

Received 10 July 2024, accepted 23 July 2024, date of publication 29 July 2024, date of current version 7 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435144

RESEARCH ARTICLE

Spatially Aware Fusion in 3D Convolutional Autoencoders for Video Anomaly Detection

ASIM NIAZ¹, SAREER UL AMIN¹, SHAFIULLAH SOOMRO², HAMZA ZIA¹,
AND KWANG NAM CHOI¹

¹Department of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea

²Department of Computer Science and Media Technology, Linnaeus University, 352 52 Växjö, Sweden

Corresponding author: Kwang Nam Choi (knchoi@cau.ac.kr)

This work was supported in part by the Ministry of Science and Information and Communication Technology (ICT), and in part by the National Information Technology (IT) Industry Promotion Agency (NIPA) through the High Performance Computing (HPC) Support Project.

ABSTRACT Surveillance videos are crucial for crime prevention and public safety, yet the challenge of defining abnormal events hinders their effectiveness, limiting the applicability of supervised methods. This paper introduces an unsupervised end-to-end architecture for video anomaly detection that applies spatial and temporal features to identify anomalies in surveillance footage. The model employs a three-dimensional (3D) convolutional autoencoder, with an encoder-decoder structure that learns spatiotemporal representations and reconstructs the input through the latent space. Skip connections linking the encoder and decoder blocks facilitate the transfer of information across various scales of feature representations, enhancing the reconstruction process and improving the overall performance. The architecture incorporates spatial attention modules that highlight informative regions in the input, enabling improved anomaly detection. Spatial and contextual dependencies are further acquired using 3D convolutional filters. The performance of the proposed model is assessed on four benchmark datasets: UCSD Pedestrian 1, UCSD Pedestrian 2, CUHK Avenue, and ShanghaiTech. Notably, the proposed model achieves frame-based Area Under the Curve (AUC) scores of 94.6% on UCSD Ped 1, 96.7% on UCSD Ped 2, 84.7% on CUHK Avenue, and 74.8% on ShanghaiTech. These results demonstrate the state-of-the-art performance of the proposed approach, highlighting its efficacy in real-world anomaly detection scenarios.

INDEX TERMS Anomaly detection, autoencoders, computer vision, intelligent surveillance systems, three-dimensional convolutional neural network (3DCNN), video-based abnormal event detection.

I. INTRODUCTION

Anomaly detection in surveillance videos has become crucial in modern society, as it helps maintain security, offering a monitoring that spans from highly secured military installations, businesses, and private properties to public spaces [1], [2], [3], [4]. Surveillance cameras continually capture and record video footage, and scientists deal with vast, continuously evolving data, requiring high-end devices for rapid processing [5], [6], [7], [8], [9], [10], [11]. In the era of artificial intelligence, manual observation of years-long

patterns or videos to alert relevant personnel in the case of an emergency or anomaly is not practical; instead, automated algorithms have been developed to raise an alert if unexpected events occurs, such as a vehicle hitting a pedestrian or a mob gathering in front of a secure site [12], [13]. Detecting anomalous events is challenging because these events cover only a minimal fraction of a large set of events [14], [15].

Supervised learning-based video anomaly detection solves such limitations by automatically identifying and flagging unusual activities or behaviors, enabling real-time alerts without human intervention [16], [17]. However, algorithms that have the potential to automate the surveillance video anomaly detection process still need densely annotated

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang¹.

datasets, which significantly increases the labor and time expenses. Unlike supervised learning approaches, unsupervised learning-based video anomaly detection algorithms do not depend on dense annotations. Instead, these models are trained using normal data, which adhere typical patterns without any deviation. Existing video anomaly detection techniques primarily employ two-stream networks. One is for spatial features and the other for motion features [18], [19], [20]; however, an extra stream increases the computational cost.

Two-dimensional (2D) convolutional neural networks (CNNs) are a popular method for video anomaly detection in literature in supervised and unsupervised surveillance [21], [22], [23], [24]. The 2D-CNN learns hierarchical spatial features well at various levels of abstraction and local contextual understanding of the anomaly. Nevertheless, the 2D-CNN offers limited temporal understanding due to the lack of motion information, which is one of the most critical criteria for distinguishing abnormal from normal behavior.

Typically, unsupervised video anomaly detection is based on autoencoders [25], a type of neural networks that processes input frames into lower dimensions followed by upsampling them back to their original dimensions in one stream while the temporal features are addressed through a second stream. The part of the autoencoder responsible for lowering the dimensions is termed the encoder. In contrast, the decoder is responsible for up-sampling input frames back to the original dimensions. Then, the reconstruction loss is computed to identify anomalies in every scene. 2D-CNNs can also expand their applicability to unsupervised autoencoder methods; however, they overgeneralize frame patterns with no temporal understanding. Hence, temporal learning blocks are appended to such models. One study [26] addressed autoencoder over generalization by reducing prediction errors for normal cases instead of increasing errors for abnormal cases, using hybrid attention and motion constraint-based autoencoder for anomaly detection.

The detection of unusual activities is crucial to ensuring safety and security in video surveillance systems. Developing a video anomaly detection system for effective deployment in real-world scenarios necessitates high performance and resource efficiency with integrated capabilities to differentiate between normal and anomalous events accurately. Achieving these requirements poses a significant challenge because integrating robust performance with efficient resource utilization in a single method remains complex.

This research follows the autoencoder stream to identify anomalies in surveillance videos. The difference in the proposed approach is that the proposed autoencoder structure is greatly inspired by the convolutional 3-dimensional neural network (C3DN) [27], unlike previous literature, to the best of our knowledge. Fig. 1 illustrates a three-dimensional convolution, highlighting spatial features X and Y for each video frame, with Z representing the temporal dimension. The C3DN-based autoencoder extracts not only the spatial features but also the temporal features, unlike the 2D-CNN.

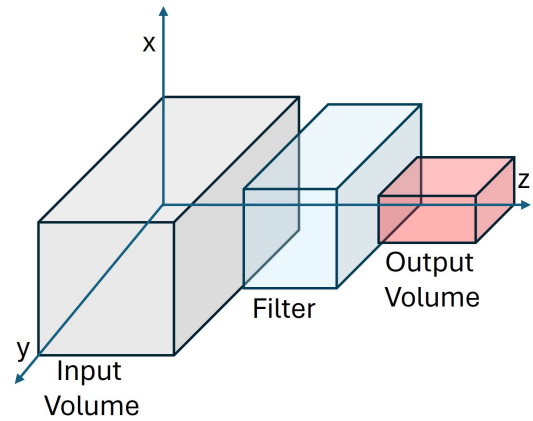


FIGURE 1. Illustration of a three-dimensional convolution with volume dimensions labeled, where X and Y are the spatial features of each video frame, and Z is the temporal dimension.

The presented autoencoder combines several advantages, including capturing spatiotemporal information, a better representation and end-to-end manner of learning, contextual understanding, adaptability to various anomaly types and robustness to noise and variations. Moreover, the proposed algorithm is computationally efficient with significantly less number of parameters compared to the literature. The main contributions of this research are summarized as follows:

- **Development of an end-to-end unsupervised anomaly detection approach:** We propose a novel 3D-CNN-based autoencoder that leverages both spatial and temporal features for effective anomaly detection in surveillance videos.
- **Utilization of normal surveillance videos for training:** The autoencoder is trained on normal video data, enabling the model to capture inherent patterns and regularities without the need for labeled anomaly data. This approach ensures a meaningful latent space representation that forms the basis to identify anomalies.
- **Performance enhancement:** Integrating skip connections and spatial attention modules enhances the performance of the autoencoder. Skip connections facilitate transferring information across scales of feature representations, whereas spatial attention modules selectively emphasize informative regions in the input data, improving overall anomaly detection.
- **Resource-efficient design:** Our approach offers a lightweight network with superior efficiency, featuring a significantly lower parameter count (5.760 million), model size (23.05 MB), and reduced time complexity per sequence (160 milliseconds). This makes it a practical solution for real-world applications where resource constraints are a concern.
- **State-of-the-art performance on benchmark datasets:** The proposed method demonstrates state-of-the-art (SOTA) performance on various benchmark datasets, including UCSD Pedestrian 1 [28], UCSD

Pedestrian 2 [29], ShanghaiTech [30], and CUHK Avenue [31]. These results underscore the efficacy and robustness of our approach in diverse surveillance environments.

In summary, this introduction provides an overview of the context and motivation behind this research. Section II explores the related work, offering insight into the existing literature and the gaps we aim to address. In addition, Section II outlines the significance of addressing the challenges in the existing work and Section III details how the proposed framework, contributes to the existing body of knowledge. Then Section IV presents the experimental findings and results, confirming the effectiveness of the proposed approach. Finally, Section V concludes this research and details the implications of this study for future research.

II. RELATED WORK

For the last decade, researchers have explored anomaly detection in surveillance videos. Surveillance video anomaly detection methods are broadly classified into two streams: prediction-based and reconstruction-based methods.

Prediction-based surveillance video anomaly detection identifies abnormal activities or events in surveillance videos by predicting the expected behavior in each scene and detecting deviations from those predictions. The prediction approach exploits the spatial and motion features of consecutive frames to identify anomalies, combining elements of predictive modeling and anomaly detection to enhance the accuracy of anomaly detection systems.

With substantial advancements in video prediction [32], researchers have integrated predictive capabilities into anomaly detection by assuming that routine or everyday events can be predicted, whereas abnormal events are unpredictable. One notable method in this domain is using Generative Adversarial Networks (GANs) for prediction-based video anomaly detection, which are trained to generate future frames in a video sequence based on the preceding frames. The generator component predicts subsequent frames, whereas the discriminator distinguishes between the actual and predicted frames. Liu et al. [33] proposed a network architecture that involves a U-Net with components for intensity and gradient distance minimization. The architecture enforces optical flow coherence, and determines abnormality based on differences between the predicted and actual future frames.

In contrast, one study [34] employed a convolutional long short-term memory (LSTM) model as a unit within a composite LSTM model, supporting an encoder-decoder approach with separate branches for reconstruction and prediction. The convolutional LSTM model applies a convolutional feature representation, predicts the latent space representation, and evaluates anomalies in a robotics application based on the prediction error [34].

Yang et al. introduced a model addressing the problem of video anomaly detection in various scenarios using patch-level loss and a segmentation map. Based on multi-scale

structural similarity, the patch-level loss emphasizes foreground boundaries during training. The segmentation map alters prediction errors between frames during the testing phase, enhancing the foreground errors while reducing background errors. This approach effectively increases the distinction between normal and anomalous classes.

Despite advancements, prediction-based video anomaly detection methods encounter challenges with noise-sensitive motion constraints, such as optical flow, which is often integrated into the framework. This incorporation can result in reduced robustness. Moreover, prediction-based video anomaly detection methods have difficulty capturing long-range temporal dependencies and frequently face high computational costs, making real-time detection challenging and potentially increasing the false positive rates (FPRs).

Reconstruction-based surveillance video anomaly detection trains a model to reconstruct input video sequences using autoencoders that comprising an encoder and a decoder. The former compresses the input feed into lower-dimensional space, whereas the latter reconstructs the output to the original dimension from the compressed dimensions. After compression and extraction, a calculation of the reconstruction error follows. After establishing a preset threshold, the sequence is categorized as either a normal or abnormal based on the degree of resemblance to the learned sequence. The reconstruction error for Anomalous events are higher to normal events.

Several reconstruction models have been proposed that were originally based on principal component analysis (PCA) and autoencoders. One study [23] proposed an autoencoder-based method that learns regular patterns in video sequences by reconstructing motion signatures with low error in normal videos but not as accurately in irregular videos. This model captures the complex distribution of appearance dynamics in normal videos. Initially, state-of-the-art handcrafted motion features were employed. However, handcrafted features may not be optimal for this purpose; therefore, an end-to-end learning model based on a fully connected CNN is employed to learn local motion features and train the autoencoder, as the handcrafted features may not be optimal for this purpose.

In [35], the authors designed a bidirectional spatiotemporal feature learning framework for video sequences, incorporating a double-encoder double-decoder network and multi-scale anomaly evaluation strategy. Another study [36] used convolutional layers to learn spatial features, and the temporal representation was learned with LSTM.

Continual learning has recently been employed in the field of video anomaly detection. Doshi et al. built a continual learning framework involving a neural network-based feature extraction module and a statistical anomaly detection module. At each time step t , the feature extraction module extracts three types of features: motion (optical flow), location (center coordinates and bounding-box area), and appearance (class probabilities) to the statistical anomaly detection module. The statistical anomaly detection module employs these features

to make online decisions and update its decision rule with new data.

A novel method [37] was proposed for anomaly detection and localization in videos, following several critical steps to achieve accurate results. Initially, the method generates local gradient and optical flow patches, representing the discriminative appearance and motion cues. The approach, it trains a two-stream multivariate Gaussian fully convolutional adversarial autoencoder (MGFC-AAE) on normal samples of gradient and optical flow patches enabling training process enables the model to learn underlying anomaly patterns. During testing, the learned models compute anomaly scores for appearance and motion cues, employing an energy-based method. Finally, the authors combined these anomaly scores from the appearance and motion cues to enhance the detection performance, leveraging their complementary information.

Similarly, another study [19] employed a two-stream framework to investigate motion and appearance anomalies, aiming to enhance anomaly detection accuracy. In Stage 1, gradient and optical flow cuboids are generated from raw video cuboids to represent appearance and motion cues, using anomaly scores with the Gaussian distribution. The fusion of these scores aids in detecting potentially suspicious abnormal cuboids by eliminating superfluous normal ones. In Stage 2, the two streams process the gradient and optical flow cuboids of the suspicious anomalous cuboids are further processed in two streams. Based on their appearance and anomaly scores, each patch is categorized as either normal or abnormal.

In another study [38], spatiotemporal representations were captured, introducing a unique motion autoencoder that inputs consecutive frames and outputs red, green, and blue (RGB) differences to lower the computational costs for optical flow calculation. This method incorporates a variance attention module to assign weights to the dynamic elements of the video clips, enhancing the convergence performance of the motion autoencoder. A deep k-means cluster was designed to encourage the autoencoder network to generate compact motion and appearance descriptors. This cluster is trained only on regular events, leading to greater distances between abnormal representations and the cluster compared to normal patterns. The anomaly assessment combines the reconstruction error and cluster distance, efficiently detecting anomalies in the videos.

Moreover, another study [39] developed a memory module, which employs attention to retrieve relevant items from memory based on the encoded representation of the input data for reconstruction, facilitating anomaly detection via reconstruction errors. Reconstruction-based video anomaly detection methods may encounter challenges with overfitting normal data, which can result in poor generalize-ability and difficulty in distinguishing anomalies, particularly in noisy and complex scenes. Overall, while these methods mentioned above represent substantial advancements, there remains a need to model temporal dependencies between temporal sequences require further modeling.

In [40], the authors used a combination of U-Net and a vision transformer for video anomaly detection to capture richer temporal information and more global contexts; however, the method compromises computational efficiency by incorporating transformers. Context Tree Based Anomaly Detection (CTBAD) [41] is not strictly reconstruction or prediction-based; instead, it employs an ensemble technique utilizing a hierarchical ensemble of image plane partitions generated by a context tree, allowing for the detection of both Indigenous anomalies and locational anomalies through linear combination of separately trained partition models, balancing between low complexity for sparse data and high complexity for improved accuracy with sufficient data.

III. PROPOSED FRAMEWORK

This section presents the proposed surveillance video anomaly detection method employing a C3DN-based autoencoder capable of exploiting spatial and temporal features. The proposed method drew inspiration from the work in [27]. We built an unsupervised learning technique to detect anomalies in surveillance videos without requiring labeled training data based on the prior work. The proposed approach aims to generate concise representations of ordinary surveillance video clips and detect irregularities by analyzing reconstruction errors.

This research proposes end-to-end anomaly detection with a video surveillance approach using a 3D-CNN. The input of the network is regular surveillance videos input to the network in small segments of consecutive frames, $S_i = f_1, f_2, \dots, f_n$. During training, the autoencoder learns to capture the underlying patterns and regularities in the normal video data. The latent space representation serves as a compressed version of the input frames, contains meaningful information pertaining to the normal data distribution. When the trained autoencoder encounters an anomalous frame during testing or deployment, the reconstruction error is expected to be higher than for the normal frames because it likely deviates from the learned patterns and regularities, making it challenging for the autoencoder to reconstruct the frame accurately.

A. DATA PREPROCESSING

In the initial preprocessing steps for surveillance video analysis, we begin by segmenting the input videos into shorter clips, where each clip has 16 consecutive frames. These clips were designed to acquire temporal patterns and dynamics effectively. Subsequently, each frame in these clips is resized to a consistent spatial resolution of 112x112 pixels, ensuring uniformity in the image size across all frames. Additionally, color channel normalization was applied to address lighting variations, enabling the neural network to focus on the content of these frames without being influenced by changes in lighting conditions. Color channel normalization is calculated using the following equation:

$$\text{Normalized Pixel Value} = \frac{\text{Original Pixel Value} - \text{Mean}}{\text{Standard Deviation}} \quad (1)$$

where Normalized Pixel Value represents the adjusted pixel value after normalization. Original Pixel Value denotes the pixel value from the original image. Mean indicates the mean value calculated for the specific color channel (e.g., Red, Green, or Blue). Standard Deviation is the standard deviation calculated for the same color channel.

$$\text{Standard Deviation} = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (2)$$

where:

- Standard Deviation is a measure of the spread or dispersion of data points from the mean. N is the total number of data points. x_i represents each individual data point. $\bar{x}$ is the mean (average) of the data points. $\sum$ denotes the summation of all the squared differences between each data point (x_i) and the mean ($\bar{x}$), divided by N .

B. PROPOSED ARCHITECTURE

The proposed C3DN-based autoencoder architecture comprises an encoder and a decoder, designed to capture salient features and reconstruct the input video frames, respectively. Fig. 2 depicts the pictorial representation of the proposed method. The architectural details follow.

1) ENCODER

The encoder is responsible for extracting discriminative features from the input video clips. It comprises several 3D convolutional layers to capture spatial and temporal information. Each 3D convolutional layer applies a set of learnable filters to local spatiotemporal regions, generating feature maps that encode relevant information. The 3D convolutional layer in network architecture can be mathematically represented as follows.

We consider an input video volume with dimensions (X, Y, Z) , where: - X represents the spatial width of each video frame, - Y denotes the spatial height of each video frame and - Z represents the temporal depth (number of video frames) in the input volume.

We assume a convolutional filter with dimensions (F, F, F) , where F represents the spatial width, height, and depth of the filter.

In the 3D convolutional layer, we apply multiple filters to the input volume, resulting in multiple output feature maps.

The output feature maps are denoted as V , where: - V represents the number of output feature maps.

The mathematical equation for the 3D convolutional layer is written as:

$$\begin{aligned} Y(i, j, k, v) \\ = b(v) + \sum_{l=0}^{F-1} \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} X(i+l, j+m, k+n, l) \cdot W(l, m, n, v) \end{aligned} \quad (3)$$

where $Y(i, j, k, v)$ represents the value at the i -th spatial row, j -th spatial column, k -th temporal depth, and v -th feature map

in the output volume, and $b(v)$ denotes the bias term for the v -th feature map. Further, $X(i+l, j+m, k+n, l)$ indicates the value at the $(i+l)$ -th spatial row, $(j+m)$ -th spatial column, $(k+n)$ -th temporal depth, and l -th channel in the input volume. $W(l, m, n, v)$ represents the weight assigned to the l -th spatial row, m -th spatial column, n -th temporal depth, and v -th feature map in the convolutional filter. In this equation, the variables i and j are iterated over the spatial width and height of the output volume, respectively, and the variable k is iterated over the temporal depth of the output volume. The variable v iterated over the number of output feature maps.

This mathematical representation captures the essence of a 3D convolutional layer in a neural network for processing spatiotemporal data, where X and Y represent the spatial dimensions of each video frame, and Z represents the temporal dimensions.

Following each convolutional layer, a rectified linear unit (ReLU) activation function

$$\text{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \quad (4)$$

is applied to introduce non-linearity to the network. Dropout regularization is also applied after the activation function to enhance generalization and mitigate overfitting.

The Spatial Attention Module introduced in our method is a crucial component within the neural network architecture. It enhances the model's ability to capture spatial dependencies within the input tensor. The spatial attention module operates by first applying a 3D convolution operation on the input tensor $\mathbf{X}$ with learnable weights $\mathbf{W}$. The output of this convolutional operation is then passed through a sigmoid activation function σ to obtain the attention map $\mathbf{A}$:

$$\mathbf{A} = \sigma(\mathbf{W} * \mathbf{X}) \quad (5)$$

Finally, the attended features are computed by element-wise multiplication of the attention map $\mathbf{A}$ with the input tensor $\mathbf{X}$, resulting in the output tensor $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{A} \quad (6)$$

Table 1 presents the spatial attention module. Moreover, the encoder applies 3D max-pooling layers to reduce the temporal dimension while preserving critical spatial information. The following max pooling operation downsamples the feature maps along the temporal dimension, recording critical information while reducing the computational complexity of the network.

$$\begin{aligned} \text{MaxPooling3D}(X, F, S) \\ = \left\{ Y(i, j, k, c) = \max_{l=0}^{F-1} \max_{m=0}^{F-1} \max_{n=0}^{F-1} X(i \cdot S + l, j \cdot S \right. \\ \left. + m, k \cdot S + n, c) \right\} \end{aligned} \quad (7)$$

Specifically, the first max-pooling layer has a kernel size of $1 \times 2 \times 2$ and a stride of $1 \times 2 \times 2$, and the second max-pooling layer has a kernel size of $2 \times 2 \times 2$ and a stride of $2 \times 2 \times 2$.

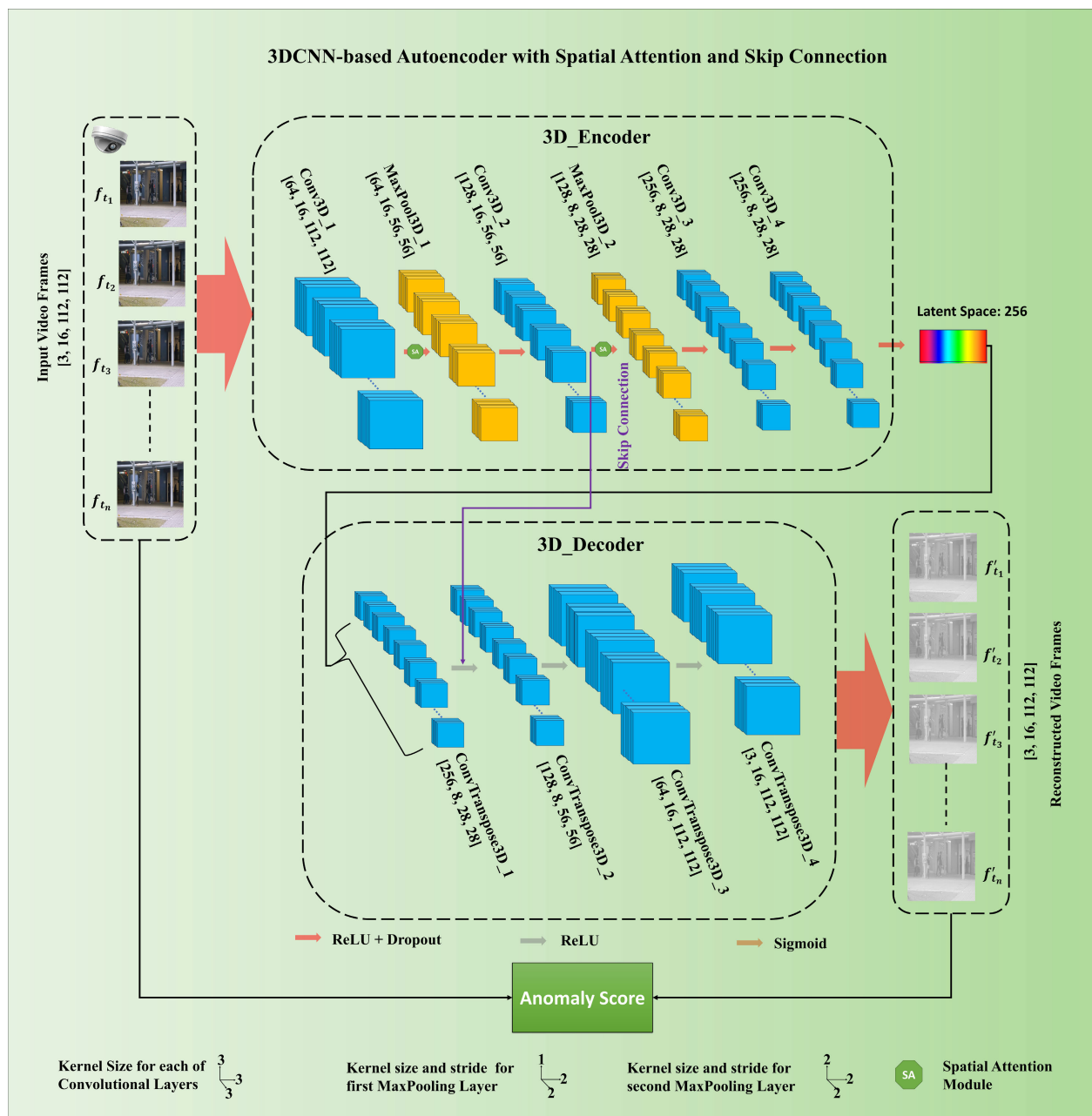


FIGURE 2. Framework diagram of the proposed 3D CNN-based autoencoder with spatial attention and skip connections for anomaly detection in surveillance videos. The autoencoder leverages spatial and temporal features from the input video feed to capture and reconstruct relevant patterns. The encoder module extracts spatial and temporal representations using 3D convolutional layers, whereas the decoder module reconstructs the video frames from the learned latent representations. This architecture enables effective anomaly detection by identifying and representing normal spatial patterns (e.g., objects, shapes, textures) present in the video frames. At the same time, it can also learn the normal temporal sequences (e.g., movement, and actions) exhibited in the video feed. The anomaly score identifies anomalies by measuring the dissimilarity between the original frames and reconstructed frames.

Table 2 summarizes the specific architecture and hyper-parameters of the encoder, including the number of convolutional layers, kernel sizes, and input volume, resulting in multiple output feature maps.

2) DECODER

The decoder aims to reconstruct the input video clips from the latent representations obtained from the encoder.

It uses transposed convolutional layers, referred to as deconvolutional layers, to upsample the latent representations and recover the spatial resolution. Each transposed convolutional layer applies a set of learnable filters to the input feature maps, producing feature maps with higher spatial dimensions. Similar to the encoder, the ReLU activation function is applied after each transposed convolutional layer to introduce nonlinearity to the network.

TABLE 1. Working of the spatial attention module in proposed network architecture.

Step	Operation	Output Dimensions
1	Input Tensor	(1, 3, 16, 112, 112)
2	3D Convolution	(1, 1, 16, 112, 112)
3	Sigmoid Activation	(1, 1, 16, 112, 112)
4	Element-wise Multiplication	Input Tensor $\times$ Sigmoid Output Result (1, 3, 16, 112, 112)
5	Output Tensor	(1, 3, 16, 112, 112)

TABLE 2. Encoder layers for C3DN_AE_Fusion.

Layer	Operation
Conv3D_1	$X^{(3,H,W)} \rightarrow Z^{(64, \frac{H}{2}, \frac{W}{2})}$
ReLU	$Z^{(64, \frac{H}{2}, \frac{W}{2})} \rightarrow Z^{(64, \frac{H}{2}, \frac{W}{2})}$
Dropout3D	$Z^{(64, \frac{H}{2}, \frac{W}{2})} \rightarrow Z^{(64, \frac{H}{2}, \frac{W}{2})}$
SpatialAttentionModule	$Z^{(64, \frac{H}{2}, \frac{W}{2})} \rightarrow Z^{(64, \frac{H}{2}, \frac{W}{2})}$
MaxPool3D_1	$Z^{(64, \frac{H}{2}, \frac{W}{2})} \rightarrow Z^{(64, \frac{H}{4}, \frac{W}{4})}$
Conv3D_2	$Z^{(64, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(128, \frac{H}{4}, \frac{W}{4})}$
ReLU	$Z^{(128, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(128, \frac{H}{4}, \frac{W}{4})}$
Dropout3D	$Z^{(128, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(128, \frac{H}{4}, \frac{W}{4})}$
SpatialAttentionModule	$Z^{(128, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(128, \frac{H}{4}, \frac{W}{4})}$
MaxPool3D_2	$Z^{(128, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(128, \frac{H}{8}, \frac{W}{8})}$
Conv3D_3	$Z^{(128, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(256, \frac{H}{8}, \frac{W}{8})}$
ReLU	$Z^{(256, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(256, \frac{H}{8}, \frac{W}{8})}$
Dropout3D	$Z^{(256, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(256, \frac{H}{8}, \frac{W}{8})}$
Conv3D_4	$Z^{(256, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})}$
ReLU	$Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})}$
Dropout3D	$Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})}$

TABLE 3. Decoder layers for C3DN_AE_Fusion.

Layer	Operation
ConvTranspose3D_1	$Z^{(\text{latent_size}, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(256, \frac{H}{8}, \frac{W}{8})}$
ReLU	$Z^{(256, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(256, \frac{H}{8}, \frac{W}{8})}$
ConvTranspose3D_2	$Z^{(256, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(128, \frac{H}{8}, \frac{W}{8})}$
ReLU	$Z^{(128, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(128, \frac{H}{8}, \frac{W}{8})}$
ConvTranspose3D_3	$Z^{(128, \frac{H}{8}, \frac{W}{8})} \rightarrow Z^{(64, \frac{H}{4}, \frac{W}{4})}$
ReLU	$Z^{(64, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(64, \frac{H}{4}, \frac{W}{4})}$
ConvTranspose3D_4	$Z^{(64, \frac{H}{4}, \frac{W}{4})} \rightarrow Z^{(3, H, W)}$
Sigmoid	$Z^{(3, H, W)} \rightarrow X^{(3, H, W)}$

The final decoder layer employs a sigmoid activation function to produce pixel-wise values ranging from 0 to 1, representing the reconstructed clips. This approach enables the autoencoder to model the pixel-level dependencies and produce visually plausible reconstructions. Upsampling in the decoder is accomplished using transpose convolutions with appropriate kernel sizes, strides, and output paddings. Table 3 summarizes the specific architecture and hyperparameters of the decode. In Tables 2 and 3, $X^{(C,H,W)}$ represents the input tensor with channels C , height H , and width W . $Z^{(C,H,W)}$ represents the intermediate tensor with C channels, H height, and W width. latent_size denotes the desired size of the latent space.

TABLE 4. Skip connections.

Connection	Description
Skip Connection 1	Between <code>encoder_block3</code> and <code>decoder_block2</code>

3) SKIP CONNECTIONS IN C3DN_AE_FUSION MODEL

In the C3DN_AE_Fusion architecture, as illustrated in Table 4, skip connections are strategically employed to enhance the flow of information between the encoder and decoder blocks. The Skip Connections list delineates these connections and explains their role in enabling feature transfer during the encoding and decoding stages.

During the forward pass, the decoder combines the output of `decoder_block2` with the skip connection from `encoder_block3`. This mechanism improves the model's ability to reconstruct input data accurately by incorporating features from multiple scales. The interconnected nature of the C3DN_AE_Fusion model, as guided by these skip connections, contributes to its effectiveness in acquiring fine-grained details and high-level context during training and inference.

Table 5 presents the output shape during each of the steps of the proposed architecture.

C. RECONSTRUCTION LOSS FOR THE C3DN_AE MODEL

The reconstruction loss of the C3DN_AE model guides the model to learn pertinent representations of the input data and produce reconstructions that closely resemble the original input. The reconstruction loss of the C3DN_AE model can be represented mathematically as the mean squared error (MSE) between the input tensor x and the decoded output tensor decoded :

$$\mathcal{L}_{\text{recon}} = \frac{1}{HW} \sum_{c=1}^3 \sum_{h=1}^H \sum_{w=1}^W (x_{c,h,w} - \text{decoded}_{c,h,w})^2 \quad (8)$$

where c denotes the channel dimension (1 to 3 for RGB channels), and h indicates the height dimension (1 to H). w signifies the width dimension (1 to W), while $x_{c,h,w}$ represents the value of the element at channel c , height h , and width w in the input tensor x . $\text{decoded}_{c,h,w}$ represents the value of the element at channel c , height h , and width w in the decoded output tensor decoded . HW denotes the total number of elements in each channel ($H \times W$).

The reconstruction loss signifies the dissimilarity between the input and the reconstructed output, and the aim is to minimize this loss during training. This loss term measures how well the C3DN_AE model can reconstruct the original input from its learned representations. During training, the goal is to minimize this reconstruction loss, encouraging the model to capture and encode critical features from the input data, ultimately leading to improved reconstruction quality.

D. TRAINING PROCEDURE

To train the proposed model we employed an approach similar to other deep learning-based learning literature as per

TABLE 5. Output shapes at each step in C3DN_AE_Fusion model.

Step	Output Shape
Input Tensor	(1, 3, 16, 112, 112)
Encoder Block 1	Conv3d: (3, 16, 112, 112) → (64, 16, 112, 112) SpatialAttentionModule: (64, 16, 112, 112) (No change)
Encoder Block 2	MaxPool3d: (64, 16, 112, 112) → (64, 16, 56, 56) Conv3d: (64, 16, 56, 56) → (128, 16, 56, 56) SpatialAttentionModule: (128, 16, 56, 56) (No change)
Encoder Block 3	MaxPool3d: (128, 16, 56, 56) → (128, 8, 28, 28) Conv3d: (128, 8, 28, 28) → (256, 8, 28, 28)
Encoder Block 4	Conv3d: (256, 8, 28, 28) → (latent_size, 8, 28, 28)
Decoder Block 1	ConvTranspose3d: (latent_size, 8, 28, 28) → (256, 8, 28, 28)
Decoder Block 2	ConvTranspose3d: (256, 8, 28, 28) → (128, 8, 28, 28) ConvTranspose3d: (128, 8, 28, 28) → (64, 16, 56, 56) ConvTranspose3d: (64, 16, 56, 56) → (out_channels, 16, 112, 112)
Output Tensor	(1, 3, 16, 112, 112)

TABLE 6. Glossary.

Term	Definition
D_{train}	Training video dataset
D_{val}	Validation video dataset
DL_{train}	Training data loader
DL_{val}	Validation data loader
C3DN_AE	Model used for training
optimizer	Optimization algorithm used
loss_function	Loss function used
best_accuracy	Best accuracy achieved during training
save_weights	Flag for saving model weights
losses	List to track losses
early_stopping_threshold	Threshold for early stopping
epochs_without_improvement	Counter for epochs without improvement
best_val_loss	Best validation loss achieved
epoch	Current epoch number
f_{extract}	Function for frame extraction
$root_dir$	Root directory of the video dataset
S_{input}	Input shape of the model

the following generalized form.

$$w \leftarrow w - \eta \nabla L(w) \quad (9)$$

where w represents the model parameters, which are iteratively updated during training, and η (eta) is the learning rate, a hyperparameter that determines the step size for parameter updates. It controls the trade-off between convergence speed and stability. $\nabla L(w)$ is the gradient of the loss function $L(w)$ with respect to the model parameters w . It indicates the direction and magnitude of the steepest ascent of the loss function. The negative gradient is subtracted from w to move in the direction of decreasing loss, thereby updating the parameters to minimize the loss.

During training, the autoencoder aims to minimize the reconstruction error between the input video clips and their reconstructed versions. We employed the mean squared error (MSE) loss function to quantify the reconstruction error. The MSE loss calculates the average squared difference between the corresponding pixels of the input and reconstructed clips.

We used gradient-based optimization techniques, such as the stochastic gradient descent (SGD) or Adam optimizer to optimize the autoencoder. The model was trained on a dataset consisting of normal surveillance video clips without explicit labels for anomalies. The training procedure

follows an unsupervised learning approach, where the autoencoder learns to capture the underlying patterns and structures of normal video clips. The training algorithm consists of the steps mentioned above in the Algorithm 1 block.

E. ANOMALY DETECTION

After the C3DN-based autoencoder was trained, we used it for anomaly detection in surveillance videos. Given a new video clip, we passed it through the trained autoencoder. The reconstruction error between the input clip and its reconstruction was calculated. Anomalies were detected using setting a threshold on the reconstruction error. Clips with higher reconstruction errors were classified as anomalies, indicating the presence of abnormal events. The testing algorithm of the proposed method is provided below.

Glossary items are defined in Table 6.

IV. EXPERIMENTS AND RESULTS

This section presents the experiments and results to assess the performance of the proposed methodology compared to the literature.

A. EXPERIMENTAL SETUP

The proposed method was implemented in the Python 3.7.1 environment, leveraging the PyTorch framework for deep learning tasks. The experimental procedures were conducted on a Windows 10-based personal computer equipped with an Nvidia GeForce RTX-3070 GPU and 32 GB of RAM. We utilized the CUDA toolkit version 11.0 and cuDNN v8.0 to optimise computational performance. These software and hardware configurations were set to ensure computational efficiency and adherence to best practices in deep learning research, contributing to the reliability of the experimental outcomes.

B. DATASETS

The proposed method was evaluated on four publically available state-of-the-art benchmark datasets: UCSD Pedestrian 1 [28], UCSD Pedestrian 2 [29], CUHK Avenue [31] and ShanghaiTech [30]. Fig. 3 presents a grid with four rows

Algorithm 1 Training Algorithm for the Proposed Approach

Procedure PROPOSED APPROACH ($S_{\text{input}} = (1, 3, 16, 112, 112)$)

Input ROOT DIRECTORY OF THE VIDEO DATASET ($root_dir$)

Output TRAINED MODEL WEIGHTS

Step 1: Frame Extraction Extract frames using the function f_{extract}

Step 2: Dataset Initialization Initialize the video dataset: $D_{\text{train}}, D_{\text{val}}$

Step 3: Transformation Pipeline Define the transformation pipeline using `transforms.Compose`

Step 4: Training Data Loader Initialize the training data loader: DL_{train}

Step 5: Model Initialization Instantiate the model: C3DN_AE_Fusion

Step 6: Optimizer and Loss Function Define the optimizer (`optimizer`) and the loss function (`loss_function`)

Step 7: Variable Initialization Initialize the tracking variables: `best_accuracy`, `save_weights`, `losses`

Step 8: Validation Data Loader Initialize the validation data loader: DL_{val}

Step 9: Early Stopping Variables Initialize the early stopping variables: `early_stopping_threshold`, `epochs_without_improvement`, `best_val_loss`

Step 10: Training Loop for each epoch do

Step 10a: Learning Rate Adjustment Adjust the learning rate of the optimizer

Step 10b: Training Phase Set the model to the training mode Iterate over the batches in the training data loader Calculate the loss, perform backpropagation, and update the model's parameters Track the running loss and the total number of processed samples

Step 10c: Validation Phase Set the model to the evaluation mode Calculate the validation loss for each batch in the validation data loader

Step 10d: Model Saving Save the model's state dictionary if the validation loss improves

Step 10e: Early Stopping Check Check for early stopping based on the validation loss **if there is no improvement for a specified number of epochs then**

 | Stop the training process

end

end

Step 11: Training Completion Training completes

and eight columns, displaying example image frames of normal and anomalous events from these datasets. The first four columns of the figure present image frames captured from UCSDPedestrian 1 [28], depicting normal pedestrian activities in a specific environment. The next four columns display frames from the same dataset, portraying anomalous events. The second grid row displays examples from UCSD Pedestrian 2 [29], depicting pedestrian behavior in a different context. The third row features frames from the CUHK Avenue [31] dataset, offering insights into crowd scenes and

Algorithm 2 Anomaly Detection Test Procedure

Input : Test Videos directory $Video_dir$, Saved weights path $saved_weights_path$

Output: Anomalous frames

`set_seed(seed)`

Step 1: Transform frames using predefined transformation pipeline

Step 2: Instantiate model

Step 3: Load model weights from $saved_weights_path$

Step 4: Compute losses and reconstructions using model and transformed segments

Step 5: Compute mean and standard deviation of losses: $\mu_{\text{loss}}, \sigma_{\text{loss}}$

Step 6: Compute threshold as p th percentile of losses: $threshold = \text{percentile}(losses, p)$

Step 7: Determine anomalous segments with losses exceeding threshold: $anomalous_segments = \{i | loss_i > threshold\}$

Step 8: Determine anomalous frames based on segment indices: $anomalous_frames = \{(i \cdot frame_size, (i + 1) \cdot frame_size - 1) | i \in anomalous_segments\}$

their potential anomalies. Finally, the fourth row presents examples from ShanghaiTech [30], highlighting anomalous events in distinctly urban settings. Table 7 presents a comparison of various datasets used for abnormal event detection in video surveillance. Each dataset is characterized by its total frame count, video count, training set size, testing set size, and the number of abnormal events present in the dataset. The datasets vary in terms of scale and composition, affording valuable resources for research and development in abnormal event detection.

1) UCSD PEDESTRIAN 1

This dataset documented in the paper referenced as [28], comprises 14,000 frames distributed among 70 video sequences. The dataset is divided into a training set with 34 video sequences and a testing set with 36 video sequences. The dataset includes 40 abnormal events, such as small carts, bikers, and mini trucks serving as a benchmark for abnormal event detection in video surveillance.

2) UCSD PEDESTRIAN 2

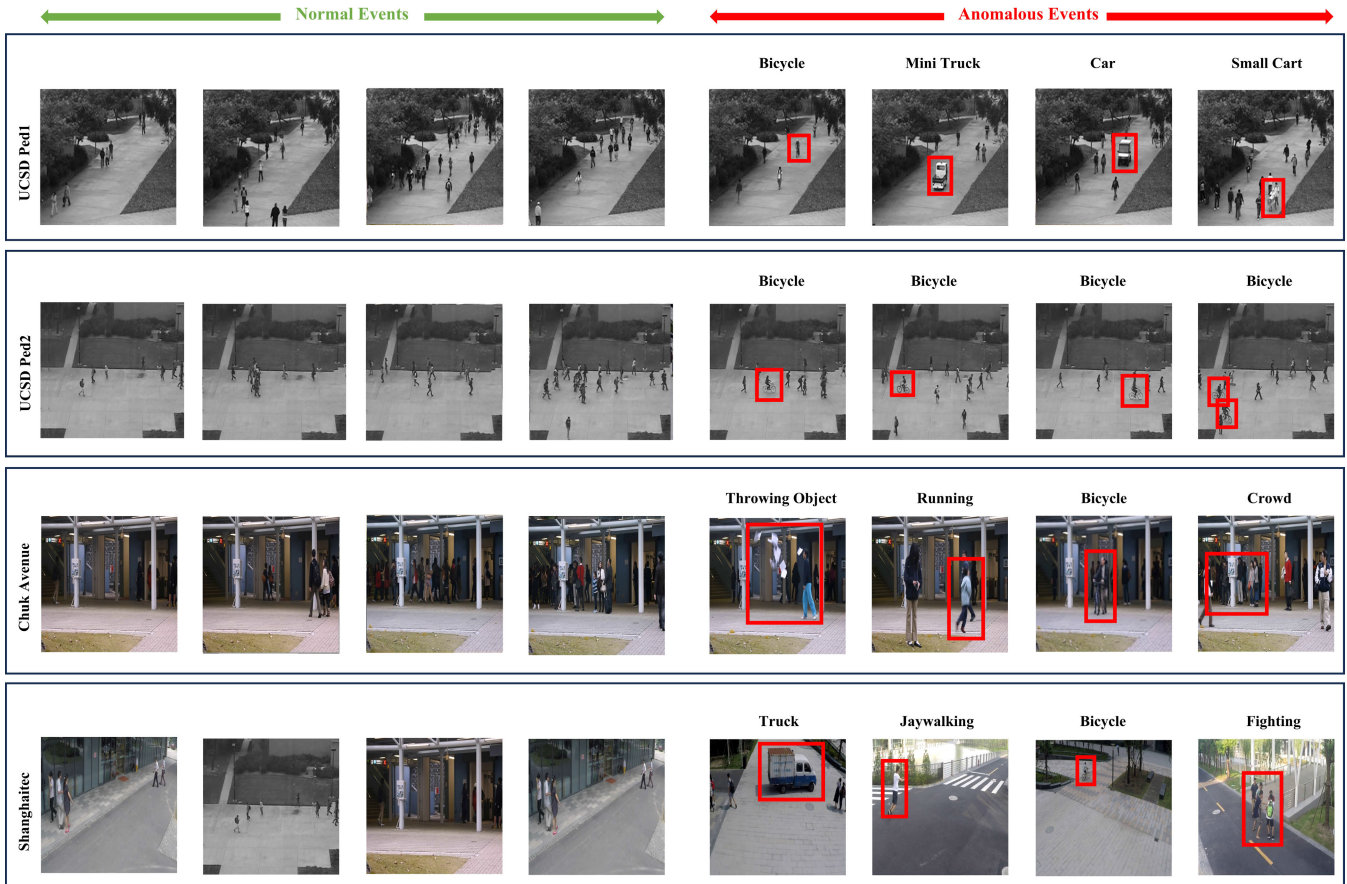
This dataset, referenced as [29], comprises 4,560 frames from 28 video sequences split into a training set with 16 video sequences and a testing set with 12 video sequences. The dataset specifically focuses on the anomaly of bikers with 12 instances of such events. Additionally, UCSD Pedestrian 2 is useful for evaluating algorithms that detect abnormal events involving bicycles.

3) CUHK AVENUE

The CUHK Avenue dataset, in [31], comprises 30,652 frames from 37 video sequences consisting of a training set with 16 video sequences and a testing set with 21 video sequences.

TABLE 7. Comparison of datasets for abnormal event detection in video surveillance.

Dataset	Total Frames	Video Count	Training Set	Testing Set	Abnormal Events	Types of Anomalies
UCSD Pedestrian 1 [28]	14,000	70	34	36	40	Small carts, bikers, mini trucks etc.
UCSD Pedestrian 2 [29]	4,560	28	16	12	12	Bikers
CUHK Avenue [31]	30,652	37	16	21	47	Trowing objects, running, crowd etc.
ShanghaiTech [30]	317,398	437	330	107	130	Trucks, jaywalking, fighting etc.


FIGURE 3. Example image frames of normal and anomalous events from surveillance video datasets. Rows represent UCSD Pedestrian 1 [28], UCSD Pedestrian 2 [29], CUHK Avenue [31], and ShanghaiTech [30], respectively.

The dataset covers 47 abnormal events, including throwing objects, running, and crowd-related anomalies. The CUHK Avenue dataset was designed to evaluate the performance of abnormal event detection methods.

4) SHANGHAITECH

From [30], this dataset is a comprehensive resource for abnormal event detection in video surveillance, encompassing 317,398 frames from 437 video sequences, with a training set containing 330 video sequences and a testing set containing 107 video sequences. The ShanghaiTech dataset includes 130 abnormal events, such as trucks, jaywalking, and fighting, and serves as a valuable benchmark for developing and evaluating abnormal event detection algorithms in crowded scenes.

Fig. 4 presents the 3D uniform manifold approximation and projection (UMAP) projection to gain visual insight into

the overall structure and relationships between the video sequences in UCSD Pedestrian 1 [28], UCSD Pedestrian 2 [29], CUHK Avenue [31], and ShanghaiTech [30]. The UMAP projection is helpful for exploratory data analysis, anomaly visualization, and understanding the distribution and patterns within the video dataset.

The proposed model is trained exclusively on videos depicting normal behavior to learn the standard patterns of data without any anomalies. This model was tested on normal and anomalous videos to assess its performance.

C. ANOMALY SCORE

In surveillance video anomaly detection, an anomaly score indicates the degree to which a specific region, frame, or event in a video deviates from what is considered normal or expected. In this context, the goal is to identify unusual or anomalous behavior, events, or objects distinct from the

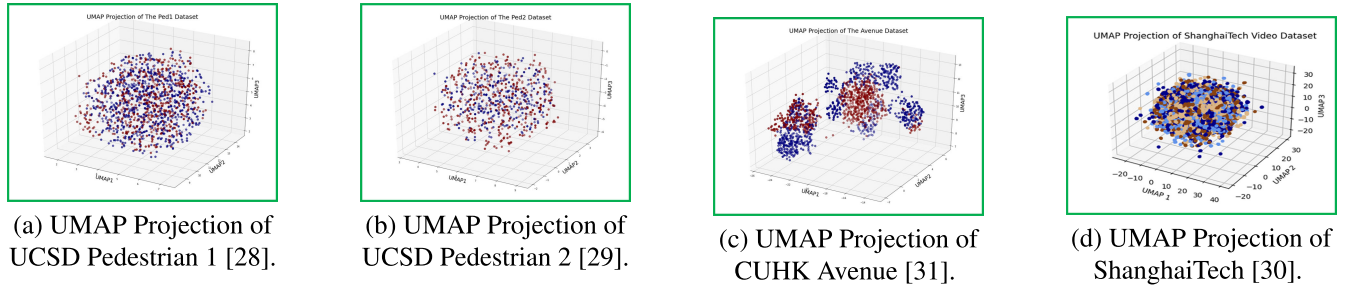


FIGURE 4. Uniform manifold approximation and projection (UMAP) data visualization of the surveillance video anomaly detection datasets used in the proposed study.

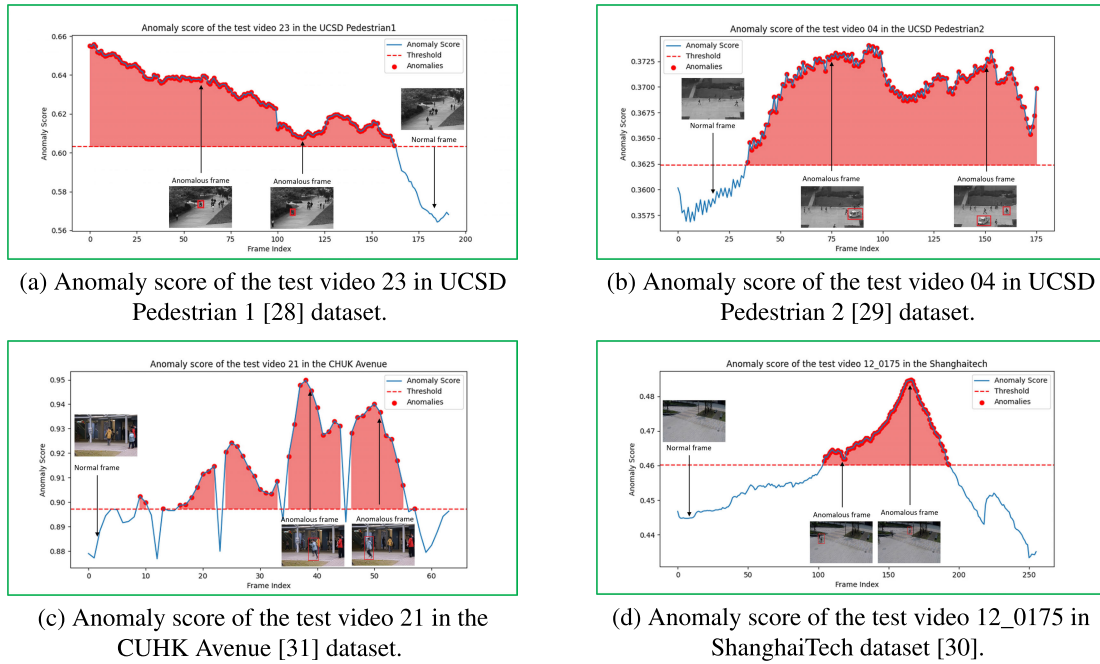


FIGURE 5. Anomaly score visualization of the proposed method over various datasets. The red bounding boxes enclose anomalies.

typical patterns of the monitored environment. Fig. 5 presents the anomaly scores of the proposed method along video frames from four datasets. Note that the anomaly score which is marked with a blue line across the y-axis and frames on the x-axis. The blue line fluctuates between anomalous and normal events in a given test video, confirming that the proposed method can distinguish abnormal events from a vast number of normal events. Fig. 5 (a) visualizes the anomaly score of the test video 23 in UCSD Pedestrian 1 dataset [28]; one frame (walking pedestrians) of the normal frames and two frames (small cart) of the anomalous frames are presented too. The anomaly starts from the beginning of the video and remains unless the anomaly score falls below the threshold, indicating the successful anomaly detection by the proposed model.

Fig. 5 (b) presents the anomaly score of the test video 05 in the UCSD Pedestrian 2 dataset [29]. Two anomalous frames and one normal frame are depicted. The normal frame displays walking pedestrians, whereas the first anomalous frame depicts a white vehicle and the second anomalous frame shows town anomalies, a white vehicle and a cyclist.

The anomaly score increases when unexpected objects appear in the scene.

Fig. 5 (c) presents the anomaly score of the test video 12_0175 in the CUHK Avenue dataset [31]. Abnormal frames are highlighted when the anomaly score exceeds a threshold.

Fig. 5 (d) shows the anomaly score of the test video 12_0175 in the ShanghaiTec dataset [30]. The anomaly score rises above the threshold when an unexpected cyclist appears on the scene. The Anomaly score falls below the threshold when the cyclist exits from the scene.

D. QUANTITATIVE ANALYSIS

We employed the receiver operating characteristic (ROC) curve [56] to demonstrate the trade-off between the true positive and the false positive rates to assess the performance of the proposed method quantitatively.

In the context of video anomaly detection, the ROC curve is a graphical representation that clarifies the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across decision thresholds employed for anomaly

TABLE 8. Frame-based AUC comparison of our (proposed) framework and existing approaches on UCSD Pedestrian 1, UCSD Pedestrian 2, CUHK Avenue, and ShanghaiTech datasets.

Methods	UCSD Pedestrian 1	UCSD Pedestrian 2	CUHK Avenue	ShanghaiTech
Lu <i>et al.</i> [31]	91.8	—	80.9	—
Radu <i>et al.</i> [42]	68.4	82.2	80.6	—
Zhou <i>et al.</i> [43]	83.5	94.9	86.1	—
Romero <i>et al.</i> [44]	—	—	—	73.4
Che <i>et al.</i> [45]	—	—	89.6	74.7
Mahmudul <i>et al.</i> [23]	81.0	90.0	70.2	60.9
Weixin <i>et al.</i> [46]	75.5	88.1	77.0	—
Weixin <i>et al.</i> [47]	—	92.2	81.7	68.0
Dong <i>et al.</i> [39]	—	94.1	83.3	71.2
Hyunjong <i>et al.</i> [48]	—	90.2	82.8	69.8
Wen <i>et al.</i> [33]	83.1	95.4	85.1	72.8
Yao <i>et al.</i> [49]	84.5	95.9	85.9	73.5
Yiwei <i>et al.</i> [50]	86.2	96.0	85.7	—
Zhou <i>et al.</i> [51]	83.9	96.0	86.0	—
Yuan <i>et al.</i> [40]	86.7	96.4	86.7	—
Pelvan <i>et al.</i> [41]	82.0	—	71.6	54.7
Saypadith <i>et al.</i> [52]	85.3	95.7	86.9	73.0
Cruz <i>et al.</i> [53]	71.1	87.3	83.3	—
Dilek <i>et al.</i> [54]	83.5	—	—	—
Li <i>et al.</i> [55]	93.2	96.3	87.1	—
Ours	94.6	96.7	84.7	74.8

detection. The primary purpose of the ROC curve is to assess the performance of an anomaly detection model or system.

The ROC-related components and pertinent formulas for video anomaly detection are elucidated below:

True Positive Rate (TPR), often referred to as Sensitivity or Recall, is computed as the proportion of correctly identified true positive predictions in relation to the total number of actual positive instances (anomalies):

$$TPR = \frac{TP}{TP + FN} \quad (10)$$

where, TP denotes True Positives, signifies the number of accurately detected anomalies, whereas FN represents False Negatives denotes the count of actual anomalies that the model failed to identify.

False Positive Rate (FPR) is determined by the ratio of falsely predicted positive outcomes to the total number of actual negative instances (non-anomalies):

$$FPR = \frac{FP}{FP + TN} \quad (11)$$

In this context, FP corresponds to False Positives, indicates the number of non-anomalies incorrectly classified as anomalies, whereas TN symbolizes True Negatives, representing the number of correctly classified non-anomalies.

ROC Curve: The ROC curve is constructed by plotting the TPR against the FPR for diverse decision thresholds. Each data point on the curve defines the model's performance at a particular threshold setting. Typically, the curve commences at the origin point (0,0) and terminates at (1,1).

$$\begin{aligned} &\text{ROC(FPR, TPR)} \\ &= \{(FPR_1, TPR_1), (FPR_2, TPR_2), \dots, (FPR_n, TPR_n)\}, \end{aligned} \quad (12)$$

Area Under the ROC Curve (AUC-ROC): The AUC-ROC is a scalar metric quantifying the overall effectiveness

of an anomaly detection model. The AUC formula,

$$AUC = \int_0^1 TPR(fpr) dfpr, \quad (13)$$

represents the integral of the True Positive Rate (TPR) as a function of the False Positive Rate (FPR) across different threshold values. In this formula: - $TPR(fpr)$ represents the True Positive Rate, also known as Sensitivity or Recall, as a function of the False Positive Rate (FPR). - The integral symbol $\int$ denotes the definite integral, which calculates the area under the ROC curve. - The integration is performed over the entire range of FPR from 0 to 1.

The AUC presents the likelihood that the model ranks a randomly selected anomaly higher than a randomly chosen non-anomaly. The AUC-ROC score varies between 0 and 1, with higher values representing superior discriminatory power. An AUC-ROC of 0.5 indicates random performance.

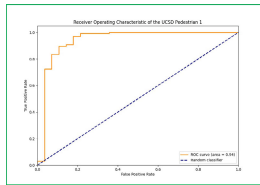
In summary, the ROC curve and AUC-ROC are useful tools for evaluating and contrasting the performance of the proposed anomaly detection model. Models with higher AUC-ROC scores are more proficient in distinguishing anomalies from non-anomalies, considering the required equilibrium between TPR and FPR. Fig. 6 represents ROC curves for the UCSD Pedestrian 1 [28], UCSD Pedestrian 2 [29], ShanghaiTech [30], and CUHK Avenue [31] datasets using different supervised and unsupervised techniques.

Table 8 presents a comprehensive comparative analysis of the frame-based Area Under the Curve (AUC) scores, a standard metric for assessing the performance of anomaly detection methods, across four benchmark datasets: UCSD Pedestrian 1, UCSD Pedestrian 2, CUHK Avenue, and ShanghaiTech. Each row in the table corresponds to a distinct anomaly detection approach, including the proposed method, labeled as "Ours."

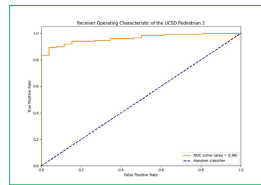
The Ref. column lists the sources of the methods under consideration, encompassing a diverse array of existing

TABLE 9. Proposed model efficiency in terms of parameter count (in millions), model size (MBs), and time complexity per sequence (ms).

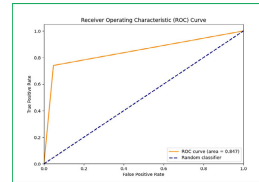
Method	Parameter Count (M)	Model Size (MB)	Time Complexity/Sequence (ms)
VGG19+ multi-layer BD-LSTM [57]	143.00	605.50	220
Inception V3+ multi-layer BD-LSTM [57]	23.00	148.50	180
VGG19 + multi-layer BD-LSTM [57]	25.00	143.00	200
ADSV [58]	14.17	54.10	260
EADN [16]	14.14	53.90	200
Ours	5.760	23.05	160



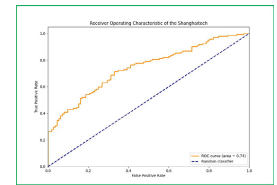
(a) ROC of UCSD Pedestrian 1 [28] dataset.



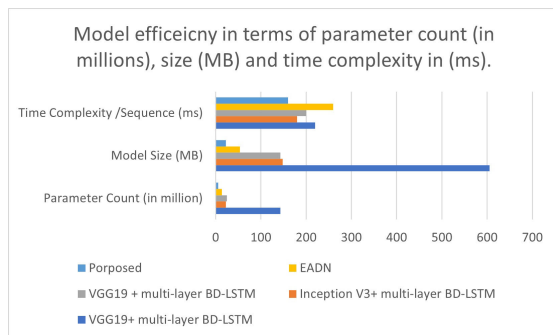
(b) ROC of UCSD Pedestrian 2 [29] dataset.



(c) ROC of the CUHK Avenue [31] dataset.



(d) ROC of ShanghaiTech [30] dataset.

FIGURE 6. Receiver operating characteristic (ROC) curves of the four datasets used in this research.**FIGURE 7.** Graphical visualization of model efficiency values in Table 9.

techniques in the domain of video anomaly detection. The subsequent columns, namely “UCSD Pedestrian 1,” “UCSD Pedestrian 2,” “CUHK Avenue,” and “ShanghaiTech,” list the AUC values achieved by each method on these datasets. Notably, in cases where a particular method was not evaluated on a specific dataset, we have indicated this with the symbol “—.”

The understanding of the AUC values is crucial to evaluate the comparative effectiveness of these techniques. A higher AUC score indicates better performance in distinguishing between normal and anomalous frames in video sequences. Therefore, methods marked with bold AUC values in a dataset are the top performers for that dataset.

The proposed method, identified as “Ours,” stands out prominently in the comparative analysis. It achieves the highest AUC values on the UCSD Pedestrian 1, UCSD Pedestrian 2, and ShanghaiTech datasets, scoring 94.6, 96.7, and 74.8, respectively. For CUHK Avenue, the proposed model archived an acceptable performance of 84.7. These results demonstrate the robustness and effectiveness of the proposed approach in detecting anomalies across diverse scenarios and datasets.

In conclusion, Table 8 offers an in-depth and data-driven perspective on the comparative performance of

various anomaly detection methods. The representation reinforces the strength of the proposed method, “Ours,” revealing that it consistently outperforms existing approaches on multiple datasets. Furthermore, the table offers transparency regarding instances where certain methods were not assessed on specific datasets, highlighting the dataset-specific nature of these evaluations and directions for future research.

E. MODEL EFFICIENCY

In addition to achieving an impressive accuracy rate, our video anomaly detector operates at a remarkable speed, processing video streams at a high frame rate of 111 frames per second (fps). To comprehensively understand the scale and resource requirements of our proposed model, we present critical metrics in Table 9. This table outlines the parameter count measured in millions and its corresponding model size in megabytes (MB). Our model comprises approximately 5.76 million learnable parameters, indicative of its complexity and capacity to acquire intricate patterns within the data. Furthermore, its practical feasibility is highlighted by its modest model size of approximately 23.05 MB, contributing to a significantly low time complexity of only 160 ms. These metrics underscore the balance between model expressiveness and resource efficiency, making our model a suitable choice for detecting anomalies in surveillance videos. Fig. 7 presents the graphical visualization of the values in Table 9.

In summary, our proposed method utilizes a C3DN-based autoencoder for surveillance video anomaly detection. The autoencoder is trained to reconstruct normal video clips and detects anomalies based on the reconstruction errors. The method operates unsupervised, eliminating the need for labelled training data.

V. CONCLUSION AND FUTURE WORK

This study introduces a lightweight, and cost-effective approach using a 3DCNN-based autoencoder to exploit

spatial and temporal features for anomaly detection in surveillance videos with promising accuracy. Our unsupervised learning end-to-end scheme was tested on four benchmark datasets, achieving state-of-the-art performance. The model captures inherent patterns and regularities by training the autoencoder on normal video data, leading to a meaningful latent space representation that serves as a basis for anomaly detection. During testing, the higher reconstruction errors observed for anomalous frames highlight deviations from learned patterns, making the proposed method a promising solution for automated surveillance anomaly detection. This research contributes to the field by offering a robust end-to-end approach with potential applications in various industries requiring continuous monitoring.

Future work will assess fine-tuning the model for specific surveillance scenarios and real-world implementation to validate its effectiveness further, ultimately enhancing security and safety measures across different domains. We also aim to embed an anomaly recognition stream in our model to improve its practicality and usability in real-time applications.

REFERENCES

- [1] J. Guo, P. Zheng, and J. Huang, "Efficient privacy-preserving anomaly detection and localization in bitstream video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 3268–3281, Sep. 2020.
- [2] Z. Zheng, W. Liu, R. Liu, L. Wang, L. Mao, Q. Qiu, and G. Ling, "Anomaly detection of metro station tracks based on sequential updatable anomaly detection framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7677–7691, Nov. 2022.
- [3] M. Khan, A. E. Saddik, W. Gueaieb, G. De Masi, and F. Karray, "VD-Net: An edge vision-based surveillance system for violence detection," *IEEE Access*, vol. 12, pp. 43796–43808, 2024.
- [4] S. Hong, S. Ahn, Y. Jo, and S. Park, "Making anomalies more anomalous: Video anomaly detection using a novel generator and destroyer," *IEEE Access*, vol. 12, pp. 36712–36726, 2024.
- [5] H. Liu, S. Chen, and N. Kubota, "Intelligent video systems and analytics: A survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1222–1233, Aug. 2013.
- [6] J. Lin, D. Liu, H. Yang, H. Li, and F. Wu, "Convolutional neural network-based block up-sampling for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3701–3715, Dec. 2019.
- [7] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 566–576, Feb. 2020.
- [8] L. A. Thomaz, E. Jardim, A. F. da Silva, E. A. B. da Silva, S. L. Netto, and H. Krim, "Anomaly detection in moving-camera video sequences using principal subspace analysis," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 3, pp. 1003–1015, Mar. 2018.
- [9] Y. Ou, Z. Chen, and F. Wu, "Multimodal local-global attention network for affective video content analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 1901–1914, May 2021.
- [10] B. Yang, J. Cao, N. Wang, and X. Liu, "Anomalous behaviors detection in moving crowds based on a weighted convolutional autoencoder-long short-term memory network," *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 4, pp. 473–482, Dec. 2019.
- [11] Y. Liu, J. Ren, J. Xu, X. Bai, R. Kaur, and F. Xia, "Multiple instance learning for cheating detection and localization in online examinations," *IEEE Trans. Cogn. Develop. Syst.*, early access, Jan. 5, 2024, doi: 10.1109/TCDS.2024.3349705.
- [12] S. Yan, J. S. Smith, W. Lu, and B. Zhang, "Abnormal event detection from videos using a two-stream recurrent variational autoencoder," *IEEE Trans. Cogn. Develop. Syst.*, vol. 12, no. 1, pp. 30–42, Mar. 2020.
- [13] N. Li, F. Chang, and C. Liu, "A self-trained spatial graph convolutional network for unsupervised human-related anomalous event detection in complex scenes," *IEEE Trans. Cogn. Develop. Syst.*, vol. 15, no. 2, pp. 737–750, Jun. 2023.
- [14] Y. Liu, J. Liu, J. Lin, M. Zhao, and L. Song, "Appearance-motion united auto-encoder framework for video anomaly detection," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 5, pp. 2498–2502, May 2022.
- [15] Y. Liu, J. Liu, M. Zhao, S. Li, and L. Song, "Collaborative normality learning framework for weakly supervised video anomaly detection," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 5, pp. 2508–2512, May 2022.
- [16] S. Ul Amin, M. Ullah, M. Sajjad, F. A. Cheikh, M. Hijji, A. Hijji, and K. Muhammad, "EADN: An efficient deep learning model for anomaly detection in videos," *Mathematics*, vol. 10, no. 9, p. 1555, May 2022.
- [17] X. Mo, V. Monga, R. Bala, and Z. Fan, "Adaptive sparse representations for video anomaly detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 4, pp. 631–645, Apr. 2014.
- [18] Y. Fan, G. Wen, D. Li, S. Qiu, M. D. Levine, and F. Xiao, "Video anomaly detection and localization via Gaussian mixture fully convolutional variational autoencoder," *Comput. Vis. Image Understand.*, vol. 195, Jun. 2020, Art. no. 102920.
- [19] N. Li, F. Chang, and C. Liu, "Spatial-temporal cascade autoencoder for video anomaly detection in crowded scenes," *IEEE Trans. Multimedia*, vol. 23, pp. 203–215, 2021.
- [20] V.-T. Le and Y.-G. Kim, "Attention-based residual autoencoder for video anomaly detection," *Int. J. Speech Technol.*, vol. 53, no. 3, pp. 3240–3254, Feb. 2023.
- [21] R. Leyva, V. Sanchez, and C.-T. Li, "Video anomaly detection with compact feature sets for online performance," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3463–3478, Jul. 2017.
- [22] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1933–1941.
- [23] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 733–742.
- [24] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *Proc. Int. Symp. Neural Netw.*, May 2017, pp. 189–196.
- [25] U. Michelucci, "An introduction to autoencoders," 2022, *arXiv:2201.03898*.
- [26] X. Zhang, J. Fang, B. Yang, S. Chen, and B. Li, "Hybrid attention and motion constraint for anomaly detection in crowded scenes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 5, pp. 2259–2274, May 2023.
- [27] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4489–4497.
- [28] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 909–926, May 2008.
- [29] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and Bayesian regression," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2160–2177, Apr. 2012.
- [30] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 589–597.
- [31] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 FPS in MATLAB," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2720–2727.
- [32] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," 2015, *arXiv:1511.05440*.
- [33] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection—A new baseline," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6536–6545.
- [34] A. Munawar, P. Vinayavekhin, and G. De Magistris, "Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 1017–1025.
- [35] Y. Zhong, X. Chen, Y. Hu, P. Tang, and F. Ren, "Bidirectional spatio-temporal feature learning with multiscale evaluation for video anomaly detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 12, pp. 8285–8296, Dec. 2022.
- [36] R. Nawaratne, D. Alahakoon, D. De Silva, and X. Yu, "Spatiotemporal anomaly detection using deep learning for real-time video surveillance," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 393–402, Jan. 2020.

- [37] N. Li and F. Chang, "Video anomaly detection and localization via multivariate Gaussian fully convolution adversarial autoencoder," *Neuro-computing*, vol. 369, pp. 92–105, Dec. 2019.
- [38] Y. Chang, Z. Tu, W. Xie, and J. Yuan, "Clustering driven deep autoencoder for video anomaly detection," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, Aug. 2020, pp. 329–345.
- [39] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. Van Den Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1705–1714.
- [40] H. Yuan, Z. Cai, H. Zhou, Y. Wang, and X. Chen, "TransAnomaly: Video anomaly detection using video vision transformer," *IEEE Access*, vol. 9, pp. 123977–123986, 2021.
- [41] S. Ö. Pelvan, B. Can, and H. Ozkan, "A hierarchical approach for improved anomaly detection in video surveillance," *IEEE Access*, vol. 11, pp. 101644–101665, 2023.
- [42] R. T. Ionescu, S. Smeureanu, B. Alexe, and M. Popescu, "Unmasking the abnormal events in video," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2914–2922.
- [43] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, "AnomalyNet: An anomaly detection network for video surveillance," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 10, pp. 2537–2550, Oct. 2019.
- [44] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh, "Learning regularity in skeleton trajectories for anomaly detection in videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11988–11996.
- [45] C. Sun, Y. Jia, Y. Hu, and Y. Wu, "Scene-aware context reasoning for unsupervised abnormal event detection in videos," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 184–192.
- [46] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional LSTM for anomaly detection," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 439–444.
- [47] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked RNN framework," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 341–349.
- [48] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14360–14369.
- [49] Y. Yang, D. Zhan, F. Yang, X.-D. Zhou, Y. Yan, and Y. Wang, "Improving video anomaly detection performance with patch-level loss and segmentation map," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2020, pp. 1832–1839.
- [50] Y. Lu, K. M. Kumar, S. S. Nabavi, and Y. Wang, "Future frame prediction using convolutional VRNN for anomaly detection," in *Proc. 16th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Sep. 2019, pp. 1–8.
- [51] J. T. Zhou, L. Zhang, Z. Fang, J. Du, X. Peng, and Y. Xiao, "Attention-driven loss for anomaly detection in video surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 12, pp. 4639–4647, Dec. 2020.
- [52] S. Saypadith and T. Onoye, "An approach to detect anomaly in video using deep generative network," *IEEE Access*, vol. 9, pp. 150903–150910, 2021.
- [53] E. Cruz-Esquivel and Z. J. Guzman-Zavaleta, "An examination on autoencoder designs for anomaly detection in video surveillance," *IEEE Access*, vol. 10, pp. 6208–6217, 2022.
- [54] E. Dilek and M. Dener, "Enhancement of video anomaly detection performance using transfer learning and fine-tuning," *IEEE Access*, vol. 12, pp. 73304–73322, 2024.
- [55] D. Li, X. Nie, X. Li, Y. Zhang, and Y. Yin, "Context-related video anomaly detection via generative adversarial network," *Pattern Recognit. Lett.*, vol. 156, pp. 183–189, Apr. 2022.
- [56] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
- [57] W. Ullah, A. Ullah, I. U. Haq, K. Muhammad, M. Sajjad, and S. W. Baik, "CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks," *Multimedia Tools Appl.*, vol. 80, no. 11, pp. 16979–16995, May 2021.
- [58] S. Ul Amin, Y. Kim, I. Sami, S. Park, and S. Seo, "An efficient attention-based strategy for anomaly detection in surveillance video," *Comput. Syst. Sci. Eng.*, vol. 46, no. 3, pp. 3939–3958, 2023.



ASIM NIAZ received the B.S. degree in electrical (computer) engineering from the COMSATS Institute of Information Technology, Islamabad, Pakistan, in 2016, and the M.S. degree in computer science and engineering from Chung-Ang University, Seoul, South Korea, in 2020.

He was a Research Assistant with the Visual Image Media Laboratory, Seoul. Later, he visited INRIA, Sophia Antipolis, France, followed by his research internship at Ericsson Canada. He is currently a Research Assistant with the AI Vision Laboratory with the Department of Computer Science and Engineering, Chung-Ang University. His research interests include action recognition, anomaly detection in images and videos, object detection, medical image analysis, image generation, and remote sensing.



SAREER UL AMIN was born in Swabi, Pakistan. He received the B.S. degree in computer science from Islamia College University Peshawar, Pakistan, and the M.S. degree in computer science and engineering from Chung-Ang University, Seoul, South Korea.

His research interests include computer vision, deep learning, machine learning, medical image analysis, anomaly detection in surveillance video, human actions and activity recognition, sequence learning, image and video analytics, and deep learning for multimedia understanding.



SHAFIULLAH SOOMRO received the Bachelor of Engineering (B.E.) degree from QUEST, Nawabshah, Sindh, Pakistan, in 2008, the Master of Engineering (M.E.) degree from MUET, Jamshoro, Sindh, in 2014, and the Ph.D. degree in computer science from Chung-Ang University, Seoul, South Korea, in 2018.

He is currently a Postdoctoral Researcher with the Department of Computer Science and Media Technology, Linnaeus University, Sweden. His research interests include motion tracking, object segmentation, and 3D image recognition.



HAMZA ZIA received the B.S. degree in computer science from COMSATS University Islamabad, in 2021. He is currently pursuing the master's degree in computer science with Chung-Ang University, Seoul, South Korea. His research interests include anomaly detection, image segmentation, and object detection.



KWANG NAM CHOI received the B.S. and M.S. degrees from the Department of Computer Science, Chung-Ang University, Seoul, South Korea, in 1988 and 1990, respectively, and the Ph.D. degree in computer science from the University of York, U.K., in 2002.

He is currently a Professor with the School of Computer Science and Engineering, Chung-Ang University. His current research interests include motion tracking, object categorization, and 3D image recognition.

...