IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 April 2024, accepted 16 May 2024, date of publication 23 May 2024, date of current version 31 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404553

RESEARCH ARTICLE

Enhancement of Video Anomaly Detection Performance Using Transfer Learning and Fine-Tuning

ESMA DİLEK AND MURAT DENER
Department of Information Security Engineering, Graduate School of Natural and Applied Sciences, Gazi University, 06560 Ankara, Turkey
Corresponding author: Murat Dener (muratdener@gazi.edu.tr)

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant Number 123E065.

ABSTRACT The use of surveillance cameras is a common solution that addresses the need to provide security and manage urban traffic that arises due to the increasing population in cities. As the number of surveillance cameras rises, video streams that create big data are recorded. The analysis of video streams collected from those traffic surveillance cameras and the automatic detection of unusual, suspicious events, as well as a range of harmful activities, have become crucial because it is impossible to observe, analyze, and comprehend the contents of these movies using human labor. Recent studies have shown that deep learning (DL)-based artificial intelligence (AI) techniques, particularly machine learning (ML) systems, are used in video anomaly detection (VAD) studies. In this study, an efficient frame-level VAD method is proposed based on transfer learning (TL) and fine-tuning (FT) approach, and anomalies were detected using 20 popular convolutional neural network (CNN)-based DL models where variants of VGG, Xception, MobileNet, Inception, EfficientNet, ResNet, DenseNet, NASNet, and ConvNeXt base models were trained via the TL and FT approaches. The proposed approach was tested using the CUHK Avenue, UCSD Ped1, and UCSD Ped2 datasets, and the performances of the models were measured via area under curve (AUC), accuracy, precision, recall, and F1-score metrics. The highest AUC scores measured were 100%, 100%, and 98.41% for the UCSD Ped1, UCSD Ped2, and CUHK Avenue datasets, respectively. Compared to existing techniques in the literature, experimental results show that the suggested method offers state-of-the-art VAD performance.

INDEX TERMS CUHK avenue, deep learning, fine-tuning, transfer learning, UCSD Ped1, UCSD Ped2, video anomaly detection.

I. INTRODUCTION

The use of traffic cameras in intelligent transportation applications is becoming more common as information and communication technologies advance for the effective operation of transportation infrastructures. In particular, cameras are used to monitor the traffic flows 24/7 in road transportation networks to detect incidents that put traffic safety at risk and to take the necessary actions. The instantaneous evaluation of information, getting updates, and taking necessary actions in intercity and urban road traffic observation systems are no longer efficient via traditional video surveillance methods that are based on human-eye tracking due to the large amount of stream data. Therefore, there is a need to develop innovative methods using technologies that can automatically detect unusual and suspicious situations that pose a risk to traffic safety from live video streams, such as artificial intelligence (AI), rather than manual detection by traffic operators.

Currently, intelligent systems using computer vision (CV) methods are used for the monitoring of traffic flow and automatic detection of anomalies occurring across road networks. The ability of these systems to make evaluations using sequential video images reduces the errors that occur during

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

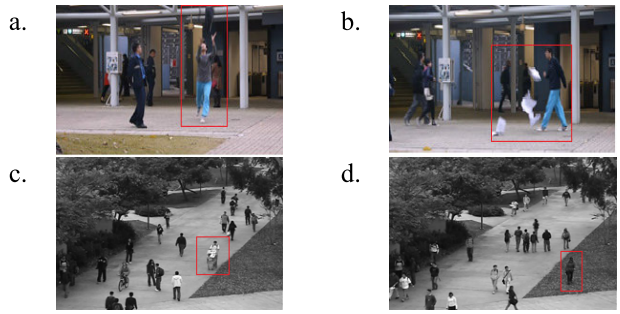


FIGURE 1. Examples of anomaly scenes a, b. Object Throwing Man (CUHK Avenue Dataset) c. Stroller (UCSD Ped1 Dataset) d. Person walking on grass (UCSD Ped1 Dataset).

human-eye analysis and increases the detection accuracy. These schemes, which use AI techniques, allow the establishment of highly successful warning systems for traffic control center operators [1].

In the context of anomaly detection, it is observed that the surveillance cameras are categorized to detect issues such as abnormal human movements, crowds, vehicle traffic, unusual objects, and unusual interactions between people and objects [2]. Some anomalies encountered in the surveillance cameras are shown in Figure 1.

Previously, anomaly detection in videos was carried out manually by utilizing handcrafted and conventional image processing techniques, but now, powerful machine learning (ML) methods are used. Deep learning (DL)-based architectures, such as convolutional neural network (CNN), deep neural network (DNN), convolutional 3D (Conv3D), generative adversarial network (GAN), long short-term memory (LSTM), and autoencoders (AEs), are widely used in video anomaly detection (VAD) studies in the literature thanks to the developments in ML techniques [2], [3].

After reviewing the literature studies, a frame-level VAD method, including transfer learning (TL) and fine-tuning (FT) approaches, which have recently gained popularity in VAD applications, was developed in this study. The following are this article's primary contributions:

1) This study explores the effectiveness of the use of Keras applications as base models in VAD, where 20 popular DL models with pre-trained weights were investigated.
2) Feature extracting capabilities of Keras applications that have pre-trained weights were utilized, which yielded promising detection rates with TL and FT approaches for real-time VAD applications.
3) This study proposes a supervised learning–based VAD approach that presents state-of-the-art (SOTA) frame-level VAD performance for the UCSD Ped1 [3], UCSD Ped2 [3], and CUHK Avenue [4] public datasets.
4) To the best of our knowledge, this is the first study in the literature that adopts the TL and FT approaches for VAD and explores the effectiveness of various Keras applications using public VAD datasets.

The following sections make up the remaining parts of this study. First, the related studies in the literature are presented in Section II. Then, in Section III, the materials and methods used in VAD studies, datasets, and data pre-processing steps utilized in the proposed method are explored. Section IV provides the proposed VAD algorithm, and the experiment results are provided in Section V. The advantages, limitations, and challenges of pre-trained DL models for VAD are discussed in Section VI. Finally, difficulties of the VAD system, the conclusions, and future studies are presented in Section VII.

II. RELATED WORKS

This section provides an overview of various AI-based methods for detecting anomalies in video cameras. VAD studies in the literature and the comparison of frame-level performances of SOTA VAD methods are given in the following sections.

A. VAD WORKS IN THE LITERATURE

Due to researchers' interest in DL methods, they started to apply them for VAD. DL-based methods have enabled high performance to be obtained for the detection of abnormalities from video streams under harsh environmental conditions [5], [6], [7], [8], [9].

A convolutional LSTM (ConvLSTM) network architecture designed as an encoder–decoder model was developed by authors of [10] for anomaly detection via prediction of subsequent frames and reconstruction. This architecture was demonstrated to be a promising technique for VAD in [11]. The ConvLSTM network was fed with input video frames for feature extraction, followed by a deconvolution step in which input frames were reconstructed. In [12] and [13], authors utilized an AE architecture with layered ConvLSTM networks to extract features from video sequence data.

A CNN and LSTM–based network was adopted in [5] for the detection of anomalies in the UCSD [3] and Subway [14] datasets. In a comparable network suggested by the authors in [15], a Conv3D with LSTM was employed for feature extraction from videos. Assuming they included abnormalities in the videos, these extracted features were then employed to look for anomalies. The AE with support vector machine was used by [16] to evaluate the performance on the UCSD [3] and CUHK Avenue [4] datasets. AEs with ConvLSTM were employed in [17], and the performance evaluation was made in the UCSD [3], Subway [14], and CUHK Avenue [4] datasets.

The stacked recurrent neural network (RNN) framework was used in the study conducted by the authors of [11]. In [6], the authors proposed the temporally coherent sparse coding (TSC) approach, where the stacked RNN was used to map similar neighboring squares to the reconstruction coefficient. TSC has been proposed as an efficient technique to detect anomalies in [3], [4], and [14] datasets.

In [18], an interesting usage of GAN was proposed for the detection of anomalies, and the proposed model's performance was evaluated using the [3] and [19] datasets.

A summary of anomaly detection methods in videos was presented in [20] by classifying the methods developed

within 2015–2018 according to the network structures and the datasets used. It was observed that DNNs with learning for hierarchical feature representation were much more efficient than handcrafted feature extraction methods employed in conventional architectures [21].

In the supervised learning–based method proposed in [22], a GAN-trained feature extraction model, which is effective even when there is not enough amount of anomaly data, and a TL method for VAD were used together. In this research, a hybrid DL-based approach, which has recently gained popularity in VAD applications, was adopted.

Transformers, which provide successful results in modeling sequence data in current studies, were applied for VAD in [23]. In the predictive anomaly detection method, where U-Net and Video Vision Transformer (ViViT) were combined, the detection performance was noticed to improve with the addition of the transformer module. A recent review of VAD approaches based on DL methods was presented in [24], and it was observed that GAN and adversarial AE-based methods achieve higher detection rates.

A thorough analysis of the latest approaches in the field of anomaly detection in the literature was presented by authors of [25]. In this research, computational models, datasets, performance metrics, and experimental results used in anomaly analysis in images and videos were discussed from a broad perspective. Similarly, in [26], the authors examined ML and DL methods in the literature for anomaly detection in video surveillance cameras; they argued about the advantages and disadvantages of the proposed methods, discussed the difficulties encountered, and provided popular datasets. In another study [27], the authors analyzed DL approaches, the architectural models used, datasets, performance metrics, and research challenges in VAD applications.

B. COMPARISON OF THE FRAME-LEVEL PERFORMANCES OF SOTA VAD METHODS

In VAD applications, if the anomaly map of a frame includes a pixel containing at least one anomaly, it is considered an anomaly detection at the frame level according to the evaluation approach specified in [28].

Several methods developed in the literature for VAD using the CUHK Avenue, UCSD Ped1, and UCSD Ped2 datasets were extensively investigated and are listed in Table 1. Area under curve (AUC) metric, which is widely used for performance evaluation at the frame level, is presented as percentages. As can be observed from Table 1, methods based on hybrid DL approaches were recently developed by researchers to obtain optimal results in VAD procedures. In addition, it is observed that the success of detection rates is increased with the use of TL and transformer modules.

III. MATERIAL AND METHODS

In the early stages of CV studies until the late 2000s, researchers benefited greatly from conventional image

TABLE 1. Comparison of the frame-level performance of SOTA methods (%).

Ref.	Method	Datasets		
		CUHK Avenue	UCSD Ped1	UCSD Ped2
[29]	Convolutional Autoencoder (ConvAE)	70.2	81.0	90.0
[30]	A method based on locality-sensitive hashing filters	-	87.0	91.0
[31]	Histogram of optical flow (HOF)	-	72.7	87.5
[5]	Spatiotemporal autoencoder (ST-AE)	80.3	89.9	87.4
[32]	3D convolutional autoencoder (3D-ConvAE)	80.9	92.3	91.2
[33]	Unmasking	80.6	68.4	82.2
[34]	Two-stream recurrent variational autoencoder (R-ConvVAE)	79.6	75.0	91.7
[35]	Spatio-temporal adversarial network (STAN)	87.2	82.1	96.5
[7]	Future frame prediction (FFP)	85.1	83.1	95.4
[36]	Optical Flow-ConvAE-LSTM (OF-ConvAE-LSTM)	89.5	92.4	92.9
[37]	MultiLevel anomaly detector (MLAD)	52.82	82.34	99.21
[38]	Adversarial attention-based autoencoder (Ada-Net)	89.2	90.4	90.3
[39]	Object-centric auto-encoders	90.4	-	97.8
[40]	Memory-augmented Deep Autoencoder (MemAE)	83.3	-	94.1
[41]	Stacked recurrent neural network-autoencoder (sRNN-AE)	83.48	-	92.21
[42]	Implicit two-path autoencoder (ITAE + NFs)	85.8	-	97.3
[43]	U-Net	87.8	89	96.6
[44]	Deep spatiotemporal translation network (DSTN) based on GAN and edge wrapping (EW)	87.9	98.5	95.5
[45]	Future frame prediction + reconstruction (IPR)	83.7	82.6	96.2
[46]	TL + Continual Learning	89.6	-	97.8
[21]	Singular value decomposition GAN (SVD-GAN)	89.82	73.26	76.98
[47]	Noise-modulated GAN (NM-GAN)	88.6	90.7	96.3
[48]	Convolutional transformer Based Dual Discriminator GAN (CT-D2GAN)	85.9	-	97.2
[23]	U-Net + ViViT	87	86.7	96.4
[49]	Memory-guided normality for anomaly detection (MNAD)	88.5	-	97.0
[50]	Multi-scale feature and temporal information fusion	87.4	-	96.8
[51]	Hybrid framework that integrates flow reconstruction + frame prediction (HF^2-VAD)	91.1	-	99.3
[52]	ROADMAP	88.3	83.4	96.3
[53]	Double-flow convolutional LSTM variational autoencoder (DF-ConvLSTM-VAE)	87.2	88.4	88.8
[54]	Self-training	86.6	74.7	88.1
[55]	Non-local U-Net frame prediction (NUFP)	85.2	83.6	95.9
[56]	Self-supervised attentive GAN (SSAGAN)	88.8	92.1	97.6
[57]	Hybrid attention + motion constraint	84.9	85.2	95.8
[58]	Spatiotemporal consistency-enhanced network (STCEN)	86.6	82.5	96.9
[59]	Context-related VAD via GAN	87.1	-	96.3
[60]	Feature trajectory–smoothed LSTM (FTS-LSTM)	91.1	83.5	98.3

TABLE 1. *(Continued.)* **Comparison of the frame-level performance of SOTA methods (%).**

[61]	Residual variational autoencoder (RVAE)	88	92	90
[62]	rpNet (reconstruction prediction network)	91.6	91.8	98.9
[63]	Inception capsule autoencoder (Inception-CAE)	99.1	-	99
[64]	Multi-scale feature memorization and multipath network (MsMp-Net)	89	83.8	97.6
[65]	Hierarchical spatio-temporal graph convolutional neural network (HSTGCNN)	87.51	83.39	97.73
[66]	Spatio-temporal dissociation	87.1	-	96.7
[67]	Deep multiplicative attention-based autoencoder (DeMAAE)	83.4	90.7	96.2
[68]	Attention mechanism	85.9	80.5	97.9
[69]	TransCNN	89.6	-	98.4
	Ours (TL-FT-based VAD)	98.41	100	100

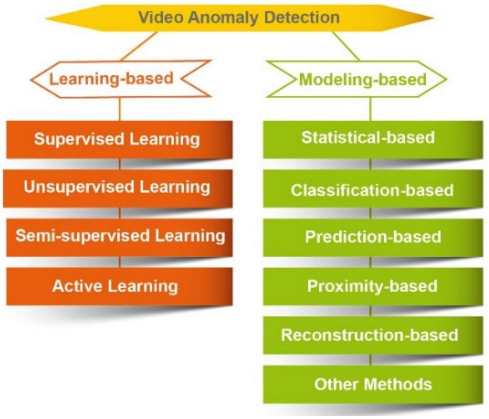


FIGURE 2. Classification of VAD methods.

processing techniques using different handcrafted spatio-temporal features. However, in recent studies, more advanced techniques using DL methods have begun to be used for the detection of anomalies in video streams. The general block diagram proposed by [2] for anomaly detection is shown in Figure 2, where the raw video images that are subjected to data pre-processing are first collected by the cameras and then put through a feature extraction process. The collected data then pass through a modeling technique, where a learning method models the behaviors and determines whether they are normal or abnormal.

A. VAD METHODS

VAD methods have been developed for more than a decade to automatically detect anomalies in videos, and they have been thoroughly examined in [2], where VAD techniques are classified as (i) learning-based and (ii) modeling-based, as shown in Figure 3. VAD methods are briefly examined in the following sections.

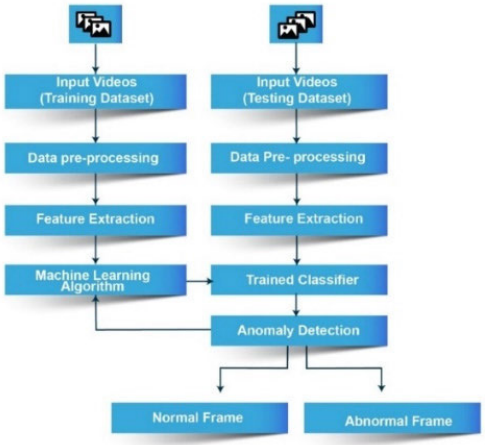


FIGURE 3. Block diagram of VAD.

1) LEARNING-BASED

Learning-based algorithms use labeled and unlabeled training data to learn normal conditions or anomalies. Based on the data and approach used, learning-based methods for VAD are categorized as (i) supervised learning, (ii) unsupervised learning, (iii) semi-supervised learning [2], and (iv) active learning [57].

In supervised learning, labeled datasets are used during the training of the algorithm to predict or classify the results. In this sort of learning, training data are processed to construct various class formulations, such as one-class, two-class, or multi-class output values are produced for various categories.

The grouping concept is the foundation for unsupervised learning and association of data in an unlabeled dataset or discovering patterns in events that occur frequently in the data. Both the normal and abnormal training data samples in this method lack labels. The majority of the samples in the dataset are likely to be typical occurrences that happen regularly, while events that are encountered infrequently are categorized as anomalies.

When just a few labeled training datasets are available, the semi-supervised learning strategy, which sits between supervised learning and unsupervised learning, is favored. A small part of labeled data and a large amount of unlabeled data are combined during the training process.

The anomaly detection model in unsupervised or semi-supervised VAD is trained offline using normal training samples, and the model is not updated when fresh data come in. This sometimes results in video representations that are inefficient. In active learning–based VAD techniques, people (or domain experts) are involved in the online framework while labeling ambiguous situations or data samples.

2) MODELING-BASED

Modeling-based VAD methods in the literature are summarized below, as traditional handcrafted methods are now replaced by DNNs.

In statistical-based approaches, model parameters are learned during training to predict anomalies, and the distribution of normal event data is modeled. Thus, according to the probability model, a higher probability is expected for normal events, while a lower probability outcome is anticipated for anomalies.

Anomalies are discovered using proximity-based VAD approaches by calculating the distance between the object and its surroundings.

In classification-based anomaly detection approaches, anomalies are distinguished by determining the separation margin.

Reconstruction-based methods rely on predicting anomalies based on the reconstruction error. In this approach, where normal samples can be accurately reconstructed using a limited set of basic functions, anomalies have greater reconstruction loss.

In predictive techniques, the probability of the desired outcome is predicted in line with the significance of input variables in the dataset, utilizing known outcomes while training the model. Several other techniques, such as fuzzy theory estimation [58], adaptive sparsity model [59], sparsity-based background subtraction method [60], use of high-frequency correlation sensors [61], particle filtering [62], and redundancy removal [63], are utilized in the literature to spot irregularities in the flow of traffic, such as accidents, dangerous driving behavior, street crimes, and traffic violations.

B. TECHNOLOGIES USED IN THE PROPOSED METHOD

A VAD method that is based on supervised learning using the TL and FT approaches is proposed in this study, where several Keras applications were used as base models. TL uses features that are learned on one problem and takes advantage of these learned features on a new, similar problem. Basically, TL consists of four basic steps and an optional step: (i) getting layers from a pre-trained model (base model), (ii) freezing the layers of the base model to prevent losing the previously learned information that they contain, (iii) adding some new, trainable layers on top of the frozen layers so that previously learned features will be turned into predictions in the new dataset, and (iv) training the new layers using the new dataset. FT is the last step that comprises unfreezing the whole model or some parts of it and training it again on the new dataset using a considerably low learning rate. This FT step aims to enhance the general performance of the model and achieve more success.

In the following sections, brief information about the Keras applications, DL architectures, programming language, technologies, and libraries used in this study is presented.

1) KERAS APPLICATIONS USED IN TL

Keras applications are DL models that come with pre-trained weights. They are made available for (i) prediction, (ii) feature extraction, and (iii) FT. These pre-trained models consist of architecture and weights. A list of selected Keras applications within the scope of this study is given in Table 2 and summarized in the following sections.

TABLE 2. List of Keras applications used as base models.

Base Models			
VGG16	VGG19	Xception	MobileNetV3Small
MobileNetV3Large	InceptionV3	EfficientNetB0	EfficientNetB1
EfficientNetB2	EfficientNetB3	EfficientNetB4	EfficientNetB5
EfficientNetB6	EfficientNetB7	EfficientNetV2 B0	ResNet101V2
DenseNet121	NASNetMobile	NASNetLarge	ConvNeXtTiny

a) *VGG16 and VGG19:* VGG16 is a CNN model that is employed for image classification. It has 16 trainable layers and regarded as one of the most successful architectures employed for vision tasks. It has three fully connected layers and 13 convolutional layers. VGG19 is a variant of VGG model that consists of 19 layers. It has 16 convolution layers, three fully connected layers, five MaxPool layers, and one SoftMax layer.

b) *Xception:*Xception is a deep CNN architecture that was developed by Google researchers and has Depthwise Separable Convolutions (DSCs). DSCs are regarded as alternatives to classical convolutions, which are much more efficient in terms of computation time.

c) *MobileNetV3:*Through the use of the NetAdapt algorithm and hardware-aware network architecture search, MobileNetV3 is a CNN architecture that is tailored to the central processing units (CPUs) of mobile phones. It is the next generation of the MobileNet architecture family and provides SOTA results for lightweight models in CV problems. MobileNetV3 has two effective models for DL operations on mobile devices—MobileNetV3Small and MobileNetV3Large—that can be used in low- and high-resource mobile devices.

d) *InceptionV3:*InceptionV3 is the third version of Google's Inception CNN architecture that is used for image analysis and object detection purposes. It is a DNN architecture that has inception blocks where the same input tensor is convolved with multiple filters and their results are concatenated.

e) *EfficientNet:*EfficientNet is a scaling method and CNN design that uniformly scales all depth, width, and resolution dimensions utilizing a compound coefficient. EfficientNet consists of a collection of image classification models that achieve SOTA accuracy while being an order-of-magnitude smaller and much faster compared to earlier models. While there are several variants of EfficientNet family from the baseline model, EfficientNetB0 to EfficientNetB7 and EfficientNetV2B0 models were used in this study.

f) *ResNet:*Residual Network (ResNet) has residual blocks and was introduced by Microsoft researchers to address the vanishing and exploding gradient problem. In this network, the skip connections method is applied to transfer output from one layer to another, which helps to mitigate

the problem of gradient vanishing, and a residual block is a layer with a skip connection. ResNet101V2 model was used in this study.

g) *DenseNet:*A variant of CNN called DenseNet uses dense connections via dense blocks, where all layers are directly connected to one another. While providing better parameter efficiency, it also yields comparable results with a low number of parameters to train and helps reduce overfitting problems. DenseNet121 model was used in this study.

h) *NASNet:* NASNet, which stands for neural architecture search (NAS) network, is also a sort of CNN model. Since substantial engineering techniques are typically needed when developing a neural network, NASNet attempts to directly train the model architectures from the relevant dataset and produces cutting-edge results with a smaller model size and level of complexity. NASNetMobile and NASNetLarge models were used in this study.

i) *ConvNeXt:*ConvNeXt is a pure convolutional model that is more accurate, performant, and scalable than vision transformers while maintaining the design simplicity of CNNs. It not only includes easy, fully-convolutional design of ResNets for both training and testing but also outperforms vision transformers and allows for straightforward implementation. ConvNeXtTiny model was used in this study.

2) LIBRARIES

A list of the libraries used in this study is given below:

a) *TensorFlow:* TensorFlow is a Python library that basically contains numerical computation routines and can be used for ML applications.

b) *Keras:*Keras is a Python-based DL application programming interface (API) running on the TensorFlow ML platform.

c) *Scikit-learn:*Scikit-learn is a free ML library for Python that contains statistical modeling and ML methods, such as classification, regression, clustering, and dimension reduction.

d) *Pandas:*Pandas is an open-source data analysis and manipulation tool built on Python programming language, which is quick, robust, versatile, and easy to use.

e) *NumPy:*NumPy is a Python library that enables working with arrays as well as offering tools for working with matrices, the Fourier transform, and linear algebra.

f) *Python Imaging Library (PIL):*The PIL package gives the Python interpreter access to image processing features by adding wide file format compatibility, effective internal representation, and robust image processing capabilities.

g) *Open-Source Computer Vision (OpenCV):* OpenCV provides a free and open-source software library for CV and ML applications.

3) GOOGLE COLAB

Google Colab is a research product of Google that enables the generation and running of Python code via the Internet.

FIGURE 4. Classification of VAD methods.

It is a platform particularly suited for ML, data analysis, and training purposes and provides a Jupyter notebook service hosting environment that requires no installation and grants access to computer resources, such as graphics processing units (GPUs).

C. DATASETS

In this study, the CUHK Avenue, UCSD Ped1, and UCSD Ped2 datasets, which are widely used in VAD benchmarking studies, were employed to evaluate the efficiency of the proposed approach.

1) CUHK AVENUE

The CUHK Avenue dataset has been widely utilized for the detection of anomalies in academic studies since 2013. As presented in Table 3, it includes 16 training and 21 test video clips recorded inside the CUHK campus. The videos consist of a total of 30,652 frames, including 15,328 training and 15,324 test frames. While training videos include normal events, test videos contain both normal and anomalous events. Anomalous events consist of unusual human movements (such as running, walking with a suspicious object, or throwing an object), pedestrians going in the wrong direction, and objects that are perceived as anomalies (such as an abandoned object or a bicycle).

The video clips in the CUHK Avenue dataset were recorded at 26 frames per second (fps). Each video consists of 640 $\times$ 360 resolution, colored (24-bit, RGB) frames. Three anomalies found in the CUHK Avenue dataset are shown in Figure 4.

2) UCSD PED1 AND UCSD PED2

The UCSD dataset is another popular dataset that is frequently utilized by researchers for VAD. The dataset was obtained using a fixed camera positioned at a height facing the pedestrian crossings and contains videos showing varying population densities on the walkways, from few to many people.

Normally, the videos only contain pedestrians walking on the road. In training videos, pedestrians either walk or stand on the sidewalk. Anomalous events are caused by the movement of non-pedestrian objects on the walkways (vehicle movements on the walkways, wheelchair movements, cycling, skating, walking on the grass, etc.) and anomalous pedestrian movement patterns. Anomalies commonly observed in the dataset include cyclists, skaters, small cars,

TABLE 3. Properties of CUHK Avenue dataset.

Dataset	Number of Videos	Number of Frames	Average Number of Frames	Number of Normal Frames	Number of Frames with Anomalies	Number of Anomalies	Examples of Anomalies	Length	Resolution
Training	16	15,328	839	26,832	-	-	-	30 min.	
Test	21	15,324			3,820	47	Running, throwing object		640 × 360
Total	37	30,652	-	26,832	3,820	47	-	30 min.	-

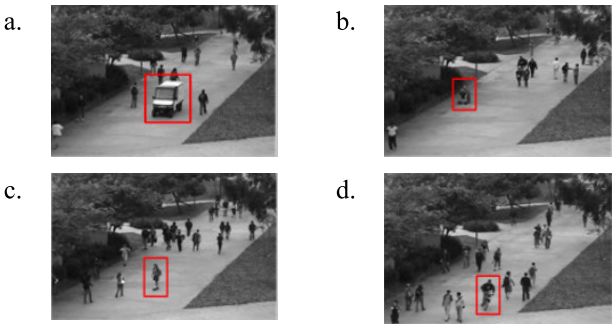


FIGURE 5. Anomalies in the UCSD Ped1 Dataset a. Van, b. Wheelchair, c. Skater, d. Cyclist.

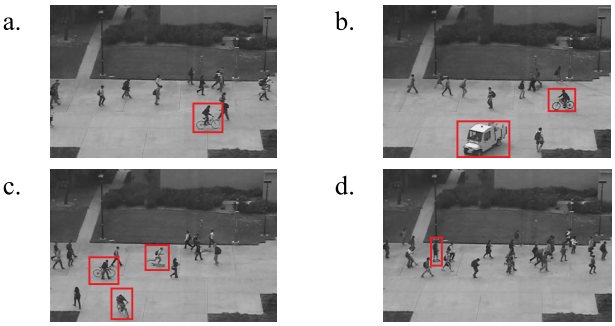


FIGURE 6. Anomalies in the UCSD Ped2 Dataset a. Cyclist, b. Van, Cyclist, c. Cyclist, Skater, d. Skater.

and people walking on grass. A few images of people in wheelchairs are also among the anomalies.

Two subgroups of the UCSD dataset were created, each of which represents a different setting. Each scene's video recording was cut into pieces that contained about 200 frames. The UCSD Ped1 and UCSD Ped2 datasets were recorded at 26 fps.

Video clips consisting of 200 frames each were generated using videos from the UCSD Ped1 dataset that has a resolution of 238 × 158 in tiff file format with 72-dpi and 8-bit grayscale images. Video clips consisting of 120–180 frames each were generated using videos from the UCSD Ped2 dataset that has a resolution of 360 × 240 in tiff file format with 72-dpi and 8-bit grayscale images.

Table 4 and Table 5 present information about the UCSD Ped1 and UCSD Ped2 datasets, respectively. Examples of some anomalies found in the UCSD Ped1 and UCSD Ped2 datasets are shown in Figure 5 and Figure 6, respectively.

D. DATA PRE-PROCESSING

The data pre-processing steps involved in this study are presented in the following sections.

1) IDENTIFICATION OF ANOMALIES

The frames containing anomalous conditions in the test datasets were read from files that were downloaded together with the datasets. All datasets and the files containing frame numbers of the anomalies were fetched from Google Colab environment and edited so that they could be utilized in ML methods and saved in Google Drive.

2) EXTRACTION OF FRAMES

The training and test videos that make up the CUHK Avenue dataset were converted into frames and saved in Google Drive as image files to be used as input for DL methods.

3) RESIZE AND NORMALIZATION

Since the video samples in the datasets have different sizes, the extracted frames were resized to 224 × 224 pixels and all the pixel values were scaled into [1, 0] range.

4) CONVERTING TO GRAYSCALE

The training and test videos that make up the CUHK Avenue dataset were converted to grayscale.

5) REORGANIZATION OF DATASETS

Since we adopted a TL and FT–based supervised learning approach for VAD, we reorganized the original training and test datasets so that we have (i) a training dataset, (ii) a validation dataset, and (iii) a test dataset for each of the three datasets. For this purpose, we applied the below steps:

a) We first merged the normal frames in both training and test datasets and created the normal frames of the new training dataset. All normal frames were combined in the normal folder.
b) We moved all abnormal frames from the test dataset into the new training dataset's anomaly folder.
c) We randomly selected 20% of the frames from the new training dataset and created a validation dataset that has both normal and abnormal frames randomly selected from the normal and anomaly folders of the new training

TABLE 4. Properties of UCSD Ped1 dataset.

Dataset	Number of Videos	Number of Frames	Average Number of Frames	Number of Normal Frames	Number of Frames with Anomalies	Number of Anomalies	Examples of Anomalies	Length	Resolution
Training	34	6,800	201	9,995	-	-	-	5 min.	
Test	36	7,200			4,005	40	Bikers, small carts, walking across walkways		238 × 158
Total	70	14,000	-	9,995	4,005	40	-	5 min.	-

TABLE 5. Properties of UCSD Ped2 dataset.

Dataset	Number of Videos	Number of Frames	Average Number of Frames	Number of Normal Frames	Number of Frames with Anomalies	Number of Anomalies	Examples of Anomalies	Length	Resolution
Training	16	2,550	163	2,924	-	-	-	5 min.	
Test	12	2,010			1,636	12	Cyclists, small carts, walking across walkways		360 × 240
Total	28	4,560	-	2,924	1,636	12	-	5 min.	-

TABLE 6. Final datasets.

Dataset	Category	Number of Frames in Training Dataset	Number of Frames in Validation Dataset	Number of Frames in Test Dataset	Total
CUHK	Normal	21,426	4,324	1,082	26,832
Avenue	Anomaly	3,096	579	145	3,820
UCSD	Normal	7,964	1,592	399	9,955
Ped1	Anomaly	3,236	647	162	4,045
UCSD	Normal	2,329	466	117	2,912
Ped2	Anomaly	1,318	264	66	1,648

dataset. We removed those selected frames from the new training dataset.

d) We randomly selected 20% of the frames from the validation dataset and created a test dataset that has both normal and abnormal frames randomly selected from the normal and anomaly folders of the validation dataset. We removed those selected frames from the validation dataset.

The final datasets used in this study for the TL-FT-based VAD method are given in Table 6.

IV. PROPOSED METHOD

In this study, a VAD method that is based on TL and FT using the supervised DL techniques is proposed and explained in the following sections.

A. TL-FT-BASED VAD ALGORITHM

The majority of ML techniques, particularly those based on DL, take a long time to train. Transferring a pre-trained model from one data domain to another that is related but distinct might be one way to solve the problem without having to retrain the model or provide new datasets. For example, without having to retrain, a model that was trained to detect cars in video footage may also detect trucks that were not seen before. This approach is known as TL, which this study takes advantage of.

Twenty publicly available Keras applications, which are DL models with pre-trained weights, were used as base models in this study. When a model is instantiated, weights were automatically downloaded and the models were built according to the image data format. The proposed method consists of a selection of basic hyper parameter values and training phases. Training phase also comprises TL phase and FT phase. All models built in this study were trained according to selected hyper parameter values.

1) IMAGE PRE-PROCESSING

Since every Keras application requires a particular type of input pre-processing, we performed pre-processing of our inputs before passing them to the base models. Before training of the models with the images in the datasets, images were pre-processed depending on the selected base model. The TensorFlow library converted the input values of the images into the format accepted by the models with the pre-processing layer that it contains. All inputs were pre-processed using the TensorFlow library available in the pre-processing layers of the selected base models.

2) SELECTION OF HYPER PARAMETERS

Since several Keras applications were used as base models in the proposed method and the DL model used in this study was applied to all selected base models, the aim was to

avoid overfitting with pre-trained base models using dropout layers. In the absence of the dropout layers, the models tend to memorize the training dataset and perform poorly in the validation dataset during the FT phase. Therefore, after a few trial-and-error steps, the dropout and dense layers used in the DL model and the other hyperparameter values were determined.

3) PROPOSED ARCHITECTURE

The general architecture of the proposed TL and FT–based DL model that is adapted from [64] and used for VAD is shown in Figure 7.

As shown in Figure 7, the DL model comprises an input layer, a pre-processing layer, a pre-trained base model, a global average pooling layer, dropout layers, and dense layers. The input layer is the layer where the images were given to the model. In the pre-processing layer, the inputs were adapted to the base model. The fully connected layers at the top of the base model were removed from the model, and a GlobalAverage2D layer was added. While the dropout layers were placed at the output of dense layers with ratios of 0.25, 0.25, 0.25, 0.50, 0.50, and 0.50, the dense layers had 512, 512, 128, 128, and 32 units in the architecture of the proposed VAD model as shown in Figure 7. While the ReLu activation function was used after the dense layers, the sigmoid activation function was selected as the output layer's activation function since a binary classification was performed where the DL model was designed to predict normal and anomalous frames. Binary cross-entropy was used as the loss function, and the Nesterov Adam function was utilized as the model's optimization function in both the TL and FT phases. The learning rate was selected as 0.001 for the TL phase and 0.0001 for the FT phase. These selected layers and parameters were applied to all models in the same way.

4) TL PHASE

Before training the proposed VAD model, all inputs were pre-processed using the pre-processing functions of the base models. Input shapes of frames were pre-processed and adapted for each of the base models. Then, base models were instantiated using weights pre-trained on ImageNet and without including the fully connected layers at the top of the base model networks. Base model layers were frozen during the TL phase, and the model was compiled using binary cross-entropy as the loss function and the Nesterov Adam function as the optimizer. After the compilation of the model, it was trained on the created training data.

Since the problem in this study is the detection of normal and anomaly classes, the weights obtained from the pre-trained base models were preserved by freezing the base model layers for training. During the TL phase, only the layers added to the output layer of the DL model, as can be observed from Figure 7, were trained. During the model training, a total of 999 epochs with a batch size of 8 were used. We applied the early stopping technique, which is a form of regularization that is used to avoid overfitting while training the learner with an iterative method. We monitored the performance of the network on our validation set, and the models that did not improve with a wait of 20 epochs after the best accuracy value, based on validation accuracy, were stopped by applying early stopping. We observed that all of the models were early stopped during the TL phase and went through the FT phase while preserving their weights.

TABLE 7. Number of layers for FT and other model parameters.

Base Model	Number of Layers in the Base Model	Number of Layers Fine-Tuned	Number of Parameters in Base Model	Number of Trainable Parameters	Total Number of Parameters in the Model
VGG16	19	3	14,714,688	611,649	15,326,337
VGG19	22	4	20,024,384	611,649	20,636,033
Xception	132	20	20,861,480	1,398,081	22,259,561
MobileNetV3Small	229	35	939,120	644,417	1,583,537
MobileNetV3Large	263	40	2,996,352	841,025	3,837,377
InceptionV3	311	47	21,802,784	1,398,081	23,200,865
EfficientNetB0	238	36	4,049,571	1,004,865	5,054,436
EfficientNetB1	340	51	6,575,239	1,004,865	7,580,104
EfficientNetB2	339	51	7,768,569	1,070,401	8,838,970
EfficientNetB3	385	58	10,783,535	1,135,937	11,919,472
EfficientNetB4	475	72	17,673,823	1,267,009	18,940,832
EfficientNetB5	577	87	28,513,527	1,398,081	29,911,608
EfficientNetB6	667	101	40,960,143	1,529,153	42,489,296
EfficientNetB7	814	123	64,097,687	1,660,225	65,757,912
EfficientNetV2B0	270	41	5,919,312	1,004,865	6,924,177
ResNet101V2	377	57	42,626,560	1,398,081	44,024,641
DenseNet121	427	65	7,037,504	873,793	7,911,297
NASNetMobile	769	116	4,269,716	890,177	5,159,893
NASNetLarge	1,039	156	84,916,818	2,413,889	87,330,707
ConvNeXtTiny	151	23	27,820,128	742,721	28,562,849

5) FT PHASE

In the FT phase, we aimed to activate the previously frozen base model layers for training by preserving the weights obtained in the TL phase of the model, thus acquiring higher accuracy values. Table 7 lists the number of layers in the base model, the number of layers that underwent FT, the number of parameters in the base model, the number of trainable parameters, and the total number of parameters in the base models used in this study.

The number of layers specified in Table 7 reflects those reported by the TensorFlow tool. The number of layers fine-tuned shows how many layers were activated for training during the FT phase in the base model backward from the output layer.

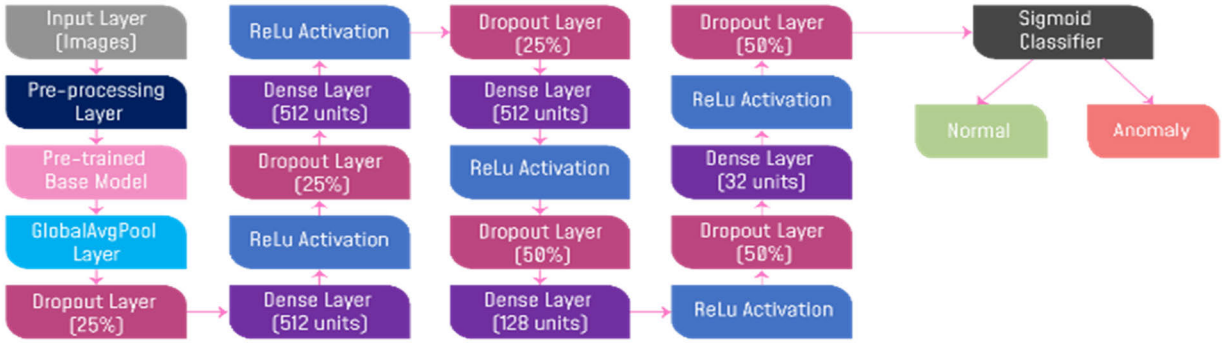


FIGURE 7. General architecture of the proposed VAD model.

TABLE 8. Number of epochs for CUHK avenue test dataset.

Model	TL Phase	FT Phase	Total
VGG16	21	51	72
VGG19	47	21	68
Xception	50	50	100
MobileNetV3Small	53	48	101
MobileNetV3Large	42	75	117
InceptionV3	21	60	81
EfficientNetB0	100	62	162
EfficientNetB1	77	54	131
EfficientNetB2	56	82	138
EfficientNetB3	68	70	138
EfficientNetB4	90	32	122
EfficientNetB5	97	62	159
EfficientNetB6	84	54	138
EfficientNetB7	35	64	99
EfficientNetV2B0	74	60	134
ResNet101V2	38	21	59
DenseNet121	46	54	100
NASNetMobile	56	58	114
NASNetLarge	84	71	155
ConvNeXtTiny	77	47	124

TABLE 9. Number of epochs for UCSD Ped1 test dataset.

Model	TL Phase	FT Phase	Total
VGG16	106	55	161
VGG19	74	21	95
Xception	59	41	100
MobileNetV3Small	108	69	177
MobileNetV3Large	95	69	164
InceptionV3	36	69	105
EfficientNetB0	92	42	134
EfficientNetB1	61	47	108
EfficientNetB2	81	48	129
EfficientNetB3	79	54	133
EfficientNetB4	74	96	170
EfficientNetB5	105	66	171
EfficientNetB6	70	37	107
EfficientNetB7	112	83	195
EfficientNetV2B0	75	41	116
ResNet101V2	100	21	121
DenseNet121	21	57	78
NASNetMobile	132	48	180
NASNetLarge	115	50	165
ConvNeXtTiny	67	50	117

The general view on the activation of the layers for training is that the first layers of the model learn only the basic features, while the layers close to the output learn more complex features. During the backpropagation of artificial neural networks, weights are updated from the output layer toward the input layer. For these two reasons, we preferred to freeze the bottom layers and train the remaining top layers that are close to the output layer. The number of layers to be fine-tuned was determined as a percentage in this study because the number of layers varies depending on the base model. We calculated 15% of the total number of layers of the base model and rounded it to an integer, and those top layers of the base model were unfrozen for training during the FT phase, while the rest of the layers were kept frozen. The DL model was recompiled using the Nesterov Adam function as the optimizer with a low learning rate, i.e., 0.0001, for the modifications to take effect. After recompilation of the model, we trained our model again on the created training data. By using the same epoch number (i.e., 999), batch size (i.e., 8), and the early stopping approach applied in the TL phase, the training of the DL model during the FT phase was carried out. Similar to the TL phase, we noticed that all the models were early stopped during the FT phase.

Table 18, Table 9, and Table 10 show the number of epochs executed in the proposed VAD model during the TL and FT phases for the CUHK Avenue, UCSD Ped1, and UCSD Ped2 test datasets, respectively. Although the number of epochs was set as 999 for both the TL and FT phases, all of the models were early stopped during both TL and FT phases, as can be seen from Table 18, Table 9, and Table 10. The total number of epochs was observed to vary within 68–162, 78–180, and 71–142 for the CUHK Avenue, UCSD Ped1, and UCSD Ped2 datasets, respectively.

B. PERFORMANCE EVALUATION METRICS

Several metrics that are widely used in assessing the performance of ML models were utilized for the evaluation of the proposed VAD model. Accuracy, precision, recall, F1-score, and AUC measure were the metrics used for the evaluation of

TABLE 10. Number of epochs for UCSD Ped2 test dataset.

Model	TL Phase	FT Phase	Total
VGG16	55	68	123
VGG19	41	34	75
Xception	61	43	104
MobileNetV3Small	62	52	114
MobileNetV3Large	59	33	92
InceptionV3	68	37	105
EfficientNetB0	68	54	122
EfficientNetB1	49	40	89
EfficientNetB2	62	44	106
EfficientNetB3	64	25	89
EfficientNetB4	54	31	85
EfficientNetB5	71	35	106
EfficientNetB6	99	43	142
EfficientNetB7	64	36	100
EfficientNetV2B0	49	28	77
ResNet101V2	63	61	124
DenseNet121	47	44	91
NASNetMobile	65	33	98
NASNetLarge	41	30	71
ConvNeXtTiny	80	38	118

TABLE 11. Model evaluation metrics.

Evaluation Metric	Calculation Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
F1-score	$\frac{2 * TP}{2 * TP + FP + FN}$
AUC	It is the whole two-dimensional area below the receiver operating characteristic (ROC) curve.

the models in this study, and they are given in Table 11, along with their calculation formulas.

True Positive, abbreviated as TP, refers to the quantity of accurately classified abnormal frames. True Negative, or TN, refers to the quantity of accurately classified normal frames. False Positive, or FP, refers to the quantity of normal frames that were misclassified. False Negative, or FN, refers to the quantity of abnormal frames that were misclassified.

V. RESULTS

A. DATASETS

The CUHK Avenue, UCSD Ped1, and UCSD Ped2 datasets, which are often used in the literature for VAD, were used in experiments to measure the performance of the proposed approach.

B. EXPERIMENTAL ENVIRONMENT

All of the experiments in this study were performed in a Google Colab environment using 83.5 GB RAM and an NVIDIA A100-SXM4 GPU that has 40 GB RAM. The proposed TL-FT-based VAD approach was developed with Python programming language, and all the models were trained using Keras and TensorFlow frameworks.

C. EXPERIMENTAL RESULTS

Results of the evaluation metrics used in this study for the created validation and test datasets are given in Table 12, Table 13, and Table 14. While validation loss and validation accuracy values were obtained using validation datasets, test loss, test accuracy, precision, recall, F1-score, and AUC scores were obtained using test datasets. As can be observed from Table 12, Table 13, and Table 14, variants of EfficientNet base models helped achieve the highest scores in different evaluation metrics for all datasets using the TL-FT-based approach proposed in this study.

1) COMPARISON OF ACCURACY VALUES

Although accuracy rates tended to rise as the number of epochs increased and the FT phase of the proposed DL model helped improve the test accuracy for most of the models, there were also some models whose accuracy rates were not improved during the FT phase. VGG19 and ResNet101V2 models were stuck in local minimums while training on the CUHK Avenue and UCSD Ped1 datasets, and the proposed approach did not improve their VAD performances. It should be noted that due to being stuck in local minimums, these models that did not improve during the FT phase reported the lowest scores in all evaluation metrics, although their validation accuracy rates were greater than 85% during the TL phase.

The highest accuracy rate of 99.10% was achieved with the Xception base model for the CUHK Avenue test dataset, the highest accuracy rate of 100% was achieved with the EfficientNetB5 base model for the UCSD Ped1 test dataset, and the highest accuracy rate of 100% was achieved with VGG16, VGG19, EfficientNetB0, EfficientNetB3, EfficientNetV2B0, and DenseNet121 base models for the UCSD Ped2 test dataset. There were several base models where the proposed TL-FT-based VAD approach provided SOTA accuracy rates for the UCSD Ped2 test dataset.

Accuracy-epoch graphs of the proposed VAD method using EfficientNetB3 as the base model for the CUHK Avenue, UCSD Ped1, and UCSD Ped2 test datasets are shown in Figure 8, Figure 9, and Figure 10, respectively. It can be observed from Figure 8, Figure 9, and Figure 10 that both training and validation accuracy rates were increased when the FT phase was applied after the TL phase. These figures illustrate that the VAD performance of our approach is further enhanced by applying the FT mechanism followed by TL. As far as we know, this is the first attempt in the literature to utilize a TL-FT-based hybrid approach to enhance the detection rate of anomalies.

2) COMPARISON OF AUC SCORES

It can be noticed from Table 12, Table 13, and Table 14 that the highest AUC rate of 98.41% was achieved with the MobileNetV3Large base model for the CUHK Avenue test dataset, the highest AUC rate of 100% was achieved with the EfficientNetB5 base model for the UCSD Ped1 test

TABLE 12. Results of the evaluation metrics for CUHK avenue dataset.

Model	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy	Precision	Recall	F1-score	AUC
VGG16	0.0486	0.9906	0.0857	0.9870	0.9640	0.9241	0.9437	0.9598
VGG19	0.3637	0.8819	0.3639	0.8818	-	0.0000	-	0.5000
Xception	0.0426	0.9902	0.0550	0.9910	0.9786	0.9448	0.9614	0.9710
MobileNetV3Small	0.0557	0.9916	0.0692	0.9886	0.9645	0.9379	0.9510	0.9667
MobileNetV3Large	0.0293	0.9929	0.0470	0.9878	0.9221	0.9793	0.9498	0.9841
InceptionV3	0.0421	0.9878	0.0397	0.9878	0.9276	0.9724	0.9495	0.9811
EfficientNetB0	0.1130	0.9763	0.1309	0.9707	0.8079	0.9862	0.8882	0.9774
EfficientNetB1	0.1059	0.9902	0.2821	0.9861	1.0000	0.8828	0.9377	0.9414
EfficientNetB2	0.0238	0.9927	0.0240	0.9902	0.9854	0.9310	0.9574	0.9646
EfficientNetB3	0.0304	0.9920	0.0478	0.9853	0.9853	0.9704	0.9034	0.9499
EfficientNetB4	0.0271	0.9910	0.0284	0.9870	0.9778	0.9103	0.9429	0.9538
EfficientNetB5	0.0292	0.9922	0.0474	0.9902	0.9650	0.9517	0.9583	0.9736
EfficientNetB6	0.0659	0.9886	0.0845	0.9747	0.8958	0.8897	0.8927	0.9379
EfficientNetB7	0.0716	0.9886	0.1352	0.9902	0.9854	0.9310	0.9574	0.9646
EfficientNetV2B0	0.0446	0.9929	0.0634	0.9861	0.9571	0.9241	0.9404	0.9593
ResNet101V2	0.3634	0.8819	0.3636	0.8818	-	0.0000	-	0.5000
DenseNet121	0.0827	0.9755	0.1079	0.9698	0.8068	0.9793	0.8847	0.9739
NASNetMobile	0.1672	0.9839	0.0937	0.9715	0.8481	0.9241	0.8845	0.9510
NASNetLarge	0.0335	0.9890	0.0401	0.9845	0.9846	0.8828	0.9309	0.9405
ConvNeXtTiny	0.0369	0.9947	0.1744	0.9943	0.9929	0.9586	0.9754	0.9788

TABLE 13. Results of the evaluation metrics for UCSD Ped1 dataset.

Model	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy	Precision	Recall	F1-score	AUC
VGG16	0.0225	0.9960	0.0246	0.9911	0.9758	0.9938	0.9847	0.9919
VGG19	0.6013	0.7110	0.6011	0.7112	-	0.0000	-	0.5000
Xception	0.0834	0.9888	0.0388	0.9857	0.9639	0.9877	0.9756	0.9863
MobileNetV3Small	0.1220	0.9946	0.0102	0.9964	0.9878	1.0000	0.9939	0.9975
MobileNetV3Large	0.0371	0.9937	0.0071	0.9964	0.9938	0.9938	0.9938	0.9957
InceptionV3	0.0353	0.9920	0.0257	0.9911	0.9816	0.9877	0.9846	0.9901
EfficientNetB0	0.0233	0.9951	0.0214	0.9947	0.9947	0.9938	0.9877	0.9907
EfficientNetB1	0.0555	0.9955	0.0677	0.9911	0.9876	0.9815	0.9845	0.9882
EfficientNetB2	0.0604	0.9937	0.0121	0.9982	0.9939	1.0000	0.9969	0.9987
EfficientNetB3	0.0222	0.9937	0.0166	0.9911	0.9816	0.9877	0.9846	0.9901
EfficientNetB4	0.0490	0.9933	0..049	0.9982	0.9939	1.0000	0.9969	0.9987
EfficientNetB5	0.0758	0.9955	0.0008	1.0000	1.0000	1.0000	1.0000	1.0000
EfficientNetB6	0.0505	0.9915	0.0170	0.9929	0.9817	0.9938	0.9877	0.9932
EfficientNetB7	0.2973	0.9960	0.0038	0.9982	0.9939	1.0000	0.9969	0.9987
EfficientNetV2B0	0.0961	0.9862	0.0222	0.9911	1.0000	0.9691	0.9843	0.9846
ResNet101V2	0.6018	0.7110	0.6016	0.7112	-	0.0000	-	0.5000
DenseNet121	0.0810	0.9795	0.1015	0.9786	0.9808	0.9444	0.9623	0.9685
NASNetMobile	0.0762	0.9830	0.0279	0.9875	0.9936	0.9630	0.9781	0.9802
NASNetLarge	0.0304	0.9929	0.0219	0.9964	0.9938	0.9938	0.9938	0.9957
ConvNeXtTiny	0.1392	0.9955	0.0318	0.9929	0.9817	0.9938	0.9877	0.9932

TABLE 14. Results of the evaluation metrics for UCSD Ped2 dataset.

Model	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy	Precision	Recall	F1-score	AUC
VGG16	0.1181	0.9959	0.0008	1.0000	1.0000	1.0000	1.0000	1.0000
VGG19	0.0503	0.9945	0.0107	1.0000	1.0000	1.0000	1.0000	1.0000
Xception	0.1779	0.9959	0.1280	0.9891	1.0000	0.9697	0.9846	0.9848
MobileNetV3Small	0.1276	0.9904	0.0185	0.9945	1.0000	0.9848	0.9924	0.9924
MobileNetV3Large	0.0399	0.9932	0.0357	0.9891	1.0000	0.9697	0.9846	0.9848
InceptionV3	0.0689	0.9932	0.0686	0.9945	1.0000	0.9848	0.9924	0.9924
EfficientNetB0	0.1284	0.9959	0.0026	1.0000	1.0000	1.0000	1.0000	1.0000
EfficientNetB1	0.1133	0.9945	0.0435	0.9891	1.0000	0.9697	0.9846	0.9848
EfficientNetB2	0.0502	0.9986	0.1423	0.9891	1.0000	0.9697	0.9846	0.9848
EfficientNetB3	0.1024	0.9932	0.0043	1.0000	1.0000	1.0000	1.0000	1.0000
EfficientNetB4	0.1331	0.9945	0.2011	0.9945	1.0000	0.9848	0.9924	0.9924
EfficientNetB5	0.0596	0.9904	0.0132	0.9891	1.0000	0.9697	0.9846	0.9848
EfficientNetB6	0.0684	0.9973	0.0062	0.9945	1.0000	0.9848	0.9924	0.9924
EfficientNetB7	0.1288	0.9863	0.0705	0.9836	0.9565	1.0000	0.9778	0.9872
EfficientNetV2B0	0.1021	0.9945	0.0032	1.0000	1.0000	1.0000	1.0000	1.0000
ResNet101V2	0.0283	0.9932	0.0301	0.9891	1.0000	0.9697	0.9846	0.9848
DenseNet121	0.1447	0.9945	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000
NASNetMobile	0.0661	0.9932	0.0462	0.9891	1.0000	0.9697	0.9846	0.9848
NASNetLarge	0.0870	0.9932	0.0305	0.9836	1.0000	0.9545	0.9767	0.9773
ConvNeXtTiny	0.1361	0.9877	0.1140	0.9781	1.0000	0.9394	0.9688	0.9697

dataset, and the highest AUC rate of 100% was achieved with VGG16, VGG19, EfficientNetB0, EfficientNetB1, EfficientNetV2B0, and DenseNet121 base models for the UCSD Ped2 test dataset. Similar to the highest accuracy rates, the same base models provided SOTA AUC scores for the UCSD Ped2 test dataset.

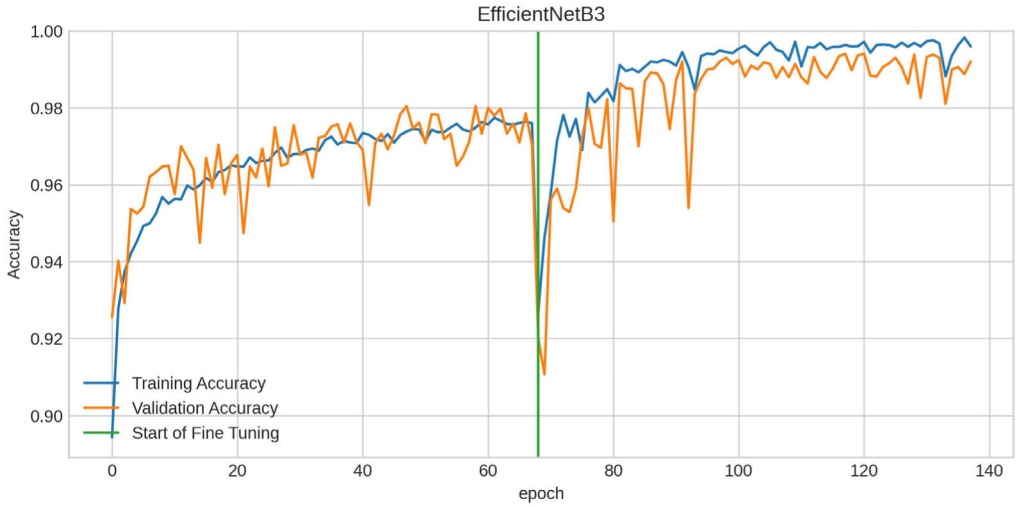


FIGURE 8. Accuracy-epoch graph of CUHK Avenue dataset.

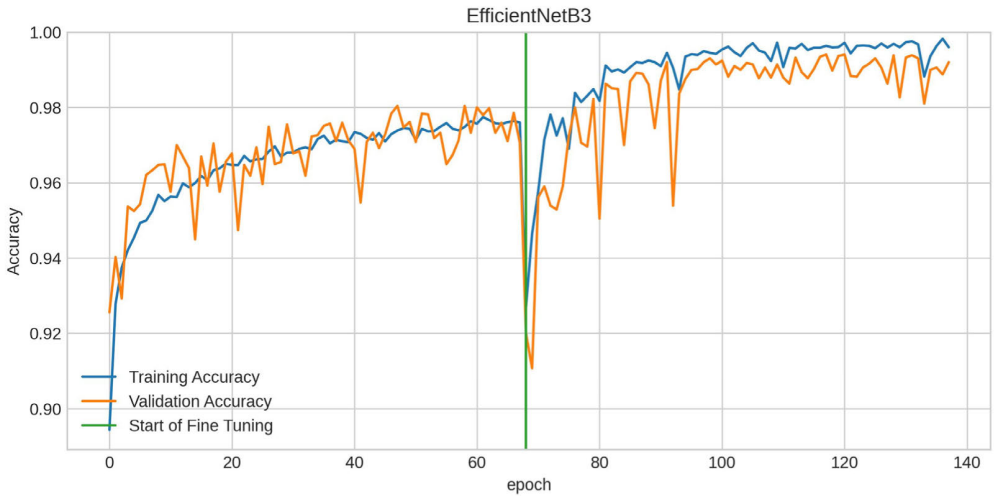


FIGURE 9. Accuracy-epoch graph of UCSD Ped1 dataset.

Since VGG19 and ResNet101V2 base models were stuck in local minimums and their VAD performance did not improve during the FT phase, the DL models where these used base models were not able to identify anomalies reported the lowest AUC scores.

3) COMPARISONS OF COMPUTATIONAL RESOURCES ANALYSES

Comparisons of the processing time analyses (seconds per frame) for the CUHK Avenue, UCSD Ped1, and UCSD Ped2 test datasets are given in Table 15, Table 16, and Table 17, respectively. While training time represents the total period of time needed for the TL and FT phases for a frame, prediction time represents the time needed to predict if a frame is normal or anomalous. The proposed approach, where VGG16 and VGG19 were used as base models, had fastest training and prediction times for all three datasets. Considering the low prediction time of a frame, it can be deduced that the proposed TL-FT-based VAD approach can be employed for real-time VAD applications.

Computational resources in terms of GPU hours for the CUHK Avenue, UCSD Ped1, and UCSD Ped2 test datasets are presented in Table 18, Table 19, and Table 20, respectively. TL phase represents the total period of GPU hours needed for TL, FT phase signifies the total GPU hours required for the FT phase, and total time denotes the total period of GPU hours taken for the TL and FT phases. As Table 18, Table 19, and Table 20 show, the proposed approach, where VGG16 and VGG19 were used as base models, was observed to have the shortest GPU hours for the TL and FT phases and reported the least total time for all three datasets. It is noticed that the highest TL phase, FT phase, and total time values were obtained where EfficientNetB5, EfficientNetB6, and NASNetMobile were used as base models for three datasets. It can further be observed that while base models with a higher number of layers needed more GPU hours for the TL and FT phases and yielded higher total time values, those with a lower number of layers needed shorter GPU hours for the TL and FT phases and yielded less total time. As can be seen from Table 18, Table 19, and

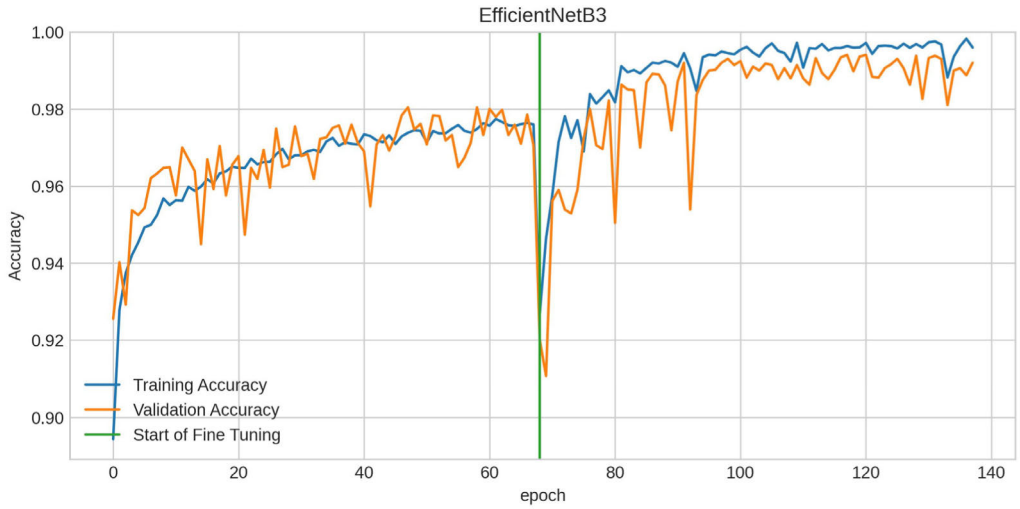


FIGURE 10. Accuracy-epoch graph of UCSD Ped2 dataset.

TABLE 15. Processing time analysis of CUHK Avenue test dataset.

Model	Training Time	Prediction Time	Prediction (fps)
VGG16	0.1072	0.0008	1,250
VGG19	0.1117	0.0008	1,250
Xception	0.2004	0.0016	625
MobileNetV3Small	0.1763	0.0016	625
MobileNetV3Large	0.2407	0.0016	625
InceptionV3	0.2111	0.0024	417
EfficientNetB0	0.3620	0.0024	417
EfficientNetB1	0.3836	0.0033	303
EfficientNetB2	0.5016	0.0033	303
EfficientNetB3	0.5231	0.0033	303
EfficientNetB4	0.4743	0.0041	244
EfficientNetB5	0.6520	0.0049	204
EfficientNetB6	0.6940	0.0057	175
EfficientNetB7	0.6699	0.0073	137
EfficientNetV2B0	0.4140	0.0024	417
ResNet101V2	0.1614	0.0033	303
DenseNet121	0.3297	0.0033	303
NASNetMobile	0.5729	0.0057	175
NASNetLarge	1.0848	0.0081	123
ConvNeXtTiny	0.5363	0.0049	204

Table 20, while the total period of GPU hours needed for the UCSD Ped2 test dataset varied within 0.19–0.70, it varied within 0.52–3.56 GPU hours for the UCSD Ped1 dataset and 0.73–7.39 GPU hours for the CUHK Avenue dataset. We noticed that the total amount of GPU time tends to grow as the dataset size and number of layers in the base model increase.

We also performed a comprehensive run-time analysis comparison of different methods in Table 21. We compared the computational efficiency of our proposed method with several methods in the literature. Our approach is able to predict if a frame is anomalous in a much less amount of time compared to other methods. Depending on the base model type, the proposed method can predict the class of a frame in the CUHK Avenue dataset within 0.0008–0.0081 seconds (i.e., between 123 and 1,250 fps), can predict the class of a frame in the UCSD Ped1 dataset within 0.0018–0.0125 seconds (i.e., between 80 and 556 fps), and can predict the class of a frame in the UCSD Ped2 dataset within 0.0055–0.0328 seconds (i.e., between 30 and 182 fps).

TABLE 16. Processing time analysis of UCSD Ped1 test dataset.

Model	Training Time	Prediction Time	Prediction (fps)
VGG16	0.3111	0.0018	556
VGG19	0.1656	0.0018	556
Xception	0.1795	0.0018	556
MobileNetV3Small	0.2975	0.0018	556
MobileNetV3Large	0.3053	0.0036	278
InceptionV3	0.2610	0.0036	278
EfficientNetB0	0.3517	0.0036	278
EfficientNetB1	0.2901	0.0053	189
EfficientNetB2	0.3335	0.0053	189
EfficientNetB3	0.3879	0.0053	189
EfficientNetB4	0.6466	0.0071	141
EfficientNetB5	0.6948	0.0089	112
EfficientNetB6	0.4879	0.0089	112
EfficientNetB7	1.1225	0.0107	93
EfficientNetV2B0	0.4207	0.0053	189
ResNet101V2	0.2780	0.0036	278
DenseNet121	0.2660	0.0053	189
NASNetMobile	0.8361	0.0089	112
NASNetLarge	1.1450	0.0125	80
ConvNeXtTiny	0.5363	0.0071	141

TABLE 17. Processing time analysis of UCSD Ped2 test dataset.

Model	Training Time	Prediction Time	Prediction (fps)
VGG16	0.1840	<0.0000	-
VGG19	0.1854	<0.0000	-
Xception	0.1977	0.0055	182
MobileNetV3Small	0.2065	0.0055	182
MobileNetV3Large	0.1782	0.0055	182
InceptionV3	0.3244	0.0109	92
EfficientNetB0	0.2975	0.0109	92
EfficientNetB1	0.2641	0.0109	92
EfficientNetB2	0.3041	0.0164	61
EfficientNetB3	0.2679	0.0164	61
EfficientNetB4	0.3153	0.0164	61
EfficientNetB5	0.4560	0.0219	46
EfficientNetB6	0.6907	0.0219	46
EfficientNetB7	0.6087	0.0273	37
EfficientNetV2B0	0.1925	0.0109	92
ResNet101V2	0.3798	0.0109	92
DenseNet121	0.3101	0.0109	92
NASNetMobile	0.4310	0.0328	30
NASNetLarge	0.4834	0.0328	30
ConvNeXtTiny	0.4974	0.0109	92

Due to the fact that higher frame rates per second, i.e., fps, correspond to quicker anomaly detection, considering the

TABLE 18. Computational resources analysis of CUHK avenue test dataset.

Model	TL Phase	FT Phase	Total Time (h)
VGG16	0.21	0.52	0.73
VGG19	0.51	0.25	0.76
Xception	0.55	0.82	1.37
MobileNetV3Small	0.53	0.67	1.20
MobileNetV3Large	0.44	1.20	1.64
InceptionV3	0.29	1.15	1.44
EfficientNetB0	1.19	1.27	2.47
EfficientNetB1	1.22	1.39	2.61
EfficientNetB2	1.37	2.05	3.42
EfficientNetB3	1.62	1.95	3.56
EfficientNetB4	2.20	1.03	3.23
EfficientNetB5	2.09	2.36	4.44
EfficientNetB6	2.27	2.46	4.73
EfficientNetB7	1.11	3.46	4.56
EfficientNetV2B0	1.51	1.31	2.82
ResNet101V2	0.57	0.53	1.10
DenseNet121	0.77	1.48	2.25
NASNetMobile	1.30	2.60	3.90
NASNetLarge	2.82	4.57	7.39
ConvNeXtTiny	2.13	1.52	3.65

TABLE 19. Computational resources analysis of UCSD Ped1 test dataset.

Model	TL Phase	FT Phase	Total Time (h)
VGG16	0.72	0.25	0.97
VGG19	0.40	0.12	0.52
Xception	0.26	0.29	0.56
MobileNetV3Small	0.48	0.44	0.93
MobileNetV3Large	0.45	0.50	0.95
InceptionV3	0.22	0.59	0.81
EfficientNetB0	0.73	0.36	1.09
EfficientNetB1	0.38	0.52	0.90
EfficientNetB2	0.50	0.54	1.04
EfficientNetB3	0.55	0.66	1.21
EfficientNetB4	0.61	1.40	2.01
EfficientNetB5	1.03	1.13	2.16
EfficientNetB6	0.81	0.71	1.52
EfficientNetB7	1.56	1.93	3.49
EfficientNetV2B0	0.87	0.44	1.31
ResNet101V2	0.64	0.22	0.87
DenseNet121	0.15	0.68	0.83
NASNetMobile	1.62	0.98	2.60
NASNetLarge	2.12	1.44	3.56
ConvNeXtTiny	0.89	0.78	1.67

achieved fps rates that are relatively higher for the three public VAD datasets, we believe that our TL-FT-based algorithm is not only fast and effective but also an efficient solution for real-time VAD. We noticed from Table 21 that running an algorithm on the GPU results in a performance improvement of up to 30 times compared to running an algorithm on the CPU. This indicates that GPU cards can operate at a speed that is around 30 times higher than a CPU's. For this reason, due to computationally intensive operations performed in DL techniques, it is recommended to run the VAD algorithm on a GPU for a shorter inference time.

VI. DISCUSSION

The majority of previous studies suggest specific DL-based CV models that need substantial amounts of data for training; however, this restricts their use in situations where there is plenty of data available. In addition, the training time of these models increases exponentially with the volume of data, which makes them unfeasible to use in environments where the models need to continuously learn. Therefore, we propose a novel TL-FT-based VAD method that utilizes TF to extract

TABLE 20. Computational resources analysis of UCSD Ped2 test dataset.

Model	TL Phase	FT Phase	Total Time (h)
VGG16	0.08	0.11	0.19
VGG19	0.13	0.06	0.19
Xception	0.10	0.10	0.20
MobileNetV3Small	0.09	0.12	0.21
MobileNetV3Large	0.10	0.08	0.18
InceptionV3	0.21	0.12	0.33
EfficientNetB0	0.13	0.17	0.30
EfficientNetB1	0.11	0.16	0.27
EfficientNetB2	0.14	0.17	0.31
EfficientNetB3	0.16	0.11	0.27
EfficientNetB4	0.15	0.17	0.32
EfficientNetB5	0.25	0.22	0.46
EfficientNetB6	0.40	0.30	0.70
EfficientNetB7	0.31	0.31	0.62
EfficientNetV2B0	0.10	0.10	0.20
ResNet101V2	0.14	0.24	0.38
DenseNet121	0.12	0.19	0.31
NASNetMobile	0.22	0.22	0.44
NASNetLarge	0.21	0.28	0.49
ConvNeXtTiny	0.32	0.18	0.50

significant features from video footage. Furthermore, since our approach takes advantage of using DL models with pre-trained weights, we managed to reduce the total amount of time needed for training the VAD method, which can be observed in Table 18, Table 19, and Table 20. The total period of training time varied within 0.19–0.70 hours for the UCSD Ped2 dataset, 0.52–3.56 hours for the UCSD Ped2 dataset, and 0.73–7.39 hours for the CUHK Avenue dataset using an NVIDIA A100-SXM4 GPU. Compared to SOTA VAD methods in the literature that take one hour for UCSD Ped2 and 15 hours for CUHK Avenue to train the model with an NVIDIA GTX TITAN Xp [49] and 16 hours to train the model using an NVIDIA Tesla V100 GPU card [23], our approach is observed to be an efficient solution in terms of training time. Moreover, as can be observed from Table 21, our TL-FT-based VAD algorithm is able to detect anomalies much faster than other methods in the literature, which demonstrates its effectiveness in terms of run-time.

In addition to the advantages of using pre-trained DL models for VAD, there are also some challenges. One of its challenges is the computational complexity of thoroughly examining various DL models to choose the best one. Researchers need to empirically test and validate to find out the most efficient model for their requirements. Studying the security implications of pre-trained DL models is also necessary, as vulnerabilities and attacks targeting these models need to be considered.

There are some limitations to using pre-trained DL models for VAD as well. Long training times when the model is run on limited hardware resources and the requirement for a large number of instances to obtain a satisfactory performance can be regarded as two major limitations. (i) The possibility of domain mismatch, (ii) large-scale pre-training data dependence, (iii) the inability to handle data quality, (iv) the absence of explainability, (v) the likelihood of shifting labels, and (vi) the risk of changing the model's focus are some other limitations [78] of using pre-trained DL models for VAD purposes.

TABLE 21. Run-time analysis of different methods.

Model	Platform/OS	CPU	CPU Time (s)	RAM Size	GPU Type	GPU Time (s)	Dataset
[28]	MATLAB	Pentium 3 GHz	25	2 GB	-	-	UCSD Ped UCSD Ped2
[79]	MATLAB	2.6 GHz	3.8	2 GB	-	-	UCSD Ped1
[4]	MATLAB 2012	3.4 GHz	0.00667–0.00714 (140–150 fps)	8 GB	-	-	CUHK Avenue UCSD Ped1
[80]	-	-	0.22 0.29	-	-	-	UCSD Ped1 UCSD Ped2
[81]	PC	Intel Q9550	0.19 0.22	4 GB	-	-	UCSD Ped1 UCSD Ped2
[5]	-	Intel Xeon E5-2620	0.2027 (~5 fps)	-	NVIDIA Maxwell Titan X	0.0070 (~143 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[33]	-	Intel Core i7 2.3 GHz	0.05 (~20 fps)	8 GB	-	-	CUHK Avenue
[82]	MATLAB and C++ based on the Caffe framework built on a PC	Multi-core 2.1 GHz	-	32 GB	NVIDIA Quadro K4000	5.2 7.5 (with foreground detection)	UCSD Ped1 UCSD Ped2
[7]	TensorFlow Samsung SSD 850 PRO	Intel Xeon(R) E-2643 3.40 GHz	-	-	NVIDIA GeForce TITAN	0.04 (~25 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[83]	Caffe framework	-	-	-	NVIDIA TITAN	0.0027 (~370 fps)	UCSD Ped2
[84]	Python	2.1 GHz	-	-	NVIDIA GTX TITAN X	0.027 0.033	UCSD Ped1 UCSD Ped2
[34]	Python with Keras built on a PC running Ubuntu 14.04 OS	2.6 GHz	-	64 GB	NVIDIA TITAN X	0.0012	CUHK Avenue UCSD Ped1 UCSD Ped2
[85]	PC	Intel i7-7700 3.6 GHz	0.048	8 GB	-	-	UCSD Ped2
[36]	Python built on a PC running Windows 10 Pro OS	Intel Core i7-7700K 4.20 GHz	0.251 (~4 fps)	16 GB DDR4	NVIDIA GeForce GTX 10170 with 256-bit 1920 CUDA cores and 8 GB GDRR5 RAM	0.024 (~42.5 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[39]	-	-	-	-	NVIDIA TITAN Xp with 12 GB RAM	0.090 (~11 fps)	CUHK Avenue UCSD Ped2
[40]	PyTorch	-	-	-	NVIDIA GeForce 1080Ti	0.0262 (~38 fps)	UCSD Ped2
[41]	Python with TensorFlow and MATLAB	-	0.10 (10 fps) *	-	-	-	CUHK Avenue UCSD Ped2
[86]	TensorFlow	Intel Xeon E5-2690 2.60 GHz	-	-	NVIDIA GeForce GTX 1080 Ti	0.10 (~10 fps)	CUHK Avenue UCSD Ped2
[39]	-	-	-	-	NVIDIA Titan Xp with 12 GB RAM	0.090 (~11 fps)	CUHK Avenue UCSD Ped2
[44]	Python and MATLAB based on Keras backend TensorFlow	Intel Core i9-7960x 2.8 GHz	0.334 0.315 0.319	24 GB	NVIDIA GeForce GTX 1080 Ti with NVIDIA CUDA Cores 3584 (used for training)	-	CUHK Avenue UCSD Ped1 UCSD Ped2
[87]	-	-	-	-	NVIDIA TITAN Xp	0.5556 (~18 fps)	CUHK Avenue UCSD Ped2
[88]	Python with Keras	Intel i7	-	32 GB	NVIDIA GeForce 1050	0.312	UCSD Ped1 UCSD Ped2
[45]	-	-	-	-	NVIDIA Tesla P40	0.0333 (30 fps)	UCSD Ped2
[46]	-	Intel i7-8700k CPU	-	-	NVIDIA GeForce RTX 2070 with 8 GB RAM	0.03125 (32 fps)	CUHK Avenue UCSD Ped2
[47]	TensorFlow running Ubuntu Mate 16.04 OS	Intel Xeon E5-2620 2.1 GHz	-	128 GB	NVIDIA TITANX	0.031 0.036	UCSD Ped1 UCSD Ped2
[21]	-	-	-	-	8-GPU machine with Quadro RTX 6000 cards with 24 GB VRAM each	0.027	CUHK Avenue UCSD Ped1 UCSD Ped2
[49]	PyTorch	-	-	-	NVIDIA GTX TITAN Xp	0.0149 (~67 fps)	UCSD Ped2
[50]	-	i7-6700	-	-	GTX 1080	0.050 (20 fps)	CUHK Avenue UCSD Ped2
[52]	PyTorch	Intel E5-2630	-	45 GB	NVIDIA Tesla P40	0.109	CUHK Avenue

TABLE 21. *(Continued.)* **Run-time analysis of different methods.**

[51]	PyTorch	Intel Core(TM) i9-7920X 2.90	-	-	NVIDIA RTX 3090	0.015 (~10 fps)	CUHK Avenue UCSD Ped2
[65]	PyTorch built on Ubuntu OS	-	-	-	NVIDIA GTX 2070	0.0703–0.1042 (9.59 and 14.22 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[58]	PyTorch	Intel(R) Xeon(R) 6230 2.10 GHz	-	-	NVIDIA Tesla V100	0.025 (~40 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[59]	-	-	0.0303 (~33 fps)*	-	-	-	CUHK Avenue UCSD Ped2
[66]	-	-	-	-	NVIDIA GeForce Titan Xp	0.0312 (~32 fps)	UCSD Ped2
[67]	Python with TensorFlow	-	-	-	8-core NVIDIA GeForce GTX 1080	0.0151 (~67 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2
[64]	PyTorch	Intel E5-2697 2.3 GHz	-	-	GeForce GTX 1080Ti with 32 GB RAM	0.0891	CUHK Avenue UCSD Ped1 UCSD Ped2
Ours	Python with Keras and TensorFlow built on Google Colab	-	-	83.5 GB	NVIDIA A100-SXM4 GPU with 40 GB RAM	0.0008–0.0081 (123–1,250 fps) 0.0018–0.0125 (80–556 fps) 0.0055–0.0328 (30–182 fps)	CUHK Avenue UCSD Ped1 UCSD Ped2

*Not explicitly indicated if it is measured on a CPU or GPU, so it is assumed to be measured on a CPU.

VII. CONCLUSION

A frame-level VAD method based on the TL and FT approaches was proposed in this study, where 20 Keras applications that are popular in the literature were used as the base models. Anomaly detection performances of the models were improved by performing FT. All experimental studies were carried out in a Google Colab environment, using the UCSD Ped1, UCSD Ped2, and CUHK Avenue public datasets. The performances of the models were measured by calculating AUC, accuracy, precision, recall, and F1-score values. It was observed that the proposed approach achieves SOTA VAD performance, where 100% AUC and accuracy values were obtained for both UCSD Ped1 and UCSD Ped2 test datasets. The highest AUC and accuracy scores for the CUHK Avenue test dataset were observed at 98.41% and 99.10%, respectively. We observed that EfficientNet architectures achieve higher performance than the other architectures for the benchmark datasets. As a result, when compared to SOTA VAD approaches, the proposed VAD framework has demonstrated better performance, considering high AUC scores and average accuracy rates for the benchmark datasets. Based on the outcomes of the experiments, it is worth mentioning that the proposed technique seems to be suitable for real-time VAD applications, considering the low prediction times.

While satisfactory VAD detection performances were achieved using VGG19 and ResNet101V2 as base models for the UCSD Ped2 dataset, these models were stuck in local minimums, and their detection performance did not improve during the FT phase for the UCSD Ped1 and Avenue datasets.

Considering the run-time analysis of different methods, our approach is not only able to quickly detect anomalies compared to SOTA methods in the literature but also effective in terms of training time.

In the future, we plan to optimize the hyperparameters to further enhance VAD performance and make additional improvements to the proposed method for anomaly detection at the pixel level.

ACKNOWLEDGMENT

The authors thank TUBITAK for its support.

REFERENCES

[1] A. Derya and Y. Sönmez, "Trafik video analiz verilerinden anomali tespit ve diferansiyel gelişim algoritmasıUç Öçrenme makinesi ile sınıflandırma," *Int. J. Innov. Eng. Appl.*, vol. 5, no. 2, pp. 115–124, 2021.

[2] D. R. Patrikar and M. R. Parate, "Anomaly detection using edge computing in video surveillance system: Review," *Int. J. Multimedia Inf. Retr.*, vol. 11, no. 2, pp. 85–110, Jun. 2022, doi: 10.1007/s13735-022-00227-8.

[3] *UCSD Anomaly Detection Dataset*. Accessed: Jan. 12, 2023. [Online]. Available: http://www.svcl.ucsd.edu/projects/anomaly/dataset.html

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. J. Samuel and F. Cuzzolin, "Unsupervised anomaly detection for a smart autonomous robotic assistant surgeon (SARAS) using a deep residual autoencoder," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7256–7261, Oct. 2021.

[9] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6479–6488.

[10] J. R. Medel, *Anomaly Detection Using Predictive Convolutional Long Short-Term Memory Units*. Rochester, NY, USA: Rochester Institute of Technology, 2016.

[11] 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.

[12] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–19.

[13] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," 2015, *arXiv:1511.06309*.

[14] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 555–560, Mar. 2008.

[15] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo, "Unsupervised extraction of video highlights via robust recurrent auto-encoders," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4633–4641.

[16] H. Tran and D. Hogg, "Anomaly detection using a convolutional winner-take-all autoencoder," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–11.

[17] J. Ryan Medel and A. Savakis, "Anomaly detection in video using predictive convolutional long short-term memory networks," 2016, *arXiv:1612.00390*.

[18] M. Ravanbakhsh, E. Sangineto, M. Nabi, and N. Sebe, "Training adversarial discriminators for cross-channel abnormal event detection in crowds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1896–1904.

[19] Univ. Minnesota. *UMN Anomaly Detection Dataset*. Accessed: Jun. 11, 2023. [Online]. Available: http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi

[20] M. Z. Zaheer, J. H. Lee, S.-I. Lee, and B.-S. Seo, "A brief survey on contemporary methods for anomaly detection in videos," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2019, pp. 472–473.

[21] S. D. Jackson and F. Cuzzolin, "SVD-GAN for real-time unsupervised video anomaly detection," in *Proc. Brit. Mach. Vis. Conf.*, 2021, pp. 22–25.

[22] W. Shin, S.-J. Bu, and S.-B. Cho, "3D-convolutional neural network with generative adversarial network and autoencoder for robust anomaly detection in video surveillance," *Int. J. Neural Syst.*, vol. 30, no. 6, Jun. 2020, Art. no. 2050034.

[23] 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.

[24] S. Chandrakala, K. Deepak, and G. Revathy, "Anomaly detection in surveillance videos: A thematic taxonomy of deep models, review and performance analysis," *Artif. Intell. Rev.*, vol. 56, no. 4, pp. 3319–3368, Apr. 2023.

[25] T. M. Tran, T. N. Vu, N. D. Vo, T. V. Nguyen, and K. Nguyen, "Anomaly analysis in images and videos: A comprehensive review," *ACM Comput. Surveys*, vol. 55, no. 7, pp. 1–37, Jul. 2023.

[26] E. Şengönül, R. Samet, Q. A. Al-Haija, A. Alqahtani, B. Alturki, and A. A. Alsulami, "An analysis of artificial intelligence techniques in surveillance video anomaly detection: A comprehensive survey," *Appl. Sci.*, vol. 13, no. 8, p. 4956, Apr. 2023.

[27] S. A. Jebur, K. A. Hussein, H. K. Hoomod, L. Alzubaidi, and J. Santamaría, "Review on deep learning approaches for anomaly event detection in video surveillance," *Electronics*, vol. 12, no. 1, p. 29, Dec. 2022.

[28] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1975–1981.

[29] 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.

[30] Y. Zhang, H. Lu, L. Zhang, X. Ruan, and S. Sakai, "Video anomaly detection based on locality sensitive hashing filters," *Pattern Recognit.*, vol. 59, pp. 302–311, Nov. 2016.

[31] R. V. H. M. Colque, C. Caetano, M. T. L. de Andrade, and W. R. Schwartz, "Histograms of optical flow orientation and magnitude and entropy to detect anomalous events in videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 3, pp. 673–682, Mar. 2017.

[32] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua, "Spatio-temporal AutoEncoder for video anomaly detection," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1933–1941.

[33] 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.

[34] S. Yan, J. S. Smith, W. Lu, and B. Zhang, "Abnormal event detection from videos using a two-stream recurrent variational autoencoder," *IEEE Trans. Cognit. Develop. Syst.*, vol. 12, no. 1, pp. 30–42, Mar. 2020.

[35] S. Lee, H. G. Kim, and Y. M. Ro, "STAN: Spatio-temporal adversarial networks for abnormal event detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 1323–1327.

[36] E. Duman and O. A. Erdem, "Anomaly detection in videos using optical flow and convolutional autoencoder," *IEEE Access*, vol. 7, pp. 183914–183923, 2019.

[37] H. Vu, T. D. Nguyen, T. Le, W. Luo, and D. Phung, "Robust anomaly detection in videos using multilevel representations," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5216–5223.

[38] H. Song, C. Sun, X. Wu, M. Chen, and Y. Jia, "Learning normal patterns via adversarial attention-based autoencoder for abnormal event detection in videos," *IEEE Trans. Multimedia*, vol. 22, no. 8, pp. 2138–2148, Aug. 2020.

[39] R. T. Ionescu, F. S. Khan, M.-I. Georgescu, and L. Shao, "Object-centric auto-encoders and dummy anomalies for abnormal event detection in video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7834–7843.

[40] 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.

[41] W. Luo, W. Liu, D. Lian, J. Tang, L. Duan, X. Peng, and S. Gao, "Video anomaly detection with sparse coding inspired deep neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 1070–1084, Mar. 2021.

[42] M. Cho, T. Kim, W. Jin Kim, S. Cho, and S. Lee, "Unsupervised video anomaly detection via normalizing flows with implicit latent features," 2020, *arXiv:2010.07524*.

[43] D. Chen, P. Wang, L. Yue, Y. Zhang, and T. Jia, "Anomaly detection in surveillance video based on bidirectional prediction," *Image Vis. Comput.*, vol. 98, Jun. 2020, Art. no. 103915.

[44] T. Ganokratanaa, S. Aramvith, and N. Sebe, "Unsupervised anomaly detection and localization based on deep spatiotemporal translation network," *IEEE Access*, vol. 8, pp. 50312–50329, 2020.

[45] Y. Tang, L. Zhao, S. Zhang, C. Gong, G. Li, and J. Yang, "Integrating prediction and reconstruction for anomaly detection," *Pattern Recognit. Lett.*, vol. 129, pp. 123–130, Jan. 2020.

[46] K. Doshi and Y. Yilmaz, "Continual learning for anomaly detection in surveillance videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1025–1034.

[47] D. Chen, L. Yue, X. Chang, M. Xu, and T. Jia, "NM-GAN: Noise-modulated generative adversarial network for video anomaly detection," *Pattern Recognit.*, vol. 116, Aug. 2021, Art. no. 107969.

[48] X. Feng, D. Song, Y. Chen, Z. Chen, J. Ni, and H. Chen, "Convolutional transformer based dual discriminator generative adversarial networks for video anomaly detection," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 5546–5554.

[49] 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.

[50] Y. Cai, J. Liu, Y. Guo, S. Hu, and S. Lang, "Video anomaly detection with multi-scale feature and temporal information fusion," *Neurocomputing*, vol. 423, pp. 264–273, Jan. 2021.

[51] Z. Liu, Y. Nie, C. Long, Q. Zhang, and G. Li, "A hybrid video anomaly detection framework via memory-augmented flow reconstruction and flow-guided frame prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 13568–13577.

[52] X. Wang, Z. Che, B. Jiang, N. Xiao, K. Yang, J. Tang, J. Ye, J. Wang, and Q. Qi, "Robust unsupervised video anomaly detection by multipath frame prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2301–2312, Jun. 2022.

[53] L. Wang, H. Tan, F. Zhou, W. Zuo, and P. Sun, "Unsupervised anomaly video detection via a double-flow ConvLSTM variational autoencoder," *IEEE Access*, vol. 10, pp. 44278–44289, 2022.

[54] A. Guo, L. Guo, R. Zhang, Y. Wang, and S. Gao, "Self-trained prediction model and novel anomaly score mechanism for video anomaly detection," *Image Vis. Comput.*, vol. 119, Mar. 2022, Art. no. 104391.

[55] Q. Zhang, G. Feng, and H. Wu, "Surveillance video anomaly detection via non-local U-Net frame prediction," *Multimedia Tools Appl.*, vol. 81, no. 19, pp. 27073–27088, Aug. 2022.

[56] C. Huang et al., "Self-supervised attentive generative adversarial networks for video anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9389–9403, Nov. 2023, doi: 10.1109/TNNLS.2022.3159538.

[57] 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.

[58] Y. Hao, J. Li, N. Wang, X. Wang, and X. Gao, "Spatiotemporal consistency-enhanced network for video anomaly detection," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108232.

[59] 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.

[60] L. Sun, Z. Wang, Y. Zhang, and G. Wang, "A feature-trajectory-smoothed high-speed model for video anomaly detection," *Sensors*, vol. 23, no. 3, p. 1612, Feb. 2023.

[61] A. Kumar and M. Khari, "Efficient video anomaly detection using residual variational autoencoder," in *Proc. Int. Conf. Commun. Syst., Comput. IT Appl. (CSCITA)*, Mar. 2023, pp. 50–55.

[62] M. H. Sharif, L. Jiao, and C. W. Omlin, "Deep crowd anomaly detection by fusing reconstruction and prediction networks," *Electronics*, vol. 12, no. 7, p. 1517, Mar. 2023.

[63] H. S. Modi and D. A. Parikh, "An intelligent unsupervised anomaly detection in videos using inception capsule auto encoder," *Imag. Sci. J.*, vol. 72, no. 2, pp. 267–284, Feb. 2024.

[64] N. Taghinezhad and M. Yazdi, "A new unsupervised video anomaly detection using multi-scale feature memorization and multipath temporal information prediction," *IEEE Access*, vol. 11, pp. 9295–9310, 2023.

[65] X. Zeng, Y. Jiang, W. Ding, H. Li, Y. Hao, and Z. Qiu, "A hierarchical spatio-temporal graph convolutional neural network for anomaly detection in videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 1, pp. 200–212, Jan. 2023.

[66] Y. Chang, Z. Tu, W. Xie, B. Luo, S. Zhang, H. Sui, and J. Yuan, "Video anomaly detection with spatio-temporal dissociation," *Pattern Recognit.*, vol. 122, Feb. 2022, Art. no. 108213.

[67] N. Aslam and M. H. Kolekar, "DeMAAE: Deep multiplicative attention-based autoencoder for identification of peculiarities in video sequences," *Vis. Comput.*, vol. 40, no. 3, pp. 1729–1743, Mar. 2024.

[68] Q. Zhang, H. Wei, J. Chen, X. Du, and J. Yu, "Video anomaly detection based on attention mechanism," *Symmetry*, vol. 15, no. 2, p. 528, Feb. 2023.

[69] W. Ullah, T. Hussain, F. U. M. Ullah, M. Y. Lee, and S. W. Baik, "TransCNN: Hybrid CNN and transformer mechanism for surveillance anomaly detection," *Eng. Appl. Artif. Intell.*, vol. 123, Aug. 2023, Art. no. 106173.

[70] R. Nayak, U. C. Pati, and S. K. Das, "A comprehensive review on deep learning-based methods for video anomaly detection," *Image Vis. Comput.*, vol. 106, Feb. 2021, Art. no. 104078.

[71] Y. Li, T. Guo, R. Xia, and W. Xie, "Road traffic anomaly detection based on fuzzy theory," *IEEE Access*, vol. 6, pp. 40281–40288, 2018.

[72] 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.

[73] L. Li, Z. Wang, Q. Hu, and Y. Dong, "Adaptive nonconvex sparsity based background subtraction for intelligent video surveillance," *IEEE Trans. Ind. Informat.*, vol. 17, no. 6, pp. 4168–4178, Jun. 2021.

[74] F. Guo, Z. Wang, S. Du, H. Li, H. Zhu, Q. Pei, Z. Cao, and J. Zhao, "Detecting vehicle anomaly in the edge via sensor consistency and frequency characteristic," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5618–5628, Jun. 2019.

[75] Ata-Ur-Rehman, S. Tariq, H. Farooq, A. Jaleel, and S. M. Wasif, "Anomaly detection with particle filtering for online video surveillance," *IEEE Access*, vol. 9, pp. 19457–19468, 2021.

[76] S. Wan, S. Ding, and C. Chen, "Edge computing enabled video segmentation for real-time traffic monitoring in Internet of Vehicles," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108146.

[77] T. Kural. (2022). *Android Kötücül Yazýlým Analizinde Derin Örenme Modellerinin Performansýnýn Karýlatýrýlmasý*. Accessed: Aug. 10, 2023. [Online]. Available: https://avesis.gazi.edu.tr/dosya?id=9fc66459-0f50-4914-a348-8402ccb8be47

[78] Z. Zhao, L. Alzubaidi, J. Zhang, Y. Duan, and Y. Gu, "A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations," *Expert Syst. Appl.*, vol. 242, May 2024, Art. no. 122807, doi: 10.1016/j.eswa.2023.122807.

[79] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *Proc. CVPR*, Jun. 2011, pp. 3449–3456.

[80] T. Xiao, C. Zhang, and H. Zha, "Learning to detect anomalies in surveillance video," *IEEE Signal Process. Lett.*, vol. 22, no. 9, pp. 1477–1481, Sep. 2015.

[81] M. Javan Roshtkhari and M. D. Levine, "An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions," *Comput. Vis. Image Understand.*, vol. 117, no. 10, pp. 1436–1452, Oct. 2013.

[82] D. Xu, Y. Yan, E. Ricci, and N. Sebe, "Detecting anomalous events in videos by learning deep representations of appearance and motion," *Comput. Vis. Image Understand.*, vol. 156, pp. 117–127, Mar. 2017.

[83] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Comput. Vis. Image Understand.*, vol. 172, pp. 88–97, Jul. 2018.

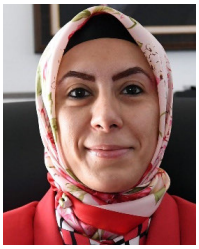
[84] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3379–3388.

[85] H. Zhu, B. Liu, Y. Lu, W. Li, and N. Yu, "Real-time anomaly detection with HMOF feature," in *Proc. 2nd Int. Conf. Video Image Process.*, Dec. 2018, pp. 49–54.

[86] M. Ye, X. Peng, W. Gan, W. Wu, and Y. Qiao, "AnoPCN: Video anomaly detection via deep predictive coding network," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 1805–1813.

[87] Y. Zhang, X. Nie, R. He, M. Chen, and Y. Yin, "Normality learning in multispace for video anomaly detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 9, pp. 3694–3706, Sep. 2021.

[88] A. Atghaei, S. Ziaeinejad, and M. Rahmati, "Abnormal event detection in urban surveillance videos using GAN and transfer learning," 2020, *arXiv:2011.09619*.



ESMA DİLEK received the B.S. degree in computer engineering from Yildiz Technical University, İstanbul, Turkey, in 2003, and the M.S. degree in computer science from Boston University, Boston, MA, USA, in 2010. She is currently pursuing the Ph.D. degree in information security engineering with Gazi University, Ankara, Turkey.

From 2010 to 2020, she was with İstanbul Metropolitan Municipality as a Senior Software Engineer, the Deputy Traffic Director, the IT Director, and the Smart City Director, respectively. She has been the General Deputy Director of the Republic of Türkiye Ministry of Transport and Infrastructure, since 2020. She has also been the President of Intelligent Transportation Systems Association of Türkiye, since 2021. Her research interests include intelligent transportation systems, computer vision, and deep learning.

Ms. Dilek was a recipient of the Certificate in Academic Excellence from Boston University Metropolitan College, in 2010, and the Best Female Government Leader of the Year Award, in 2018, from International Data Corporation Turkey.

MURAT DENER received the bachelor's degree from the Department of Electronic Computer Education, Gazi University, in 2005, and the master's and Ph.D. degrees from the Informatics Institute, Gazi University, in 2008 and 2012, respectively. He continues his academic career as a Professor with the Information Security Engineering Department, Gazi University. He is also the Head of the Information Security Engineering Department. He has been working in the field of the artificial intelligence, big data, cyber security, and the Internet of Things for nearly 20 years. He has published more than 140 international and national studies.