Anomaly Detection for Video Surveillance

Mrs. Prakruthi G R [1], Mahim HM[2], M Shamil [3], Chiraag R Mishra[4]

[1]Assistant Professor, Dept. of ISE, East West Institute Of Technology, Bengaluru
[2]Student, Dept. of AI&DS, East West Institute Of Technology, Bengaluru
[3] Student, Dept. of AI&DS, East West Institute Of Technology, Bengaluru
[4] Student, Dept. of AI&DS, East West Institute Of Technology, Bengaluru
[5] Student, Dept. of AI&DS, East West Institute Of Technology, Bengaluru

---***---

Abstract - Anomaly detection in video surveillance is critical for identifying abnormal behaviours or events that deviate from typical patterns. This abstract explores various techniques used to detect anomalies in surveillance videos, including traditional methods like motion detection and background subtraction, as well as advanced approaches such as deep learning and anomaly scoring algorithms. By leveraging these techniques, surveillance systems can automatically detect suspicious activities such as trespassing, theft, or violence, enabling timely intervention and improved security measures. Highlighting the challenges in anomaly detection, such as dealing with complex scenes, occlusions and discusses ongoing research efforts to enhance the accuracy and efficiency of anomaly detection systems in real-world surveillance scenarios.

Key Words: anomaly, suspicious, pattern, motion, detection, security measure

1.INTRODUCTION

In today's increasingly interconnected and technologically driven world, the deployment of video surveillance systems has become ubiquitous aoss various sectors, ranging from public safety and transportation to retail and industrial environments. These systems play a crucial role in monitoring and securing physical spaces, deterring illicit activities, and providing valuable insights for situational awareness and decision-making. However, the sheer volume of video data generated by these systems poses significant challenges for effective monitoring and analysis, necessitating the development of automated anomaly detection techniques to augment human surveillance efforts.

Anomaly detection in video surveillance refers to the process of identifying abnormal behaviors, events, or patterns that deviate from established norms or expected behaviors. These anomalies can manifest in various forms, including suspicious movements, unauthorized access, loitering, vandalism, and other illicit activities. Detecting such anomalies in real-time or post-event analysis is essential for prompt intervention, mitigating potential threats, and enhancing overall security measures.

This research paper explores the various techniques used for anomaly detection in video surveillance, ranging from traditional methods to state-of-the-art approaches. We delve into the underlying principles behind these techniques, their strengths and limitations, and their applicability in real-world surveillance scenarios. Additionally, we discuss the challenges inherent in anomaly detection, such as dealing with complex scenes, occlusions, and real-time processing constraints, and examine ongoing research efforts aimed at overcoming these challenges.

By shedding light on the current landscape of anomaly detection in video surveillance and highlighting emerging trends and technologies, this paper aims to provide insights into the advancements and future directions of surveillance systems for improved security and situational awareness in diverse environments. Through a comprehensive understanding of the principles and techniques involved, stakeholders can make informed decisions regarding the implementation and optimization of anomaly detection systems tailored to their specific needs and operational requirements.

2. LITERATURE REVIEW

Venkata Ramana, Lakshmi Prasanna “Human activity recognition using opencv”, International journal of creative research thoughts(IJCRT),(2021)

Low cost and is fast enough because it process low resolution frames and it is able to recognize the occurrence of uncommon events such as overcrowding and fight in the low resolution video without using any classifier and training datasets initially. The application of this proposed method is to enhance the ATM security.

Lamiyah Khattar, Garima Aggarwal “Analysis of human activity recognition using deep learning.” 2021 11th International Conference on Cloud Computing

Data Science & Engineering Now a days we come across theft and robbery attacks on public in atms. Many a time it becomes difficult for in investigating agencies to track the cases. We are proposing an enhanced ATM security system by proposing a system to detect unusual event detection even in the low resolution videos using ARM 7 LPC 2148. The low resolution camera fitted inside ATM can be used. We are using the inbuilt web camera of the PC or laptop in designing of the prototype.

Long Cheng,Yani Guan "Recognition of human activities using machine learning methods with wearable sensors" IEEE Members Research and Development Department 2017.

This cost again rises if we are handling with event detection. Recently, an algorithm has been developed which is capable of detecting unusual events and for enhancement of atms security where simple low resolution cameras are used because of their lower cost. Our state-of- the-art technique is able to detect the existence of unusual events like face masking. Camera masking, fight or overcrowding in the low resolution video merely by means of standard deviation, statistical method of moving target objects.

The reliance on low-resolution video quality in anomaly detection methods may limit detail and clarity, reducing the discriminative power and accuracy of detection. Environmental factors such as varying lighting conditions and occlusions pose challenges, leading to decreased performance in complex scenes. Techniques may be hardware-specific, hindering scalability and adaptability to emerging threats. Moreover, privacy concerns arise due to the intrusive nature of surveillance systems. These limitations highlight the need for ongoing research to address challenges in accuracy, scalability, adaptability, and privacy, ensuring the effectiveness and ethical deployment of anomaly detection systems in video surveillance.

3. METHODOLOGY

3.1 TensorFlow

TensorFlow is an open-source deep learning framework developed by the Google Brain team. It is designed to facilitate the development and deployment of machine learning models, with a primary focus on neural networks for deep learning tasks.

Here's an overview of key aspects of TensorFlow:

1. Graph-Based Computation: TensorFlow operates on a computational graph, where nodes represent mathematical operations, and edges represent the flow of data (tensors) between these operations. This graph-based approach allows for efficient parallelization and optimization of computations.
2. Flexibility and Extensibility: TensorFlow is known for its flexibility, supporting a wide range of machine learning tasks, including neural networks, natural language processing, and computer vision. It provides a comprehensive set of APIs and tools for building and deploying machine learning models.
3. High-Level APIs: TensorFlow offers high-level APIs, such as Keras, which simplify the process of building and training deep learning models. Keras, originally an independent library, has been integrated seamlessly into TensorFlow, providing a user-friendly interface for model development.
4. Eager Execution: TensorFlow supports eager execution, a mode that allows developers to execute operations immediately as they are called, facilitating easier debugging and a more interactive development experience.
5. TensorBoard: TensorFlow comes with TensorBoard, a visualization toolkit that helps developers visualize and understand the structure and performance of their models. It can display training metrics, model graphs, and more, aiding in the debugging and optimization process.
6. Wide Adoption: TensorFlow has gained widespread adoption in both academia and industry. It is used by researchers, engineers, and data scientists for a variety of applications, including image and speech recognition, natural language processing, and reinforcement learning.

3.2 OpenCV

OpenCV, or Open-Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by the OpenCV community. OpenCV provides a comprehensive set of tools and functions for image and video processing, enabling developers to create applications that involve computer vision tasks. Here's a brief introduction to OpenCV:

1. Computer Vision and Image Processing: OpenCV is designed for computer vision applications, which involve the interpretation of visual data from the world. It provides a wide range of functions for image and video processing, including image filtering, feature detection, object recognition, and more.
2. Cross-Platform: OpenCV is cross-platform and can be used on various operating systems, including Windows, Linux, macOS, and Android. This cross-platform nature makes it versatile and widely applicable for different projects and environments.
3. Programming Language Support: While originally written in C++, OpenCV has interfaces for various programming languages, including Python, Java, and MATLAB. This makes it accessible to a broad audience of developers with different language preferences.
4. Image and Video Input/Output: OpenCV supports reading and writing images and videos in various formats. It can capture and process video streams from cameras, process image files, and handle real-time video input.
5. Computer Vision Algorithms: OpenCV includes a rich set of computer vision algorithms. This includes image filtering, edge detection, object recognition, facial recognition, feature matching, camera calibration, and

more. These algorithms serve as building blocks for developing complex computer vision applications.
6. Machine Learning Integration: OpenCV has been extended to include machine learning functionalities. It supports integration with machine learning libraries such as scikit-learn and TensorFlow, making it a powerful tool for developing applications that involve both computer vision and machine learning.
7. Community and Documentation: OpenCV has a large and active community of developers and researchers who contribute to its development. The library is well-documented, providing extensive resources, tutorials, and examples for developers to learn and utilize its capabilities.

3.3 Flask

Flask is a lightweight and extensible web framework for Python. It is designed to be simple, easy to use, and flexible, making it an excellent choice for building web applications and APIs. Here's a brief overview of Flask:
1. Micro-framework Philosophy: Flask follows a micro-framework philosophy, which means it provides the essentials for building web applications without imposing a rigid structure. Developers have the flexibility to choose and integrate components based on their project requirements.
2. Routing: Flask uses a simple and intuitive routing system, allowing developers to define URL patterns and associate them with specific functions (views) to handle requests. This makes it easy to map URLs to the corresponding logic in the application.
3. Templates: Flask includes a template engine (Jinja2) for rendering dynamic HTML content. This allows developers to separate the presentation layer from the application logic, promoting clean and maintainable code.
4. HTTP Request and Response Handling: Flask simplifies handling HTTP requests and responses. It provides convenient objects for accessing request data (e.g., form data, URL parameters) and for constructing responses (e.g., rendering templates, returning JSON).
5. Extensions: Flask has a modular architecture, and developers can easily extend its functionality using various extensions. These extensions cover a wide range of features, such as authentication, database integration, and form validation.
6. Integrated Development Server: Flask comes with a built-in development server, making it easy to test and debug applications during the development process. However, for production use, it is recommended to deploy Flask applications using production-ready servers.
7. Werkzeug and Jinja2: Flask is built on top of the Werkzeug WSGI (Web Server Gateway Interface) toolkit and uses the Jinja2 template engine. Werkzeug provides utilities for handling WSGI details, while Jinja2 is a powerful and expressive template engine.
8. RESTful Support: Flask is well-suited for building RESTful APIs. Its simplicity and flexibility make it easy to define routes that return JSON responses, making it popular among developers building web services.

4. EXPERIMENTAL SETUP

Constructing a benchmark dataset for anomaly detection in video surveillance requires diverse scenes, including urban, rural, indoor, and outdoor environments, capturing various anomaly types such as altercations, loitering, and unattended objects. Ground truth annotations should detail spatial and temporal anomaly locations for accurate evaluation. Ensure dataset scalability with a balance between size and computational feasibility. Adhere to privacy regulations by anonymizing sensitive information. Capture realistic variations in lighting, weather, and scene dynamics. Publicly release the dataset with comprehensive documentation to facilitate reproducibility and comparison of anomaly detection algorithms. Continuously update and expand the dataset to reflect evolving surveillance scenarios and technological advancements, incorporating feedback from the research community to improve dataset quality and utility.

By utilizing multiple datasets, we aimed to ensure the generalizability and robustness of our proposed approach across different image conditions

4.2 Evaluation Metrics

Evaluation metrics are essential for quantitatively assessing the performance of anomaly detection algorithms in video surveillance. Key metrics include detection accuracy, precision, recall, and F1-score, which measure the algorithm's ability to correctly identify anomalies while minimizing false alarms. The area under the receiver operating characteristic curve (AUC-ROC) and precision-recall curve (AUC-PR) provide comprehensive measures of algorithm performance across different operating points. Mean Average Precision (mAP) is particularly useful for multi-class anomaly detection tasks, computing the average precision across all classes. Additionally, false positive rate (FPR) and false negative rate (FNR) offer insights into the algorithm's error rates. Beyond these, computational efficiency metrics such as processing time and memory usage are also crucial, ensuring that algorithms can scale to real-world surveillance scenarios while maintaining acceptable performance levels.

4.3 Preprocessing Steps and Parameter Tuning

Preprocessing steps are crucial for preparing video data before anomaly detection. Common preprocessing techniques include standardizing video resolutions and frame rates, noise reduction, contrast enhancement, and frame stabilization to improve data quality and reduce noise. Additionally, background modeling and subtraction can help remove static elements and highlight moving objects, simplifying anomaly detection. Parameter tuning involves optimizing algorithm parameters to maximize detection performance. This may include fine-tuning hyperparameters such as learning rates, regularization parameters, and feature selection criteria for machine learning models. For deep learning architectures, tuning parameters such as network depth, layer sizes, activation functions, and dropout rates can significantly impact model performance. Cross-validation techniques, such as k-fold cross-validation, are often used to assess parameter sensitivity and select optimal parameter configurations. By carefully preprocessing data and tuning algorithm parameters, researchers can enhance the effectiveness and efficiency of anomaly detection systems in video surveillance.

5. IMPLEMENTATION

5.1 Programming Environment

Python with libraries such as OpenCV, TensorFlow, PyTorch, and scikit-learn is commonly used for developing anomaly detection algorithms in video surveillance. These libraries offer comprehensive support for image and video processing, deep learning, machine learning, and statistical analysis, providing a versatile programming environment for researchers and practitioners in the field.

5.2 Model Implementation

Model implementation for anomaly detection in video surveillance typically involves using deep learning architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs). Researchers preprocess video data, feed it into the network, and train the model on annotated datasets. Transfer learning techniques may be employed to leverage pre-trained models for feature extraction. Hyperparameters are fine-tuned through techniques like grid search or random search. After training, the model's performance is evaluated using appropriate evaluation metrics. Finally, the trained model is deployed for real-time or post-event anomaly detection in surveillance systems.

5.3 Training Procedure

The training procedure for anomaly detection in video surveillance begins by partitioning the dataset into training, validation, and test sets. Preprocessed video data is fed into the chosen deep learning architecture, such as a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), and trained using annotated examples of normal and anomalous activities. During training, model parameters are optimized using gradient-based optimization algorithms like stochastic gradient descent (SGD) or Adam. Hyperparameters, including learning rates and regularization parameters, are fine-tuned using techniques like grid search or random search. Training continues until convergence or a predefined number of epochs, with validation data used for early stopping.

5.4 Hardware Infrastructure

For implementing anomaly detection in video surveillance, a hardware infrastructure with suitable computational resources is essential. High-performance CPUs or GPUs are commonly used for training deep learning models due to their parallel processing capabilities. A multi-core CPU or a GPU with CUDA support can accelerate model training and inference tasks significantly. Sufficient RAM is necessary to handle large datasets and model parameters. Additionally, storage capacity is required to store datasets, pre-trained models, and intermediate results. Cloud-based platforms like AWS, Google Cloud, or Azure offer scalable computing resources for researchers and practitioners lacking dedicated hardware infrastructure.

6. EXPERIMENTAL RESULTS

6.1 Model Performance

Model performance in anomaly detection for video surveillance is assessed through various metrics including detection accuracy, precision, recall, and F1-score. Additionally, metrics like AUC-ROC and AUC-PR offer comprehensive evaluations across different operating points. Computational efficiency metrics such as processing time and memory usage are also vital for scalability and real-world deployment. The goal is to achieve high detection accuracy while minimizing false alarms and computational overhead, ensuring effective surveillance and timely intervention in detecting and mitigating potential threats or abnormal events.

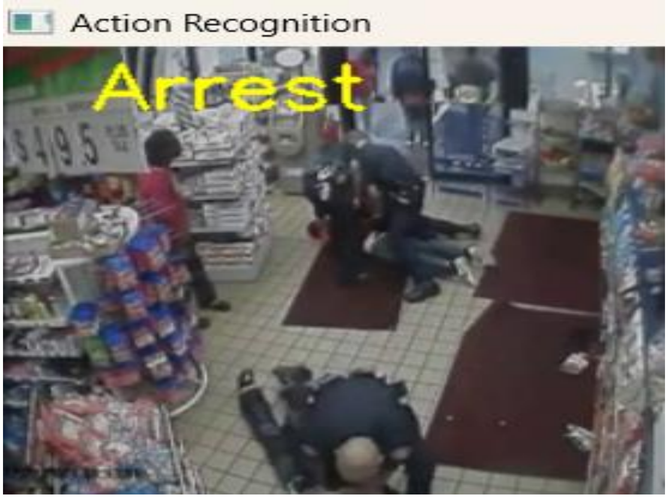

Fig 1: Recognizing the act of arrest

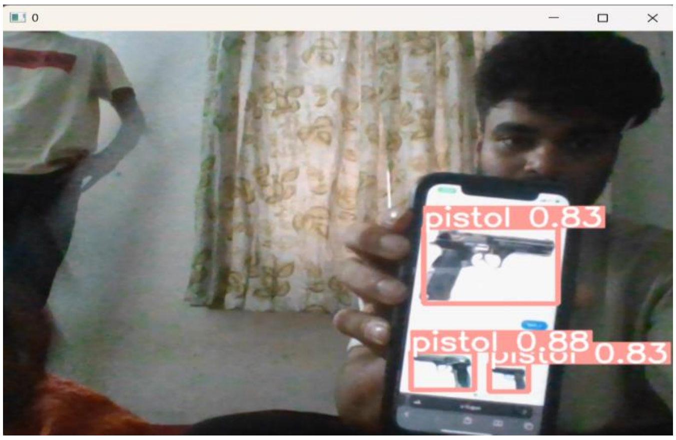

Fig 2: Recognition of the weapon gun

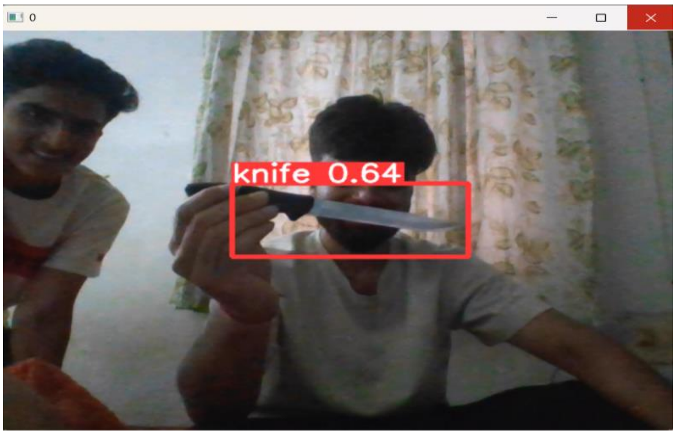

Fig 3: Recognition of the weapon knife in real life

6.2 Discussion of Result

The discussion of results in anomaly detection for video surveillance entails interpreting the performance metrics obtained from model evaluation. Analyzing detection accuracy, precision, recall, and F1-score provides insights into the model's ability to correctly identify anomalies while minimizing false alarms. AUC-ROC and AUC-PR curves offer a comprehensive view of the model's performance across different thresholds. Moreover, considerations of computational efficiency metrics such as processing time and memory usage are crucial for practical deployment. By comparing results across different models and datasets, researchers can identify strengths and weaknesses, highlight areas for improvement, and guide future research directions. Additionally, discussing the implications of the findings on real-world surveillance scenarios and potential applications helps contextualize the significance of the results for security practitioners and system developers.

7. CONCLUSION

In conclusion, anomaly detection in video surveillance plays a crucial role in enhancing security measures by identifying abnormal behaviors or events in real-time. Traditional methods like motion detection are being augmented with advanced techniques such as deep learning to improve accuracy and efficiency. Despite challenges such as complex scenes and computational constraints, ongoing research efforts are focused on addressing these limitations. By leveraging diverse datasets, rigorous evaluation metrics, and scalable hardware infrastructure, significant progress has been made in developing robust anomaly detection systems. The findings underscore the importance of continuous innovation and collaboration between academia and industry to meet the evolving demands of surveillance technology. Moving forward, integrating these advancements into practical applications will empower security personnel with effective tools for detecting and mitigating potential threats, ultimately enhancing public safety and security in various environments.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. We would like to take this opportunity to thank them all.

First and foremost, we would like to thank Dr. K Channakeshavalu, Principal, EWIT, Bangalore, for his moral support towards completing our project work.

We would like to thank, Dr. Suresh M B, Professor and Head of Department of ISE, EWIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide Mrs. Prakruthi G R, Assistant Professor, Department of ISE, EWIT, Bangalore for her able guidance throughout the project work and guiding me to organize the report in a systematic manner.

REFERENCES

[1] Venkata Ramana, Lakshmi Prasanna "Human activity recognition using opencv", International journal of creative research thoughts(IJCRT),(2021).

[2] Lamiyah Khattar, Garima Aggarwal "Analysis of human activity recognition using deep learning." 2021 11th International Conference on Colud Computing, Data Science & Engineering.

[3] Long Cheng,Yani Guan "Recognition of human activities using machine learning methods with wearable sensors" IEEE Members Research and Development Departmentin 2017.

[4] Ran He, Zhenan Sun "Adversarial cross spectral face completion for NIR-VIS face recognition." IEEE paper received on January 2019.

[5] Jing Wang, Yu cheng "Walk and learn: facial attribute representation learning",2016 IEEE Conference on Computer Vision and Pattern

Recognition.

[6] Yongjing lin,Huosheng xie "Face gender recognition based on face recognition feature vectors", International Conference on Information Systems and Computer Aided Education (ICISCAE),(2020).

[7] Mohanad Babiker, Muhamed Zaharadeen "Automated Daily Human Activity Recognition for Video Surveillance Using Neural Network." International Conference on Smart Instrumentation, Measurement and Applications(ICSIMA) 28- 30 November 2017.

[8] Neha Sana Ghosh, Anupam Ghosh "Detection of Human Activity by Widget." 2020 8th International Conference on Reliability, Infocom Technologies and Optimization(ICRITO) June 4-5,2020.

[9] AN YANG, WANG KAN "Segmentation and Recognition of Basic and Transitional Activities for Continuous Physical Human Activity" IEEE paper on 2016.

[10] AbdullahAl Fahim and Ki H. Chon, "Smartphone Based Human Activity Recognition with Feature Selection and Dense Neural Network" International Conference on Reliability, Infocom Technologies and Optimization (ICRITO),(2020).