TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

PROJECT REPORT

ON

ANOMALY DETECTION IN SURVILLENCE VIDEOS

SUBMITTED BY:

JIWAN PRASAD GURAGAIN (PUL075BCT041)

KUSHAL SHRESTHA (PUL075BCT045)

LAXMAN KUNWAR (PUL075BCT046)

YAMAN SUBEDI (PUL075BCT044)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

May,2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Anomaly Detection in Surveillance Videos"** submitted by **Jiwan Prasad Guragain**, **Kushal Shrestha**, **Laxman Kunwar**, **Yaman Subedi** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
Prof. Dr. Nanda Bikram Adhikari
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner

Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgement

We are grateful to have had the opportunity to work on this project, and We would like to extend our sincere thanks to all those who have supported us along the way.

We would like to begin by expressing our deepest appreciations to our project supervisor, **Prof. Dr. Nanda Bikram Adhikari**, for his exceptional guidance, encouragement, and invaluable insights throughout the project. His unwavering support and timely feedback have been instrumental in the successful completion of this project.

Our sincere gratitude goes to our **colleagues**, for their support, feedback, and encouragement throughout the project. Their constructive criticism and suggestions have been critical in shaping the final outcome of this project.

Finally, we extend our gratitude to the management and staff of **Department of Electronics and Computer Engineering(DOECE) IOE Pulchowk** , for their support, resources, and facilities that have enabled us to complete this project successfully.

Thank you all once again for your invaluable support, guidance, and encouragement throughout this project.

Abstract

Anomaly detection with weakly supervised video-level labels is typically formulated as a multiple instance learning (MIL) problem, in which we aim to identify snippets containing abnormal events, with each video represented as a bag of video snippets. Although current methods show effective detection performance, their recognition of the positive instances, i.e., rare abnormal snippets in the abnormal videos, is largely biased by the dominant negative instances, especially when the abnormal events are subtle anomalies that exhibit only small differences compared with normal events. This issue is exacerbated in many methods that ignore important video temporal dependencies. To address this issue, we use add information from the optical flow which captures the temporal relation between successive frames in a video.

In this project, we explored the field of video anomaly detection and reviewed existing literature on the subject, as well as related topics such as action recognition and optical flow extraction. Our analysis led us to conclude that semi-supervised MIL-based approaches outperformed unsupervised techniques and required fewer labeled datasets compared to supervised ones. In the area of action recognition, inflated 3D architectures proved popular and effective, utilizing parameters from trained 2D counterparts to recognize useful features from images. Optical flow extraction also proved useful in various fields, with deep learning architectures such as FastFlowNet demonstrating the ability to achieve real-time processing speeds while maintaining sufficient accuracy.

Our research involved training a model to provide anomalous scores to input video. We utilized a pretrained action recognition model to extract features from both RGB and optical flow frames, which were combined into a feature vector for input into our anomaly score prediction model. Our anomaly score prediction model was trained on the UCF101 dataset using the combined feature vector from RGB and optical flow frames. The model achieved good training and validation accuracy, but suffered from a high false negative rate (FNR). We believe that using more recent action recognition models and better techniques for optical flow can improve the performance of the model.

Keywords: *Computer Vision, Deep Learning, Video Processing,Surveillance*

Contents

Page of Approval **ii**

Copyright **iii**

Acknowledgement **v**

Abstract **v**

Table of Contents **viii**

List of Figures **viii**

List of Abbreviations **xi**

1 Introduction **1**

1.1 Background 1

1.2 Objectives 2

1.3 Scope 2

2 Literature Review **3**

2.1 Related work 3

2.1.1 Unsupervised Anomaly Detection. 3

2.1.2 Weakly Supervised Anomaly Detection. 3

2.1.3 Optical Flow Extraction 4

2.2 Related theory 4

2.2.1 Theoretical Motivation of RTFM 5

2.2.2 Multi-scale Temporal Feature Learning 5

2.2.3 Feature Magnitude Learning 6

2.2.4 RTFM-enabled Snippet Classifier Learning 6

3 Theory **8**

3.1 Anomaly Detection 8

3.1.1 Supervised Anomaly Detection 8

3.1.2 Unsupervised Anomaly Detection 8

3.1.3 Semi-supervised Anomaly Detection . . . 9
3.1.4 Transfer Learning-Based Anomaly Detection . . . 10
3.1.5 Deep Reinforcement Learning-Based Anomaly Detection . . . 10
3.1.6 Deep Hybrid Models-Based Anomaly Detection for video . . . 10
3.2 Optical Flow . . . 11
3.2.1 Applications . . . 12
3.2.2 Sparse and Dense Optical flows . . . 12
3.2.3 FastFlowNet . . . 13
3.3 Action Recognition . . . 14
3.3.1 Two-Stream Inflated 3D ConvNets . . . 16

4 Methodology **18**
4.1 Feasibility Study . . . 18
4.1.1 Technical Feasibility . . . 18
4.1.2 Operational Feasibility . . . 19
4.1.3 Economic Feasibility . . . 19
4.1.4 Scheduling Feasibility . . . 19
4.2 Requirements Analysis . . . 20
4.2.1 Hardware Requirements . . . 20
4.2.2 Software Requirements . . . 20
4.2.3 Functional Requirements . . . 23
4.2.4 Non-Functional Requirements . . . 23
4.3 Dataset . . . 23
4.4 Data Collection . . . 24
4.4.1 Annotation . . . 24
4.5 Data Preprocessing . . . 24
4.6 Implementation Details . . . 26
4.7 Evaluation Metric . . . 26

5 System design **27**
5.1 System Design . . . 27
5.1.1 UML Diagrams . . . 28

6 Results **34**
6.1 Model Performance . . . 34

7 Discussion **41**

8 Limitations **42**

9 Future Improvements **43**

10 Project Schedule **44**

10.1 Gantt Chart 44

11 References **45**

12 Appendix **50**

12.1 Appendix A 50

12.2 Appendix B 51

12.3 Appendix C 52

List of Figures

3.1 Categories of Unsupervised AD Techniques 9
3.2 The Optical Flow Problem 11
3.3 Sparse(a) and Dense(b) Optical Flows 13
3.4 Block Digram of the FastFlowNet Architecture 14
3.5 General Flow for Action Recognition Process 15
3.6 Examples of classes in various datasets 16
3.7 Various architectures for video action recognition 16
3.8 Block diagram of Inflated Inception-v1 architecture 17

4.1 UCF101 dataset 24
4.2 Feature Extraction Process 25

5.1 High Level System block Diagram 27
5.2 Usecase Diagram 28
5.3 Class Diagram 29
5.4 Object Diagram 30
5.5 Deployment Diagram 31
5.6 Activity Diagram 32
5.7 Sequence Diagram 33

6.1 Training and Validation Losses 34
6.2 ROC curve(a) and Confusion matrix(c) for test set, and F-scores for various thresholds(b) 35
6.3 AUC for each video in the test set 36
6.4 Normal output-1 37
6.5 Normal output-2 37
6.6 Normal output-3 38
6.7 Explosion output-1 38
6.8 Arrest output-2 39
6.9 Abuse output-3 39
6.10 Fighting output-4 40

10.1 Gantt Chart 44

12.1 Input video-1 50
12.2 Input Video-2 51
12.3 Web application 51
12.4 Input video-1 52
12.5 Output 52

List of Abbreviations

LAH	List of Abbreviations Here
RTFM	Robust Temporal Feature Magnitude
PDC	Pyramid of Dilated Convolutions
TSA	Temporal Self-Attention
C3D	3D ConvNet
I3D	Inflated 3D CNN
AD	Anomaly Detection

1. Introduction

1.1 Background

Video anomaly detection has been intensively studied because of its potential to be used in autonomous surveillance systems[15, 25]. The goal of video anomaly detection is to identify the time window when an anomalous event happened – in the context of surveillance, examples of anomaly are bullying, shoplifting, violence, etc. Although one-class classifiers (OCCs, also called unsupervised anomaly detection) trained exclusively with normal videos have been explored in this context [15,17,28,31], the best performing approaches explore a Weakly-Supervised setup using training samples with video-level label annotations of normal or abnormal . This weakly-supervised setup targets a better anomaly classification accuracy at the expense of a relatively small human annotation effort, compared with OCC approaches.One of the major challenges of weakly supervised anomaly detection is how to identify anomalous snippets from a whole video labelled as abnormal. This is due to two reasons, namely: 1) the majority of snippets from an abnormal video consist of normal events, which can overwhelm the training process and challenge the fitting of the few abnormal snippets; and 2) abnormal snippets may not be sufficiently different from normal ones, making a clear separation between normal and abnormal snippets challenging.Anomaly detection trained with multiple-instance learning (MIL) approaches mitigates the issues above by balancing the training set with the same number of abnormal and normal snippets, where normal snippets are randomly selected from the normal videos and abnormal snippets are the ones with the top anomaly scores from abnormal videos.Although partly addressing the issues above,MIL introduces many problems such as the top anomaly score in an abnormal video may not be from an abnormal snippet; Training convergence as easy to fit randomly selected the normal snippets from video; The use of classification score does not necessarily enable a good separation between normal and abnormal snippets. To address the MIL problems above, we propose a novel method, named Robust Temporal Feature Magnitude (RTFM) learning. In RTFM, we rely on the temporal feature magnitude of video snippets, where features with low magnitude represent normal (i.e., negative) snippets and high magnitude features denote abnormal (i.e., positive)) snippets. RTFM solves the MIL issues above by increasing the probability of selecting abnormal snippets from abnormal video, improving training convergence, possible to include more abnormal snippets per abnormal video.

1.2 Objectives

Our primary objective is to develop a deep learning model that recognizes abnormal behaviours in surveillance videos. Apart from this, the additional objectives of the project are as follows:

- Build an AI aided system useful for organizations.
- Detect different types of anomalies that may occur in real life.
- Analytically calculate and compare the performance metrics of our model and the current state-of-the-art
- Apply the current state-of-the-art methods to a variety of relevant problems.

1.3 Scope

Our proposed system performs anomaly recognition on the given surveillance videos and tries to predict if it contains any such actions. This system helps respective personnel to predict if any anomaly activities have occurred. It can be used by organizations to track any such behaviours occurring in surveillance cameras.

2. Literature Review

2.1 Related work

2.1.1 Unsupervised Anomaly Detection.

Traditional anomaly detection methods assume the availability of normal training data only and address the problem with one-class classification using handcrafted features [2, 33]. With the advent of deep learning, more recent approaches use the features from pre-trained deep neural networks [11, 19, 41]. Others apply constraints on the latent space of normal manifold to learn compact normality representations [1, 3–5, 8–10, 13, 29, 32]. Alternatively, some approaches depend on data reconstruction using generative models to learn the representations of normal samples by (adversarially) minimising the reconstruction error [6, 14, 18, 18, 28]. These approaches assume that unseen anomalous videos/images often cannot be reconstructed well and consider samples of high reconstruction errors to be anomalies. However, due to the lack of prior knowledge of abnormality, these approaches can overfit the training data and fail to distinguish abnormal from normal events. Readers are referred to [39] for a comprehensive review of those anomaly detection approaches.

2.1.2 Weakly Supervised Anomaly Detection.

Leveraging some labelled abnormal samples has shown substantially improved performance over the unsupervised approaches [27, 38]. However, large-scale frame-level label annotation is too expensive to obtain.Hence, current SOTA video anomaly detection approaches rely on weakly supervised training that uses cheaper video-level annotations. Sultani et al. [14] proposed the use of video-level labels and introduced the large-scale weakly-supervised video anomaly detection data set, UCF-Crime. Since then, this direction has attracted the attention of the research community. Weakly-supervised video anomaly detection methods are mainly based on the MIL framework. However, most MIL-based methods fail to leverage abnormal video labels as they can be affected by the label noise in the positive bag caused by a normal snippet mistakenly selected as the top abnormal event in an anomaly video. To deal with this problem, Zhong et al. reformulated this problem as a binary classification under noisy label problem and used a Graph Convolution Neural (GCN) network to clear the label noise. Although this paper shows more accurate results than [14], the training of GCN and MIL is computationally costly, and it can lead to unconstrained latent space (i.e.,

normal and abnormal features can lie at any place of the feature space) that can cause unstable performance. By contrast, our method has trivial computational overheads compared to the original MIL formulation. Moreover, our method unifies the representation learning and anomaly score learning by an '2-norm-based temporal feature ranking loss, enabling better separation between normal and abnormal feature representations, improving the exploration of weak labels compared to previous MIL methods.

2.1.3 Optical Flow Extraction

Traditional methods such as Horn-Schunck[23] Lucas-Kanade[18] assume smoothness of motion and brightness constancy of the image intensity. However, these methods may fail in the presence of large displacements or occlusions. To overcome these limitations, variational methods and robust statistics have been introduced. Variational methods use energy functionals to compute optical flow, while robust statistics are used to handle noise and outliers. One of these methods is $TV - L^1$[20].

In recent years, deep learning-based approaches have shown remarkable results in optical flow estimation. These methods use convolutional neural networks to learn features from image patches and predict the optical flow between them. Some of the popular deep learning-based optical flow algorithms include FastFlowNet, PWC-Net, and LiteFlowNet. These methods have achieved state-of-the-art performance on benchmark datasets such as Flying Chairs and Sintel. However, they can be computationally expensive and require large amounts of memory, making them unsuitable for real-time applications.

2.2 Related theory

Our proposed robust temporal feature magnitude (RTFM) approach aims to differentiate between abnormal and normal snippets using weakly labelled videos for training. Given a set of weakly-labelled training videos, we know only if the video has anomaly or not. There is no idea about the nature of anomaly or where at the point of video anomaly occurs. The model used by RTFM returns a T-dimensional feature representing the classification of the T video snippets into abnormal or normal. The training of this model comprises a joint optimization of an end-to-end multi-scale temporal feature learning, and feature magnitude learning and an RTFM-enabled MIL classifier training, with the loss equation

$$\min_{\theta,\phi} \sum_{i,j=1}^{|D|} l_s(s_\theta(F_i), (s_\theta(F_j)), y_i, y_j) + l_f(f_\phi(s_\theta(F_i)), y_i)$$

where $s_\theta : F \to X$ is the temporal feature extractor, $f_\phi : X \to [0,1]^T$, is the snippet classifier, l_s denotes a loss function that maximises the separability separability between the top-k snippet features from normal and abnormal videos.

2.2.1 Theoretical Motivation of RTFM

Top-k MIL in [25] extends MIL to an environment where positive bags contain a minimum number of positive samples and negative bags also contain positive samples, but to a lesser extent, and it assumes that a classifier can separate positive and negative samples. Our problem is different because negative bags do not contain positive samples, and we do not make the classification separability assumption. Following the nomenclature above, the temporal feature extracted from a video is denoted by $X = s_\theta(F)$, where snippet features are denotes by the rows x_t of X. An abnormal snippet is denoted by $x^+ \sim P_x^+(x)$ and a normal snippet is denoted as $x^- \sim P_x^-(x)$ To learn a function that can classify videos and snippets as normal or abnormal, we define a function that classifies a snippet using its magnitude (i.e., we use l_2 norm to compute the feature magnitude), where instead of assuming classification separability between normal and abnormal snippets (as assumed in [25]), we make a milder assumption that by learning the snippet feature, such that normal ones have smaller feature magnitude than abnormal ones, we can satisfy this assumption. To enable such learning, we rely on an optimisation based on the mean feature magnitude of the top k snippets from a video [25], defined by equation

$$g_{\theta,k}(X) = \max_{\Omega_k(X) \subseteq \{x_t\}_{t=1}^T} \frac{1}{k} \sum_{x_t \in \Omega_k(k)} ||x_t||_t$$

Where $g_{\theta,k}(.)$ is parameterised by θ to indicate its dependency on $s_\theta(.)$ to produce $x_t\Omega_t(X)$ which consist of k-snippets. The separability between abnormal and normal videos is denoted by:

$$d_{\theta,k}(X^+, X^-) = g_\theta, k(X^+) - g_\theta, k(X^-)$$

Our proposed RTFM receives a T × D feature matrix F extracted from a video containing T snippets. Then, MTN captures the long and short-range temporal dependencies between snippet features to produce $X = s_\theta(F)$. Next, we maximise the separability between abnormal and normal video features and train a snippet classifier using the top-k largest magnitude feature snippets from abnormal and normal videos.

2.2.2 Multi-scale Temporal Feature Learning

Inspired by the attention techniques used in video understanding [26], our proposed multi-scale temporal network (MTN) captures the multi-resolution local temporal dependencies and the global temporal dependencies between video snippets. MTN uses a pyramid of dilated convolutions over the time domain to learn multi-scale representations for video snippets. Dilated convolution is usually applied in the spatial domain with the goal of expanding

the receptive field without losing resolution. Here we propose to use dilated convolutions over the temporal dimension as it is important to capture the multi-scale temporal dependencies of neighbouring video snippets for anomaly detection. The global temporal dependencies between video snippets is achieved with a self-attention module, which has shown promising performance on capturing the long-range spatial dependency on video understanding ,image classification and object detection . Motivated by the previous works using GCN to model global temporal information [27], we re-formulate the spatial self-attention technique to work on the time dimension and capture global temporal context modelling. We aim to produce attention map that estimates the pairwise correlation between snippets. Our temporal self-attention (TSA) module first uses a 1X1 convolution to reduce the spatial dimension. We, then apply three separate 1X1 convolution layers to produce F^{c1}, F^{c2} and F^{c3}. A skip connection is added after this final 1×1 convolutional layer, as in

$$F^{(TSA)} = F^{(c4)} + F^{c}$$

The output from the MTN is formed with a concatenation of the outputs from the PDC and MTN modules. A skip connection using the original features F produces the final temporal feature representation.

2.2.3 Feature Magnitude Learning

we propose a loss function to model , where the top k largest snippet feature magnitudes from normal videos are minimised and the top k largest snippet feature magnitudes from abnormal videos are maximised. More specifically, we propose the loss $l_s(.)$ that maximises the separability between normal and abnormal videos:

$$l_s(s_\theta(F_i), s_\theta(F_j), y_i, y_j) = \begin{cases} \max(0, m - d - \theta, k(X_i, X_j)), & \text{if } y_i = 1, y_j = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

where m is a pre-defined margin , $X_i = s_\theta(F_i)$ is the abnormal video feature(similarly for X_j for a normal video), and $d_{\theta,k}(.)$ represents separability function that computes the difference between the score of the top k instance, from $g_{\theta,k}(.)$, of the abnormal and normal videos.

2.2.4 RTFM-enabled Snippet Classifier Learning

To learn the snippet classifier, we train a binary cross-entropy-based classification loss function using the set $\Omega_k(X)$ that contains k snippets with the largest l_2-norm features from $s_\theta(F)$. In particular, the loss $l_f(.)$ is defined as:

$$l_f(f_\phi(s_\theta(F)), y == \sum_{x \in \Omega_k(X)} -y \log(f_\phi(x)) + (1 - y) \log(1 - f_\phi(x))$$

where $x = s_\theta(f)$. The l_f is accompanied by the temporal smoothness and sparsity regularisation, with the temporal smoothness defined as $f_\phi(s_\theta(f_t)) - f_\phi(s_\theta(f_{t-1}))^2$ to enforce similar anomaly score for neighbouring snippets, while the sparsity regualisation defined as $\sum_{t=1}^{T} |f_\phi(s_\theta(f_t)|$ to impose a prior that abnormal events are rare in each abnormal video.

3. Theory

3.1 Anomaly Detection

Anomaly detection is a field of study concerned with identifying unusual or abnormal behavior in data. It is used across a wide range of applications, including fraud detection, network intrusion detection, medical diagnosis, and sensor monitoring. All these applications can have various forms of input data such as user behaviour logs, network logs, video, etc. The goal of anomaly detection is to identify data points that deviate significantly from the expected or normal pattern of behavior.
Anomaly detection can be classified into three main categories: supervised, unsupervised, and semi-supervised methods. Following section gives an overview of each of them.

3.1.1 Supervised Anomaly Detection

Supervised methods include classification algorithms such as decision trees, support vector machines, and neural networks. These algorithms are trained on a labeled dataset and learn to identify anomalies based on the features and characteristics of the data. They can be effective in detecting known anomalies(ones which have been labelled in the dataset), but may not be suitable for detecting new or unknown anomalies. So, this setup is practically not very relevant due to the assumption that anomalies are known and labeled correctly, as for many applications, anomalies are not known in advance or may occur spontaneously as novelties during the test phase. They also cannot deal with unbalanced dataset, which is the usual case, as anamolous contents occur in much less amount in the usual datasets. Some of supervised anomaly detection techniques are:

- Decision Trees
- Support Vector Machine
- Classification-based Neural Network Architectures

3.1.2 Unsupervised Anomaly Detection

Unsupervised techniques allow one to perform the learning process in the absence of labeled data. They learn the inherent data features such as distance or density to distinguish between normal and abnormal data to facilitate AD by finding commonalities within the data.

Unsupervised techniques require the availability of large video datasets and high computational resources compared to other techniques, otherwise they don't give good accuracy. One of the benefits of these methods is that it is a low-cost way to find outliers because the algorithms do not need to be trained with annotated data. Some of supervised anomaly detection techniques are:

- Auto Encoders (AEs)
- Restricted Boltzmann Machines (RBMs)
- One class SVMs
- Deep Belief Networks (DBNs)
- Generative Adverserial Networks(GANs)
- RNN techniques like GRUs and LSTM

The following figure shows the unsupervised AD algorithms grouped in categories:

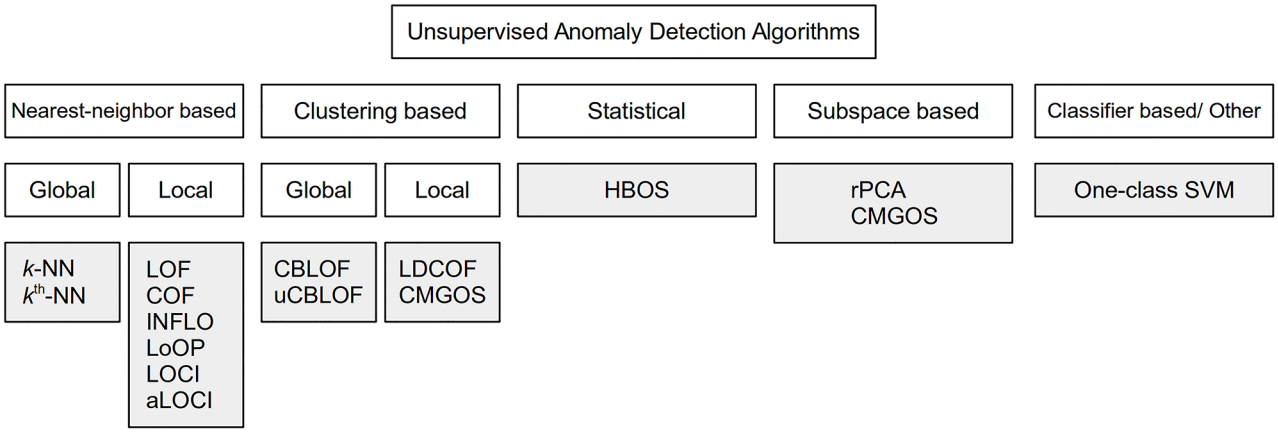


Figure 3.1: Categories of Unsupervised AD Techniques

3.1.3 Semi-supervised Anomaly Detection

In semi-supervised learning, the training process utilizes datasets with weakly labeled instances, where it is assumed that all instances belong to a single class. This technique develops a discriminative boundary around the normal instances and marks test instances as anomalous when they do not belong to the majority class. This approach combines the benefits of both supervised and unsupervised methods.

One of the key advantages of this technique is that it requires minimal labeled data. However, one of the main drawbacks is that it may be susceptible to overfitting, and irrelevant input features in the training data may lead to incorrect classifications.

3.1.4 Transfer Learning-Based Anomaly Detection

Neural networks (NNs) can be trained in two ways: scratch learning (SL) and transfer learning (TL). In SL, a network is trained with random initial weights, requiring vast amounts of data, high computing power, and a significant processing time. To overcome these challenges, TL is proposed as a technique that leverages knowledge from a related task to improve performance. TL uses the learned weights and parameters from the first task to initialize the second task, reducing the need for extensive training data and computational resources while achieving satisfactory results.

3.1.5 Deep Reinforcement Learning-Based Anomaly Detection

The use of deep reinforcement learning (DRL) in AD involves actively seeking out novel classes of anomalies that are not covered by labeled training data. This strategy strikes a balance between utilizing the current data model and exploring new classes of anomalies. Consequently, it can leverage labeled anomaly data to enhance detection accuracy without limiting the set of anomalies it can identify to those in the training set.

The underlying concept behind using Reinforcement Learning (RL) for decision-making is that an agent can learn from its environment by interacting with it and receiving rewards for certain actions, much like humans learn through experience. DRL combines RL and deep learning (DL) to address more complex problems, such as high-dimensional data and environments, sparse reward signals, and uncertainty in observations.

In particular, the DRL-based AD approach uses labeled anomaly data to improve detection accuracy while exploring new classes of anomalies. This is achieved by creating an environment using the training data and interacting with it to identify anomalies.

3.1.6 Deep Hybrid Models-Based Anomaly Detection for video

Hybrid models in AD for video streaming combine multiple models to improve performance. These models are particularly effective for high-dimensional input data, such as video. Deep hybrid models typically use DNNs for feature extraction and traditional ML algorithms for anomaly detection. For instance, Al-Dhamari et al. [**?**] proposed a hybrid approach that uses transfer deep learning along with binary support vector machines for abnormal behavior detection. In this approach, a CNN is used for feature extraction and the feature vector is passed into a binary SVM to construct the abnormal event detection model. Emad et al. [31] introduced a model that integrates two methods, GAN and multi-instance learning, to predict future anomalies in surveillance cameras. GAN is used for future frame prediction and multi-instance learning for anomaly detection. Zhao et al. [29] used 3D-ConVNet and AEs for video anomaly detection. 3D-ConVNet learns video representation automatically

and extracts features from both spatial and temporal dimensions, while AEs predict the future frames and calculate the anomaly score based on the reconstruction error.

3.2 Optical Flow

The concept of optical flow was introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world. Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene.Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image.
The optical flow problem may be expressed as:

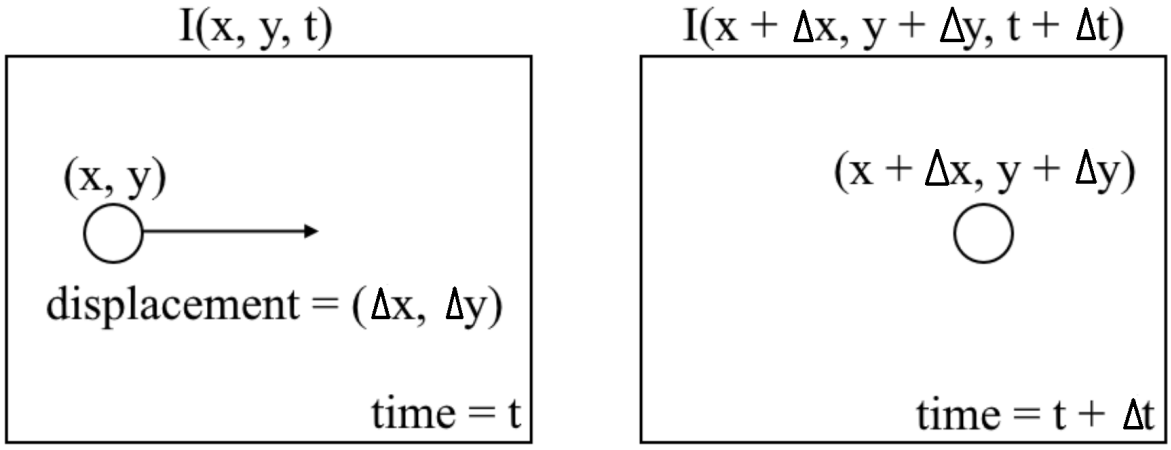


Figure 3.2: The Optical Flow Problem

Differential optical flow methods aim to compute the motion between two image frames captured at times t and $t + \Delta t$ at each voxel position. These methods are referred to as "differential" as they are based on local Taylor series approximations of the image signal, utilizing partial derivatives with respect to the spatial and temporal coordinates. For a (2D + t)-dimensional case (similar for 3D or n-D cases), a voxel at location (x, y, t) with intensity $I(x, y, t)$ will have moved by Δx, Δy, and Δt between the two image frames. The brightness constancy constraint is expressed as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{3.1}$$

Assuming the movement to be small, the image constraint at I(x,y,t) with Taylor series can be developed to get:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta + \text{higher order terms} \tag{3.2}$$

By truncating the higher order terms (which performs a linearization) it follows that:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.3}$$

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \tag{3.4}$$

which results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{3.5}$$

where V_x, V_y are the x and y components of the velocity or optical flow of I(x,y,t) and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x,y,t) in the corresponding directions. I_x, I_y and I_t can be written for the derivatives . Thus,

$$I_x V_x + I_y V_y = -I_t \tag{3.6}$$

or

$$\nabla I.\vec{V} = -I_t \tag{3.7}$$

This is an equation in two unknowns and cannot be solved as such. This is known as the aperture problem of the optical flow algorithms. To find the optical flow another set of equations is needed, given by some additional constraint. All optical flow methods introduce additional conditions for estimating the actual flow.

3.2.1 Applications

Optical flow has a wide range of applications in computer vision, including object tracking, activity recognition, video stabilization, robotics, medical imaging, etc.. In object tracking, optical flow can be used to track the motion of objects in a video sequence. In *activity recognition*, optical flow can be used to identify human actions from a video sequence. In *video stabilization*, optical flow can be used to correct camera motion and stabilize the video. In *robotics*, optical flow can be used for navigation and obstacle avoidance. In *medical imaging*, optical flow can be used in medical imaging for tasks such as blood flow analysis.

3.2.2 Sparse and Dense Optical flows

Sparse optical flow gives the flow vectors of some "interesting features" (say few pixels depicting the edges or corners of an object) within the frame while Dense optical flow, which gives the flow vectors of the entire frame (all pixels) - up to one flow vector per pixel. Dense optical flow has higher accuracy at the cost of being slow/computationally expensive.
An algorithm for calculating sparse optical flow is Lucas-Kanade algorithm, and another one for calculating dense optical flow is $TV - L^1$ algorithm. Following figures show some examples of sparse and dense optical flows:

(a) Dense Optical Flow for a moment in a clip

(b) Sparse Optical Flow for a moment in a clip

Figure 3.3: Sparse(a) and Dense(b) Optical Flows

3.2.3 FastFlowNet

Many optical flow algorithms are computationally intensive and unable to run in real-time due to the high number of operations required to estimate motion vectors between frames. These methods often rely on iterative optimization algorithms, such as the Lucas-Kanade method or the Horn-Schunck method, which require solving large systems of equations at every pixel location in the image. These algorithms are slow and often failed to accurately estimate motion in regions with large displacements, occlusions, or textureless areas. As a result, they are unsuitable for many real-world applications that require fast and reliable optical flow estimation, such as robotics, autonomous driving, and video compression.

FastFlowNet is a lightweight deep learning model for optical flow estimation. The FastFlowNet architecture is based on a fully convolutional network (FCN) that consists of a few convolutional and pooling layers followed by a series of deconvolutional layers that upsample the feature maps to the original resolution. The network takes as input a stack of two consecutive frames and outputs a dense optical flow field between them.

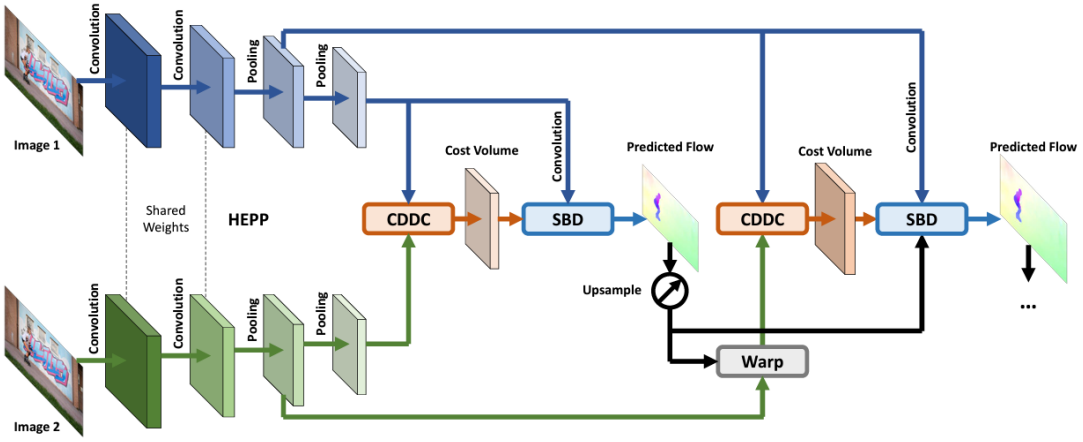


Figure 3.4: Block Digram of the FastFlowNet Architecture

Figure 3.4 shows the FastFlowNet architecture. One of the key contributions of FastFlowNet is the use of a lightweight feature extraction module that consists of a single 3x3 convolutional layer with a small number of filters. This module is shared between the two input frames and reduces the computational cost of the network significantly while maintaining high accuracy.

To further improve the performance of the model, the authors introduce a new loss function that takes into account the local smoothness of the optical flow field. This loss function penalizes large spatial and temporal gradients in the flow field, which encourages the network to output smoother and more plausible flow fields.

The experimental evaluation of FastFlowNet on several benchmark datasets shows that it achieves state-of-the-art accuracy while being significantly faster and more lightweight than previous methods. The proposed model is especially useful for real-time applications on resource-constrained devices such as mobile phones and drones.

3.3 Action Recognition

Action recognition refers to detecting and labeling actions of one or more agents in an input observation(such as video). Video action recognition is a challenging but crucial task in computer vision, with applications in fields such as healthcare, law enforcement, and sports injury prevention. Despite significant improvements in recent years, recognition accuracy is still limited by factors such as perspective shifts, camera proximity, context, and motion speed. One of the most challenging aspects of the process is finding suitable features that can capture the relevant information in a computationally efficient manner. Many existing methods rely on handcrafted features extracted from RGB video, but these may not be suf-

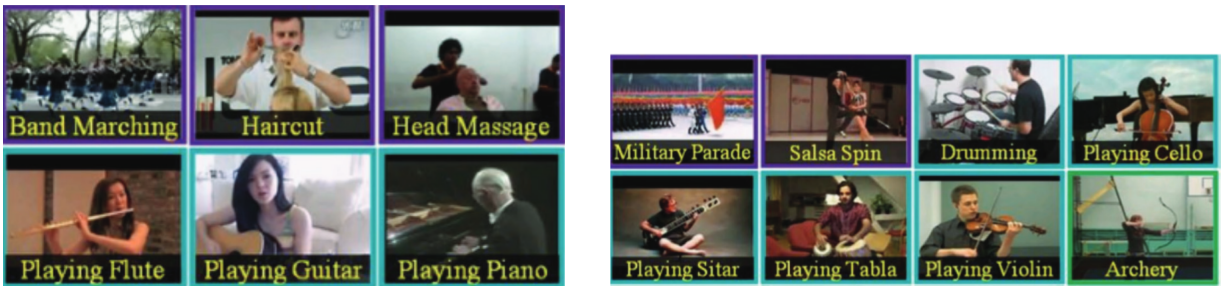
ficient for recognizing complex actions. To address motion tracking problems, researchers have turned to computer vision techniques such as *Optical Flow*, Scale Invariant Feature Transformation (SIFT), Histogram of Oriented Gradients (HOG), and Mean Shift. These methods combine object appearance and motion to recognize actions from video. One application of video action recognition is in remote monitoring of patient activity, which can be used to advise patients on diet, exercise, and medication adherence. The beneficial robotic field is another area where video action recognition plays a crucial role.
Recent strategies for identifying human activities victimization are often assisted in two main stages: feature extraction and action classification, as shown in figure 3.5.

Figure 3.5: General Flow for Action Recognition Process

Human Action Recognition algorithms involve two primary tasks with different evaluation procedures. The action distribution task is a multiclass classification problem, where the action performed in a specific scene is recognized. While most datasets use multiclass accuracy as the metric, unbalanced and long-tailed datasets use alternative measurements such as precision, recall, and F score. Certain datasets permit covering activity labels both in space and time, which transforms the problem into a label binary-level compilation task. The activity expectation task involves action segment annotation, and mean Average Precision (mAP) calculated over the Intersection over Union (IoU) is used as prescribed. The challenges and issues in this task are highlighted in previous research. Other tasks include Spatio-temporal localization, which involves predicting the extent of a task in both bounding boxes and time, and providing localized event labeling. Benchmark datasets are known to have built-in biases that constrain the study, and cross-dataset testing can be used to ensure the algorithms truly generalize on unseen data.
Some of the popular benchmark datasets for human action recognition are:

- UCF101
- HMDB-51

- Kinetics-400
- Something Something V2
- Aslan



(a) UCF101 (b) Aslan dataset

Figure 3.6: Examples of classes in various datasets

3.3.1 Two-Stream Inflated 3D ConvNets

Inflated 3D architectures replace the 2D components of image related architectures, like 2D convolutions, with 3D components for supporting an extra time dimensions of videos. They often use pre-trained version of their 2D counterparts(copied along the slices of the extra dimension) as a starting point and are trained of video action recognition datasets.

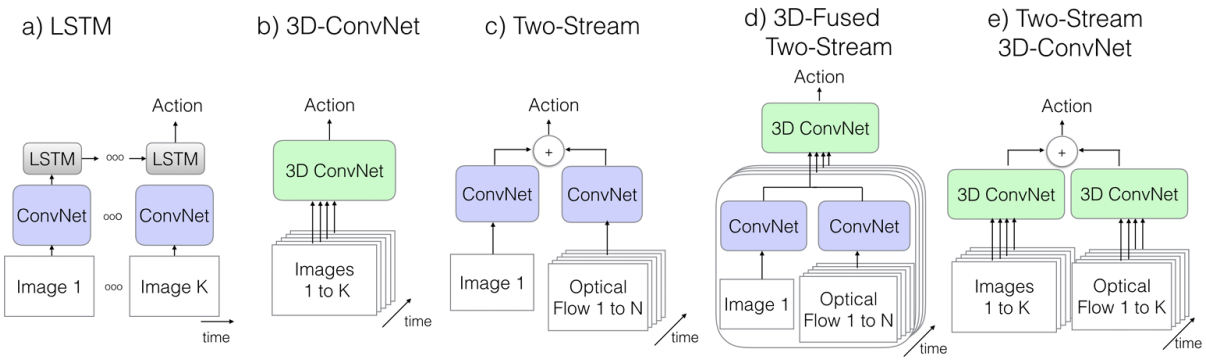


Figure 3.7: Various architectures for video action recognition

Video action recognition architectures also use Optical flow frames as input for capturing the temporal relationship between successive frames. Thus using both RGB frames and Optical Flow frames, two stream architectures have shown better performance with relatibely lower training cost(Training data size, and training iterations).

Two-Stream Inflated 3D ConvNets utilize successful 2D ConvNet designs from ImageNet to improve 3D ConvNets performance. This architecture includes an RGB stream and an optical-flow stream, and inflates 2D filters and pooling kernels into 3D filters and kernels with an additional temporal dimension. These inflated architectures can use the parameters learned by their 2D counterparts(from pre-trained ImageNet models) by bootstrapping their parameters by converting an image into a video sequence, repeating the weights of 2D filters N times along the time dimension, and rescaling them by dividing by N. Additionally, considering an increased dimension in [**?**] the authors propose pacing receptive field growth in space, time, and network depth by inflating pooling operators along the time dimension and setting convolutional/pooling temporal stride. In their experiments, the authors[**?**] found it helpful to have a two-stream configuration, with one I3D network trained on RGB inputs and another on flow inputs, and averaged their predictions at test time.

Inception architecture is one of the popular 2D convnets. There have been several versions of inception architectures(upto v-4), with the higher versions introducing some efficiency optimizations on the lower versions.

We have used Inflated Inception-v1 architecture pretrained on Kinetics dataset as an action recognition model for feature extraction. A block diagram for this architecture has been shown in figure 3.8.

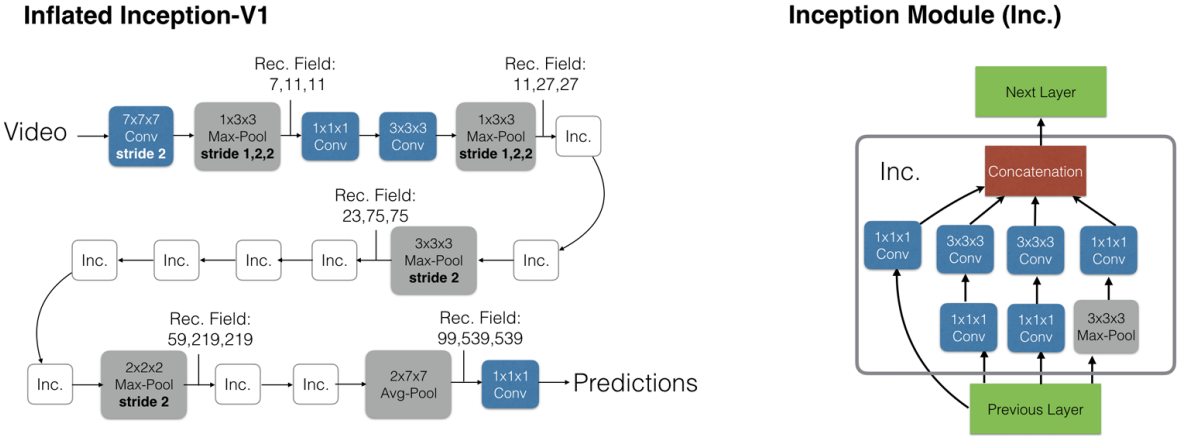


Figure 3.8: Block diagram of Inflated Inception-v1 architecture

4. Methodology

4.1 Feasibility Study

4.1.1 Technical Feasibility

Model

The model to be used for our system consists of pretrained video-feature-extraction models like I3D or C3D, fully convolutional(FC) networks, and a classification model.

Dataset

To train and test our model we will be using four multi-scene benchmark datasets, created for the weakly supervised video anomaly detection task: ShanghaiTech , UCF-Crime [57], XDViolence [67] and UCSD-Peds [69].A brief description of these datasets is given below:

- *UCF-Crime(size 4.5 GB)* is a large-scale anomaly detection data set [57] that contains 1900 untrimmed videos with a total duration of 128 hours from real-world street and indoor surveillance cameras. Unlike the static backgrounds in ShanghaiTech, UCF-Crime consists of complicated and diverse backgrounds. Both training and testing sets contain the same number of normal and abnormal videos. The data set covers 13 classes of anomalies in 1,610 training videos with video-level labels and 290 test videos with frame-level labels.

- *XD-Violence* is a recently proposed large-scale multiscene anomaly detection data set, collected from real life movies, online videos, sport streaming, surveillance cameras and CCTVs. The total duration of this data set is over 217 hours, containing 4754 untrimmed videos with video-level labels in the training set and frame-level labels in the testing set. It is currently the largest publicly available video anomaly detection data set.

- *ShanghaiTech* is a medium-scale data set from fixed-angle street video-surveillance. It has 13 different background scenes and 437 videos, including 307 normal videos and 130 anomaly videos. The original data set [28] is a popular benchmark for the anomaly detection task that assumes the availability of normal training data. Zhong et al. [79] reorganised the data set by selecting a subset of anomalous testing videos into training

data to build a weakly supervised training set, so that both training and testing sets cover all 13 background scenes. We use exactly the same procedure as in [79] to convert ShanghaiTech for the weakly supervised setting.

- *UCSD-Peds* is a small-scale dataset combined by two sub-datasets – Ped1 with 70 videos and Peds2 with 28 videos. In the dataset the default training set does not contain anomaly videos. Following He et al. [20], 6 anomaly videos and 4 normal ones on UCSD-Peds2 are randomly included into training data, and the remaining videos constitute the test set. We also repeat this process 10 times and report the average performance.

Training

Our options for training our model are to use PCs with sufficient GPU capabilities or paying for commercial compute platforms. If our PCs and google collab(free tier) prove to be incapable/impractical to train our model, during our experimenting phase we can shift to use the commercial compute platforms as the prices seem low enough(e.g. $0.42 per hour for AWS ML),

4.1.2 Operational Feasibility

The model that we will be implementing has already been used to demonstrate good performance on benchmark datasets(AUC-ROC score of 97.21 on ShanghaiTech Weakly Supervised dataset). On the basis of this, we can be confident that our system can achieve acceptable performance on real life applications like monitoring on public property, markets, etc.

4.1.3 Economic Feasibility

The only source of expense, as far as we realize, is using commercial compute platforms. If we need to use them, we will need to seek funding from the department.

4.1.4 Scheduling Feasibility

The final deadline for the completion of this project is near the end of next semester. So we will have about 3 months of time to complete this project, though that time includes the time we need to give to our lectures and exams. So deducting times for those tasks we will have about 1.5 months to work on this project. We believe we can complete our project within before the deadline.

4.2 Requirements Analysis

4.2.1 Hardware Requirements

- A working PC
- A CUDA compatible GPU for model training and feature extraction.

4.2.2 Software Requirements

- *Python* is a high-level, interpreted programming language that is widely used in many different fields such as web development, data science, artificial intelligence, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world due to its simplicity, readability, and versatility. Python has a large standard library that provides many built-in functions and modules, making it easy to perform common tasks without needing to write additional code. It also supports a wide range of third-party libraries and frameworks that can be used for more specialized tasks.Python is an interpreted language, which means that the code is executed directly, without needing to be compiled first. Some popular Python libraries and frameworks include NumPy, Pandas, Flask, Django, TensorFlow, and PyTorch. These tools allow developers to work with data, build web applications, and create machine learning models, among many other things.

- *OpenCV*(Open Source Computer Vision) is a popular open-source computer vision library that is written in C++ and has Python bindings. It provides developers with a range of computer vision algorithms and tools to help build applications that can analyze and understand visual data from cameras or images and videos. OpenCV can be used for a variety of tasks such as object detection, face recognition, image segmentation, feature extraction, motion detection, and more. It provides a large set of pre-trained models that can be used out-of-the-box for common tasks such as detecting faces or detecting objects. The library is widely used in fields such as robotics, autonomous vehicles, augmented reality, and more. It has a large and active community of developers who contribute to its development and provide support for its users. OpenCV supports a range of platforms including Windows, Linux, Mac OS, Android, and iOS. It also has bindings for popular programming languages such as Python, Java, and MATLAB, making it easy to use in a wide range of applications.

- *Django* is a popular open-source web framework that is written in Python. It provides developers with a set of tools and libraries that help them build web applications quickly

and efficiently. Django follows the Model-View-Controller (MVC) architectural pattern, which separates the data models, views, and controllers into distinct components. Django comes with a range of features and components that make it easy to build web applications, including an Object-Relational Mapping (ORM) system for working with databases, a templating engine for building dynamic web pages, and a built-in admin interface for managing site content. Django is known for its robustness, scalability, and security. It includes built-in security features such as cross-site scripting (XSS) protection, cross-site request forgery (CSRF) protection, and SQL injection protection. Django also includes support for caching, internationalization, and localization, making it easy to build applications that can be used in multiple languages.Django is widely used by developers and organizations around the world, including large-scale applications such as Instagram, Pinterest, and Mozilla. It has a large and active community of developers who contribute to its development and provide support for its users.

- *Jupyter Notebook* is an open-source web application that allows users to create and share interactive documents containing live code, equations, visualizations, and narrative text. It supports a wide range of programming languages, including Python, R, and Julia. Jupyter Notebook provides an interactive computing environment where users can write and execute code, view and manipulate data, and create visualizations all within the same document. This makes it an ideal tool for data analysis, machine learning, and scientific computing. One of the key features of Jupyter Notebook is the ability to create "notebooks," which are interactive documents that can include code, text, and visualizations. Notebooks can be saved and shared with others, making it easy to collaborate on data analysis and other projects. Jupyter Notebook supports a wide range of libraries and frameworks, including NumPy, Pandas, Matplotlib, Scikit-learn, and TensorFlow. These tools allow developers and data scientists to work with data, build models, and create visualizations within the same environment. Jupyter Notebook can be run locally on a computer or hosted on the cloud using services such as Google Colab or Microsoft Azure Notebooks. It has a large and active community of users and developers who contribute to its development and provide support for its users.

- *Pytorch* is an open-source machine learning library for Python that is widely used for developing and training deep learning models. It was developed primarily by Facebook's AI research team and is known for its simplicity, ease of use, and flexibility.PyTorch provides a range of features and tools for building deep learning models,

including dynamic computational graphs, automatic differentiation, and a large set of built-in modules and functions. It supports a wide range of neural network architectures and can be used for tasks such as image and speech recognition, natural language processing, and more. One of the key features of PyTorch is its support for dynamic computational graphs, which allows for more flexible and efficient model building compared to static computational graphs used in other deep learning frameworks. This allows developers to build models with dynamic behavior, such as recurrent neural networks or models with variable-length inputs. PyTorch also provides a range of tools for debugging and visualizing deep learning models, including the ability to visualize computational graphs, track model performance, and inspect gradients during training. PyTorch is widely used by researchers and developers in the deep learning community and has a large and active community of contributors who provide support and contribute to its development.

- *Conda* is an open-source package management system and environment management system that is widely used in Python programming. It allows users to easily install, manage, and update packages and dependencies in their Python environment, as well as manage multiple Python environments on the same system.Conda provides a range of features that make it easy to work with Python packages and environments, including:

  - Package management: Conda allows users to easily install, manage, and update packages and dependencies in their Python environment. It provides access to a large repository of packages, including popular data science packages like NumPy, Pandas, and Matplotlib.
  - Environment management: Conda allows users to create and manage multiple Python environments on the same system, each with its own set of packages and dependencies. This allows developers to work on multiple projects with different requirements without interfering with each other.
  - Cross-platform support: Conda is available on all major platforms, including Windows, macOS, and Linux, making it easy to work with Python packages and environments on any system.
  - Virtualization: Conda allows users to create virtual environments, which are isolated Python environments that can have their own set of packages and dependencies. This allows developers to create reproducible environments and easily share their work with others.

  Conda is widely used in the Python community for data science, machine learning, and

scientific computing. It has a large and active community of contributors who provide support and contribute to its development.

- *JavaScript* is a high-level, interpreted programming language that is widely used for creating dynamic web content and applications. It is primarily used for client-side web development, meaning that it is executed on the client's computer rather than on the server. JavaScript allows developers to add interactivity and dynamic behavior to web pages, including animations, forms, and user interfaces. It can be used in conjunction with HTML and CSS to create rich, interactive web experiences. JavaScript is also commonly used for server-side web development, using technologies such as Node.js. This allows developers to create server-side applications and APIs using the same language that is used on the client side. JavaScript has a large and active community of developers who contribute to its development and provide support for its users. It has a wide range of frameworks and libraries, such as React, Angular, and Vue.js, that can be used to build web applications quickly and efficiently. JavaScript is supported by all major web browsers, including Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, making it a widely adopted and accessible programming language for web development.

4.2.3 Functional Requirements

- Make predictions accurately.
- Minimize false positive predictions.

4.2.4 Non-Functional Requirements

- Response time: The system must be able to make predictions as quickly as possible.
- Reliability: Every functionality on code must be able to work smoothly without failure under given conditions.
- Extendibility: More functionalities must be added easily.

4.3 Dataset

The UCF Crime Dataset is a collection of crime data reported by the University of Central Florida Police Department. The data set contains detailed information about crimes reported on and around the UCF campus from 2003 to 2015. The data set includes information about the type of crime, the location of the crime, the time and date of the crime, and other relevant details. Figure 4.1 shows a visualization of the dataset through bar plot.

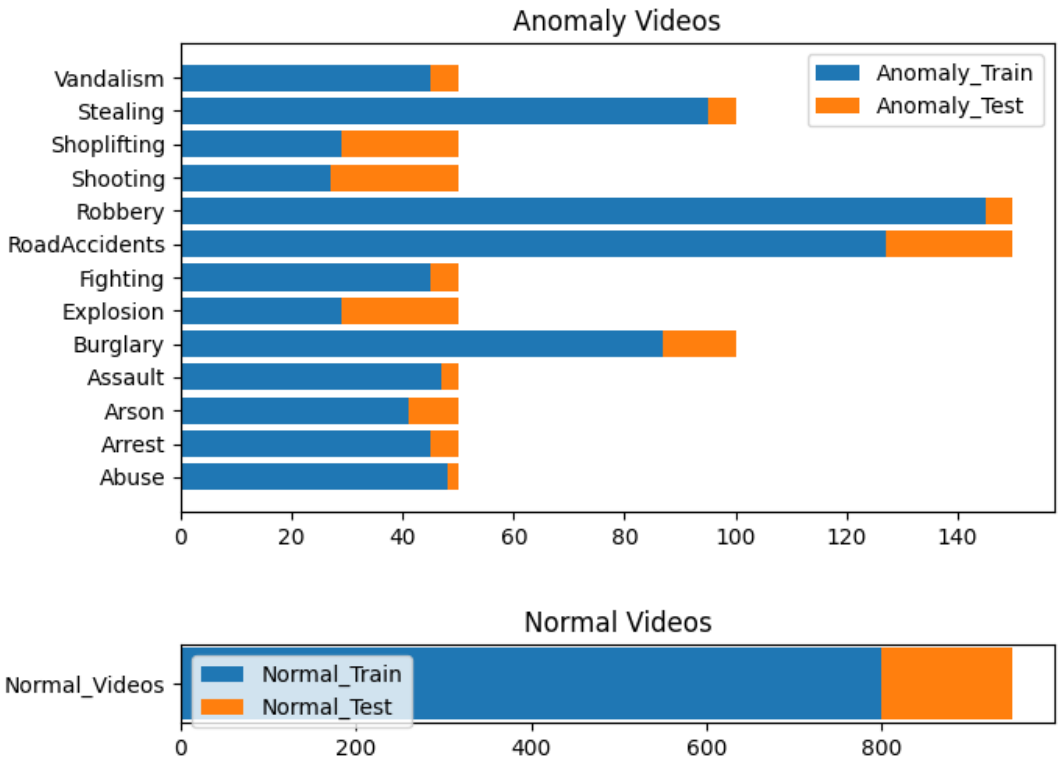


Figure 4.1: UCF101 dataset

Data collection for the UCF Crime Dataset was done by the UCF Police Department. When a crime is reported to the police department, officers fill out a report that includes all relevant information about the incident. This information is then entered into a database, which is used to create the UCF Crime Dataset.

4.4 Data Collection

4.4.1 Annotation

For our anomaly detection method, only video-level labels are required for training. However, in order to evaluate its performance on testing videos, we need to know the temporal annotations, i.e. the start and ending frames of the anomalous event in each testing anomalous video. To this end, we assign the same videos to multiple annotators to label the temporal extent of each anomaly. The final temporal annotations are obtained by averaging annotations of different annotators. The complete dataset is finalized after intense efforts of several months.

4.5 Data Preprocessing

Data pre-processing is a crucial step in any machine learning pipeline, and it is especially important when working with video data. In this particular case, a video has been divided into

32 segments, and the RGB flows and optical flows have been extracted from each segment. RGB features have been extracted using an I3D feature extractor that has been pre-trained on the Kinetic-400 dataset, and optical flow features have been extracted using the Fast-FlowNet network. The output of each feature extraction process is a 2-dimensional matrix of shape (32,1024). Figure 4.2 gives an overview of the preprocessing/feature extraction pipiline.

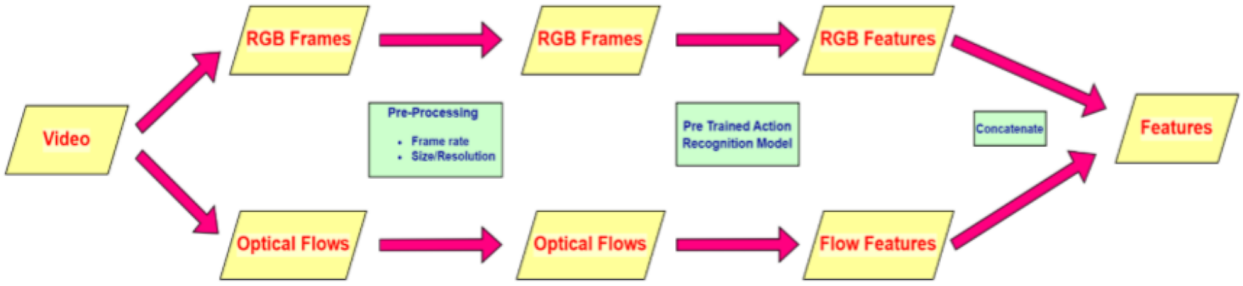


Figure 4.2: Feature Extraction Process

The first step in data pre-processing for this task is to divide the video into the 32 segments. This can be done using a variety of techniques, such as segmenting the video based on time or based on content. Once the video has been segmented, the RGB flows and optical flows can be extracted from each segment.

RGB features are extracted using the I3D feature extractor. This is a popular deep learning architecture for video analysis that has been pre-trained on the Kinetic-400 dataset, which contains over 400,000 video clips from 400 action categories. The I3D feature extractor processes each frame of the video segment and outputs a 2-dimensional matrix of shape (32,1024) that represents the RGB features for that segment.

Optical flow features are extracted using the TVL1 algorithm, which is a popular method for estimating optical flow in video. Optical flow refers to the motion of objects in the video between frames, and it can be a powerful feature for understanding the content of the video. The TVL1 algorithm estimates the optical flow between each pair of frames in the video segment and outputs a 2-dimensional matrix of shape (32,1024) that represents the optical flow features for that segment.

Once the RGB and optical flow features have been extracted for each segment, the resulting matrices can be concatenated into a single tensor of shape (32, 2048). This tensor can then be used as input to a machine learning model for further analysis.

4.6 Implementation Details

We extract visual features from the fully connected (FC) layer FC6 of the I3D network. Before computing features, we re-size each video frame to 240 × 320 pixels and fix the frame rate to 30 fps. We compute I3D RGB features for every 16-frame video clip followed by l2 normalization. To obtain features for a video segment, we take the average of all 16-frame clip features within that segment. We input these features (2048D) to a 3-layer FC neural network. The first FC layer has 512 units followed by 32 units and 1 unit FC layers. 60% dropout regularization is used between FC layers. We use ReLU [19] activation and Sigmoid activation for the first and the last FC layers respectively, and employ Adagrad [14] optimizer with the initial learning rate of 0.001. The parameters of sparsity and smoothness constraints in the MIL ranking loss are set to $\lambda 1{=}\lambda 2 = 8{\times}10^5 and \lambda 3 = 0.01$ for the best performance.

4.7 Evaluation Metric

Following previous works on anomaly detection [27], we use frame based receiver operating characteristic (ROC) curve and corresponding area under the curve (AUC) to evaluate the performance of our method. We do not use equal error rate (EER) [27], as it does not measure anomaly correctly, specifically if only a small portion of a long video contains anomalous behavior. The AUC (Area Under the Curve) curve is a graphical representation of the performance of a binary classification model. It is commonly used to evaluate and compare the performance of different models. The AUC curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds.

To understand the AUC curve, it's important to first understand TPR and FPR. TPR is the ratio of true positives to the total number of actual positive instances in a dataset. In other words, it measures the proportion of actual positive instances that the model correctly identifies as positive. FPR is the ratio of false positives to the total number of actual negative instances in a dataset. It measures the proportion of actual negative instances that the model incorrectly identifies as positive

5. System design

5.1 System Design

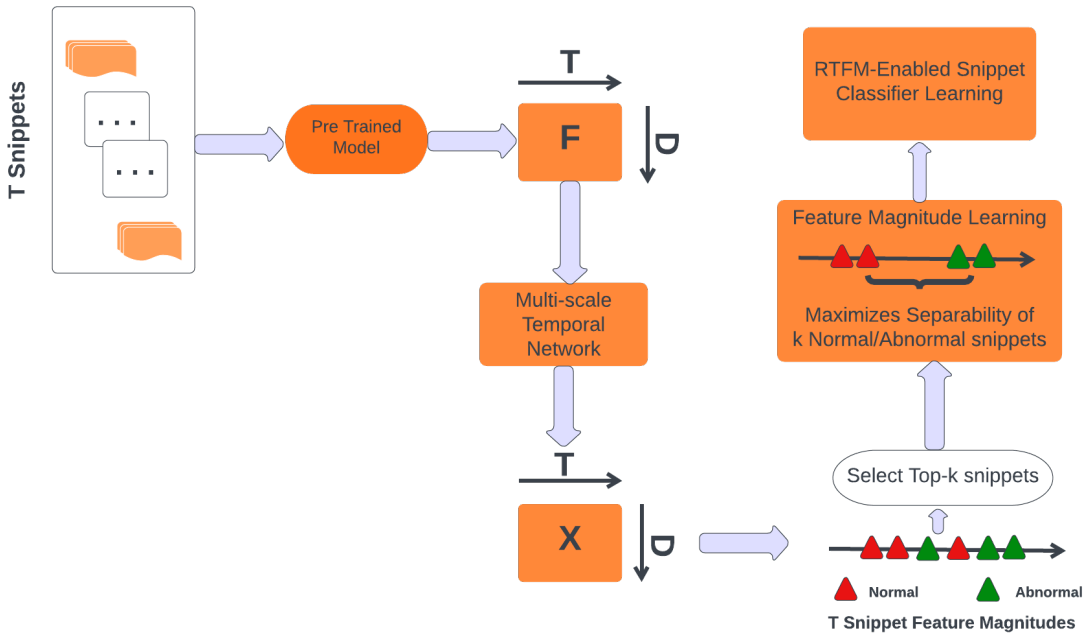


Figure 5.1: High Level System block Diagram

5.1.1 UML Diagrams

Usecase Diagram

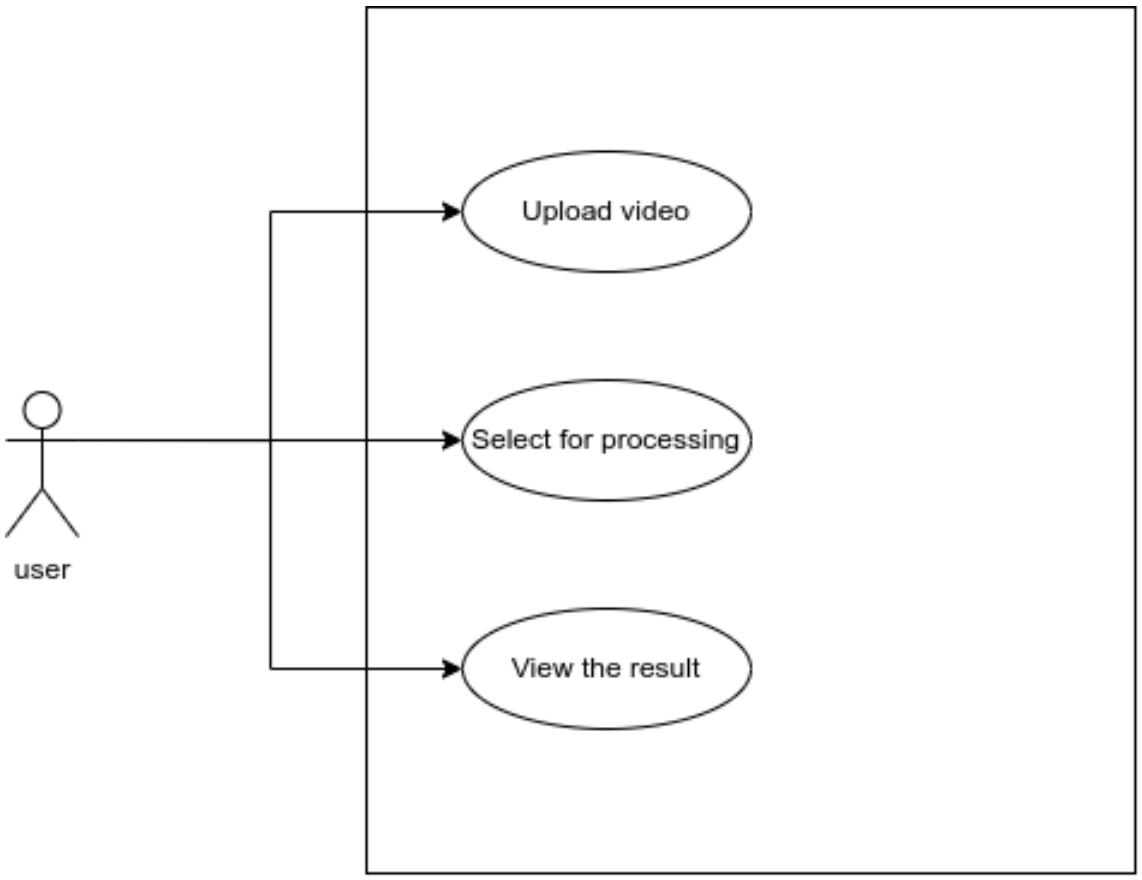


Figure 5.2: Usecase Diagram

Class diagram

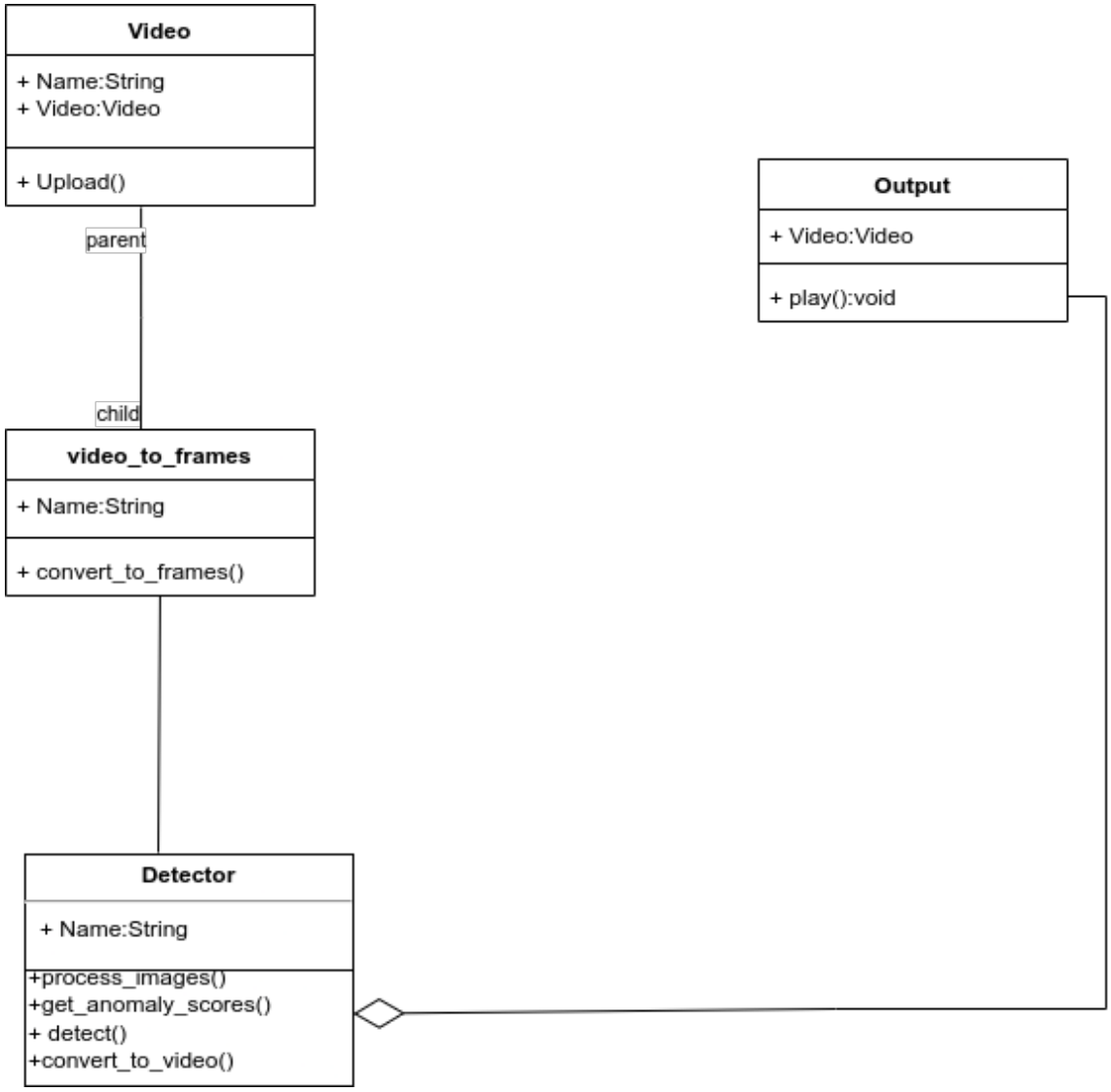


Figure 5.3: Class Diagram

Object diagram

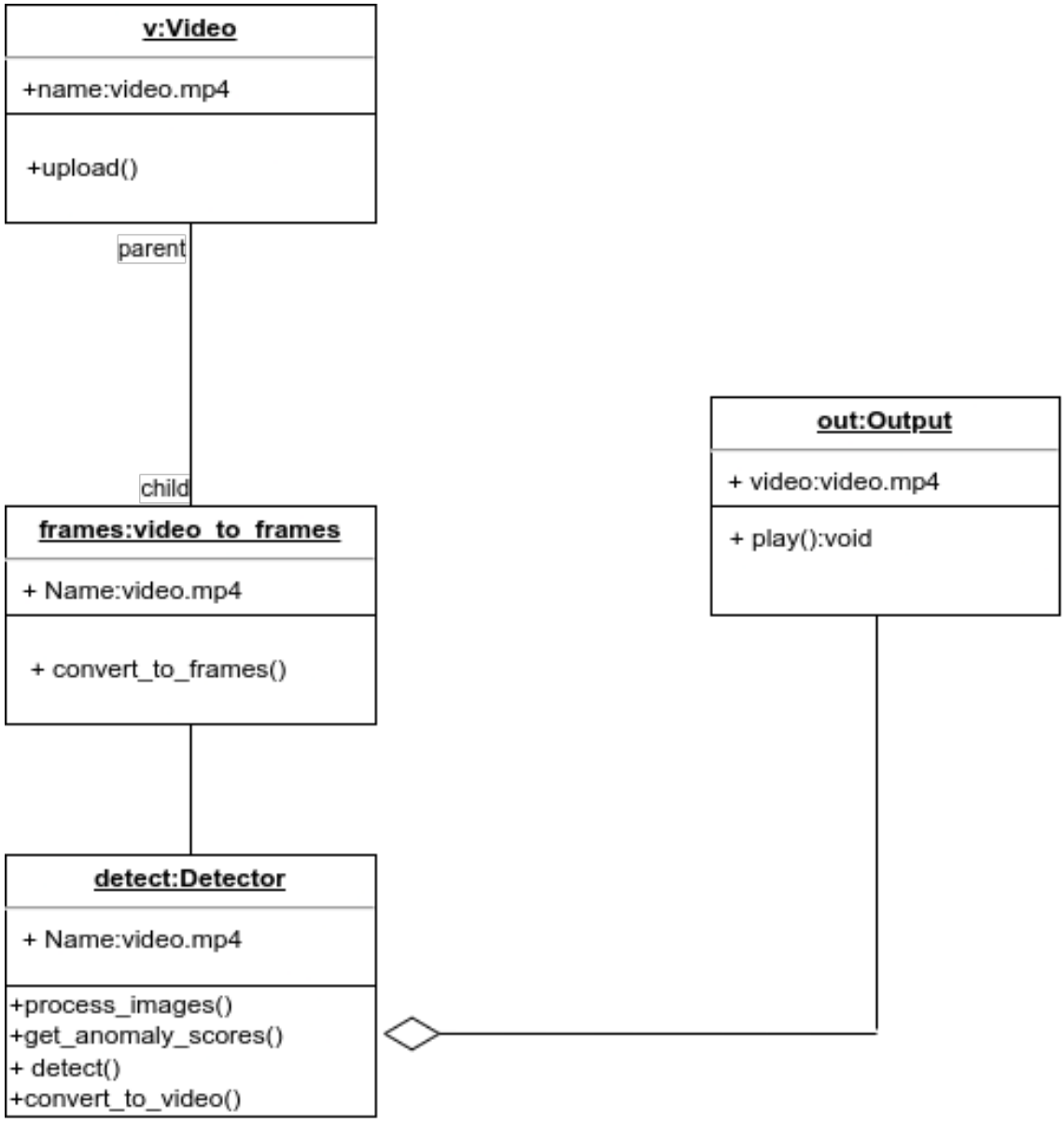


Figure 5.4: Object Diagram

Deployment diagram

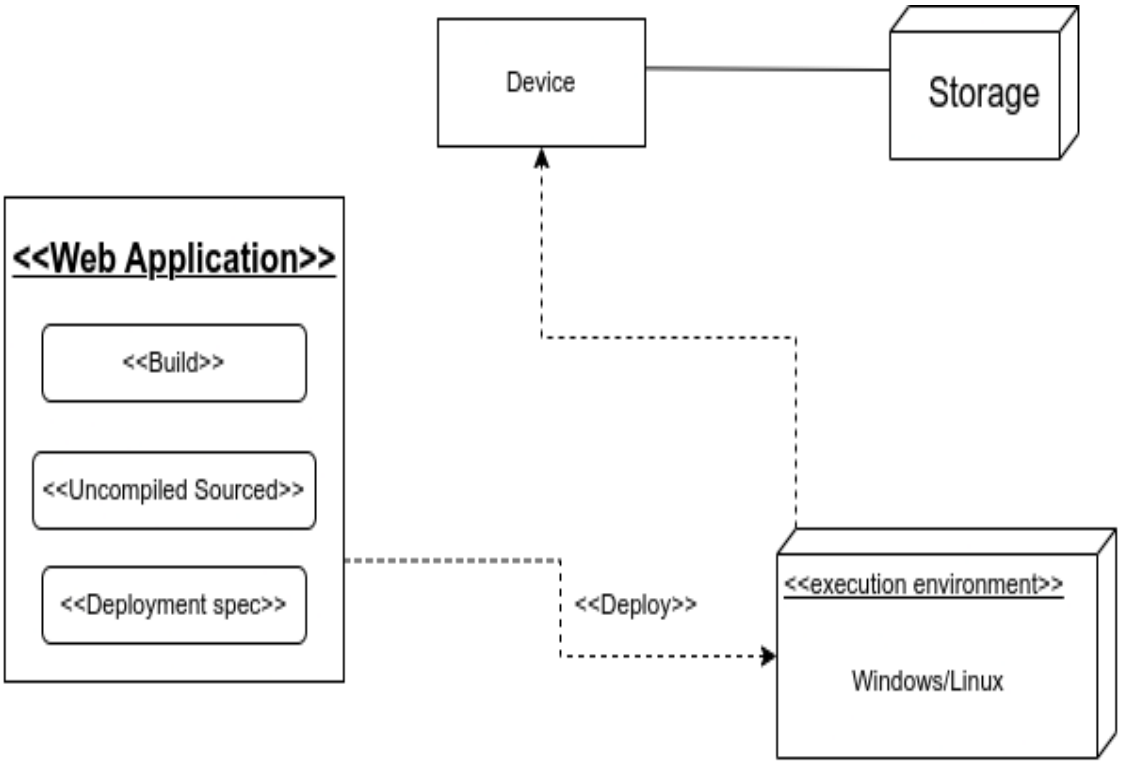


Figure 5.5: Deployment Diagram

Activity diagram

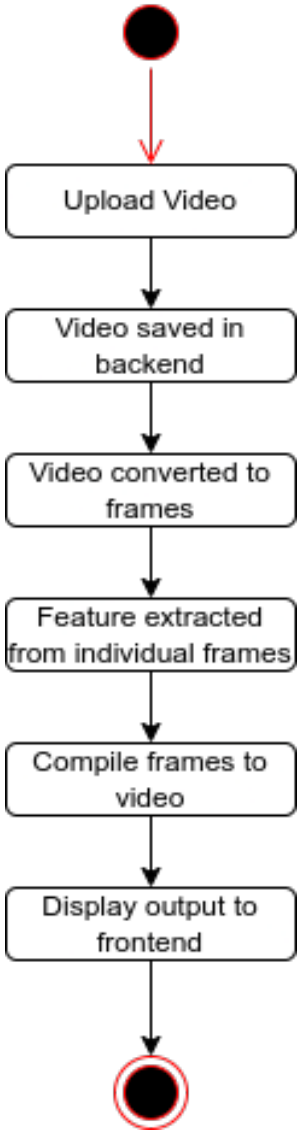


Figure 5.6: Activity Diagram

Sequence diagram

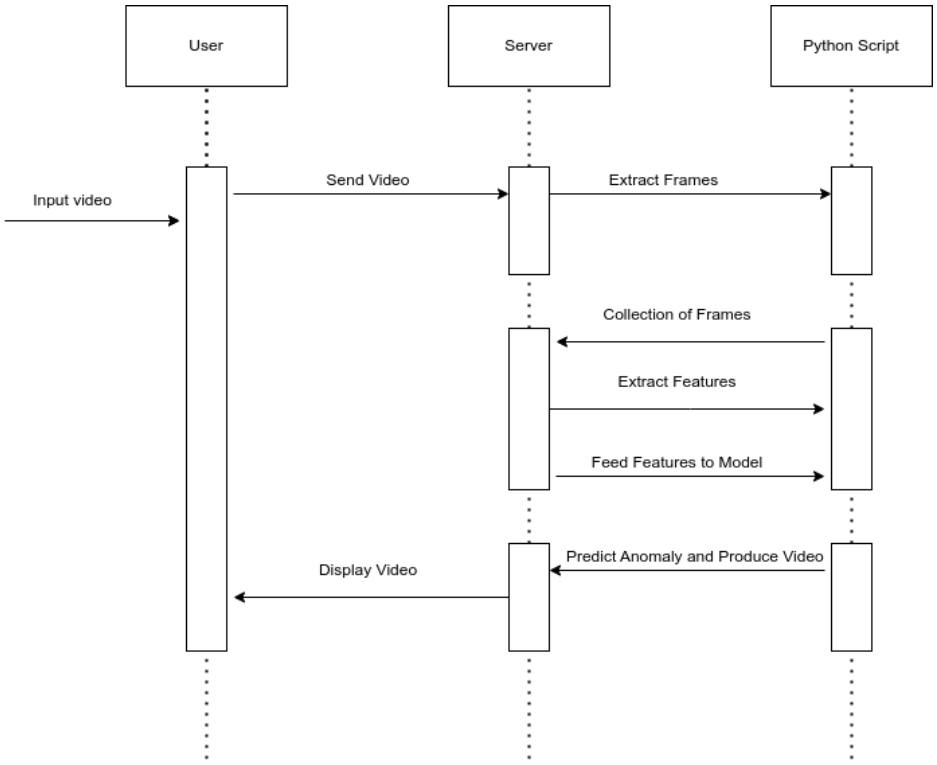


Figure 5.7: Sequence Diagram

6. Results

6.1 Model Performance

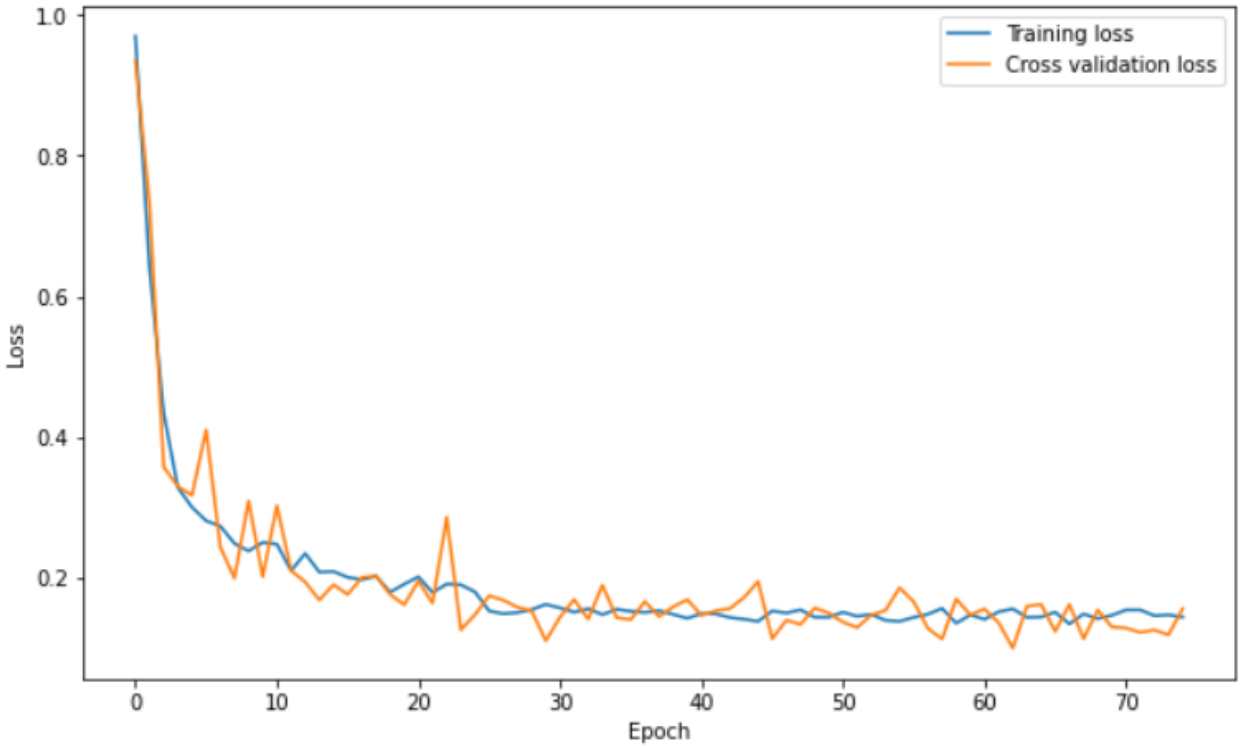
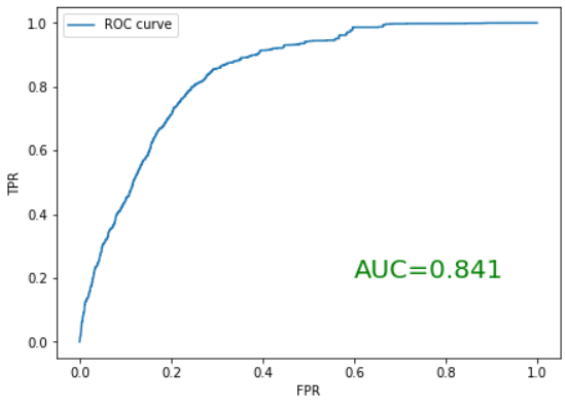
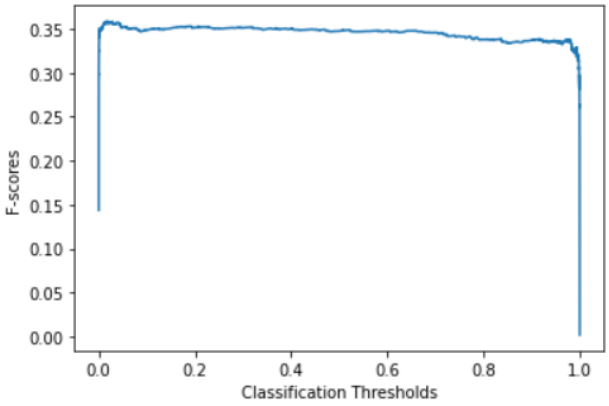


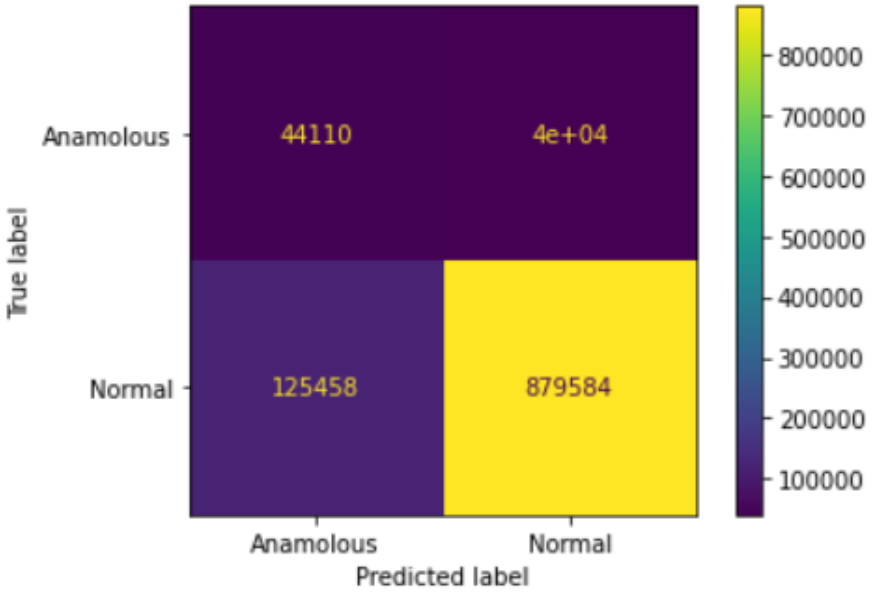
Figure 6.1: Training and Validation Losses



(a) ROC curve for the test set



(b) Fscores for various thresholds



(c) Confusion Matrix for test set

Figure 6.2: ROC curve(a) and Confusion matrix(c) for test set, and F-scores for various thresholds(b)

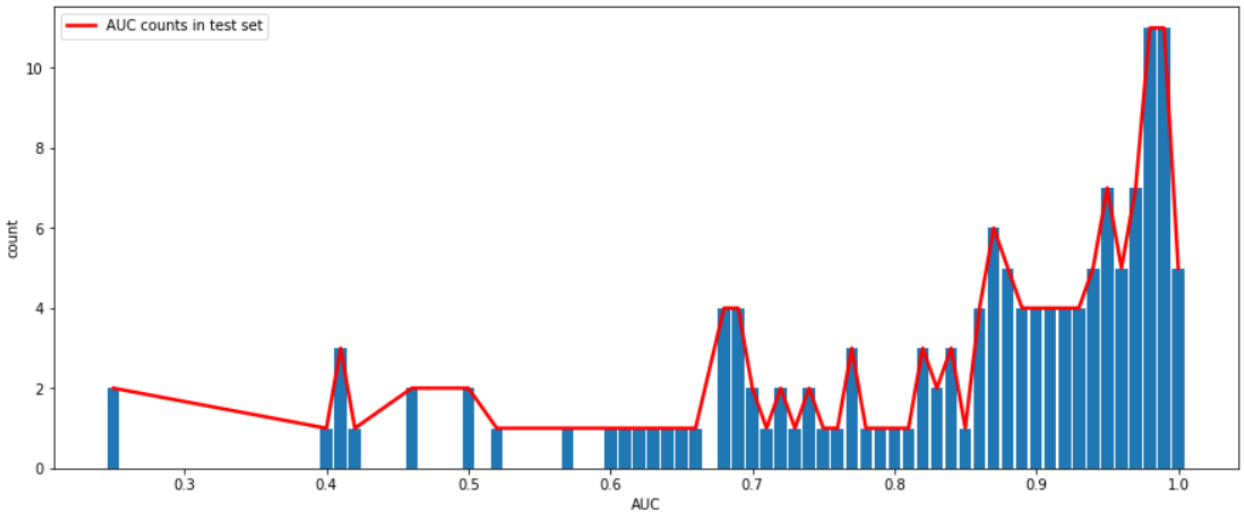


Figure 6.3: AUC for each video in the test set

The webpage consists of a upload button from which an anomaly video can be uploaded. On submitting, the video is processed which may take several minutes depending on the size. After the video is processed, the application is redirected to a video player page where the uploaded video is displayed with obtained anomaly identifications.

Figure 6.4: Normal output-1

Figure 6.5: Normal output-2

Figure 6.6: Normal output-3

Figure 6.7: Explosion output-1

Figure 6.8: Arrest output-2

Figure 6.9: Abuse output-3

Figure 6.10: Fighting output-4

7. Discussion

In this project we reviewed many existing literatures in the field of video anomaly detection, action recognition and Optical Flow extraction. We found that semi-supervised MIL based approach performed the best, as they perform better than unsupervised techniques and require datasets with minimal labels compared to supervised datasets. In action recognition inflated 3D architectures were popular, and performed well due to the use of parameters from their trained 2D counterparts which have learned to recognize useful features from images. Kinetics 400 was one of the popular datasets on which most action recognition models were trained. Optical Flow extraction was useful in many fields but most of the algorithms are computationally expensive and can't achieve real time speed. However, there are several DL architectures which are computationally cheaper and sufficiently accurate that can operate in real time. We use the DL architecture TVL1 for this purpose. By observing its output on some videos, We found that it performed reasonably well.

We also trained a model to give anamolous scores to input video. We used the pretrained action recognition model to extract features from the *mix5c* layer for both RGB and Optical flow frames, each of which had the dimensions (1,1024). We used combined them into a (1, 2048) feature vector and used them as input to our anomaly score prediction model. We trained the model on UCF101 dataset and used cross validation process for training, due to the relatively small size of the dataset. As we see in figure 6.1, the training and validation accuracy are quite good, but the loss curve is not smooth. This might be due to the value of learning rate being too high which causes the model to overshoot at each iteration. In figures 6.2c and 6.2b we see that the model suffers from high False Negative Rate(FNR). This might be caused by the noise being inserted by the weak labels, and large size of MIL bags(32 in our case) which have only few anamolous portions. We are confident that by using better, more robust techniques for optical flow, more recent action recognition models, and a more complex anamoly detection model, we can achieve lower FNR and higher F-scores.

8. Limitations

Little dataset available. Requires extreme computation. Difficult to identify some anomaly in dataset.

- Little dataset available.
- Requires extreme computation.
- Difficult to identify some anomaly in dataset.
- Minimal dataset is available

9. Future Improvements

- Extract video clips
- Automatically list out time intervals of incidents
- Improve UI further
- Current processing speed with GPU is 10s per 1s of clip. Add multithreading and asynchronous processing to improve speeds (3-4x gain)
- Use transformer for better feature extraction
- Can be hosted on cloud for better computation.

10. Project Schedule

10.1 Gantt Chart

S.N	Tasks	Month 1	Month 2	Month 3	Month 3-6	Month 7	Month 8
1	Research						
2	Requirement Analysis						
3	Coding						
4	Testing						
5	Implementation						
6	Documentation						

Figure 10.1: Gantt Chart

11. References

[1] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] Arslan Basharat, Alexei Gritai, and Mubarak Shah. Learning object motion patterns for anomaly detection and improved object detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[3] Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. arXiv preprint arXiv:2005.02359, 2020.

[4] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad – a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] Philippe Burlina, Neil Joshi, and I-Jeng Wang. Where's wally now? deep generative and discriminative embeddings for novelty detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 6299–6308, 2017.

[8] Yuanhong Chen, Yu Tian, Guansong Pang, and Gustavo Carneiro. Unsupervised anomaly detection with multiscale interpolated gaussian descriptors. In *arXiv preprint arXiv:2101.10043*, 2021.

[9] and Yie-Tarng Chen Kai-Wen Cheng and Wen-Hsien Fang. Video anomaly detection and localization using hierarchical feature representation and gaussian process regression.

In *n Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] and J Andrew Bagnell Allison Del Giorno and Martial Hebert. A discriminative framework for anomaly detection in large videos. In *European Conference on Computer Vision*, number 334-349.Springer, 2016.

[11] Zhiwen Fang, Jiafei Liang, Joey Tianyi Zhou, Yang Xiao, and Feng Yang. Anomaly detection with bidirectional consistency in videos. In *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[12] Mariana-Iuliana Georgescu, Antonio Barbalau, Radu Tudor Ionescu, Fahad Shahbaz Khan, Marius Popescu, and Mubarak Shah. Anomaly detection in video via self-supervised and multi-task learning. *arXiv preprint arXiv:2011.07491*, 2020.

[13] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *In Advances in Neural Information Processing Systems*, pages 9758—-9769, 2018.

[14] Dong Gong andLingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 1705—-1714, 2019.

[15] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 733—-742, 2016.

[16] He Chengkun, Shao Jie, and Jiayu Sun. An anomalyintroduced learning method for abnormal event detection. *Multimedia Tools and Applications*, 77(22):29573—-29588, 2018.

[17] Hinami Ryota, Mei Tao, and Shin'ichi Satoh. Joint detection and recounting of abnormal events by learning deep generic knowledge. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 3619—3627, 2017.

[18] and Fahad Shahbaz Khan Radu Tudor Ionescu, Mariana-Iuliana Georgescu, and Ling Shao. Object-centric auto-encoders and dummy anomalies for abnormal event detection in video. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7842—-7851, 2019.

[19] Radu Tudor Ionescu, Sorina Smeureanu, Bogdan Alexe, and Marius Popescu. Unmasking the abnormal events in video. *In Proceedings of the IEEE International Conference on*, pages 2895—2903,, 2017.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. *In CVPR*, 2014.

[21] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. *CVPR*, 2014.

[22] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, and Fabio Viola. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Louis Kratz and Ko Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1446—-1453, 2009.

[25] Weixin Li, Vijay Mahadevan, and Nuno Vasconcelos. ultiple instance learning for soft bags via top instances. *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[26] Weixin Li and Nuno Vasconcelos. Transfer learning based image visualization using cnn. *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4277—-4285, 2015.

[27] C. Liu, X. Xu, and Y. Zhang. emporal attention network for action proposal. *25th IEEE International Conference on Image Processing (ICIP)*, pages 2281—-2285, 2018.

[28] Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[29] Wen Liu, Weixin Luo, Zhengxin Li, and Peilin Zhao. Margin learning embedded prediction for video anomaly detection with a few anomalies. *EEE/CVF Conference on Computer Vision and Pattern Recognition*, 10(4):12173—12182, 2020.

[30] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection–a new baseline. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6536—6545, 2018.

[31] Yuyuan Liu, Yu Tian, Gabriel Maicas, and Leonardo Zorron Cheng Tao Pu. Transfer learning based image visualization using cnn. *IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 2020.

12. Appendix

12.1 Appendix A

Figure 12.1: Input video-1

Figure 12.2: Input Video-2

12.2 Appendix B

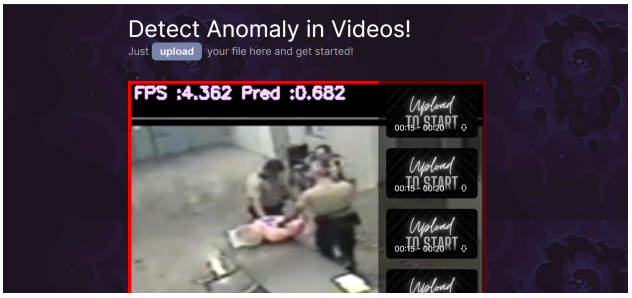


Figure 12.3: Web application

Figure 12.4: Input video-1

12.3 Appendix C

Figure 12.5: Output