.

CHAPTER 3

ANOMALOUS BEHAVIOR DETECTION AND ANALYSIS IN VIDEO SURVEILLANCE USING CNN AND MODIFIED YOLOv4 MODEL

3.1 INTRODUCTION

Detecting anomalous behavior in public spaces is essential for security and emerging trends in Deep Learning (DL) and Computer Vision (CV) have improved automated surveillance systems for Video Anomaly Detection (VAD). This work presents a modified You Only Look Once (YOLO) model for Anomaly Detection (AD), incorporating optimization techniques to improve accuracy and evaluates its performance in detecting and analyzing unusual activities.

3.2 DATASET DESCRIPTION

UCSD Dataset: The University of California, San Diego (UCSD) Anomaly Detection Dataset comprises of grayscale video sequences recorded in 2008 using stationary, elevated camera overlooking pedestrian walkways. It is available for public use at http://www.svcl.ucsd.edu/projects/anomaly/dataset.html.

The dataset captures varying crowd densities, varying from low to densely populated scenarios. Under normal conditions, the video contains only pedestrians, while anomalies arise from the existence of non-pedestrian entities or unusual motion patterns. These anomalies occur naturally and were not staged for dataset creation. Each video stream is segmented into clips of approximately 200 frames for analysis.

Anomalous events recorded in the dataset fall into two main categories:

- **Non-pedestrian entities:** Includes cyclists, skateboarders, small carts and individuals crossing walkways or moving through the surrounding grassy areas.

- **Unusual pedestrian behaviors:** Encompasses irregular movement patterns, such as people in wheelchairs or pedestrians exhibiting non-standard motion.

The dataset comprises of two subsets, each characterized by varying camera perspectives and pedestrian movement patterns.

UCSD Pedestrian1 (Peds1): The Peds1 subset captures groups of pedestrians walking towards and away from the camera, introducing perspective distortion. It contains 34 training samples and 36 testing samples, with 1,307 training frames and 231 testing frames.

UCSD Pedestrian2 (Peds2): The Peds2 subset features pedestrian motion parallel to the camera plane, eliminating perspective distortion. The dataset comprises 16 training samples and 12 testing samples which include 2,550 training frames and 2,010 testing frames.

For both the subsets, ground truth annotations include binary flags per frame to indicate the presence of anomalies. Additionally, 10 clips in Peds1 and 12 clips in Peds2 contain pixel-level binary masks that highlight regions with anomalies, facilitating anomaly localization evaluation. Figure 3.1 and Figure 3.2 illustrates samples frames of normal and anomalous events in the Peds1 and Peds2 datasets.

(a) **(b)**

Figure 3.1 Sample Frame from UCSD Peds1 Dataset: (a) Normal Frame, (b) Anomalous Frame

(a) **(b)**

Figure 3.2 Sample Frame from UCSD Peds2 Dataset: (a) Normal Frame, (b) Anomalous Frame

Figures 3.1(a) and Figure 3.2(a) depicts a normal scene with people walking along the walkway. Figures 3.1(b) and Figure 3.2(b) depicts anomalous events featuring a boy riding a bicycle.

Although the modern surveillance systems capture color videos, the grayscale UCSD dataset remains a benchmark in VAD research due to its historical significance and well-documented anomalies. Its simplified format allows for controlled algorithm testing, while VAD models often convert color images to grayscale during preprocessing to reduce computational complexity and enhance structural and motion-based anomaly detection, reinforcing the dataset's relevance in modern research.

The implementation was executed on a system featuring an NVIDIA GeForce RTX 3050 Graphics Processing Unit (GPU), an 11th Generation Intel Core i7 processor running at speed of 3.30 GHz, 16 GB RAM and a 512 GB SSD. The system runs a 64-bit version of Windows 11.

3.3 CONVOLUTIONAL NEURAL NETWORK(CNN) ARCHITECTURE

CNN a subset of DL model inspired by biological visual processing heavily leveraged in the realm of image recognition, video analysis and object detection due to its capability to uncover patterns like edges, textures and shapes. The structure of the CNN is specified in Figure 3.3.

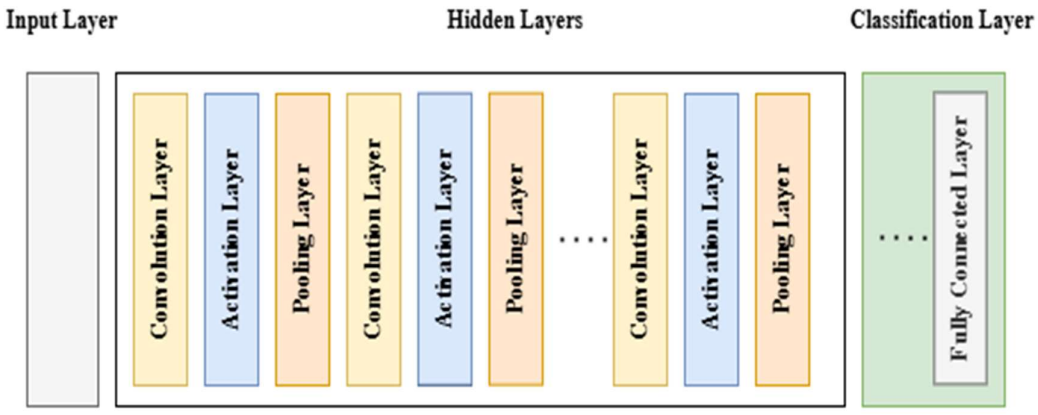

Figure 3.3 Structure of CNN

The CNN model encompasses three main segments: the input layer, hidden layers and the classification layer. The input layer acquires raw data, like images and processes it for further analysis. The hidden layers encompass of convolutional layers, activation

functions and pooling layers that work together to extract, fine-tune and process multi-level features from the source data. Convolutional layers focus on capturing features, activation layers incorporate non-linear transformations and pooling layers diminish the spatial dimensions, promoting efficient representation and learning.

Convolutional layers apply filters of size $m \times m$ across input $n \times n$ producing output $(n-m+1) \times (n-m+1)$ and capturing patterns like edges and textures. These filters, also called as kernels, are small matrices used for feature extraction, sliding over the input with a specific stride to determine the step size, directly influencing the feature map's resolution. Parameters such as stride and padding adjust overlap, preserve spatial information and enhance feature extraction. The kernel and stride utilized for capturing edges and texture patterns of the convolution layer are represented in Figure 3.4.

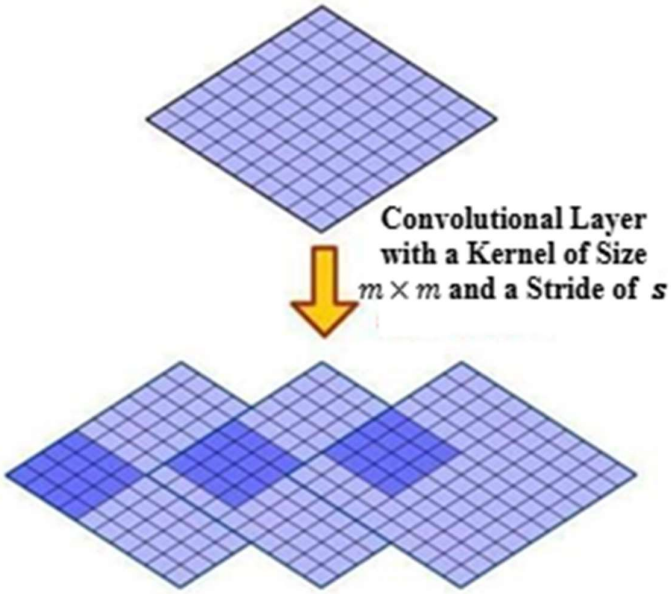

Figure 3.4 Illustration of kernel and stride of convolutional layer (Ahlawat et al., 2020)

To enable non-linearity and to capture complex features in the deep learning task the activation function namely Rectified Linear Unit (ReLU) can be used. ReLU provides output for the input value in case of positive and 0 in case of negative, defined by Equation (3.1):

$$ReLU(x) = max(0, x) \tag{3.1}$$

To improve the robustness of the model, the down sampling of spatial features can be performed using pooling layers like max-pooling or average-pooling (Ahlawat et al., 2020).

The pooling layer ensures efficient feature learning and helps to prevent overfitting. The example of reducing the dimensions using max-pooling is provide in the Figure 3.5. (Scherer et al., 2010).

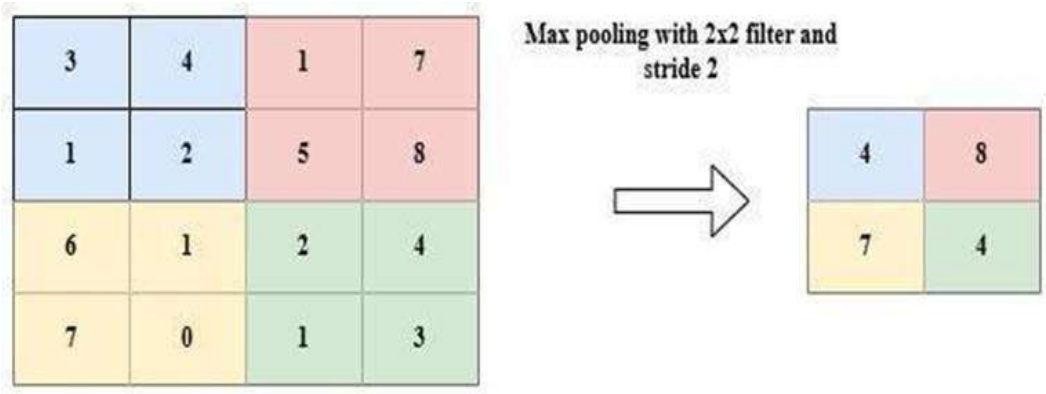

Figure 3.5 Stride size and max pooling with filter 2x2

The final classification layer of CNN is a fully connected feed forward network which is used for categorization of input. By combining the extracted features from the previous the CNN model accurately learns features to classifies correctly.

A Softmax activation function helps to map learned features to corresponding classes of classification. The raw model outputs or logits determines the probability of a specific class. CNNs can be used in applications like image recognition, video analysis and AD.

3.4 YOLOv4 ARCHITECTURE

YOLOv4 (Bochkovskiy et al., 2020) is a cutting-edge object detection model designed to balance high accuracy and computational efficiency. It enables training and inference on conventional Graphical Processing Units (GPUs), making advanced object detection assessable to a broader audience. Figure 3.6 provides the components of YOLOv4.

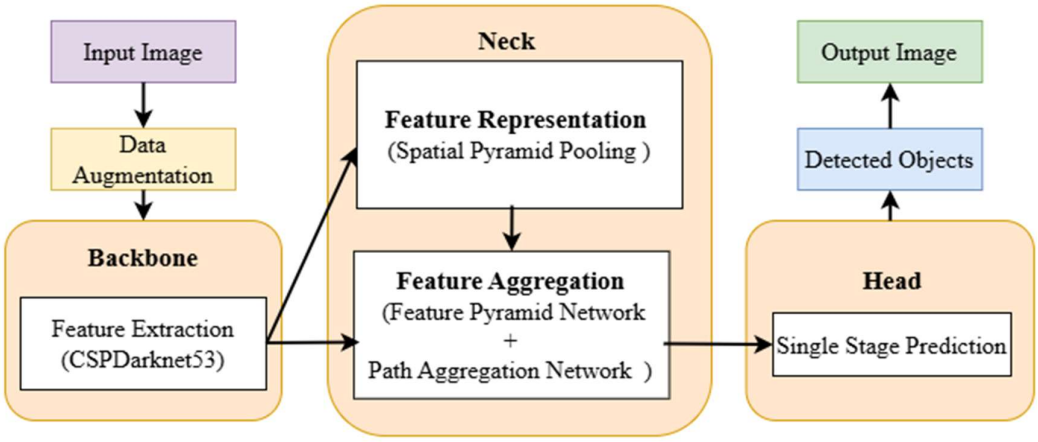

Figure 3.6 YOLOv4 Components

The YOLOv4 architecture comprises of three primary modules: the Backbone, Neck and Head, each serving a vital function in object detection. The Backbone, is used for feature extraction, is built upon CSPDarkNet53 (Cross- Stage Partial DarkNet53), an optimized CNN architecture derived from DarkNet for real-time object detection. It enhances gradient flow and efficiency by splitting feature maps, with one path undergoing convolutional operations for feature extraction while the other bypasses convolution and merges with the output using CSP connection. This structure integrates convolutional layers, batch normalization and shortcut connections, improving computational efficiency, gradient propagation and feature learning, making it a robust backbone for YOLO-based detection.

The Neck acts as an intermediate layer, enhancing feature representation and aggregation before final detection. It includes Spatial Pyramid Pooling (SPP) for capturing multiscale features, Feature Pyramid Network (FPNet) for extracting hierarchical representations and Path Aggregation Network (PAN) for refining feature fusion and improving localization accuracy.

The Head consists of a single-stage object detector that utilizes three YOLOv3-based detection heads to boost multiscale detection, enabling accurate identification of objects of different sizes. The Head serves as the final prediction layer, responsible for localizing objects, assigning bounding boxes and computing confidence scores. This design allows YOLOv4 to efficiently detect multiple objects within an image while ensuring real-time performance.

The YOLOv4 detection process begins with input processing, where the frame is segmented into a grid layout, in every individual cell tasked with identifying objects within its corresponding area. Next, the CSPDarkNet53 backbone extracts essential features, identifying object patterns and spatial characteristics. The Neck (SPP, FPNet, PAN) further sharpens these features by amplifying spatial details, pyramidal and hierarchical representations. The Head performs object detection and classification, assigning bounding boxes, confidence score and object probability values. Finally, post-processing models, like Non-Maximum Suppression (NMS), eliminate duplicate detections and refine concluding predictions (Ghahremannezhad et al., 2022). This structure architecture enhances accuracy, speed and computational efficiency, making YOLOv4 a robust and scalable object detection framework suitable for real-time applications like surveillance, autonomous systems and classification.

3.5 HYBRID CNN – YOLO MODEL

The CNN and YOLO models are extensively used in CV tasks because of their effectiveness in processing image and video data. CNNs excel at feature extraction and classification through hierarchical layers, while YOLO specializes in real-time object detection, predicting object locations and classes directly from input data. Together, these models form a robust framework such as AD, video analysis and object tracking, as pictured in Figure 3.7.

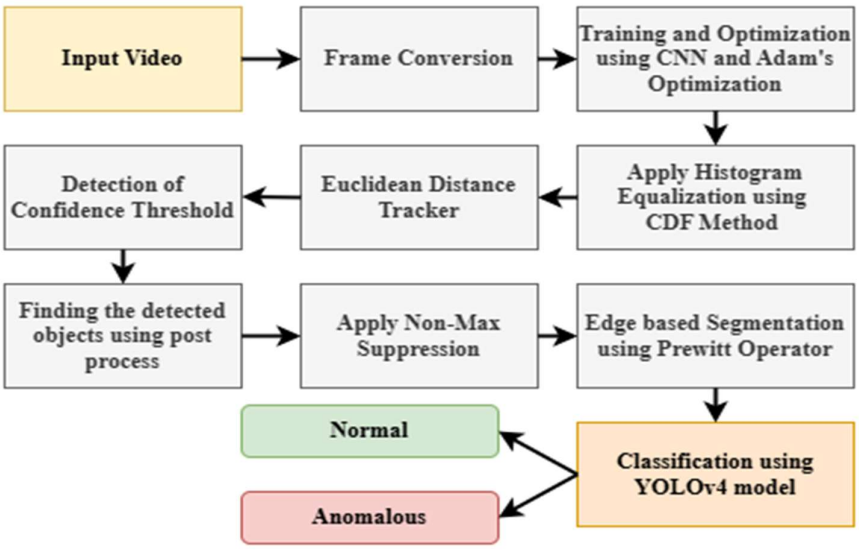

Figure 3.7 Overview of CNN-YOLO Model

The model is trained utilizing the UCSD video dataset, with CNN employed for feature extraction and the Adam optimization algorithm ensuring efficient learning. After training, the model is assessed using the Improved YOLOv4 framework. The testing phase involves preprocessing, feature extraction, post-processing steps and anomaly detection, enabling the precise identification of irregular events in the dataset.

- **Model Training and Construction using CNN**

This phase involves constructing and training the model for accurate anomaly detection. CNN is used for feature extraction, while the Adam optimization algorithm provides stable and efficient learning. Training includes converting video data into frames, preprocessing input and fine-tuning weights using backpropagation for improved performance. The video from the UCSD dataset is segmented into individual frames, which are then utilized for in depth analysis and CNN training.

The CNN architecture includes:

- **Data Layer:** Handles raw input without learnable parameters.
- **Convolutional Layers:** Extract features like edges and patterns that are essential for effective detection.
- **Pooling Layers:** Lower the dimensionality of feature maps, improving computational efficiency.
- **Fully Connected Layers:** Merge the identified features to produce the ultimate predictions.

Weights are adjusted using backpropagation to enhance accuracy and performance during training.

- **Adam Optimization Algorithm**

The Adaptive Moment Estimation or Adam optimization algorithm, (Kingma & Ba, 2014), is a stochastic optimization method designed to efficiently train deep learning models. It enhances standard Stochastic Gradient Descent (SGD) by dynamically tuning the learning rate for every individual parameter depending on the first and second moment estimates of the gradients. Adam fuses the strengths of the Adaptive Gradient (AdaGrad) algorithm, which is particularly effective for sparse gradients, with those of Root Mean Square Propagation (RMSProp), which performs well with non-stationary objectives. This

algorithm is computationally efficient, memory-friendly and has step-size invariance to gradient scaling, making it suitable for large-scale deep learning models.

Adam maintains two moving averages: the first-moment estimate m_t, which represents the mean of gradients and the second-moment estimate n_t, which approximates the uncentered change of gradients. At every time interval t, the algorithm updates the first-moment estimate using Equation (3.2).

$$m_t = u_1 * m_{t-1} + (1 - u_1) * g_t \tag{3.2}$$

where, m_t is the biased first-moment estimate at time interval t, m_{t-1} is the estimate from the preceding step, g_t is the gradient of the objective function at t *and* u_1 (typically set to 0.9) is a decay factor controlling the moving average of past gradients. Simultaneously, the second-moment estimate is updated as Equation (3.3).

$$n_t = u_2 * n_{t-1} + (1 - u_2) * g_t^2 \tag{3.3}$$

where, n_t represents the biased second-moment estimate, n_{t-1} is the previous estimate, g_t^2 is the element-wise square of the gradient and u_2 (typically set to 0.999) controls the moving average of squared gradients.

Since these moment estimates are biased towards zero in the early iterations, bias correction is applied as Equations (3.4) and (3.5).

$$m'_t = \frac{m_t}{1-u_1^t} \tag{3.4}$$

$$n'_t = \frac{n_t}{1-u_2^t} \tag{3.5}$$

where m'_t and n'_t are the bias-corrected values of the first and second moments, in order. The final parameter update is computed using the Equation (3.6):

$$\theta_t = \theta_{t-1} - \frac{\alpha m'_t}{\sqrt{n'_t + \epsilon}} \tag{3.6}$$

where θ_t is the updated parameter, θ_{t-1} is the previous parameter value, α is the learning rate (default 0.001) and ϵ (default 10^{-8}) is a minor constant to avoid division by zero.

Adam is a powerful and versatile optimization method that balances computational efficiency, adaptability and performance. It converges quickly, making it effective for deep learning applications and its adaptive learning rates enable strong performance across

various problems, including those with sparse gradients and non-stationary objectives. Additionally, Adam is robust in practice, requiring minimal hyperparameter tuning, making it well-suited for a diverse set of Machine Learning (ML) and DL projects.

- **Model Evaluation Using Improved YOLOv4**

Once training is accomplished, the model's performance is assessed using Improved YOLOv4, where the remaining data is evaluated through preprocessing, feature extraction and post-processing before anomaly detection.

- **Histogram Equalization using Cumulative Distribution Function**

Histogram equalization is an image processing method used to increase contrast by reordering pixel intensity values. This method improves visibility in low-contrast images, which is applicable in areas like computer vision and neural network-driven image compression.

The Cumulative Distribution Function (CDF) plays a vital role in histogram equalization by mapping pixel values to a normalized scale, ensuring a more uniform distribution of intensity levels. Given an image with intensity stages spanning from *0* to $L-1$ (where $L=256$ for an 8-bit image), the histogram denotes the frequency of each intensity level. The CDF of the histogram is computed using Equation (3.7).

$$c(k) = \text{round}\left((L-1)\sum_{i=0}^{k} p(i)\right), \quad k \in L \tag{3.7}$$

where:

— $c(k)$ is the CDF at intensity level k.

— L represents the summation of intensity stages (256 for an 8-bit image).

— $p(i)$ is the Probability Mass Function (PMF), which gives the probability of occurrence of intensity i, calculated using Equation (3.8):

$$p(k) = \frac{h(k)}{mn} \tag{3.8}$$

where $h(k)$ is the histogram count for intensity k and mn is the summation of pixels comprising the image.

— The summation of *p(i)* accumulates the probability mass up to intensity level *k*.

— The round function ensures discrete intensity values are preserved in the equalized histogram.

This transformation adjusts pixel values, making dark regions brighter and bright regions darker, thereby enhancing overall contrast. Unlike traditional naive histogram equalization, modern approaches address sparsity and intensity quantization errors, leading to better visual clarity and improved performance in DL models for image analysis tasks (Dyke & Hormann, 2023).

- **Euclidean Distance Tracker**

A grayscale image can be expressed in vector form of pixel intensity values in the Equation (3.9).

$$Fx = (x1, x2, \ldots, x_{MN}) \tag{3.9}$$

where *MN* is the overall sum of pixels in the image.

The Euclidean distance between two image vectors, x_1 and x_2, is given by Equation (3.10)

$$d_E^2(x_1, x_2) = \sum_{k=1}^{MN}\left(x_1^k - x_2^k\right)^2 = (x_1 - x_2)^T(x_1 - x_2) \tag{3.10}$$

where x_1^k and x_2^k denotes the k^{th} pixel values in the respective image vectors. Transpose *(T)* converts this column vector into a row vector.

Standard Euclidean distance treats pixels as independent and orthogonal, ignoring spatial relationships, making it useful for image similarity measurement and change detection but less effective for tasks that rely on structural or perceptual accuracy due to its inability to consider neighboring pixel dependencies.

- **Threshold Functions**

Threshold functions are essential in image processing for segmenting images based on pixel intensity, improving contrast and extracting relevant features. These functions categorize pixel values into distinct groups, aiding in tasks such as object detection, edge enhancement and noise reduction. Various thresholding techniques exist, including global, adaptive and statistical methods, each suited for different applications.

Global thresholding applies a fixed threshold, converting pixels above a defined intensity value to white (255) and those below to black (0). Adaptive mean thresholding calculates the threshold dynamically based on the local mean intensity within a defined neighborhood, improving segmentation in images with varying lighting conditions. Adaptive Gaussian thresholding, in contrast, assigns weights using a Gaussian function to compute a locally adaptive threshold, effectively enhancing fine details. These methods enable efficient feature extraction and preprocessing in the video analysis pipeline, facilitating object detection and tracking in real-world scenarios.

- **Identifying the Detected Objects using Post-Processing Techniques**

Various post-processing techniques are applied to enhance the quality of foreground masks, which are binary or grayscale images separating moving objects from the background. These methods refine object detection by reducing noise, identifying connected regions and segmenting meaningful structures in the scene.

— **Noise Removal**

The foreground mask contains unwanted noise blobs due to camera artifacts or background subtraction limitations. To eliminate these disturbances, the implementation applies morphological operations, specifically opening and closing transformations, using a structured kernel. The opening operation removes small noise particles, while closing helps gaps in detected objects, ensuring accurate segmentation.

— **Blob Processing**

To identify distinct objects in the scene, the implementation employs connected component labeling or Blob processing, which detects continuous regions in the processed image. This method assigns unique labels to each detected object, allowing further analysis such as bounding box extraction and object tracking. The quantity of connected components represents the total number of detected objects.

- **Non-Max Suppression (NMS)**

NMS is post-processing method deployed to refine edges by eliminating weaker gradient magnitudes, ensuring only the most prominent edges are retained. The method compares each pixel's gradient magnitude with its neighboring values along the gradient direction and suppresses non-maximum values to improve edge clarity. Figure 3.8 provides classification of Gradient Angle into Eight Sectors.

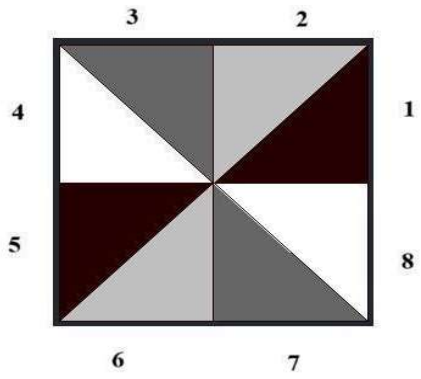

Figure 3.8 Gradient Angle Classifications into Eight Sectors

In the model, the gradient angles are classified into eight sectors and linear interpolation is applied to compute neighboring gradient magnitudes (Ding & Goshtasby, 2001). The interpolating parameter $t=dy/dx$ helps to accurately suppress weak edges while preserving significant boundaries. This approach enhances edge detection precision, making it essential for contour refinement and object boundary detection.

- **Edge-Based Segmentation using the Prewitt Algorithm**

The Prewitt operator for edge detection utilizes horizontal and vertical gradient filters to identify structural boundaries and enhance object segmentation. This method effectively highlights edges but is limited to detecting only horizontal and vertical orientations.

-1	-1	-1
0	0	0
1	1	1

90 Degrees

-1	0	1
-1	0	1
-1	0	1

180 Degrees

1	1	1
0	0	0
-1	-1	-1

270 Degrees

1	0	-1
1	0	-1
1	0	-1

0 Degrees

-1	-1	0
-1	0	1
0	1	1

135 Degrees

0	1	1
-1	0	1
-1	-1	0

225 Degrees

1	1	0
1	0	-1
0	-1	-1

315 Degrees

0	-1	-1
1	0	-1
1	1	0

45 Degrees

Figure 3.9 Eight templates of different directions

The standard Prewitt operator employs a pair of 3×3 matrices to compute gradients in these two primary directions. To improve directional sensitivity, an extended Prewitt operator introduces eight-directional templates, capturing additional edge variations as shown in Figure 3.9.

However, without a predefined threshold, the algorithm remains susceptible to noise, as it computes gradients across all directions and assigns them directly as pixel values in the edge image. Despite its limitations, the Prewitt operator remains an effective method for basic edge-based segmentation, particularly when combined with noise reduction techniques to enhance accuracy in feature extraction.

- **Anomalous Behavior Detection Using Modified YOLOv4 Framework**

The Modified YOLOv4 architecture utilizes the existing YOLOv4 Backbone and Neck, including the PAN and SPP to enhance feature extraction and integration efficiently. These components improve the capability of the system to retrieve detailed and high-level contextual features, which is essential for accurate AD. Figure 3.10 imprints the modified YOLOv4 Architecture.

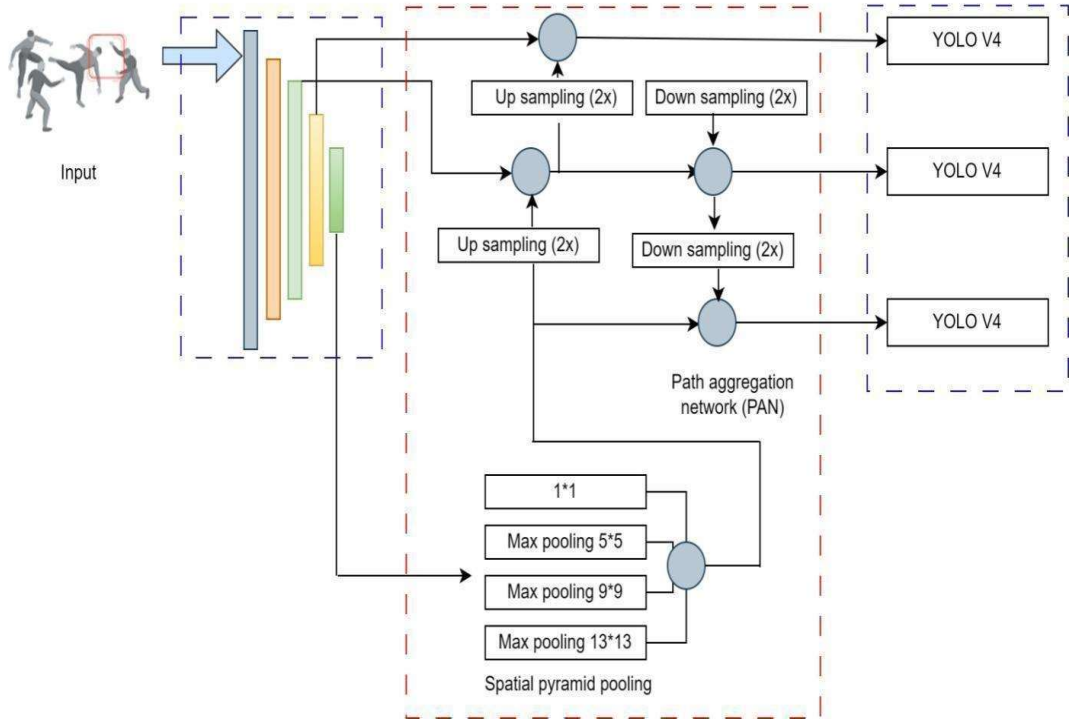

Figure 3.10 Modified YOLOv4 Architecture

The standard Prewitt operator employs a pair of *3×3* matrices to compute gradients in these two primary directions. To improve directional sensitivity, an extended Prewitt operator introduces eight-directional templates, capturing additional edge variations as shown in Figure 3.9.

In this modified architecture, the conventional detection heads are replaced with YOLOv4-specific detection heads, designed to optimize feature aggregation and improve real-time detection capabilities. This modification enhances localization accuracy and ensures robust detection across multiple scales, making it particularly effective in identifying anomalous behaviors in surveillance environments.

The Pseudo code of the CNN - YOLO model is outlined in Algorithm 3.1.

Algorithm 3.1: Pseudocode for CNN - YOLO model

Model Training and Construction

1. **Frame Conversion:** Convert the video dataset into individual frames for processing.
2. **CNN Model Training:**
   - Extract spatial features using 3×3 filters, stride=1, ReLU activation in convolution layers.
   - Reduce feature map size using 2×2 max-pooling with stride=2 to retain essential features.
   - Normalize activations using batch normalization after each convolutional layer.
   - Flatten the feature maps and fed to fully connected layers with ReLU activation and dropout for regularization.
   - Optimize weight updates using Adam optimizer with categorical cross-entropy loss.
3. **Optimization Using Adam Algorithm:**
   - Compute first and second gradient moments for adaptive learning rate adjustment (initial learning rate = 0.001).

Model Evaluation Using Improved YOLOv4

4. **Histogram Equalization:**
    - Normalize image intensities using contrast-based equalization to enhance object visibility.
5. **Euclidean Distance Tracking:**
    - Track object positions using vector-based distance calculations between frames.
6. **Thresholding for Object Detection:**
    - Apply global thresholding to segment objects.
    - Use adaptive mean thresholding with a kernel size suited for handling varying illumination conditions.
    - Apply adaptive Gaussian thresholding, weighted by the local intensity distribution, to achieve fine-grained segmentation.
7. **Post-Processing for Object Detection:**
    - Remove noise using morphological opening and closing operations.
    - Identify and segment objects using connected component labeling**.**
8. **Non-Maximum Suppression (NMS):**
    - Retain the most confident bounding boxes by applying IoU filtering.
9. **Edge-Based Segmentation:**
    - Detect edges using the Prewitt operator (3×3 kernel, stride=1) in horizontal and vertical directions.
    - Extend to eight-directional templates (diagonal, vertical, horizontal) for improved edge detection.
10. **Anomalous Behavior Detection Using Modified YOLOv4:**
    - Extract hierarchical features using YOLOv4 Backbone with CSPDarkNet.

— Improve multiscale feature fusion using PAN and SPP.

— Replace traditional YOLO detection heads with YOLOv4-specific detection heads for improved Accuracy and Recall.

3.6 PERFORMANCE METRICS

The performance of the model is assessed using the metrics such as Accuracy, Precision, Recall, F1 Score, Area Under Curve (AUC), Equal Error Rate (EER) and Peak Signal-to-Noise Ratio (PSNR). These metrics are inferred from the confusion matrix presented in Table 3.1, which compares predicted outcomes with actual classifications, offering intuitions into the model's efficiency and possible refinements.

Table 3.1 Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

where,

- **True Positive (TP):** The count of actual positive cases that the model correctly identifies as positive.
- **False Positive (FP):** The count of actual negative cases that the model incorrectly identifies as positive.
- **True Negative (TN):** The count of actual negative cases that the model correctly identifies as negative.
- **False Negative (FN):** The count of actual positive cases that the model incorrectly classifies as negative.

The performance metrics are mathematically defined in Equations 3.11 to 3.16.

Accuracy measures the proportion of correctly classified instances among all cases.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{3.11}$$

Precision indicates the proportion of correctly predicted positive cases among all predicted positives.

$$Precision = \frac{TP}{TP+FP} \tag{3.12}$$

Recall (Sensitivity/ True Positive Rate) represents the percentage of actual positives correctly identified by the model.

$$Recall = \frac{TP}{(TP+F\ )} \tag{3.13}$$

F1 Score provides a balance between Precision and Recall, calculated as their harmonic mean.

$$F1\ Score = 2 \times \frac{FN}{TP+FN} \tag{3.14}$$

Area Under the Curve (AUC) evaluates the model's ability to distinguish between positive and negative instances, with higher values indicating better discrimination.

$$\text{AUC} = \sum \left(\frac{\text{TPR}_i + \text{TPR}_{i+1}}{2} \times (\text{FPR}_{i+1} - \text{FPR}_i) \right) \tag{3.15}$$

where, TPR is the True Positive Rate and FPR is the False Positive Rate.

Equal Error Rate (EER) is the point where the False Acceptance Rate (FAR)equals the False Rejection Rate (FRR) on the Receiver Operating Characteristic (ROC) curve, providing a single metric to assess model reliability. A lower EER indicates better performance.

$$EER=FAR=FRR \tag{3.16}$$

PSNR measures the quality of a compressed or reconstructed image/video by comparing it to the original. Higher PSNR values indicate lower distortion, with scores above 40dB (Decibel) considered excellent, 30-40dB good, 20-30dB fair and below 20dB poor.

3.7 RESULTS AND DISCUSSIONS

CNN-YOLO's performance is examined in this section and its effectiveness is contrasted with both a conventional CNN and the YOLOv4 architecture. The input video is converted into a masked dataset and the human positions are plotted using a red boxplot, as represented in Figure 3.11.

Figure 3.11 Boxplot with humans

Image features are extracted and anomalous events are detected using the proposed methodology are illustrated in Figures 3.12 and 3.13, correspondingly.

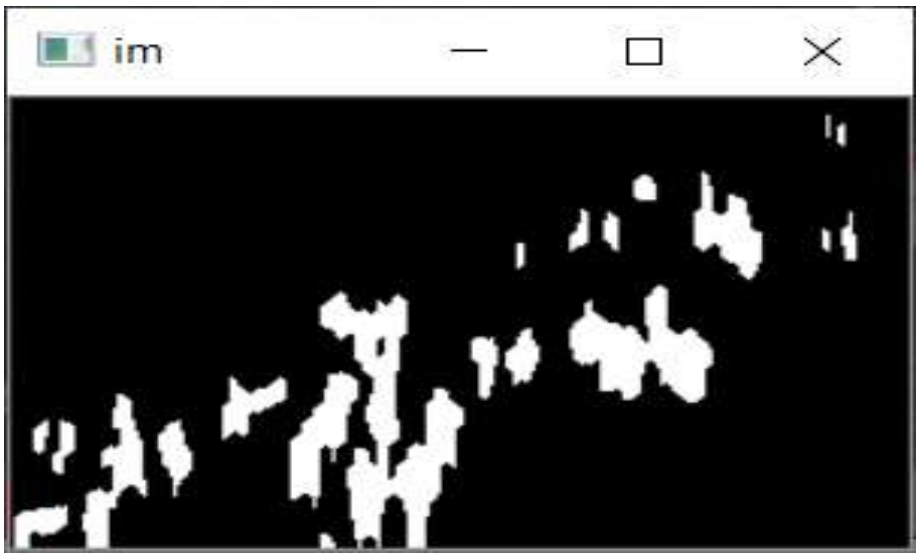

Figure 3.12 Feature Extraction

The Figure 3.12 represents a binary mask generated from the input video, where white regions indicate detected motion or foreground objects and black regions represent the background, aiding in feature extraction for anomaly detection.

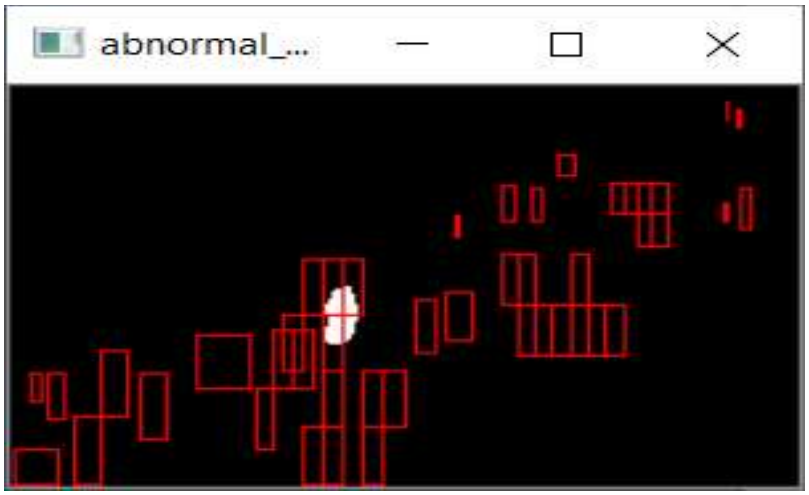

Figure 3.13 Anomalous Event Detection

The Figure 3.13 illustrates detected anomalies in the surveillance video, where red bounding boxes highlight regions of interest and the white region represents an identified abnormal event based on the proposed anomaly detection methodology.

The CNN - YOLO models offer significant advantages, including their compact size, high-speed computation and streamlined architecture, which enable them to instantly provide bounding box positions and classifications. By processing the entire image for detection, YOLO effectively captures the global context, reducing errors in misclassifying background regions as objects. Additionally, its ability to learn general features ensures adaptability across diverse applications.

3.7.1 Performance Evaluation

Table 3.2 provides the performance of hybrid CNN-YOLO model for the object detection.

Table 3.2 Performance of CNN-YOLO Model

Performance Metrics	CNN-YOLO Model
Accuracy	99.46 (%)
Precision	99.24 (%)
Recall	98.79 (%)
F1 Score	99.26 (%)
AUC	0.9926
PSNR	22.34 (dB)
EER	2.1 (%)

The results indicate a substantial enhancement in performance improvement when using the CNN-YOLO model related to the CNN and YOLOv4 models for VAD on the UCSD dataset. The CNN-YOLO model attains an accuracy of 99.46%, outperforming CNN model and YOLOv4 model. Similarly, it exhibits a higher F1 Score of 99.26%, representing a better harmony among Precision and Recall. The AUC value of 0.9926 confirms its superior ability to classify anomalies effectively.

Furthermore, the CNN-YOLO model significantly reduces the EER to 2.1%, minimizing misclassification errors compared to the other models. Additionally, the PSNR value of 22.34dB validates its robustness in processing video frames while maintaining image quality. These results establish that the CNN-YOLO model excels the CNN and YOLOv4 models for accuracy, reliability and anomaly detection, converting it a more efficient solution.

Figure 3.14 demonstrates that the accuracy during training reaches an impressive 99.46%, comparable to the level achieved at the 25th epoch. The Validation accuracy, however, fluctuates slightly but remains stable around 98.5%, indicating generalization with minor variations.

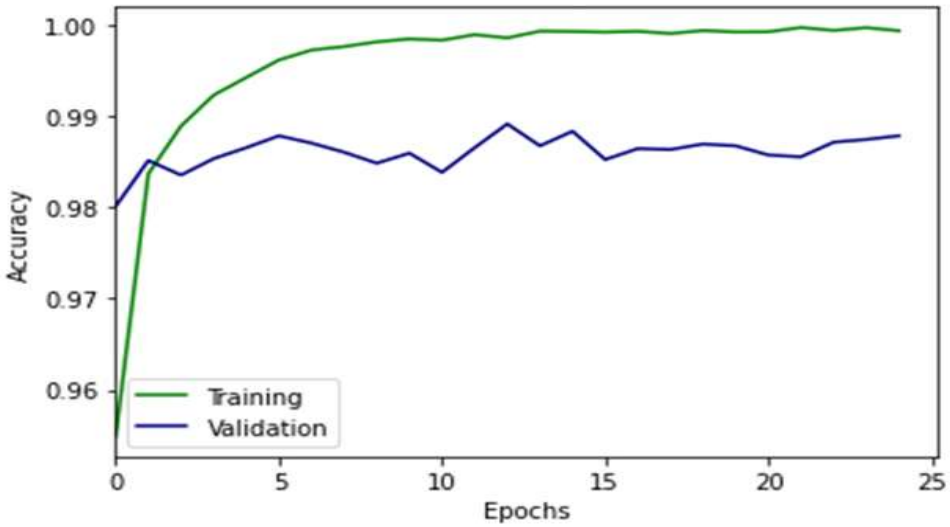

Figure 3.14 Training Progress of Accuracy Analysis

Figure 3.15 reveals minimal loss during training, particularly around the 25th epoch, where the training loss approaches near zero. However, the validation loss shows fluctuations, suggesting potential overfitting, in which the model exhibits remarkable performance on training data but exhibits some instability on validation data.

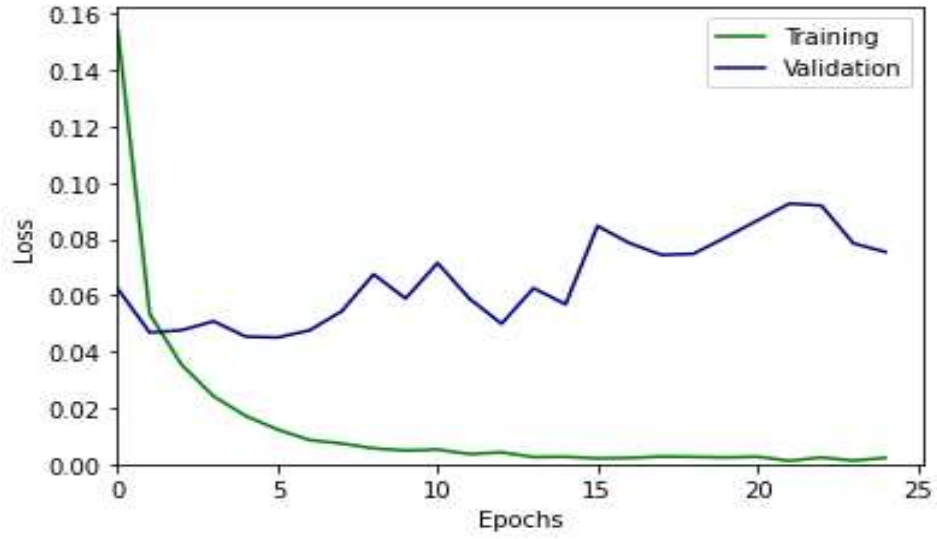

Figure 3.15 Training Progress of Loss Analysis

Figures 3.16 to 3.19 illustrate the performance comparison of CNN - YOLO with CNN, YOLOv4 models for Accuracy, Precision, Recall and F1 Score. Figure 3.16 and Figure 3.17 depicts the comparison of accuracy and precision.

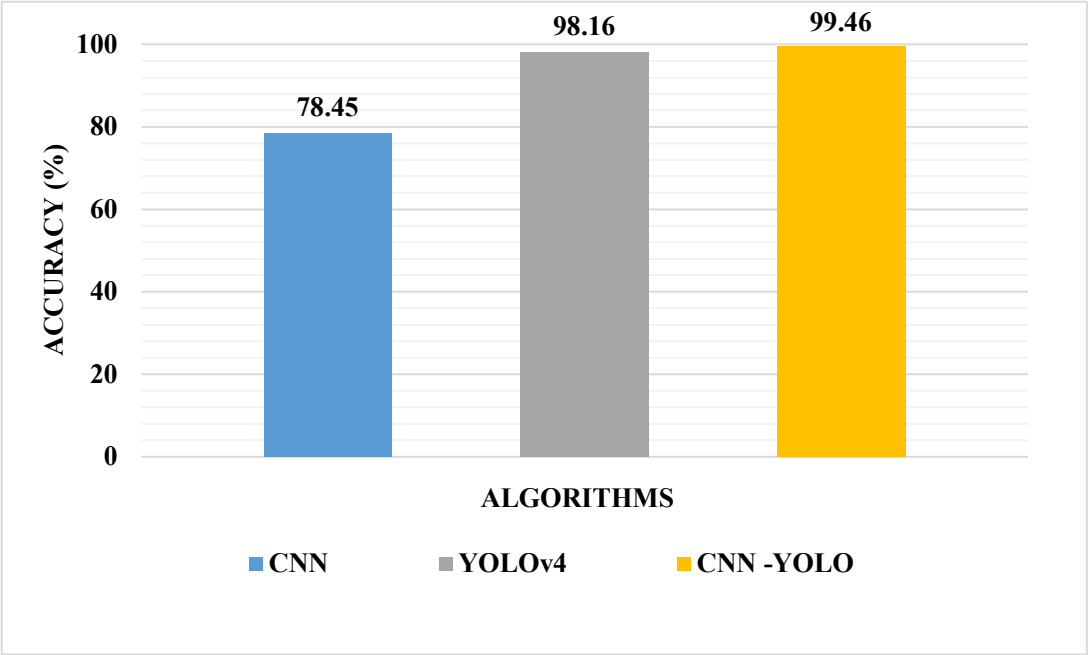

Figure 3.16 Performance Comparison of Accuracy

The output in Figure 3.16 designates that an accuracy of 78.45% is achieved by CNN, 98.16% by YOLOv4 and 99.46% by CNN-YOLO. It indicates that YOLOv4 achieves an improvement of 25.12% over CNN and CNN-YOLO outperforms both, with a 26.77% improvement over CNN and 1.32% over YOLOv4. These outcomes highlight the success of the suggested CNN-YOLO method in improving the classification accuracy as deep learning-based object detection model.

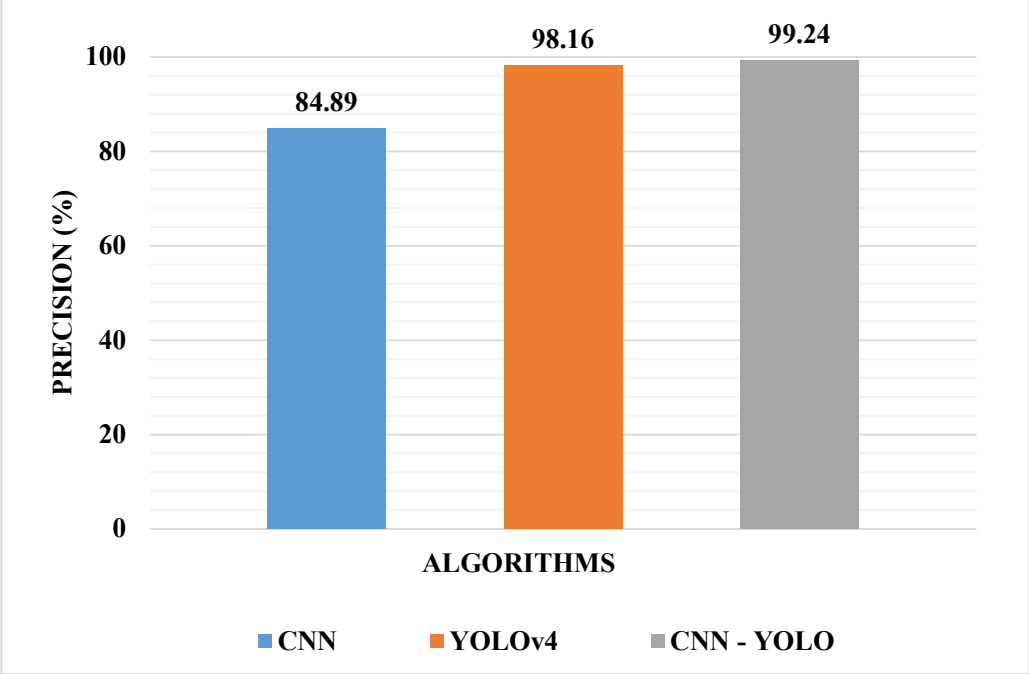

Figure 3.17 Performance Comparison of Precision

Figure 3.17 compares Precision, showing that CNN achieves 84.89%, while YOLOv4 reaches 98.16% and CNN-YOLO attains 99.24%. CNN-YOLO demonstrates the highest Precision, improving by 16.90% over CNN and 1.10% over YOLOv4, ensuring minimal false positives. YOLOv4 also achieves a 15.61% gain over CNN, reinforcing the impact of deep learning in achieving precise classifications. Figure 3.18 and Figure 3.19 presents the comparison of Recall and F1 Score respectively.

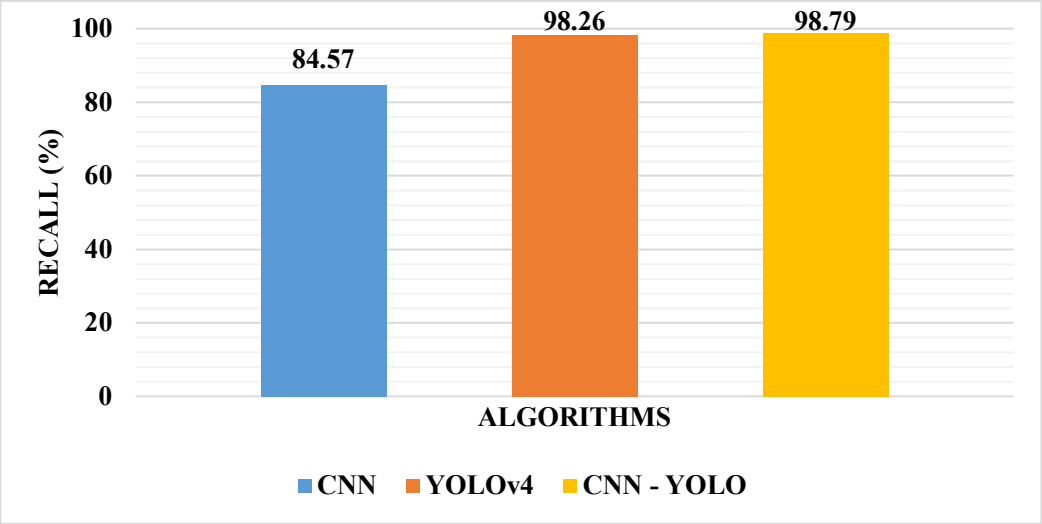

Figure 3.18 Performance Comparison of Recall

Figure 3.18 compares Recall, indicating that CNN achieves 84.57%, whereas YOLOv4 reaches 98.26% and CNN-YOLO records 98.79%. CNN-YOLO exhibits the highest Recall, with a 16.84% improvement over CNN and a 0.54% increase over YOLOv4, effectively reducing false negatives. YOLOv4 also shows a 16.19% gain over CNN, further validating its effectiveness in capturing positive instances.

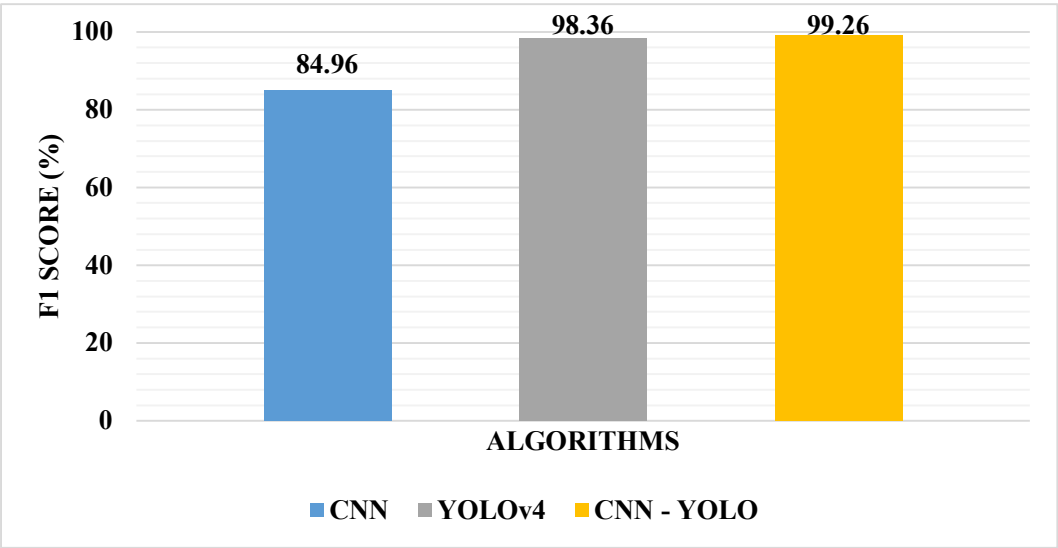

Figure 3.19 Performance Comparison of F1 Score

Figure 3.19 compares the F1 Score, highlighting that CNN attains 84.96%, while YOLOv4 achieves 98.36% and CNN-YOLO reaches 99.26%. CNN-YOLO achieves the highest balance of Precision and Recall, improving by 16.79% over CNN and 0.91% over YOLOv4. YOLOv4 also surpasses CNN by 15.73%, emphasizing the advantages of deep learning in optimizing classification performance. Figure 3.20 presents the comparison the AUC.

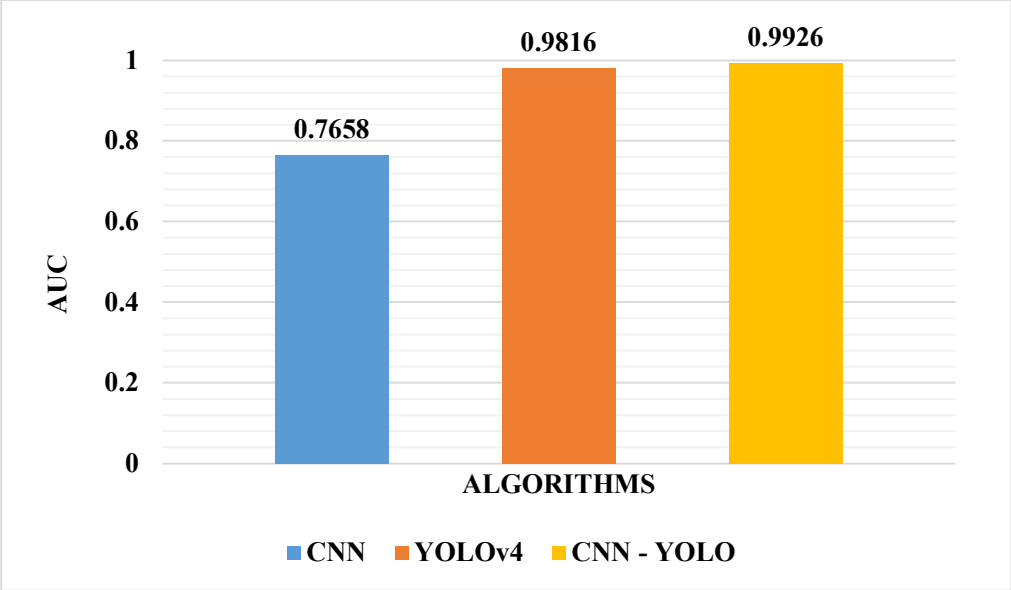

Figure 3.20 Performance Comparison of AUC

Figure 3.20 evaluates the AUC, demonstrating that CNN achieves 76.58%, while YOLOv4 records 98.16% and CNN-YOLO reaches 99.26%. CNN-YOLO outperforms both, with a 29.65% improvement over CNN and a 1.12% gain over YOLOv4, showcasing its superior ability to distinguish between normal and anomalous instances. YOLOv4 also improves upon CNN by 28.16%, further validating the efficacy of deep learning-based detection models. Figure 3.21 presents the comparison the PSNR.

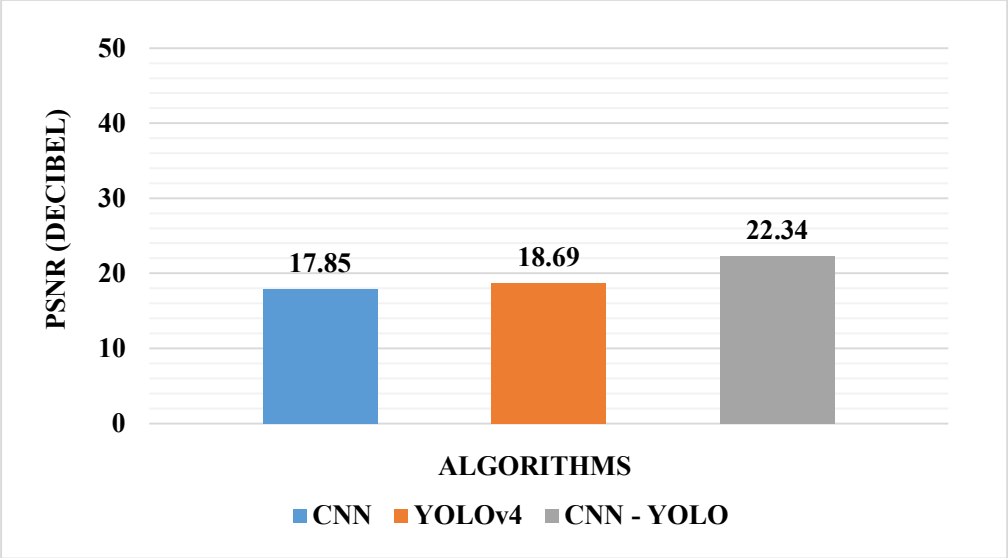

Figure 3.21 Performance Comparison of PSNR

Figure 3.21 compares PSNR, showing that CNN attains 17.85dB, while YOLOv4 achieves 18.69dB and CNN-YOLO reaches 22.34dB. CNN-YOLO exhibits the highest PSNR, improving by 25.13% over CNN and 19.52% over YOLOv4, indicating superior image quality in anomaly detection. YOLOv4 also surpasses CNN by 4.71%, demonstrating its effectiveness in maintaining better signal quality. Figure 3.22 depicts the comparison the EER.

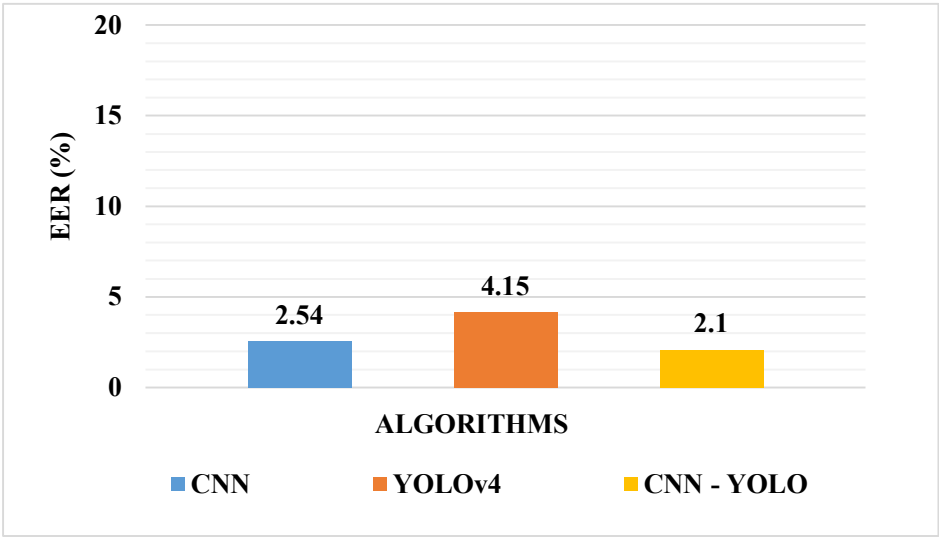

Figure 3.22 Performance Comparison of EER

Figure 3.22 compares EER, revealing that CNN records 2.54%, YOLOv4 registers 4.15% and CNN-YOLO achieves 2.1%. CNN-YOLO attains the lowest EER, reducing errors by 17.32% compared to CNN and 49.40% compared to YOLOv4, ensuring a proper equilibrium between false positives and false negatives. YOLOv4, despite an increase of 63.39% over CNN, still performs well, demonstrating deep learning's impact on error reduction strategies. The CNN-YOLO model has the lowest EER, indicating minimal misclassification errors, while CNN performs slightly worse. YOLOv4 shows a significantly higher error rate, reducing its reliability.

These results confirm its exceptional ability to detect and classify objects accurately, making it the most reliable model for anomaly detection tasks. Additionally, the CNN-YOLO model demonstrates superior anomaly differentiation and better image quality retention related to the other models.

3.8 SUMMARY

The proposed CNN-YOLO model achieves real-time anomalous behavior detection with 99.46% Accuracy, 98.79% Recall, 99.24% Precision and a 99.26% F1 Score, along with a high AUC of 0.9926. It enhances image quality with a PSNR of 22.34dB and minimizes errors with an EER of 2.1%. The CNN-YOLO model proves to be the most efficient approach for VAD, offering high accuracy, reliability and superior image quality preservation.