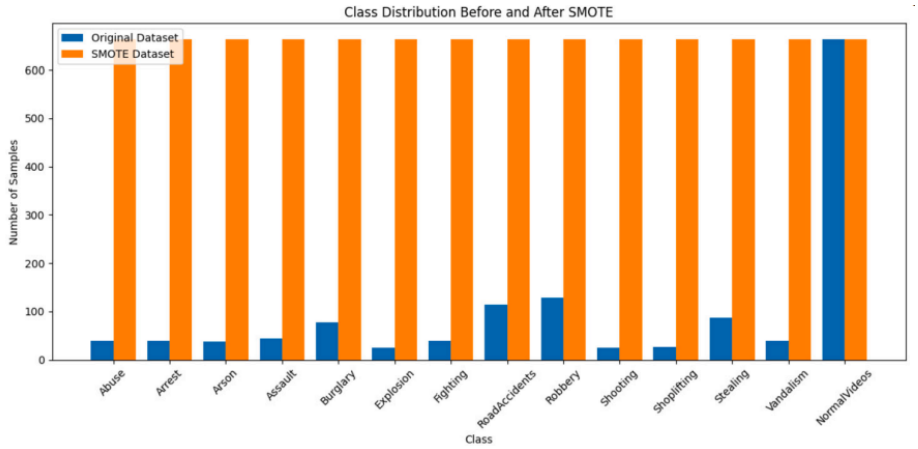


Figure 3.21: Comparison of Dataset Before and After Applying SMOTE (Zvereva, Kaprielova, and Andrey Grabovoy, 2025)

3.3.8 Loss Functions

In this model, the loss functions are vital in addressing the class imbalance issue and improving the model's performance in detecting anomalies in video sequences. Two primary loss functions were utilised: Weighted Cross-Entropy Loss and Focal Loss.

The Weighted Cross-Entropy Loss is utilised to address the issue of class imbalance by assigning higher weights to the underrepresented classes. This is especially important in datasets where the "Normal" class dominates the data, making it harder for the model to detect anomalies. The formula for this loss is:

$$L = -\sum_{i=1}^{C} \omega_i \cdot y_i \cdot \log(p_i) \tag{3.1}$$

Where ω_i is the weight assigned to class i, y_i is the true label of class i (either 0 or 1), and p_i is the predicted probability for class i. The weight ω_i is calculated inversely proportional to the frequency of the class, meaning that

classes with fewer examples will have higher weights. This ensures that the model pays more attention to the underrepresented anomaly classes, thus mitigating the bias toward the normal class. The compute_class_weight function from scikit-learn is used to calculate these weights, which are then incorporated into the loss function during training.

Focal Loss was implemented to emphasize difficult-to-classify samples and enhance the model's responsiveness to minority classes. Focal Loss is designed to improve model performance on challenging samples by reducing the influence of easily classified examples. It achieves this through down-weighting, which shifts the model's focus toward harder instances and their associated errors, rather than relying solely on prediction confidence. The formula of this loss is:

$$L = -(1 - p_t)^{\gamma} \log(p_t) \tag{3.2}$$

The parameter γ, also known as the focusing or relaxation parameter, is adjusted through cross-validation and determines how strongly the model concentrates on difficult, misclassified samples during training. Higher values of γ place greater emphasis on these challenging cases, whereas lower values maintain a more even focus between easy and hard examples. Despite this intent, it resulted in a lower ROC AUC and accuracy, indicating suboptimal performance for this task. Consequently, CrossEntropyLoss remained the preferred loss function for the model.

3.3.9 Evaluation Metric

Performance evaluation is essential for understanding how well a model differentiates between normal and abnormal events in video anomaly detection. Area Under the ROC Curve (AUC) and Average Precision (AP) are the two main evaluation metrics that are covered in this context. This is a thorough discussion of these measurements along with the necessary calculations.

A common metric to evaluate a binary classifier's performance (normal vs. aberrant) is the AUC. Plotting the Receiver Operating Characteristic (ROC) curve, which contrasts the True Positive Rate (TPR) with the False Positive Rate (FPR) at different threshold values, provides a summary of the model's

performance. The Receiver Operating Characteristic (ROC) curve shows the performance of a classification model, and the Area Under the Curve (AUC) reflects its capability to separate different categories. (Narkhede, 2018). True Positive Rate (TPR), often referred to as recall or sensitivity, represents the percentage of real positive instances, such as abnormal video frames, that are accurately classified by the model:

$$TPR = \frac{TP}{TP + FP} \tag{3.3}$$

False Positive Rate (FPR) measures the proportion of negatives (normal frames) that were incorrectly classified as positives (abnormal):

$$FPR = \frac{FP}{FP + TN} \tag{3.4}$$

The second evaluation metric is the average precision (AP), which summarises the precision–recall trade-off of the model. It is useful for imbalanced datasets, like anomaly detection, where there are fewer positive(abnormal) examples.

Precision measures the proportion of positive identifications that are correct:

$$P = \frac{TP}{TP + FP} \tag{3.5}$$

Recall evaluates the fraction of true positives that are accurately identified. AP is calculated by integrating the Precision-Recall (PR) curve, which visualizes precision versus recall across various threshold values. The actual area under the curve can be defined as:

$$AP = \int_0^1 p(r)dr \tag{3.6}$$

where $p(r)$ is the precision at recall r.

In this model, various metrics were used to evaluate performance, particularly for multiclass classification, where the task involves identifying anomalies across multiple classes. These metrics include F1-Micro, F1-Macro,

and Accuracy, each offering a unique viewpoint on how effectively the model distinguishes between different classes.

The F1-Macro score refers to the unweighted average of the F1-scores across all types of classes. It is calculated by:

$$F1 - Macro = \frac{1}{C}\sum_{i=1}^{C} F1 - Score_i \tag{3.7}$$

Where C is the number of classes, and $F1 - Score_i$ is the F1 score for class i. This metric gives an equal weight to all classes, regardless of their frequency, making it particularly useful when there is class imbalance, as it does not favor the majority class. The F1-Micro score combines the TPs, FPs, and FNs from all classes into a single metric, offering an overall assessment of the model's precision and recall across all decision thresholds. It is calculated as:

$$F1 - Micro = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.8}$$

This approach is useful for situations where all classes need to be treated equally, regardless of their individual sizes. The F1-Micro score tends to be higher when the model performs well on the majority class, but it can mask poor performance on minority classes.

Accuracy quantifies the overall correctness of the model by computing the ratio of correctly predicted instances to the total number of predictions made across all classes. It is calculated by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

3.3.10 Model Optimization

In this study, dynamic quantization was employed to optimize the AnomLite model for efficient inference, particularly on resource-constrained devices such as edge processors or CPUs. Quantization is a model compression technique that reduces the precision of the weights and, in some cases, activations from FP32 to a lower bit-width format, INT8. Specifically, dynamic quantization was chosen due to its simplicity and effectiveness for models with recurrent

components like LSTMs, which are central to AnomLite ’s architecture for processing temporal video data.

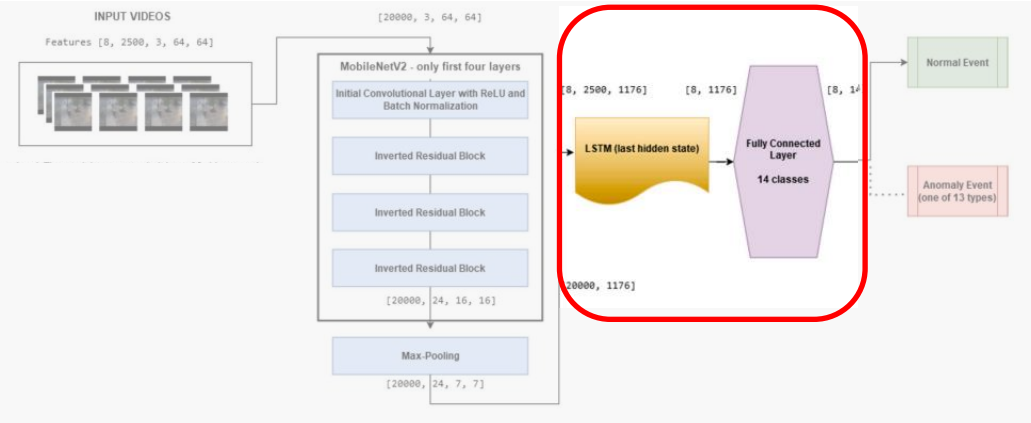


Figure 3.22: Layers Selected for Quantization

As shown in Figure 3.22, dynamic quantization is a post-training quantization (PTQ) that quantizes the weights of selected layers, which are the LSTM and Linear layers, at runtime while keeping the activations in floating point. This contrasts with quantization-aware training (QAT), which simulates quantization effects during the training process and typically yields higher accuracy but requires retraining with added complexity. These layers were selectively quantized to 8-bit integers (torch.qint8), while the convolutional layers and activation functions were preserved in their original floating-point 32-bit precision (FP32). This selective quantization approach allows the model to benefit from reduced memory and computational costs while maintaining the integrity of key components crucial for performance. This process involving PTQ did not require changes to the training pipeline or loss of model compatibility.

After quantization, the model exhibited reduced memory usage and inference time, while maintaining performance within acceptable limits. The use of dynamic quantization allowed for faster execution on CPU-based systems.

3.3.11 Inference Implementation

The sliding window approach is particularly advantageous for video anomaly detection models, such as the one used in this inference pipeline, because it allows the model to maintain temporal context while processing video in smaller, more manageable chunks. In anomaly detection, detecting abnormal events often relies on the model's ability to understand the progression of normal and

abnormal behaviors over time. The sliding window approach ensures that the model evaluates a consistent sequence of frames, which is essential for capturing the temporal patterns of normal and anomalous events.

Moreover, this method helps balance computational efficiency with accuracy. Instead of processing the entire video at once, which would be computationally expensive and slow, the sliding window allows for real-time processing by analyzing only a subset of frames at any given time. This not only reduces memory and processing time but also allows for continuous, incremental predictions as the video progresses. For anomaly detection, where subtle variations in behavior may appear over time, this approach is ideal as it ensures that the model can continuously learn from the evolving sequence of frames, making it more responsive to dynamic changes in the video stream

3.4 Gantt Chart

This section presents the Gantt charts for both semesters, with Figure 3.23 showing the chart for Semester 1 and Figure 3.24 for Semester 2.

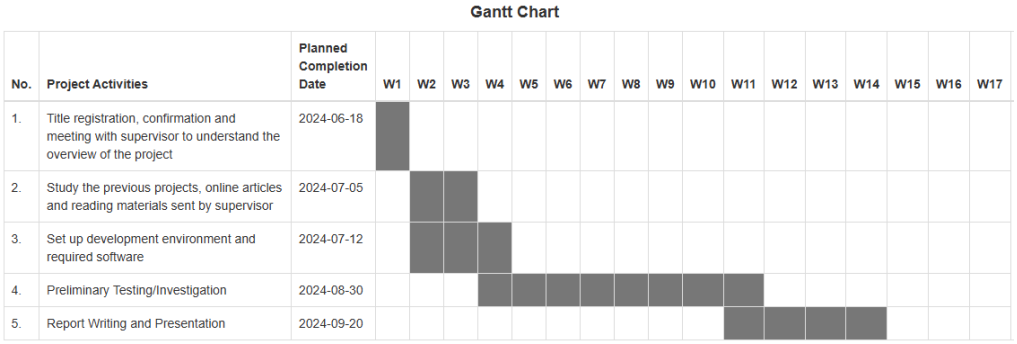


Figure 3.23: Gantt Chart for FYP1

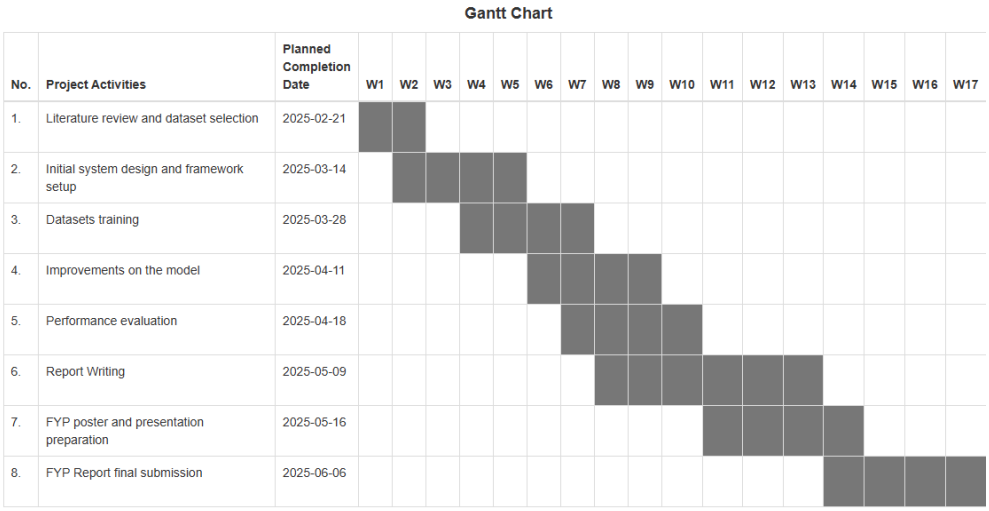


Figure 3.24: Gantt Chart for FYP2

3.5 Summary

This chapter details the comprehensive methodology used to develop the video anomaly detection system, encompassing dataset selection, preprocessing steps, model architecture design, training strategies, and the evaluation protocol. The study utilized two well-known benchmark datasets, including UCF-Crime, which contains real-world CCTV footage with 13 anomaly classes, and XD-Violence, composed of violence-related video clips from movies and YouTube featuring 6 categories. To ensure consistency and effectiveness, several key preprocessing steps were implemented. This included frame extraction using a 1-in-10 sampling rate for XD-Violence, sequence reconstruction by organizing frames in temporal order, and class balancing strategies such as trimming lengthy normal videos and applying SMOTE for oversampling underrepresented classes. To enhance the model's performance, a range of data augmentation methods were employed, such as adding Gaussian noise and blur, adjusting color through jittering, and rotating images.

The proposed framework, AnomLite, integrates MobileNetV2 and LSTM in a lightweight yet effective design. Specifically, the first four layers of MobileNetV2 were used for spatial feature extraction, reducing computational load while preserving visual features. The extracted features, represented as 1176-dimensional vectors, were input into an LSTM module to capture temporal dependencies. This was followed by a fully connected classification head that mapped the sequences into 14 separate classes. To address class imbalance and enhance the model's ability to generalize, the training process incorporated a combination of weighted cross-entropy loss. Additional regularization methods included dropout with a probability of 0.3, batch normalization, and early stopping based on validation performance. The model was trained using Google Colab's A100 GPU with 40GB of VRAM to ensure efficient computation and faster convergence.

For performance evaluation, the protocol included a mix of binary and multiclass metrics to fully assess the model’s effectiveness. This covered common evaluation standards such as AUC-ROC and Average Precision for detecting anomalies, as well as F1-Micro and F1-Macro scores for analyzing multiclass classification accuracy.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter outlines the training and testing processes for the video anomaly detection and classification model, their performance on two datasets, performance during inference, model optimization, and further model comparison. The model's effectiveness is demonstrated through metrics like AP, AUC, accuracy, and other performance metrics. The evaluation reflects the model’s accuracy in detecting normal and abnormal video segments.

4.2 Performance Evaluation on UCF-Crime

The AnomLite model is assessed using a range of critical performance metrics, evaluating different scenarios, binary and multi-class classification. These evaluation measures offer a thorough insight into the model's effectiveness in distinguishing between non-violence and violence video segments, even in the presence of class imbalance. The corresponding metric outcomes are presented below.

Table 4.1: Performance Metrics of AnomLite on UCF-Crime

Key metrics	Value
AP	0.99
AUC	0.99
Accuracy	0.94
F1-Macro	0.93
F1-Micro	0.93
F1-Weighted Avg	0.94
Recall (Macro Avg)	0.94
Precision (Macro Avg)	0.95

4.2.1 Confusion Matrix of AnomLite on UCF-Crime

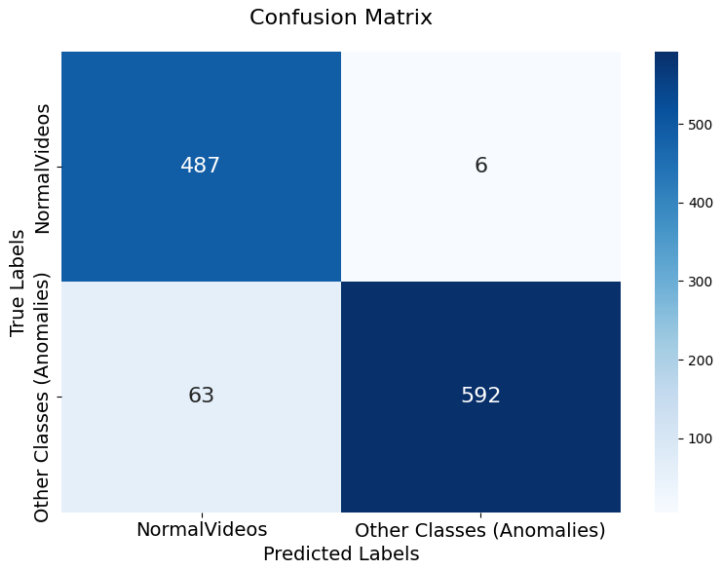


Figure 4.1: Confusion Matrix of AnomLite on UCF-Crime

Figure 4.1 shows the confusion matrix of the model on UCF-Crime, whereas the first rows refer to the normal class and the second row refers to the anomaly class. It is seen that the model correctly identified 487 normal videos (True positives), demonstrating strong specificity in recognizing non-anomalous events. The model also classified 592 actual instances of anomalies (True negatives). The low false positive rates, showing only 6 normal videos were misclassified as anomalous, and the low false negative rates, showing 63 anomalous videos were incorrectly flagged as normal.

4.2.2 ROC AUC of AnomLite on UCF-Crime

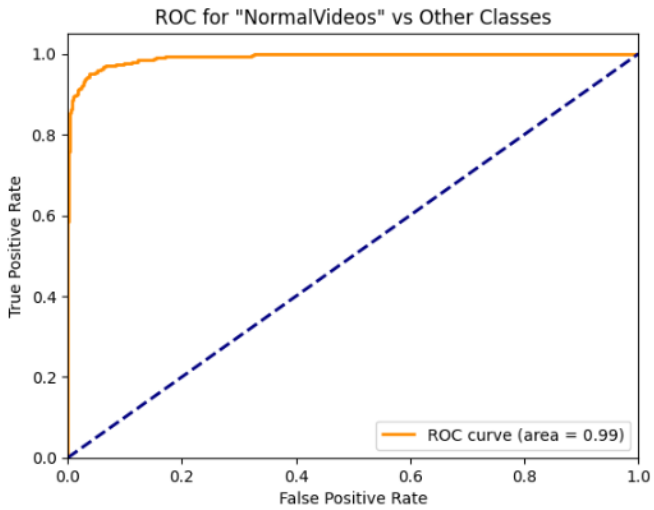


Figure 4.2: ROC AUC on UCF-Crime

Figure 4.2 illustrates the result of AnomLite on the UCF-Crime dataset, the ROC curve, which serves as a visual tool for evaluating the performance of a binary classification model across various threshold settings. The graph trends toward the upper left region, signifying that the model achieves a strong true positive rate alongside a low false positive rate across different thresholds. Additionally, the area under the ROC curve (AUC) is nearly 1, reflecting the model's strong ability to differentiate between normal and abnormal instances, and indicating reliable and consistent performance, while maintaining system robustness.

4.2.3 PR Curve of AnomLite on UCF-Crime

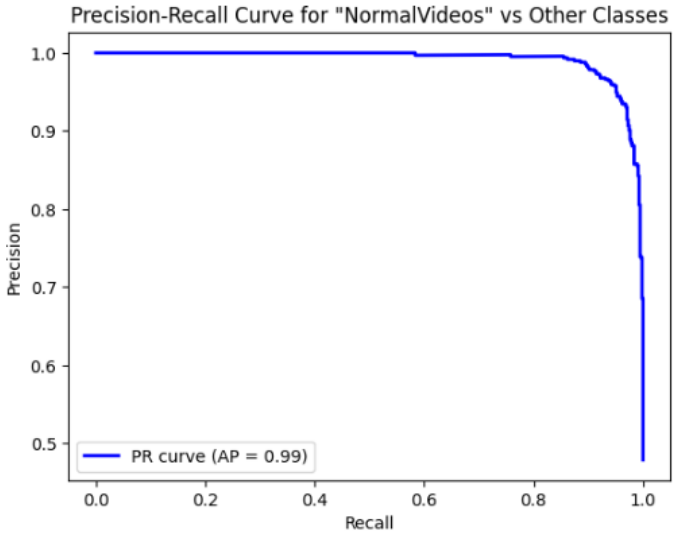


Figure 4.3: PR Curve on UCF-Crime

Figure 4.3 shows the PR Curve. It can be observed that a strong upward trend is curved towards the top-right corner, indicating that the model can distinguish anomalies from normal events well. Moreover, the area under the PR Curve (AP) provides a comprehensive summary of the model's effectiveness across the full range of recall values. An average precision of 0.99 indicates that the model maintains high precision while simultaneously detecting true anomalies, even under inherent class imbalance of video anomaly datasets.

4.2.4 F1 Scores and Loss

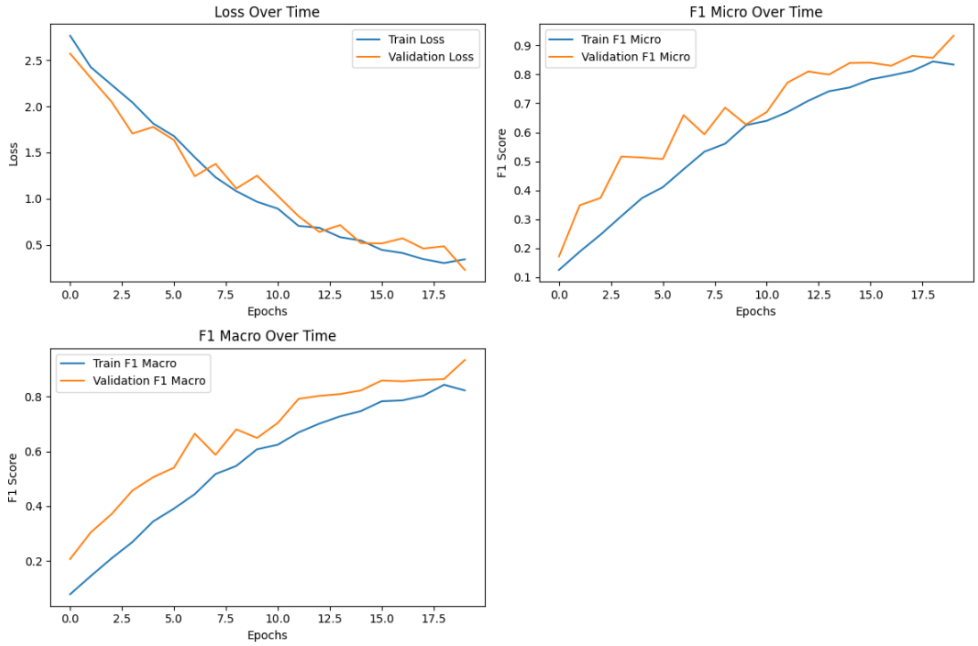


Figure 4.4: F1 Scores and Loss

Figure 4.4 shows the F1 Scores and loss over time. It can be observed that during training, both Training F1-micro and Validation F1-micro scores steadily increase, indicating that the model became more effective at correctly identifying both normal and anomalous events across all samples, regardless of class imbalance. Similarly, the growth in Training F1-macro and Validation F1-macro scores reflects improved performance across each class equally, suggesting that the model not only learned to detect dominant classes but also performed better on minority classes. Meanwhile, the gradual decrease in both training and validation loss demonstrates that the model continuously minimized prediction errors and improved its confidence over time.

4.2.5 Per-Class Accuracy

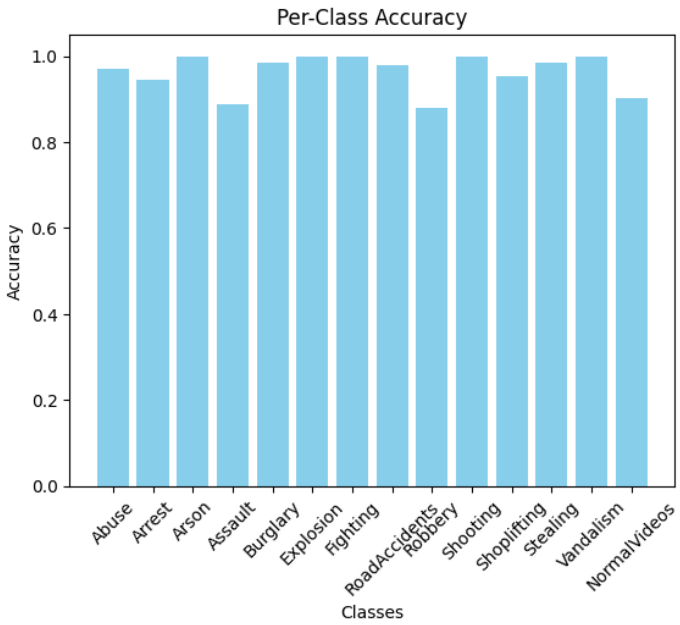


Figure 4.5: Per-Class Accuracy on 14 Classes in UCF-Crime Dataset

Table 4.2: Per-Class Accuracy on 14 Classes in UCF-Crime Dataset

Class	Accuracy
0 (Abuse)	0.97
1 (Arrest)	0.94
2 (Arson)	1.00
3 (Assault)	0.89
4 (Burglary)	0.98
5 (Explosion)	1.00
6 (Fighting)	1.00
7 (Road Accidents)	0.98
8 (Robbery)	0.88
9 (Shooting)	1.00
10 (Shoplifting)	0.95
11 (Stealing)	0.99
12 (Vandalism)	1.00
13 (Normal Videos)	0.90

Figure 4.5 and Table 4.2 present the per-class accuracy for 14 classes in the UCF-Crime dataset. The high per-class accuracy demonstrates that the model is effective at distinguishing between different categories of video anomalies.

4.3 Performance Evaluation on XD-Violence

This section demonstrates the model's performance on the XD-Violence dataset, which comprises a wide variety of video scenes collected from sources like YouTube, movies, and online platforms. Different from UCF-Crime dataset, which primarily consists of real-world CCTV surveillance footage, XD-Violence focuses on a wide range of environments, camera angles, and scene dynamics. The performance evaluation of the model AnomLite offers a thorough insight into the model's effectiveness. The results of these metrics are as follows:

Table 4.3: Performance Metrics of AnomLite on XD-Violence

Key metrics	Value
AP	0.97
ROC AUC	0.98
Accuracy	0.93
F1-Macro	0.87
F1-Micro	0.89
F1-Weighted Avg	0.93
Recall (Macro Avg)	0.91
Precision (Macro Avg)	0.94

4.3.1 Confusion Matrix of AnomLite on XD-Violence

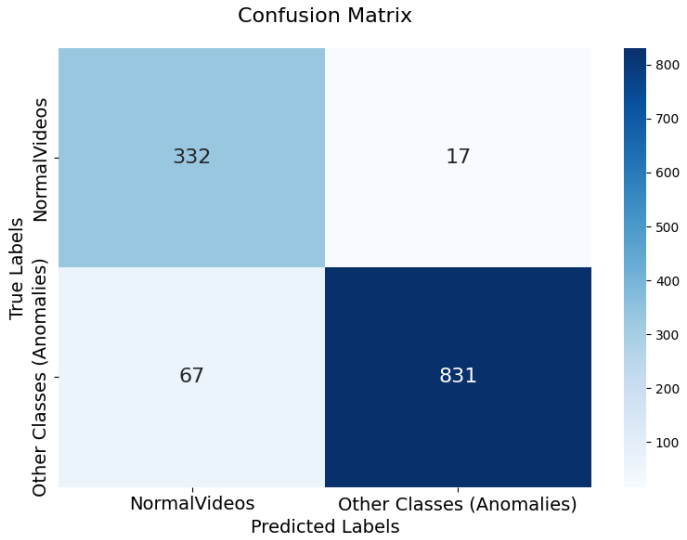


Figure 4.6: Confusion Matrix of AnomLite on XD-Violence

Figure 4.6 shows the confusion matrix of the model on the second dataset, XD-Violence, in which 332 normal videos are correctly identified (True Positives) and 831 anomalous videos were correctly flagged as anomalies (True Negatives). The low false positive of 17 indicates 17 normal videos were wrongly labelled as anomalous, and false negative of 67 indicates 67 anomalous videos were incorrectly flagged as normal.

4.3.2 ROC AUC of AnomLite on XD-Violence

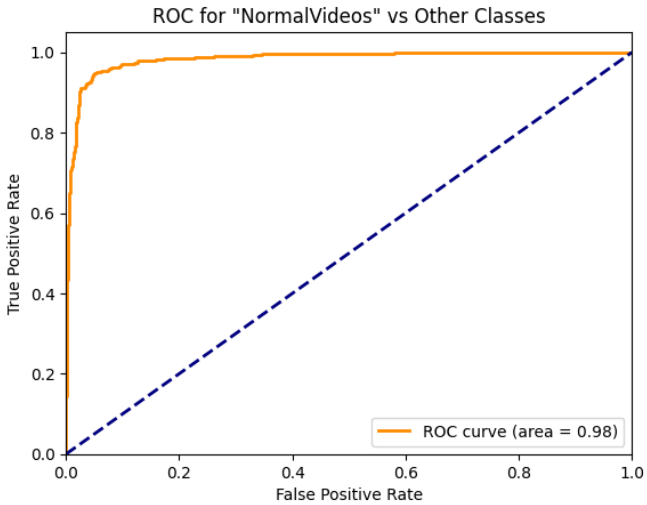


Figure 4.7: ROC AUC Curve of AnomLite on XD-Violence

Figure 4.7 displays the ROC curve, where the line bends toward the top-left corner. This shows that the model achieves a high true positive rate and a low false positive rate across different threshold values. In addition, the area under

the ROC curve (AUC) is 0.98, which indicates the model performs well in telling apart normal and abnormal events, while staying consistent and reliable.

4.3.3 Average Precision (AP) of AnomLite on XD-Violence

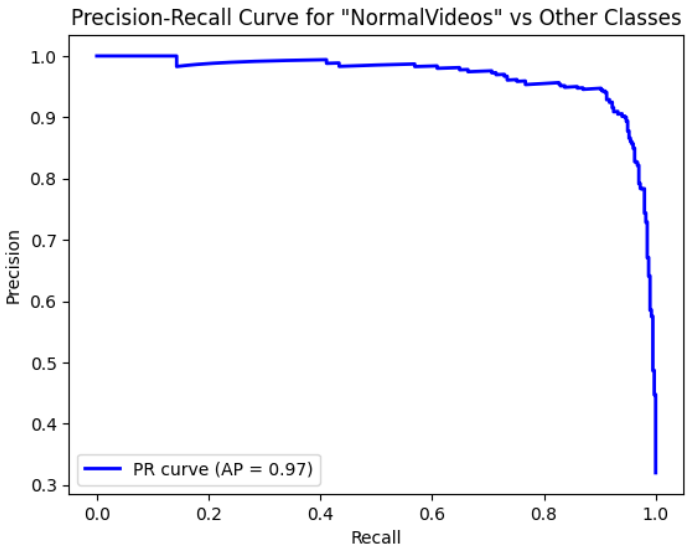


Figure 4.8: PR Curve on XD-Violence dataset

Figure 4.8 illustrates the Precision-Recall (PR) Curve for the XD-Violence dataset. A noticeable upward curvature towards the top-right corner reflects the model's strong capability in differentiating anomalous events from normal ones. Additionally, the area under the PR Curve, represented by the Average Precision (AP), captures the model's performance across varying recall thresholds. With an AP of 0.97, the model demonstrates high precision while effectively identifying true anomalies, even in the presence of the class imbalance typical of video anomaly detection datasets.

4.3.4 F1 Scores and Losses

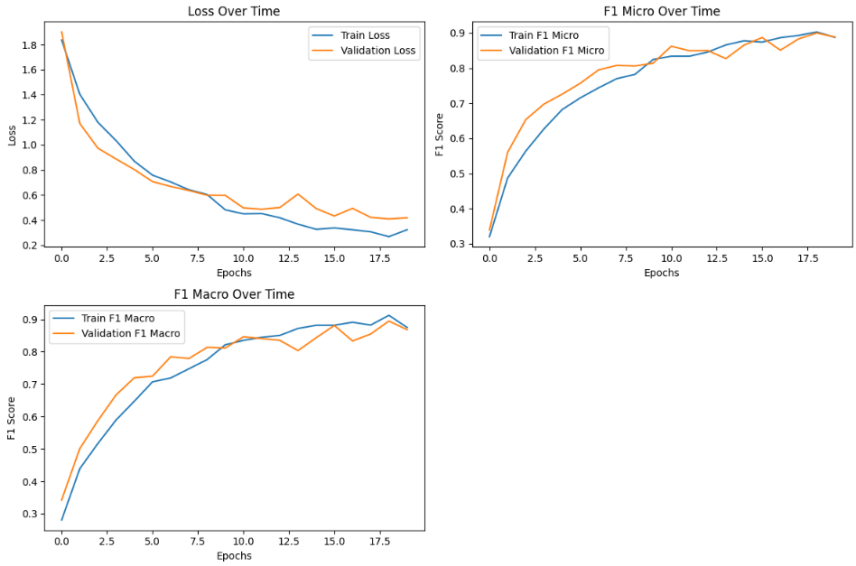


Figure 4.9: F1 scores and losses on XD-Violence dataset

Figure 4.9 shows the F1 Scores and loss trend during training on the XD-Violence dataset. Both Training and Validation F1-micro steadily increased and demonstrated the model's improved ability to identify normal and anomalous events overall. The improvement in F1-macro scores indicates better performance across all classes, including less frequent ones. In the meantime, the steady drop in loss points to fewer errors in predictions and increasing model confidence.

4.3.5 Per-Class Accuracy

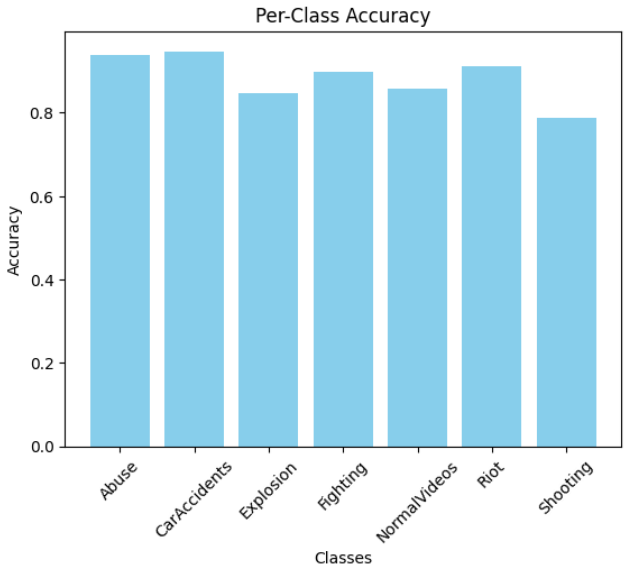


Figure 4.10: Per-Class Accuracy on XD-Violence Dataset

Table 4.4: Per-Class Accuracy on XD-Violence

Class	Accuracy
Class 0 (Abuse)	0.94
Class 1 (Car Accidents)	0.95
Class 2 (Explosion)	0.85
Class 3 (Fighting)	0.90
Class 4 (Normal Videos)	0.86
Class 5 (Riot)	0.91
Class 6 (Shooting)	0.79

Figure 4.10 and Table 4.4 present the per-class accuracy for 6 classes in the XD-Violence dataset. The high per-class accuracy demonstrates that the model is effective at distinguishing between 6 different categories of video anomalies.

4.4 Performance of AnomLite model (Inference)

This section presents the outcome of the model's prediction on previously unseen real-world videos, alongside its performance during real-time inference.

4.4.1 Prerecorded video

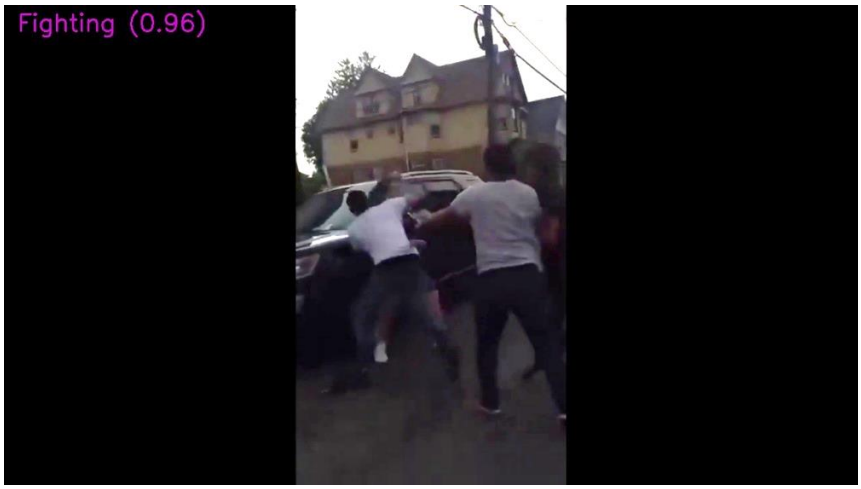
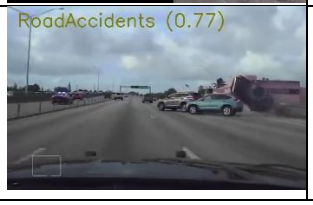
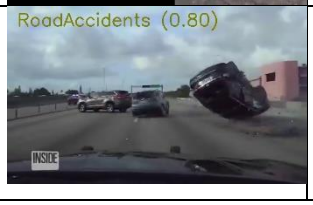


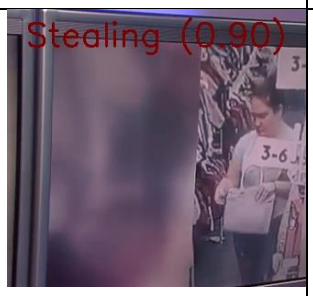
Figure 4.11: A Detected Frame classified as 'Fighting' on Unseen Data

Figure 4.12: Prediction of the model on ‘Fighting’ frames by frames on a video sequence

Figure 4.11 shows an example of a detected frame classified as 'Fighting', whereas Figure 4.12 illustrates the frame-by-frame prediction of the model on a video sequence depicting a ‘Fighting’ event. Each frame’s classification output indicates how the model detects and localizes the anomalous event activity over time. This visualization helps assess the model’s temporal consistency and sensitivity in identifying suspicious behaviour as it unfolds within a continuous video stream.

Table 4.5: Model predictions on unseen real-world videos, multiclass labelled

Arson			
Abuse			
Arrest			
Burglary			
Explosion			
Fighting			
Road Accidents			

Robbery			
Stealing			
Vandalism			

The model's predictions on unseen real-world videos are shown in Table 4.5, along with multiclass labels for different anomalous events. Each row represents a distinct class of event, such as 'Arson,' 'Abuse,' 'Arrest,' and others, showing how effectively the model identifies different types of incidents. The table demonstrates how the model can generalize to a variety of situations, correctly categorizing films that are not included in the training set. This evaluation is critical to understanding the model's robustness and its potential for real-world deployment in surveillance systems.

4.4.2 Real-Time Streaming

This section demonstrates the model's detection results for the 'Fighting' event captured from various angles in the campus library, as illustrated in Figures 4.13, 4.14, 4.15, and 4.16.

Figure 4.13: Fighting detection from bottom view

Figure 4.14: Detection of Fighting at the corner

Figure 4.15: Detection of Fighting from another side view

Figure 4.16: Detection of Fighting from another higher view

This section demonstrates the model's performance in detecting 'Fighting' events from multiple perspectives within a campus library environment, as shown in Figures 4.13 to 4.16. The model accurately identifies the event from different camera angles, including bottom, corner, side, and higher vantage points, thus illustrating its robustness across varying environmental conditions.

4.4.3 Summary of Inference

In short, two testing strategies were used in this study to evaluate the model's inference abilities in real-world situations. In the first method, pre-recorded videos were uploaded, especially for unusual occurrences like arson, explosions, or armed robberies that are dangerous or difficult to reproduce in real life. These videos, often sourced from public datasets or online platforms, ensured the inclusion of rare and complex events during evaluation. The second strategy focused on incidents that are comparatively simpler and safer to mimic, including fighting, and used real-time video capture via a webcam. This real-time testing made it possible to evaluate the model's flexibility and reactivity in dynamic, real-time environments. The results of this approach are shown in Section 4.4.2, which shows that fighting incidents that were captured from different perspectives inside the campus library setting were successfully detected.

4.5 Model Optimization

4.5.1 Results of Model Optimization

```
================================================================================
Quantization Verification
================================================================================
Is rnn quantized? True
Is fc layer quantized? True
FC layer dtype: torch.qint8
```

Figure 4.17: Quantization Verification

```
    (3): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=144, bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (rnn): DynamicQuantizedLSTM(1176, 1176, batch_first=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (batch_norm): BatchNorm1d(1176, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc): DynamicQuantizedLinear(in_features=1176, out_features=14, dtype=torch.qint8, qscheme=torch.per_tensor_affine)
)
```

Figure 4.18: Parts of Quantized Model Architecture

```
Quantized Layer Details:

rnn: <class 'torch.ao.nn.quantized.dynamic.modules.rnn.LSTM'>
  - Input size: 1176
  - Hidden size: 1176
  - All quantization parameters are calculated dynamically during inference

fc: <class 'torch.ao.nn.quantized.dynamic.modules.linear.Linear'>
  - Dtype: torch.qint8
```

Figure 4.19: Quantized Layer Details

Figures 4.17, 4.18, and 4.19 show that dynamic post-training quantization approach has been successfully applied to the model. The model's architecture mainly consists of LSTM (RNN) layers and fully connected (FC) layers. As shown in Figures 4.18 and 4.19, these layers were selectively quantized to 8-bit integers (torch.qint8), while the convolutional layers and activation functions were preserved in their original floating-point 32-bit precision (FP32). This selective quantization approach allows the model to benefit from reduced memory and computational costs while maintaining the integrity of key components crucial for performance.

The quantized layers demonstrated a significant reduction in memory usage, resulting in an approximately 4× decrease in overall model size. This

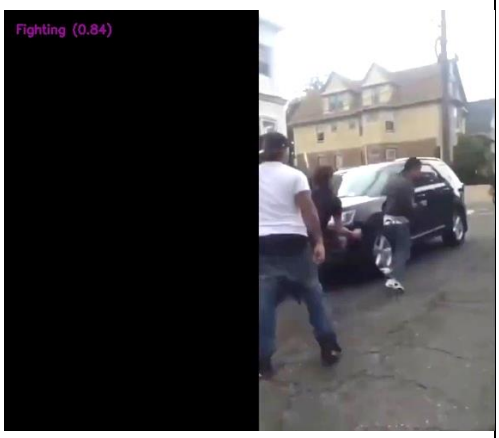
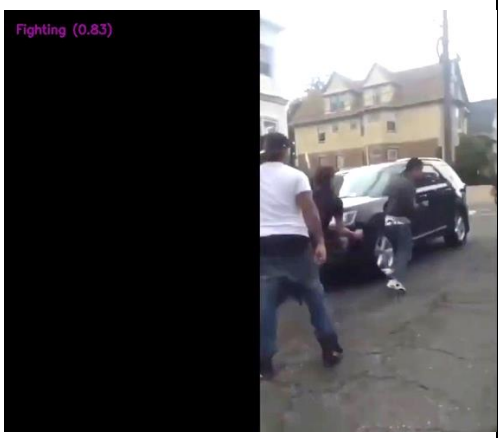
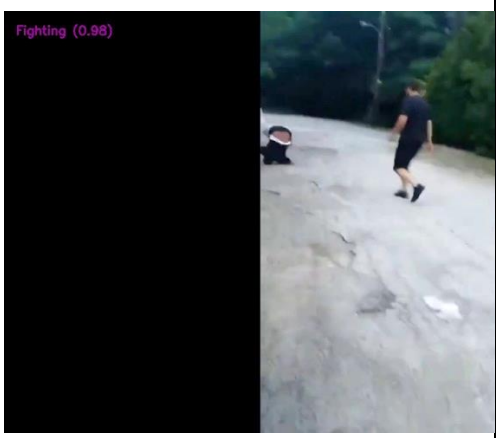
optimization significantly enhances the model's deployability on edge devices with limited storage capacity. Table 4.6 shows the comparative analysis of the original and quantized models.

Table 4.6: Comparative Analysis of Original vs. Quantized Model

Feature	Original Model	Quantized Model
LSTM Type	LSTM	DynamicQuantizedLSTM
Linear Layer (FC)	FP32	torch.qint8
Weight Storage	32-bit	8-bit
Activation Precision	FP32	Dynamic FP32 converted to int8 on-the-fly

4.5.2 Performance of Quantized Model

Table 4.7: Comparison of performance of both models on the same video frames

Original Model	Quantized Model
Confidence: Fighting (0.84)	Confidence: Fighting (0.83)
	
Confidence: Fighting (0.84)	Confidence: Fighting (0.83)
	
Confidence: Fighting (0.98)	Confidence: Fighting (0.97)
	
Confidence: Arson (0.74)	Confidence: Arson (0.73)

To compare the performance between the original model and the quantized model, random frames were selected. Table 4.7 shows the results of performance results of both models on the same video frames. It is observed that there is only a minimal impact on the model's output confidence scores. When evaluated on identical input frames, the quantized model demonstrated an average confidence reduction of only 0.01 relative to the original full precision (FP32) model, as illustrated in Figure 4.20.

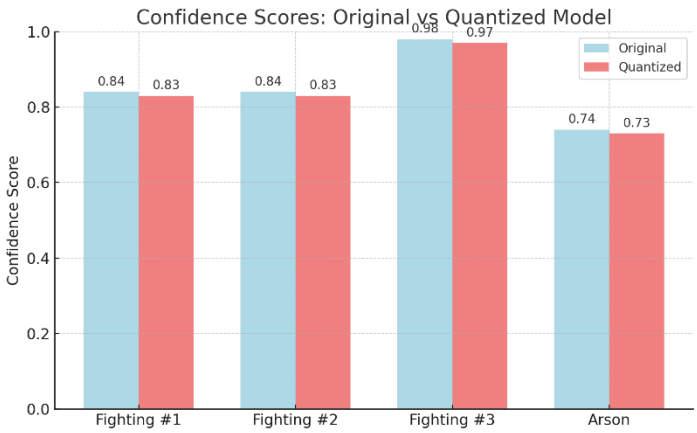


Figure 4.20: Comparison of Model Confidence for Both Models

Importantly, the quantized model retained the ability to detect anomalous events with almost identical certainty, ensuring consistent prediction quality. Thus, the selective dynamic quantization, especially for LSTM and fully connected layers, retained the performance and enables substantial efficiency gains without compromising model accuracy.

4.6 Performance Evaluation of Original and Optimized Model

Quantization not only reduces the precision of model weights and activations but also significantly enhances the model's computational efficiency and deployability. As shown in Table 4.8, the original full-precision model occupies approximately 42.4 MB, whereas the quantized model is reduced to just 10.7 MB, resulting in a compression ratio of nearly 4×. This substantial reduction in memory footprint is critical for deployment on resource-constrained devices, particularly those without dedicated GPUs.

Table 4.8: Comparison of Computational Resources before and after Quantization

Aspect	Before	After
Model Size	42.4 MB	10.7 MB
Hardware Requirement	A100 GPU (40GB VRAM)	CPU Only

Moreover, the quantized model no longer depends on high-end hardware. The original model required an NVIDIA A100 GPU with 40GB VRAM for inference, whereas the quantized version operates efficiently on a CPU-only environment, significantly expanding its potential for real-time applications in edge computing and embedded systems.

Table 4.9 further highlights the runtime performance, particularly the frames per second (FPS) throughput for both the original and quantized models. When evaluated on two video sources, the original FP32 model, executed on a high-end GPU (A100 GPU), achieved 44.67 FPS and 39.17 FPS, respectively. In contrast, the quantized model running on CPU achieved 15.16 FPS and 9.21 FPS, which, while lower than the GPU version, remains viable for near real-time processing. Given the absence of GPU dependency, this performance represents a favourable trade-off between speed and resource efficiency.

Table 4.9: Comparison of FPS on Both Models

Source	FPS (on GPU)	FPS (on CPU)
Video 1	44.67	15.16
Video 2	39.17	9.21

Together, these results emphasize the practical benefits of quantization in terms of model size reduction, hardware flexibility, and computational efficiency, all while maintaining acceptable real-time processing speeds and

detection accuracy. This makes the quantized model a compelling choice for low-power deployments such as surveillance cameras, mobile devices, or embedded systems.

4.7 Performance Comparison with BN-WVAD

This section presents the comparative analysis between the AnomLite model and BN-WVAD, one of the top-performing methods with a high ranking on Papers with Code. BN-WVAD incorporates batch normalization to enhance training stability and has demonstrated strong performance in the weakly supervised video anomaly detection task. Therefore, the model is implemented to facilitate a comparison with the previous approach. The performance of the BN-WVAD model is shown on WandB, an AI developer platform used to track machine learning work.

4.7.1 Performance on XD-Violence

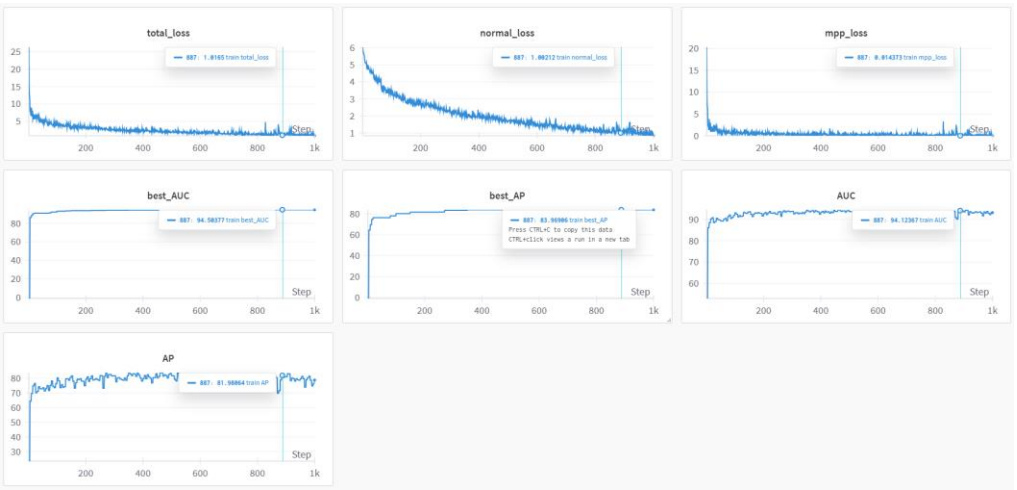


Figure 4.21: Charts of Performance on Wandb (XD-Violence)

Table 4.10: Performance Metrics of BN-WVAD on XD-Violence

Metrics	Result
AP	78.78
AUC	93.18
best_AP	83.97
best_AUC	94.50

Figures 4.21 and Table 4.10 present the results of BN-WVAD on the XD-Violence dataset. While the performance is quite good, achieving a high AUC of 94.50 and AP of 83.97, it does not surpass the results achieved by AnomLite, indicating there is still room for improvement in capturing certain types of anomalies.

4.7.2 Performance on UCF-Crime

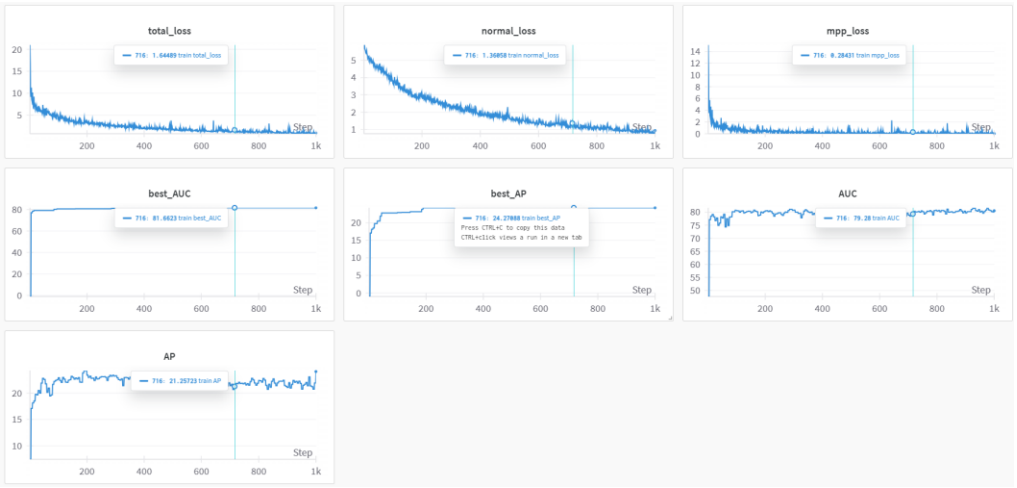


Figure 4.22: Charts of Performance on Wandb (UCF-Crime)

Table 4.11: Performance Metrics of BN-WVAD on UCF-Crime

Metrics	Result
AP	36.26
AUC	87.24
best_AP	38.13
best_AUC	87.24

Figure 4.22 and Table 4.11 present the performance of BN-WVAD on the UCF-Crime dataset. The model achieves a good AUC of 87.24, indicating strong discrimination capability. However, the AP of 36.26 is relatively low, suggesting that while the model can distinguish between normal and anomalous events, it struggles with the precise localization of anomalies. Compared to AnomLite, the performance is not as strong, particularly in terms of AP, highlighting areas where further improvement is needed.

4.7.3 Overview of Comparison

Table 4.12: Comparison of Performance Metrics of Both Models

Dataset	Metrics	BN-WVAD	MobileNetV2-LSTM
UCF-Crime	AP	0.36	0.99
	AUC	0.87	0.99
XD-Violence	AP	0.79	0.97
	AUC	0.93	0.98

This comparison highlights that AnomLite outperforms BN-WVAD on both the UCF-Crime and XD-Violence datasets concerning Average Precision (AP) and Area Under Curve (AUC). For UCF-Crime dataset, AnomLite achieves a high AP and AUC of 0.99, while BN-WVAD lags with a much lower AP of 0.36 and AUC of 0.87. For XD-Violence dataset, AnomLite maintains strong performance, AP of 0.97 and AUC of 0.98, still ahead of BN-WVAD, which has an AP of 0.79 and AUC of 0.93. Although BN-WVAD introduces a unique methodology, it showed lower performance in addressing imbalanced datasets in comparison to AnomLite, which utilizes a weighted cross-entropy loss.

4.8 Summary

Chapter 4 provides an in-depth assessment of video anomaly detection through the AnomLite architecture. The results are organized to compare baseline performance from the baseline model and enhancements introduced in this work. Section 4.2 reports the performance of the baseline model on the UCF-Crime dataset. Evaluation includes the confusion matrix, ROC AUC, precision-recall (PR) curves, F1 scores, and per-class accuracy. These results serve as a benchmark for subsequent comparisons.

Starting from Section 4.3, the chapter transitions to this work's contributions, applying the AnomLite model to the XD-Violence dataset, which is one of the well-known datasets in video anomaly detection. This section mirrors the evaluation methodology used earlier, presenting the confusion matrix, ROC AUC, average precision (AP), F1 scores, and per-class accuracy, thereby demonstrating the model's effectiveness on a more complex, multimodal dataset.

Section 4.4 evaluates the inference capabilities of the trained model on both prerecorded and real-time video streams, highlighting its potential for real-world deployment. A summary of inference behavior is provided.

Section 4.5 introduces model optimization techniques, particularly quantization, to reduce computational overhead. Experimental results show that these optimizations preserve performance while improving efficiency, enabling the model to perform effectively in resource-constrained settings, optimizing the model for accessibility on lower-end GPUs, even CPUs, making it more user-friendly. In Section 4.6, the optimized model's inference performance is further analyzed to ensure its robustness under real-time constraints.

Finally, Section 4.7 offers a detailed performance comparison between this AnomLite model and the BN-WVAD model. The evaluation across both XD-Violence and UCF-Crime datasets provides a comprehensive overview of the improvements achieved, emphasizing gains in accuracy, inference, and deployment viability.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, this research presents an optimized deep learning framework for video anomaly detection, addressing key challenges in computational efficiency and class imbalance across diverse surveillance scenarios. The lightweight AnomLite architecture, with only 11 million parameters, achieved good performance on both of the datasets, UCF-Crime and XD-Violence, with ROC AUC of 0.99 and average precision of 0.99 on UCF-Crime and a ROC AUC of 0.98 and average precision of 0.97 on XD-Violence. The model also achieves high accuracy of 94% on UCF-Crime and 93% on XD-Violence, with strong F1 scores across both datasets (F1-Micro 0.93 on UCF-Crime, 0.89 on XD-Violence. Moreover, inference on prerecorded videos and real-time capturing of different angles successfully captured and identified the correct anomalous events that happened. Moreover, further optimizations achieved remarkable efficiency gains. Memory optimization has been successfully implemented, resulting in a 70% reduction in model size, from 42.4MB to 10.7MB, through strategic quantization. This approach balances accuracy and inference speed, making it well-suited for deployment on low-resource edge devices using only CPU.

Despite the challenges, such as imbalanced dataset, memory hardware requirement, cross cross-entropy loss function with SMOTE was applied to deal with the imbalanced dataset problem. With further data augmentations such as Gaussian noise, blur, and colour jitter augmentations, this enhances the robustness of the model.

This research contributes to advancing the anomaly detection field by balancing efficiency and effectiveness, particularly in the challenging domain of multi-class anomaly recognition.

5.2 Recommendations for future work

In future work, the model can be further optimized by expanding dataset coverage, that is, collecting additional video samples of visually similar but distinct anomaly classes (e.g., *burglary vs. robbery*, *shoplifting vs. stealing*) to refine the model's ability to differentiate subtle behavioral differences. Besides, future work should focus on developing its capability to detect and classify multiple anomalous events within the same video frame. This enhancement would involve developing a more comprehensive dataset containing complex scenarios where multiple anomalies co-occur, such as a robbery taking place while a fight erupts nearby, or shoplifting occurring simultaneously with property vandalism. Moreover, future work could explore deploying the model using Intel OpenVINO to further improve real-time inference performance, particularly on Intel-based edge devices. Building on the current use of dynamic quantization, the model can be further optimized through techniques like layer fusion and hardware-aware acceleration.

REFERENCES

Allard, M. (2020). *What is a Transformer?* [online] Medium. Available at: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04.

Carreira, J. and Zisserman, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:https://doi.org/10.1109/cvpr.2017.502.

Feng, J.-C., Hong, F.-T. and Zheng, W.-S. (2021). MIST: Multiple Instance Self-Training Framework for Video Anomaly Detection. *arXiv (Cornell University)*. doi:https://doi.org/10.1109/cvpr46437.2021.01379.

Georgescu, M.-I., Barbalau, A., Radu Tudor Ionescu, Fahad Shahbaz Khan, Popescu, M. and Shah, M. (2021). Anomaly Detection in Video via Self-Supervised and Multi-Task Learning. *Computer Vision and Pattern Recognition*. doi:https://doi.org/10.1109/cvpr46437.2021.01255.

Jaime-Rodrigo González-Rodríguez, Diana-Margarita Córdova-Esparza, Terven, J. and Julio-Alejandro Romero-González (2024). Towards a Bidirectional Mexican Sign Language–Spanish Translation System: A Deep Learning Approach. Technologies (Basel), 12(1), pp.7–7. doi:https://doi.org/10.3390/technologies12010007.

Kalita, D. (2022). *Basics of CNN in Deep Learning*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/.

Kanjilal, J. (2022) *An introduction to weakly supervised learning*. https://blog.paperspace.com/an-introduction-to-weakly-supervised-learning/.

Kim, Y., Kwak, G.-H., Lee, K.-D., Na, S.-I., Park, C.-W. and Park, N.-W. (2018). Performance Evaluation of Machine Learning and Deep Learning Algorithms in Crop Classification: Impact of Hyperparameters and Training Sample Size. *Korean Journal of Remote Sensing*, [online] 34(5), pp.811–827. doi:https://doi.org/10.7780/kjrs.2018.34.5.9.

Kolena.com. (2024). *Average Precision - Testing with Kolena*. [online] Available at: https://docs.kolena.com/metrics/average-precision/#:~:text=Average%20precision%20(AP)%20summarizes%20a [Accessed 15 Sep. 2024].

Kulhary, R. (2019). *OpenCV - Overview*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/opencv-overview/.

Kwok, R. (2022). *GNN notes series — Explain Graph Convolutional Networks (GCN) with knowledge in CNN*. [online] Medium. Available at: https://medium.com/@rmwkwok/gnn-notes-series-explain-graph-convolutional-networks-gcn-with-knowledge-in-cnn-b827be1c872b [Accessed 16 Sep. 2024].

Luo, W., Liu, W. and Gao, S. (2017). A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework. *2017 IEEE International Conference on Computer Vision (ICCV)*. doi:https://doi.org/10.1109/iccv.2017.45.

Mansour, R.F., Escorcia-Gutierrez, J., Gamarra, M., Villanueva, J.A. and Leal, N. (2021). Intelligent video anomaly detection and classification using faster RCNN with deep reinforcement learning model. *Image and Vision Computing*, 112, p.104229. doi:https://doi.org/10.1016/j.imavis.2021.104229.

Matthew N. Bernstein. (2023). *Graph convolutional neural networks*. [online] Available at: https://mbernste.github.io/posts/gcn/.

Narkhede, S. (2018). *Understanding AUC - ROC Curve*. [online] Medium. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

NVIDIA (n.d.). *What is PyTorch?* [online] NVIDIA Data Science Glossary. Available at: https://www.nvidia.com/en-us/glossary/pytorch/.

Overload, D. (2022). *Sliding Window Technique — reduce the complexity of your algorithm*. [online] Medium. Available at: https://medium.com/@data-overload/sliding-window-technique-reduce-the-complexity-of-your-algorithm-5badb2cf432f.

Pytorch.org. (2024). *Dynamic Quantization — PyTorch Tutorials 2.7.0+cu126 documentation*. [online] Available at: https://docs.pytorch.org/tutorials/recipes/recipes/dynamic_quantization.html.

Ray, J. (2024). *Quantization Aware Training (QAT) vs. Post-Training Quantization (PTQ)*. [online] Medium. Available at: https://medium.com/better-ml/quantization-aware-training-qat-vs-post-training-quantization-ptq-cd3244f43d9a.

Raziyeh Pourdarbani, Sajad Sabzi, Reihaneh Zohrabi, Ginés García-Mateos, Fernandez-Beltran, R., José Miguel Molina-Martínez and Rohban, M.H. (2023). Comparison of 2D and 3D convolutional neural networks in hyperspectral image analysis of fruits applied to orange bruise detection. *Journal of food science*, 88(12), pp.5149–5163. doi:https://doi.org/10.1111/1750-3841.16801.

Sertis (2023). *Video Anomaly Detection: An Introduction - Sertis - Medium*. [online] Medium. Available at: https://sertiscorp.medium.com/video-anomaly-detection-an-introduction-232bf48c9a8d.

Shah, D. (2022). *Mean Average Precision (mAP) Explained: Everything You Need to Know*. [online] www.v7labs.com. Available at: https://www.v7labs.com/blog/mean-average-precision.

Steen, D. (2020). *Precision-Recall Curves*. [online] Medium. Available at: https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248.

Sultani, W., Chen, C. and Shah, M. (2018). Real-World Anomaly Detection in Surveillance Videos. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. doi:https://doi.org/10.1109/cvpr.2018.00678.

Teki, S. (2022). *Knowledge Distillation: Principles, Algorithms, Applications*. [online] neptune.ai. Available at: https://neptune.ai/blog/knowledge-distillation.

Tian, Y., Pang, G., Chen, Y., Singh, R., Johan Verjans and Carneiro, G. (2021). Weakly-supervised Video Anomaly Detection with Robust Temporal Feature Magnitude Learning. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. doi:https://doi.org/10.1109/iccv48922.2021.00493.

Transactions on Circuits and Systems for Video Technology, pp.1–1. doi:https://doi.org/10.1109/tcsvt.2024.3450734.

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. and Polosukhin, I. (2017). *Attention Is All You Need*. [online] Available at: https://arxiv.org/pdf/1706.03762.

Vina, A. (2024). *What is Model Optimization? A Quick Guide*. [online] Ultralytics.com. Available at: https://www.ultralytics.com/blog/what-is-model-optimization-a-quick-guide.

VK (2024). *Model Optimization Techniques (Pruning, Quantization, Knowledge Distillation, Sparsity, OpenVino Toolkit)*. [online] Medium. Available at: https://medium.com/@VK_Venkatkumar/model-optimization-techniques-pruning-quantization-knowledge-distillation-sparsity-2d95aa34ea05.

Williamson, B. (2021). *What Is Deep Learning With Python?* [online] Flatiron School. Available at: https://flatironschool.com/blog/what-is-deep-learning-with-python/.

Wu, P., Liu, J., Shi, Y., Sun, Y., Shao, F., Wu, Z. and Yang, Z. (2020). *Not only Look, but also Listen: Learning Multimodal Violence Detection under Weak Supervision*. [online] Available at: https://arxiv.org/pdf/2007.04687 [Accessed 14 Sep. 2024].

Zhong, J.-X., Li, N., Kong, W., Liu, S., Li, T.H. and Li, G. (2019). Graph Convolutional Label Noise Cleaner: Train a Plug-And-Play Action Classifier for Anomaly Detection. doi:https://doi.org/10.1109/cvpr.2019.00133.

Zhou, H., Liu, H., and Wu, X., 2024. *Video Anomaly Detection in 10 Years: A Survey and Outlook*. [online] Available at: https://arxiv.org/abs/2405.19387v1 [Accessed 14 September 2024].

Zhou, H., Yu, J. and Yang, W. (2023). Dual Memory Units with Uncertainty Regulation for Weakly Supervised Video Anomaly Detection. *Proceedings of the ... AAAI Conference on Artificial Intelligence*, 37(3), pp.3769–3777. doi:https://doi.org/10.1609/aaai.v37i3.25489.

Zvereva, A.K., Kaprielova, M. and Andrey Grabovoy (2025). AnomLite : Efficient Binary and Multiclass Video Anomaly Detection. *Results in Engineering*, pp.104162–104162. doi:https://doi.org/10.1016/j.rineng.2025.104162.

Zhou, Y., Qu, Y., Xu, X., Shen, F., Song, J. and Heng Tao Shen (2024). BatchNorm-based Weakly Supervised Video Anomaly Detection. *IEEE*