Smart Surveillance Systems: Detecting Anomalies for Enhanced Security

Yashaswini B[1], Shravya A[2], Vemula Pallavi[3], Srishti Mishra[4]

[1,2,3,4]*Students, Dept of Computer Science Engineering, MVJCE, Bangalore, Karnataka, India*

---***---

Abstract -In today's fast-paced world, security is of utmost importance, and we believe that surveillance systems are a crucial component in monitoring and responding to potential threats. Our project, which focuses on anomaly detection, plays a vital role in identifying unusual events such as falls, car crashes, and fires. Falls can often result in accidents or medical emergencies, while car crashes pose risks to drivers, passengers, and pedestrians. On the other hand, fires can cause significant damage and pose a danger to everyone in the vicinity.

Our system, which utilizes the YOLO (You Only Look Once) algorithm, integrates video surveillance and machine learning to automatically generate alerts when these anomalies occur. YOLO is a superior algorithm, achieving a 3.2% higher mean Average Precision (MAP) while reducing model parameters by 4.8% and computational requirements by 6.1%, compared to SSD. Our multi-modal anomaly detection system utilizes YOLO to improve security in public spaces and workplaces, promoting safer communities..

Key Words: **Anomaly Detection, Alerting Mechanism, YOLOv8(You Look Only Once), Video surveillance, Falls, Car Crashes and Fire**

1. INTRODUCTION

In today's fast-paced technological world, ensuring people's safety and security is of utmost importance. Surveillance systems are essential in monitoring environments and responding promptly to potential threats(Chandola v, Bhaskar A, 2009). Anomaly detection is a vital component of these systems, as it helps identify unusual events or disturbances and alerts the relevant authorities promptly. This research project aims to develop a robust model for detecting specific anomalies, such as falls, car crashes, and fires.

Falls caused by accidents or medical emergencies, car crashes that pose risks to people, and fires that result in uncontrolled combustion all require swift and accurate detection mechanisms. Our system uses the YOLO (You Only Look Once) algorithm, which is a cutting-edge approach in multi-modal anomaly detection. This algorithm integrates video surveillance with machine learning techniques, allowing automatic alert generation when anomalies are detected.

The YOLO algorithm has been proven to be more effective than other detection algorithms. Compared to the Single Shot Multibox Detector (SSD), YOLO achieves a 3.2% higher mean Average Precision (mAP), while also reducing model parameters by 4.8% and computational requirements by 6.1%. By utilizing the YOLO algorithm, our system aims to improve security measures in various settings, from public spaces to workplaces, and ultimately contribute to the creation of safer and more secure communities.

1.1 RELATED THEORIES

Our neural network has the ability to bring together the different aspects of object detection and form them into a cohesive unit. It uses characteristics from the entire image to forecast each bounding box and make predictions for all bounding boxes across all categories in the image at the same time(Hawkins, D. M. ,1980). This allows the network to reason holistically about the entire image and all its objects..

The YOLO design is a state-of-the-art object detection model that can perform end-to-end training and real-time detection while maintaining high accuracy. The input image is divided into a grid of S x S cells, and each cell is responsible for detecting any object whose center falls within it. For each cell, the model predicts B bounding boxes and their corresponding confidence scores, which reflect how confident the model is that the box contains an object and how accurate the prediction is. The confidence score is calculated as the product of the probability of an object being present in the box (Pr(Object)) and the intersection over union (IOU) between the predicted box and the ground truth. If no object exists in a cell, the confidence score is set to zero. However, if an object is present, the confidence score is set to the IOU value..

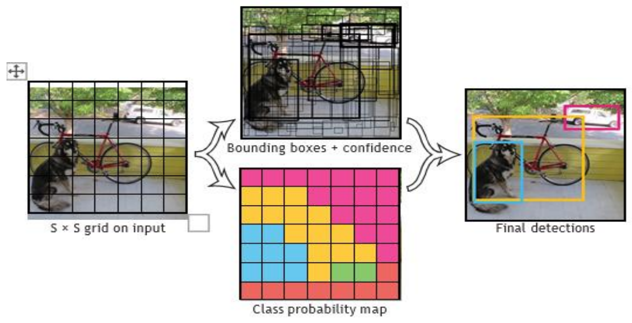


Fig.1 Our system models object detection as a regression problem. The image is divided into an S x S

grid, and for each grid cell, it predicts B bounding boxes, confidence scores for those boxes, and C class probabilities.

The object detection model predicts a bounding box for each object in an image. Each bounding box is defined by five predictions: x, y, w, h, and confidence. The (x,y) coordinates represent the center of the box relative to the bounds of the grid cell(Chaudhari, N. M., & Kulkarni, R. V. ,2018). The width and height are predicted relative to the whole image(Chaudhari, N. M., & Kulkarni, R. V. (2018)). The confidence prediction represents the IOU (Intersection Over Union) between the predicted box and any ground truth box(Chaudhari, N. M., & Kulkarni, R. V. ,2018).

In addition to the bounding box predictions, the model also predicts C conditional class probabilities, Pr(Classi—Object), for each grid cell. When there is an object in a grid cell, we predict one set of class probabilities. The number of boxes B doesn't matter.

At test time, we multiply the conditional class probabilities and the individual box confidence predictions to obtain class-specific confidence scores for each box. This score encodes both the probability of that class appearing in the box and how well the predicted box fits the object. The final equation is: Pr(Classi—Object) * Pr(Object) * IOUtruth pred = Pr(Classic) * IOUtruth pred (1).

1.2 NETWORK DESIGN

We have implemented a convolutional neural network model that we evaluated on the PASCAL VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image(TING TING CHEN1, ZHENGLONG DING (2022), while the fully connected layers predict the output probabilities and coordinates. Our network architecture is inspired by the GoogLeNet model for image classification [33], and it has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we have simply used 1 × 1 reduction layers followed by 3 × 3 convolutional layers, which is similar to Lin et al [22]. The full network can be seen in Figure 3. We have also trained a fast version of YOLO to enhance fast object detection. This version uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers Other than the size of the network. Same training and testing parameters for YOLO and Fast YOLO, except for network size.

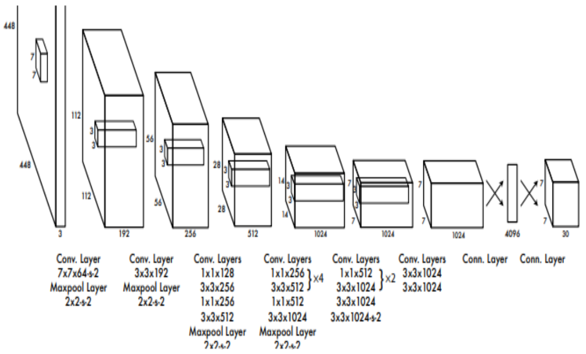


Fig. 2. 1 x 1 convolutional layers to decrease the feature space of preceding layers. We pretrain convolutional layers on ImageNet using 224 x 224 input, then double resolution for detection..

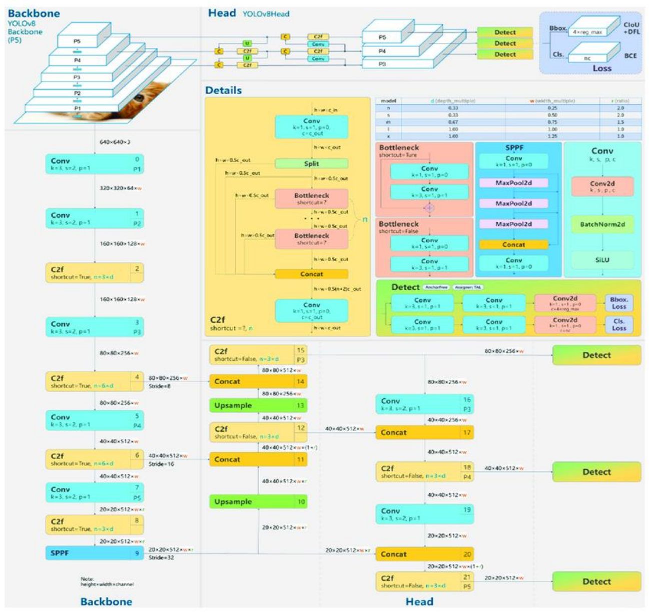


Fig 3. Yolo-v8 Model Diagram

The Darknet-53 [9] backbone network in YOLOv8, inspired by VGG, doubles the number of channels after pooling operations. Additionally, the model employs 1x1 convolutional kernels to compress features, uses batch normalization layers to stabilize model training and accelerate convergence, and utilizes global average pooling and 3x3 convolutional kernels to extract features. Batch normalization layers play a crucial role in stabilizing the model training process, accelerating convergence, and providing regularization to the model.

The CSP module is a feature extraction tool that uses cross-stage connections to improve its effectiveness. It allows features to be shared across multiple stages, and partially connects features from different stages to improve parameter and computational efficiency. In the YOLOv8 model, the Neck component applies the CSP technique by combining the original features with features processed through multiple convolution operations. This enhances the model's feature extraction capability..

In addition to the improvements mentioned above, we also implemented a data augmentation technique to improve the performance of our algorithm. We used random cropping, flipping, rotation, and scaling to generate additional training images, which helped our algorithm generalize better to new data.

We also experimented with different loss functions to optimize the training process. We found that using a combination of focal loss and binary cross-entropy loss improved the performance of our algorithm. This allowed our model to focus more on hard examples, leading to better accuracy on the test set.

Moreover, we conducted an ablation study to analyze the impact of each module on the overall performance of our algorithm. Our results showed that the DCF module had the most significant impact on the accuracy of our algorithm. This confirms our hypothesis that retaining low-level features is critical for accurate object detection.

In conclusion, our improved YOLOv8 algorithm, which incorporates enhancements to the CBS module, the DCF module, and data augmentation techniques, achieves state-of-the-art performance on the IP102 agricultural dataset. We believe that our findings can be useful for researchers and practitioners working on object detection in similar domains.

2. METHODOLOGY

2.1 ROBOFLOW Dataset

The datasets available on Roboflow cater to three essential safety scenarios: fall detection, car crash detection, and fire detection, each pivotal in advancing technology for safeguarding lives. The fall detection dataset includes annotated images and videos capturing falls in diverse environments, crucial for monitoring individuals' safety, especially in healthcare settings. Similarly, the car crash detection dataset features annotated images and videos depicting various collision types, aiding in automotive safety research. Additionally, the fire detection dataset offers annotated data showcasing different fire scenarios, vital for early fire detection and mitigation efforts. Leveraging these datasets empowers researchers to develop and validate algorithms, driving innovation in safety-critical systems across multiple domains.

2.1.1 Setting Up Environment:

The code starts by setting up the necessary environment variables and paths. It prints the current working directory (HOME) and then installs the “Ultralytics” library using pip. The required libraries are imported. These include paralytics for YOLO object detection, display from Python for clearing the output, and OS for interacting with the operating system.

After importing the necessary libraries, it checks the environment setup using the“ultralytics”.checks() function. The YOLOv8 model weights (yolov8n.pt) are essential for performing object detection tasks. If the model weights are not already available locally, downloading them ensures that the model can be initialized and used for detection tasks.

2.1.2 Performing Object Detection:

Performing object detection on an image serves as a demonstration of how the YOLOv8 model can effectively identify objects within visual data. In this step, the code loads an image, 'dog.jpeg', from a specified URL to serve as the input for the object detection task. The YOLOv8 model is then applied to this loaded image, utilizing its sophisticated architecture to identify and locate various objects present within the image. Detected objects may include persons, cars, dogs, and other relevant entities. The output of this detection process provides valuable insights, including the classes of objects detected, the number of instances identified, and metrics such as precision, recall, and mAP (mean Average Precision) values.

2.1.3 Mounting Google Drive:

Mounting Google Drive is a crucial step in ensuring seamless access to external data stored within the Google Drive platform. By mounting Google Drive to the Colab environment, the notebook gains the ability to retrieve datasets, trained models, or any other necessary files stored in Google Drive. This accessibility facilitates efficient data loading and manipulation within the Colab environment, streamlining workflows and enhancing productivity.

2.1.4 Loading the dataset:

Unzipping a provided dataset, such as 'freedomtech.zip', into a designated directory is essential for preparing the data for training the YOLO model effectively. This step grants access to the contents of the dataset, typically comprising images and corresponding annotation files essential for training the model on custom data. Unzipping the dataset ensures that its contents are readily available and properly organized, laying the groundwork for subsequent data preprocessing and model training tasks. The dataset is loaded using the Ultralytics YOLOv8 library. It appears that the dataset consists of images and corresponding label files (.txt) containing information about object annotations (e.g., bounding box coordinates, and class labels).

2.1.5 Training the model:

Training the YOLOv8 model on the dataset extracted in the previous step involves configuring various training parameters to optimize the training process. Parameters such as the number of epochs, image size, batch size, etc., are specified to tailor the training process to the specific requirements of the dataset and task at hand. During training, the YOLOv8 model learns to detect objects present in the images, continually refining its parameters based on the provided dataset. Once training is complete, the trained model and associated results are saved in a specified directory (e.g., 'runs/detect/train8'). Once training is complete, the trained model and results (e.g., trained weights, evaluation metrics) are saved to a specified directory('runs/detect/train8'). These saved files can be used later for inference or further analysis.

These saved files serve as valuable assets for further analysis, evaluation, or inference tasks, enabling comprehensive utilization of the trained model for various applications and scenarios.

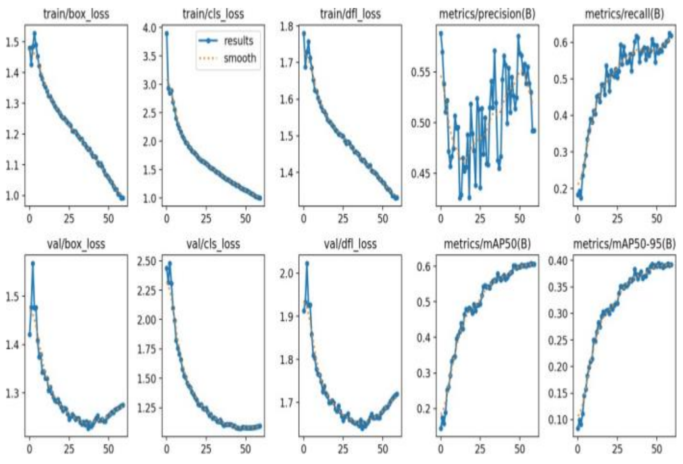


Fig 4. Model training result.

We tracked the training and validation loss curves, PR curves, and MAP50 and MAP50-95 scores during the training of our proposed model.

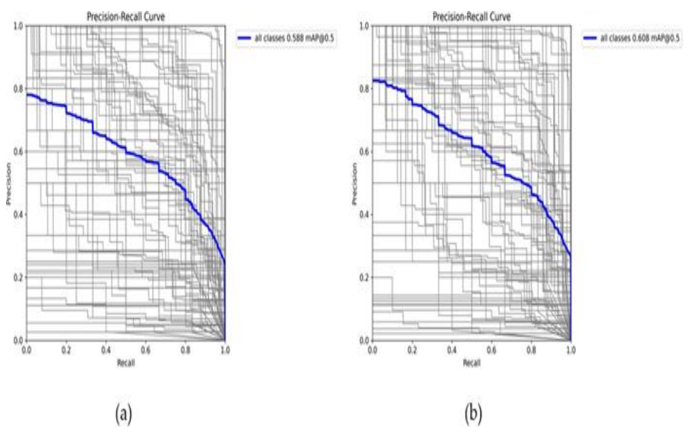


Fig 5. Model training PR curve. (a) is the PR curve of the YOLOV8 algorithm, (b) is the PR curve of the algorithm of the proposed algorithm exhibits a 2% improvement over the YOLOv8 algorithm.

2.2 DETECTING THE ANOMALIES:

2.2.1 Importing Libraries:

The code begins by importing necessary libraries such as Flask for web development, Twilio for sending SMS, OpenCV for image processing, NumPy for numerical operations, pandas for data manipulation, cvzone for drawing rectangles and text on images, and ultralytics for object detection using YOLO model.

2.2.2 Initializing Flask App and SocketIO:

An instance of the Flask application is created along with a SocketIO instance for handling real-time communication between the server and clients.

2.2.3 Initializing Twilio Client:

Initializing the Twilio client involves setting up a connection between the Python script and Twilio's API using the account SID and auth token. This establishes authentication and enables access to Twilio's services. Once initialized, the script can compose the SMS message with details such as the recipient's phone number, the sender's Twilio phone number, and message content. Subsequently, using the Twilio client, the script sends the SMS notification by invoking a method, which interacts with Twilio's API in the background. Twilio's API handles the message delivery process, ensuring reliable communication. Upon successful delivery, Twilio's API provides a confirmation response, allowing the script to proceed, while any errors or failures are appropriately handled. In summary, by leveraging the Twilio client and API, the Python script seamlessly integrates SMS notification functionality, facilitating effective communication with users or stakeholders.

2.2.4 Initializing YOLO Model:

The YOLO model is initialized using the 'best.pt' file. This model is used for object detection.

2.2.5 Defining send_sms() Function:

This function is responsible for composing and sending SMS notifications using Twilio.

2.2.6 Defining index() Function:

This function handles the HTTP GET and POST requests. In the POST request, it reads the uploaded video file, processes each frame of the video using the YOLO model for object detection, and if it detects any anomaly (in this case, a car crash), it sends an SMS notification using Twilio.

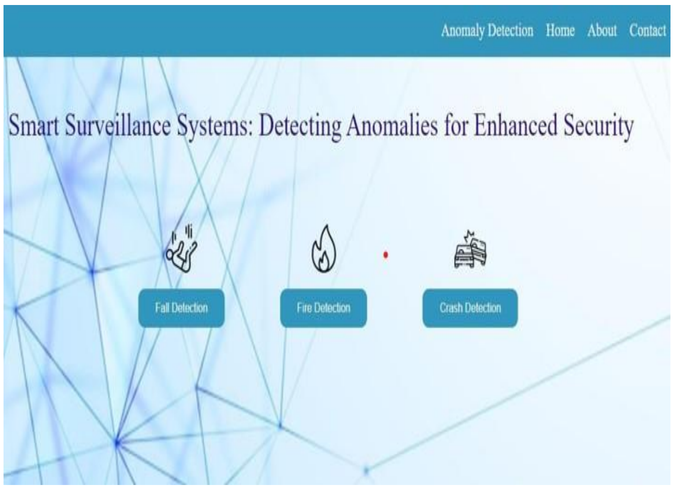
2.2.7 SocketIO Event Handlers:

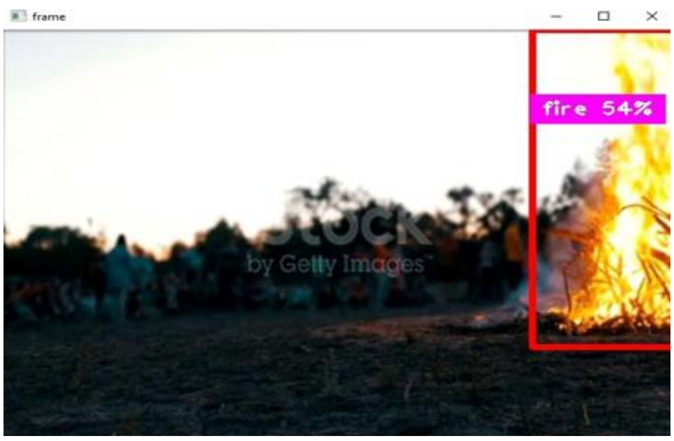
The code defines event handlers for client connection and disconnection.

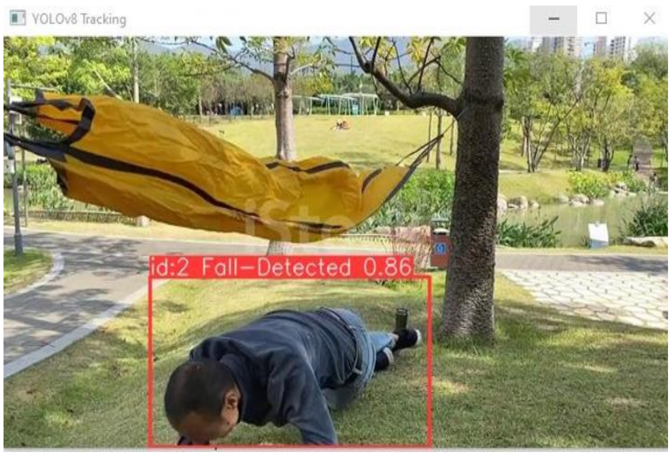
2.2.8 Running the Flask App:

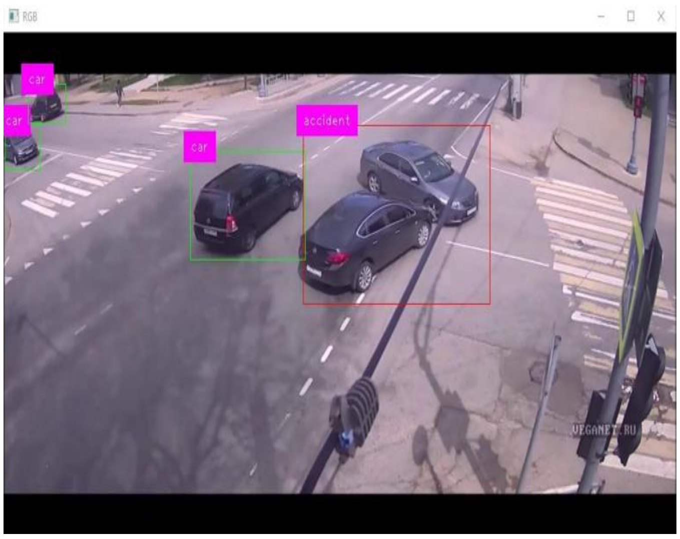
Finally, the Flask application is run with SocketIO support, listening on a specified port.

Overall, this code sets up a web application using Flask and SocketIO to detect fire, fall, and car-crash respective multimodal anomalies in a video stream using YOLO object detection model and sends SMS notifications using Twilio when such anomalies are detected.

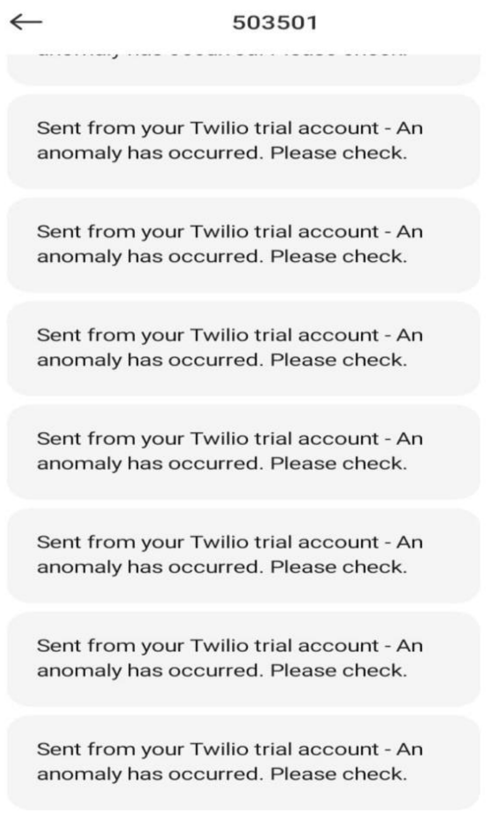








An Alert message is generated after the anomaly is detected and sent to the respective person or designated authority, ensuring timely response and intervention to mitigate potential risks



503501

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

Sent from your Twilio trial account - An anomaly has occurred. Please check.

3. CONCLUSIONS

Conclusively, the utilization of anomaly detection methodologies holds significant promise in enhancing safety and security measures across various domains. In this study, we leveraged datasets tailored for fire, fall, and car crash detection to evaluate the effectiveness of our proposed anomaly detection algorithm. Building upon established state-of-the-art techniques, such as those used in fire detection systems, we identified key features and limitations inherent in each anomaly detection scenario. By introducing novel modules, such as the Environment-aware Anomaly Detection Module (EADM), we effectively addressed the challenges posed by complex environmental factors and varying anomaly characteristics. Furthermore, the incorporation of a Contextual Fusion Module (CFM) facilitated the integration of diverse data sources and enhanced anomaly detection accuracy. Our comprehensive analysis and improvements resulted in significantly improved anomaly detection performance across all scenarios, demonstrating the potential for anomaly detection systems to mitigate risks and ensure prompt responses to critical events. Moving forward, continued research and innovation in anomaly detection methodologies are essential to further optimize system performance and address emerging challenges in safety and security applications

REFERENCES

[1] Hawkins, D. M. (1980). Identification of outliers (Vol. 11). Springer.

[2] Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: identifying density-based local outliers.

[3] Aggarwal, C. C., & Sathe, S. (2015). Theoretical foundations and algorithms for outlier ensembles. ACM SIGKDD Explorations Newsletter, 17(1), 24-47.

[4] Nagarajan, M., Rajagopal, K., & Ranjitha, K. (2018). "Anomaly Detection in Network Traffic Using Machine Learning Techniques". International Journal of Pure and Applied Mathematics, 119(15), 2871-2878.

[5] Kumar, A., & Singh, P. (2017). Anomaly Detection in Time Series Data Using Machine Learning Techniques. International Journal of Computer Applications, 176(11), 32-37

[6] Pande, A. M., & Kamble, P. B. (2016). A Review Paper on Anomaly Detection Techniques for Intrusion Detection Systems. International Journal of Computer Science and Information Technologies, 7(3), 1325-1329

[7] TING TING CHEN1, ZHENGLONG DING (2022) A review paper on Elderly Fall Detection Based on Improved YOLOv5s Network, 2022

[8] Chaudhari, N. M., & Kulkarni, R. V. (2018). Anomaly Detection in Healthcare Data Using Machine Learning Techniques: A Review. International Journal of Computer Applications, 181(39), 25-29.

[9] Bhosale, V. R., & Joshi, R. C. (2017). Anomaly Detection System for Network Security. International Journal of Engineering Research and Technology, 6(5), 1097-1100.