Passenger Flow Tracking

Smart Surveillance system

Gamal Ahmed
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
gamal20191700197@cis.asu.edu.eg

Said Ashour
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
saied20191700285@cis.asu.edu.eg

Anas Ahmed
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
anas20191700146@cis.asu.edu.eg

Mohammed Zien El-abdine
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
mohammed20191700532@cis.asu.edu.eg

Mahmoud Ali
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
mahmoud20191700604@cis.asu.edu.eg

Marwa Shams
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
marwa_shams@cis.asu.edu.eg

Dina AlSayad
Scientific Computing Deartment
Fauculty of computer and information Sciences
Ain Shams University
Cairo, Egypt
DinaAlSayyad@cis.asu.edu.eg

Abstract

This paper presents the development and implementation of a real-time anomaly surveillance system. The system leverages advanced techniques from computer vision, machine learning, and artificial intelligence to enable proactive monitoring and detection of anomalies in various environments. Through the utilization of state-of-the-art models and algorithms, the system provides real-time analysis of multiple camera feeds, automatically detects anomalies, and triggers appropriate actions for timely response. The system offers a user-friendly interface, customizable settings, and integration with email notifications for efficient anomaly management. Extensive experiments and evaluations demonstrate the effectiveness and practicality of the system in enhancing situational awareness and improving overall security.

I. Introduction

In today's rapidly evolving world, ensuring the safety and security of our surroundings is of paramount importance. As technological advancements continue to shape our lives, the need for effective surveillance systems that can quickly detect and respond to anomalies becomes increasingly crucial. Real-time anomaly surveillance systems have emerged as a powerful solution, providing proactive monitoring and alerting capabilities to mitigate potential threats and risks. This project focuses on developing an advanced real-time anomaly surveillance system that leverages state-of-the-art technologies to enhance situational awareness and enable timely responses to abnormal events.

A. Problem Definition

The problem at hand is the inherent challenge in efficiently monitoring large-scale environments and swiftly identifying anomalies that deviate from normal patterns or behaviors. Traditional surveillance systems often rely on manual monitoring, making it labor-intensive, time-consuming, and prone to human error. Moreover, detecting anomalies in real-time poses an additional challenge, as the sheer volume of data generated by multiple cameras requires intelligent processing algorithms to analyze and identify abnormal activities accurately. Thus, there is a need for an automated and intelligent system that can continuously monitor multiple camera feeds, detect anomalies in real-time, and provide timely alerts for proactive decision making and response.

B. Motivation

The motivation behind this project stems from the pressing need for effective anomaly surveillance systems to bolster security measures in various domains. By harnessing the power of advanced computer vision, machine learning, and artificial intelligence techniques, we aim to develop a system that can autonomously monitor diverse environments, such as public spaces, transportation hubs, and critical infrastructure. Our goal is to enable early detection of anomalies, ranging from suspicious activities to potential safety hazards, thereby minimizing the risk of incidents, improving emergency response times, and ultimately enhancing overall safety and security for individuals and organizations alike. By developing an intelligent real-time anomaly surveillance system, we aim to contribute to the advancement of surveillance technologies and their practical implementation in real-world scenarios.

II. Related Works

This chapter explores the field of anomaly detection, which is crucial in domains such as cybersecurity, fraud detection, and surveillance. Anomalies are abnormal instances in a dataset, and their detection is essential for system integrity and operational efficiency. The chapter discusses various approaches, including supervised learning, unsupervised learning, weakly supervised learning, and multiple instance learning, for detecting anomalies. It explores the use of labeled training data, handcrafted features, and deep neural networks in supervised learning. It also covers unsupervised methods that model normal data distribution and recent advances in deep learning. Weakly supervised learning leverages limited labeled abnormal samples, incorporating prior knowledge and statistical deviations. Multiple-instance learning operates at the bag level, assigning labels to collections of instances, and focuses on discriminative patterns between normal and abnormal bags. By examining these approaches, the chapter aims to provide a comprehensive understanding of anomaly detection methods and help identify suitable techniques for different application domains.

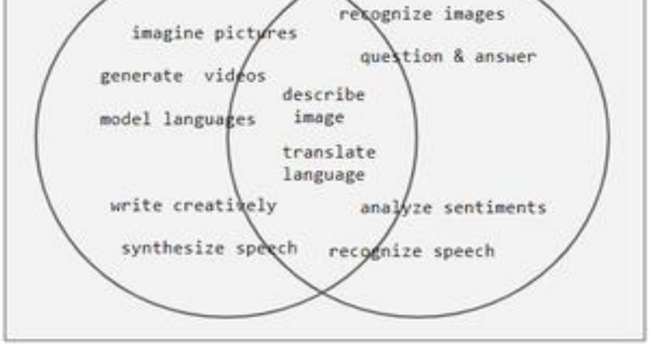
A. Supervised Learning

Supervised learning is a machine learning approach where data consists of labeled examples. The goal is to learn a function that maps input features to output labels based on training examples. The algorithm analyzes the training data and produces a function for mapping new examples. The accuracy of the learned function depends on the representation of input features. The learning algorithm and its structure are determined based on the problem at hand. After the design, the learned function's accuracy is evaluated on a separate test set. However, applying supervised learning to anomaly detection is challenging due to the need for frame-level labeling, which requires significant human effort and time. Anomaly detection datasets are often unbalanced, with a low ratio of anomaly frames, making it difficult for fully supervised approaches to perform well. [1]

B. Unsupervised Learning

Unsupervised learning involves algorithms that extract patterns and information from unlabeled data. Unlike supervised learning, which relies on labeled examples to learn how to map input to output (e.g., classifying images as "cat" or "fish"), unsupervised methods focus on learning compact representations of the input data. These representations can be utilized for various purposes such as data exploration, data analysis, or even generating new data. On the spectrum of supervision levels, there are other approaches like reinforcement learning, where the machine receives guidance in the form of a "performance score" to optimize its behavior, and semi-supervised learning, where only a subset of the training data is labeled while the rest remains unlabeled. [2]

1) Neural Networks*: Tasks vs. Methods*



The division between supervised and unsupervised methods in learning tasks is becoming less clear in today's schemes. Neural network tasks are often categorized as discriminative (supervised) or generative (unsupervised), but there are exceptions and a hazy separation. For example, object recognition can be done with both supervised and unsupervised methods. Some tasks now employ a combination of both methods, and there is a shift from one to another. During the learning phase, unsupervised networks mimic the given data and correct themselves based on the error in their output. Unsupervised learning employs various methods such as Hopfield learning, Boltzmann learning, and Contrastive Divergence, while supervised methods mainly use backpropagation.

C. Unsupervised Anomaly Detection

Traditional methods for anomaly detection typically assume that only normal training data is available. They tackle the problem through one-class classification using manually designed features [3]. In more recent approaches, deep learning techniques have been employed, utilizing features extracted from pre-trained deep neural networks. Another set of methods impose constraints on the latent space of normal patterns to learn condensed representations of normality [4,5,6,7,8]. Alternatively, some approaches rely on generative models and data reconstruction to learn representations of normal samples by minimizing the reconstruction error through adversarial techniques [9,10,11]. These methods assume that unseen anomalous videos or images cannot be effectively reconstructed and identify samples with high reconstruction errors as anomalies. However, these approaches can suffer from overfitting to the training data and struggle to accurately distinguish abnormal events from normal ones, primarily due to the lack of prior knowledge about abnormality.

D. Weakly-Supervised Learning

Weak supervision, or semi-supervised learning, combines a small amount of labeled data with a larger amount of unlabeled data during training. It bridges the gap between unsupervised learning and supervised learning. Semi-supervised learning is useful when labeled data is scarce and expensive to obtain. Unlabeled data, when used alongside labeled data, can significantly improve learning accuracy. Acquiring labeled data often requires human expertise or costly experiments, making fully labeled training sets impractical. Semi-supervised learning assumes some relationship to the underlying data distribution and makes use of various assumptions to leverage unlabeled data. It is both practically valuable and of theoretical interest in machine learning and human learning models [12,13,14,15,16,17].

E. Multiple -Instance Learning

Multiple-instance learning (MIL) is a form of supervised learning where the learner receives labeled bags, each containing multiple instances, instead of individually labeled instances. In binary classification, a bag is labeled negative if all instances in it are negative, and positive if it contains at least one positive instance. The goal is to either induce a concept that labels individual instances correctly or learn how to label bags without explicitly inducing the concept. An example of MIL is predicting whether a key or key chain can grant access to a room, where the task involves identifying the common key among positive key chains to classify entire key chains accordingly [18,19].

III. System Architecture

This chapter provides a detailed overview of the architecture used in surveillance systems for efficient video analysis. The architecture consists of three stages: anomaly prediction, object detection, and feature extraction. By integrating these components, the system enables comprehensive video analysis, facilitating real-time insights and proactive decision-making for security threats.

A. Feature Extraction with I3D Model

The feature extraction stage utilizes the I3D model to extract essential spatio-temporal features from video frames. The I3D model, pre-trained on large-scale datasets such as Kinetics, captures both spatial and temporal information by incorporating two-stream inflated 3D ConvNets (I3D) architecture. It processes the raw video frames and generates rich feature representations, which encode motion and appearance cues that are crucial for subsequent analysis.

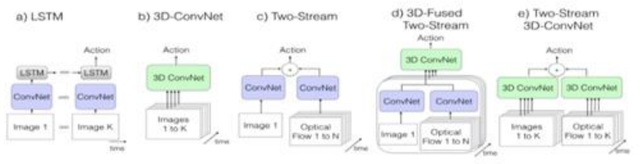


Figure 1: I3D Architecture.

B. Anomaly Prediction with Robust Temporal Feature Magnitude Learning (RTFM) Model

The anomaly prediction stage employs the RTFM model, which utilizes deep learning techniques to compute anomaly scores for each frame, enabling the identification of abnormal events. RTFM improves multiple-instance learning (MIL) by training a feature magnitude learning function to effectively recognize positive instances and enhance the robustness of MIL to negative instances in abnormal videos. It incorporates dilated convolutions and self-attention mechanisms to capture both long- and short-range temporal dependencies, resulting in more accurate learning of feature magnitudes. Extensive experiments demonstrate that the RTFM-enabled MIL model outperforms state-of-the-art methods by a large margin on benchmark datasets, showcasing improved discriminability of subtle anomalies and sample efficiency.

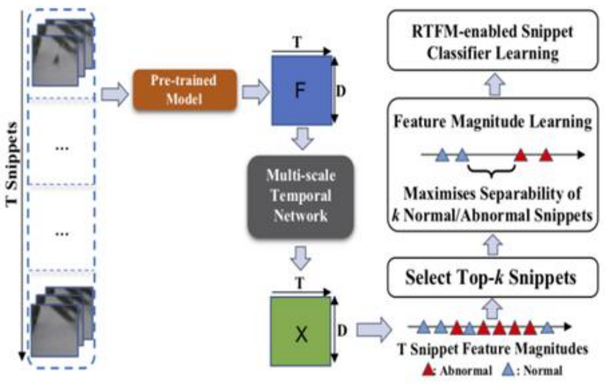

T Snippets
Pre-trained Model
F
T
D
Multi-scale Temporal Network
X
RTFM-enabled Snippet Classifier Learning
Feature Magnitude Learning
Maximises Separability of k Normal/Abnormal Snippets
Select Top-k Snippets
T Snippet Feature Magnitudes
▲: Abnormal ▲: Normal

Figure 2: RTFM Architecture.

B. Object Detection with YOLOv8 Model

The object detection stage uses the YOLOv8 model to detect and localize individuals within the scene. The YOLOv8 model operates on the raw video frames, leveraging its advanced architecture and trained weights to accurately identify individuals in real-time. By employing a single unified model, YOLOv8 performs simultaneous object detection and classification, providing boundi ng box information and relevant object attributes.

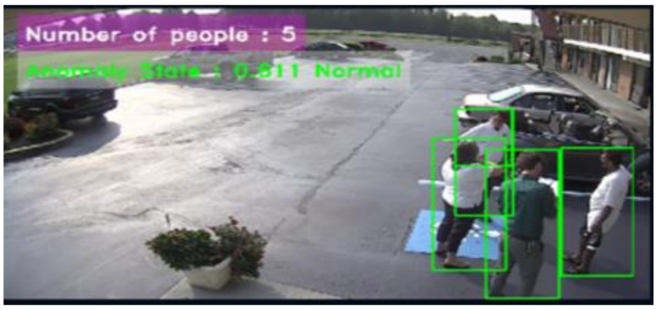


Figure 3: YOLO Results.

C. Data Flow Diagram

The data flow diagram represents the flow of data within an architecture, illustrating how information is processed, transformed, and exchanged between various components. I chose to use a data flow diagram for my architecture to gain a clear understanding of the system's data flow and identify any potential bottlenecks or inefficiencies. By visualizing the movement of data through different stages and components, I can analyze the interactions and dependencies, ensuring that data is properly handled and utilized throughout the architecture. The data flow diagram helps in identifying potential areas of improvement, optimizing data processing, and enhancing overall system performance.

1) **DFD Context Level:** The system operates by utilizing two external sources: cameras and an observer. The cameras capture frames, which are then input into the system. These frames are processed within the system, which subsequently generates results. These results are then communicated back to the observer for further analysis or action.

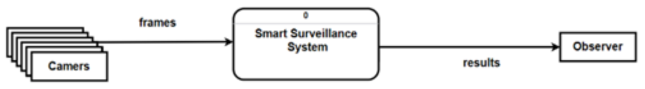


Figure 4: DFD Context Level.

2) **DFD Level 0:** At this level, we delve into the internal components of the system to understand the flow of data and operations. As depicted in Figure [x], the camera frames (Process 1.0) serve as the primary input to the system. These frames are then simultaneously received by two key processes within the system. Process 2.0, responsible for anomaly detection, analyzes the frames and generates an anomaly score that represents th e likelihood of each frame being considered an anomaly. The anomaly scores serve as one of the outputs of Process 2.0. Simultaneously, Process 3.0 focuses on detecting individuals within the frames. By utilizing advanced algorithms, such as the Ultralytics YOLOv8 model, it identifies individuals and provides detection results, including bounding boxes or other relevant information. The detection results constitute the second output of Process 3.0. To consolidate the outputs from Processes 2.0 and 3.0, the system employs Process 4.0, which merges the anomaly scores and detection results. By combining these pieces of information, the system enhances the overall understanding of the observed frames. Subsequently, the merged outputs are directed to Process 5.0, which is responsible for displaying the results in the graphical user interface (GUI). The GUI serves as the interface between the system and the observer, presenting the consolidated information in a visually accessible manner. The displayed results in the GUI are then communicated back to the observer, enabling real-time monitoring and analysis. This feedback loop ensures that the observer has access to the processed information, allowing them to make informed decisions or take appropriate actions based on the system's outputs.

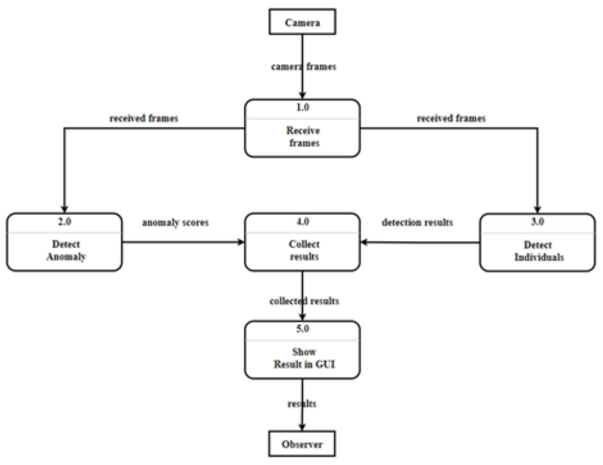


Figure 5: DFD Level 0.

3) **DFD Level 1-RTFM:** In this Level, we focus on the detailed process of detecting anomalies and computing anomaly scores. The frames received from Process 1.0 undergo several preprocessing steps before being analyzed.
Process 2.1 is responsible for preprocessing the frames. It applies various techniques to enhance the quality and prepare the frames for further analysis. Once preprocessed, the frames are passed to Process 2.2, which applies a 10-crop augmentation technique. This augmentation method involves creating multiple variations of each frame by taking ten different crops, resulting in an increased diversity of data for improved anomaly detection accuracy. The augmented frames are then forwarded to Process 2.3, where the I3D model is utilized. The I3D model extracts essential features from the frames, capturing temporal information and spatial details that are crucial for anomaly detection. Subsequently, the extracted features are fed into Process 2.4, which employs the RTFM model. This model utilizes the extracted features to compute anomaly scores for each frame. The anomaly scores quantify the likelihood of a frame being classified as an anomaly based on its unique characteristics and patterns.
The computed anomaly scores are then collected in Process 4.0, where they are stored or further processed for subsequent analysis or visualization. Process 4.0 serves as a central point for aggregating the anomaly scores from the previous steps, facilitating efficient management and utilization of the anomaly detection results. By examining this Level 1 DFD diagram, we gain a more detailed understanding of the internal processes involved in detecting anomalies and computing anomaly scores. The diagram highlights the specific steps of frame preprocessing, augmentation, feature extraction, anomaly score computation, and result collection.

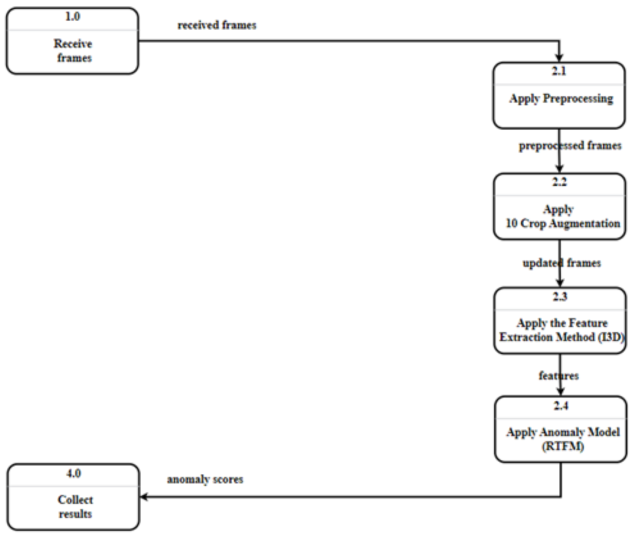


Figure 6: DFD Level 1-RTFM.

4) **DFD Level 1-YOLO:** In this Level 1 DFD diagram, we focus on the details of detecting individuals and extracting their boundary box information using the YOLO detection model. The frames received from Process 1.0 are passed to Process 3.1 for analysis. Process 3.1 utilizes the YOLO detection model to identify and locate individuals within the frames. The model processes the frames and outputs boundary box information, specifying the coordinates and dimensions of each detected individual. The boundary box information is then simultaneously fed into two processes: Process 3.2 and Process 3.3. Process 3.2 is responsible for drawing the boundary boxes on the frames. This process overlays the boundary boxes onto the corresponding frames, visually highlighting the detected individuals for better visualization and understanding. Meanwhile, Process 3.3 counts the number of individuals represented by the boundary boxes. It analyzes the boundary box information and determines the total count of individuals detected within the frames. Both the frames with drawn boundary boxes from Process 3.2 and the count of individuals from Process 3.3 are then forwarded to Process 3.4. Process 3.4 serves as a central hub, gathering the results from the previous steps. Finally, the gathered results from Process 3.4 are sent to Process 4.0. Process 4.0 is responsible for collecting and further processing the results for subsequent actions, such as displaying the collected information in the GUI or communicating it to other parts of the system. By examining this Level 1 DFD diagram, we gain a more detailed understanding of the internal processes involved in detecting individuals, extracting boundary box information, drawing the boxes on frames, counting the number of individuals, and collecting the results. The diagram showcases the flow of data and interactions between the various processes in this specific level of the system architecture.

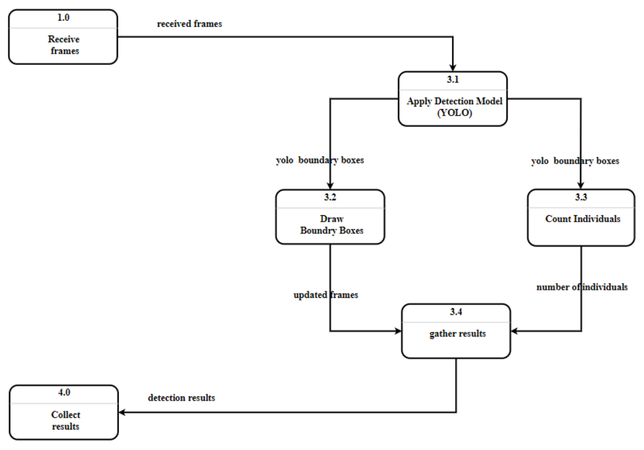


Figure 7: DFD Level 1-YOLO

IV. System Implementation and Results

A. Hardware and Software Setup

The surveillance system was implemented on a machine equipped with an NVIDIA GeForce GTX 1660 GPU for

model training and inference. The system was running on the Ubuntu Linux operating system. The software tools and libraries used in the implementation include:

- Python programming language
- PyTorch deep learning framework
- OpenCV for video frame capture and preprocessing
- PyQT5 library for creating the graphical user interface (GUI)

B. Dataset Description

The surveillance system utilized the ShanghaiTech dataset, which is a medium-scale dataset consisting of fixed-angle street video surveillance. The dataset includes 437 videos with 13 different background scenes. It contains a combination of 307 normal videos and 130 anomaly videos. The dataset was reorganized using the procedure outlined in [21], which involved selecting a subset of anomalous testing videos into the training data to create a weakly supervised training set. This reorganization ensured that both the training and testing sets covered all 13 background scenes.

C. Training Procedure

The RTFM model was trained from scratch for 15,000 epochs using the ShanghaiTech dataset. The training process was performed on the NVIDIA GeForce GTX 1660 GPU. After training, the RTFM model achieved an Area Under the Curve (AUC) of 95.6%, indicating its ability to accurately identify anomalies in the surveillance videos. The RTFM model was trained using the following parameters:
Adam Optimizer with a Weight Decay of 0.0005 and a Batch Size of 64 with a Learning Rate of 0.001.

During training, the model learned to compute anomaly scores for each frame, enabling the identification of abnormal events in surveillance videos. The training process was performed on the NVIDIA GeForce GTX 1660 GPU.

D. Integration Details

The integration of the different components in the surveillance system is as follows:

Frame Acquisition and Preprocessing:

Frames are captured from the video feed or camera using OpenCV. The system captures 20 frames at a time to ensure efficient processing. Preprocessing techniques are applied to enhance the quality of the frames and prepare them for further analysis.

Feature Extraction using I3D Model:

One thread receives the preprocessed frames and applies ten-crop augmentation, generating multiple variations of each frame. The augmented frames are fed into the I3D model, which extracts high-level features. Features are extracted from the 'mix 5' layer of the I3D model, resulting in feature representations of size 1x10x2048 for each batch of 20 frames. The feature extraction process takes approximately 2 seconds per batch of 20 frames.

Anomaly Prediction using RTFM Model:

The extracted features from the I3D model are passed to the RTFM model. The RTFM model utilizes deep learning techniques to compute anomaly scores for the batch of 20 frames. The anomaly scores represent the likelihood of each frame in the batch being considered an anomaly based on its unique characteristics and patterns. The RTFM model provides anomaly scores almost instantaneously for the batch of 20 frames.

Object Detection using YOLO Model:

Another thread receives the raw frames captured by the camera.

The frames are passed to the YOLOv8 model, which performs object detection and localization. The YOLOv8 model detects and localizes individuals within the frames, generating bounding box coordinates. The object detection process takes approximately 0.25 seconds per frame.

E. Integration of Results:

The anomaly scores, bounding box coordinates, and the count of individuals are collected from the respective threads. The main camera thread combines the results and displays them in the application's graphical user interface (GUI). The GUI serves as the interface between the system and the observer, providing real-time monitoring and analysis of the surveillance environment.

F. Results

The performance of the surveillance system is measured in terms of processing time and the AUC achieved by the RTFM model. The following metrics were observed during the system's operation:

YOLOv8 took approximately 0.25 seconds to output the object detection results per frame. The I3D model took around 2 seconds to extract features from each batch of 20 frames. The RTFM model provided anomaly scores almost instantaneously for each batch of 20 frames. After training on the ShanghaiTech dataset, the RTFM model achieved an AUC of 95.6%, indicating its effectiveness in detecting anomalies in the surveillance videos.

References

[1] https://en.wikipedia.org/wiki/Supervised_learning.

[2] https://en.wikipedia.org/wiki/Unsupervised_learning.

[3] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1386–1393, 2014.

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

[5] Guansong Pang, Cheng Yan, Chunhua Shen, Anton van den Hengel, and Xiao Bai. Self-trained deep ordinal regression for end-to-end video anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12173–12182, 2020.

[6] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10076–10085, 2020.

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

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

[9] 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.

[10] Radu Tudor Ionescu, Fahad Shahbaz Khan, 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.

[11] Xinyang Feng, Dongjin Song, Yuncong Chen, Zhengzhang Chen, Jingchao Ni, Haifeng Chen. Convolutional Transformer based Dual Discriminator Generative Adversarial Networks for Video Anomaly Detection.

[12] https://en.wikipedia.org/wiki/Weak_supervision

[13] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 353 – 362, 2019.

[14] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6479–6488, 2018.

[15] Yu Tian, Gabriel Maicas, Leonardo Zorron Cheng Tao Pu, Rajvinder Singh, Johan W Verjans, and Gustavo Carneiro. Few-shot anomaly detection for polyp frames from colonoscopy. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VI 23, pages 274–284. Springer, 2020.

[16] Peng Wu, jing Liu, Yujia Shi, Yujia Sun, Fangtao Shao, Zhaoyang Wu, and Zhiwei Yang. Not only look, but also listen: Learning multimodal violence detection under weak supervision. In European Conference on Computer Vision (ECCV), 2020.

[17] Muhammad Zaigham Zaheer, Jin-ha Lee, Marcella Astrid, Arif Mahmood, and Seung-Ik Lee. Cleaning label noise with clusters for minimally supervised anomaly detection. arXiv preprint arXiv:2104.14770, 2021.

[18] https://en.wikipedia.org/wiki/Multiple_instance_learnin g

[19] Yu Tian, Guansong Pang, Yuanhong Chen, Rajvinder Singh, Johan W. Verjans, Gustavo Carneiro. Weakly-supervised Video Anomaly Detection with Robust Temporal Feature Magnitude Learning arXiv:2101.10030.

[20] Joao Carreira, Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset arXiv:1705.07750.

[21] Jia-Xing Zhong, Nannan Li, Weijie Kong, Shan Liu, Thomas H Li, and Ge Li. Graph convolutional label noise cleaner: Train a plug-and-play action classifier