IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 20, 2021, accepted August 31, 2021, date of publication September 3, 2021, date of current version September 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109904

Deep Edge Computing for Videos

JUN-HWA KIM, NAMHO KIM, AND CHEE SUN WON, (Member, IEEE)
Department of Electronics and Electrical Engineering, Dongguk University, Seoul 04620, South Korea

Corresponding author: Chee Sun Won (cswon@dongguk.edu)

This work was supported by the Basic Science Research Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1D1A1B07043542.

ABSTRACT This paper provides a modular architecture with deep neural networks as a solution for real-time video analytics in an edge-computing environment. The modular architecture consists of two networks of Front-CNN (Convolutional Neural Network) and Back-CNN, where we adopt Shallow 3D CNN (S3D) as the Front-CNN and a pre-trained 2D CNN as the Back-CNN. The S3D (i.e., the Front CNN) is in charge of condensing a sequence of video frames into a feature map with three channels. That is, the S3D takes a set of sequential frames in the video shot as input and yields a learned 3 channel feature map (3CFM) as output. Since the 3CFM is compatible with the three-channel RGB color image format, we can use the output of the S3D (i.e., the 3CFM) as the input to a pre-trained 2D CNN of the Back-CNN for the transfer-learning. This serial connection of Front-CNN and Back-CNN architecture is end-to-end trainable to learn both spatial and temporal information of videos. Experimental results on the public datasets of UCF-Crime and UR-Fall Detection show that the proposed S3D-2DCNN model outperforms the existing methods and achieves state-of-the-art performance. Moreover, since our Front-CNN and Back-CNN modules have a shallow S3D and a light-weighted 2D CNN, respectively, it is suitable for real-time video recognition in edge-computing environments. We have implemented our CNN model on *NVIDIA* Jetson Nano Developer as an edge-computing device to show its real-time execution.

INDEX TERMS Edge computing, CNN, the IoT, anomaly detection, video recognition.

I. INTRODUCTION

Surveillance cameras have been increasingly deployed in public places for the purpose of monitoring abnormal events such as criminal activities and medical emergencies [1], [2]. In reality, falls are commonly occurred for the elderly and hospital inpatient, generally ranging from 3 to 11 falls per 1,000 bed days [3]. Approximately 25% of inpatient falls result in injury, including fracture, subdural hematoma, excessive bleeding, and even death [3].

Anomaly detection in a video is one of the challenging problems and has been studied for a long time in the computer vision community. The traditional anomaly detection mainly relied on the motion information between two consecutive frames extracted by optical flow [4] or dynamic Bayesian Network (DBN) [5]. Recently, deep neural networks (DNN) have been exploited to learn motion information in videos. For a video with the time-domain extension, a Convolutional 3D (C3D) neural network architecture with 3D convolutional filters was adopted as a natural extension of 2D filters of the 2D CNN [6]–[9]. The C3D learns the temporal motions as well as the spatial features from video frames. This requires for the C3D to execute complex 3D convolutions with the kernel dimension of $R^{c\times d\times d\times T}$, where c is the number of channels, d is the spatial size (i.e., $d \times d$) of the filter, and T is the number of frames in the video clips. That is, each filter in the inner layers of the C3D takes a 3D volume input and produces another 3D volume output, which requires a lot of computations and memory space. This motivates the researchers to simplify the C3D structure. A plausible approach is to expand already-trained 2D CNN (Convolutional Neural Network) coefficients into 3D space. For example, in I3D (Inflated 3D) CNN [9], the pre-trained filter coefficients of 2D CNNs are copied over the temporal direction to form a 3D CNN structure. However, the size of I3D is still $R^{c\times d\times d\times T}$, which makes no change in the inference complexity. Recently, in [10], it has been shown that a video recognition task can be done by using pre-trained 2D CNNs only. That is, we can select three frames in a video shot and change them into gray-scale images, forming a SG3I (Stacked Grayscale 3-channel Image) [10]. The SG3I is now compatible with a color image of RGB (Red, Green, Blue)

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

channels and is used to fine-tune a pre-trained 2D CNN for the video shot. Since the dimension of 2D CNN with SG3I is only of $R^{c \times d \times d}$ without the extension to the time-domain, it is much lighter than C3D and I3D. Also, with the SG3I, no optical flow computations are required, enabling fast training and testing for real-time applications.

Recently, the integration of IoT (Internet of Things) and video surveillance has led to a steep rise in the demand of IP (Internet Protocol) cameras. However, since an IP camera usually has a limited computing power, it may be necessary to send the video data to the cloud server to execute the C3D neural network (see Fig. 1-A). In this case, the network traffic may hamper the timely detection of any anomaly at the cloud. To solve this problem, we can cut the amount of video data at the edge-computing device before the transmission. For example, we can adopt the SG3I scheme [10] at the edge computing side, which needs only a simple process of the frame selection in a video shot. However, as addressed in [10], multiple SG3Is are required at the inference step to guarantee comparable results to the state-of-the-art performance. Specifically, in [10], the CNN outputs for all 10 SG3Is inputs were fused to make the final decision. So, as shown in Fig. 1-B, this requires to send multiple SG3Is to the cloud, which may also cause some delays in the recognition process. Certainly, the best solution is to complete the anomaly detection at the edge-side (see Fig. 1-C and 1-D). To this end, in this paper, we propose a joint CNN with a shallow 3D CNN, which has fewer convolutional layers and network parameters compared to conventional deep neural networks [11], and a light-weighted 2D CNN for a fast anomaly detection at the edge-computing environment.

Since a typical CNN model has stacked layers of sub-networks, we may separate the trained CNN into two parts, Front-CNN and Back-CNN. Also, we can combine two CNNs for the Front-CNN and the Back-CNN, where the Front-CNN is in charge of the pre-processing for the input of the Back-CNN. In this paper, as the Front-CNN, we use a Shallow 3D CNN (S3D), which is trained to condense multiple video frames into a single one with three channels of feature maps. As a result, the amount of the video data is reduced and the output of the S3D becomes compatible with the input of the conventional 2D CNNs. This allows us to use any pre-trained CNN for the Back-CNN with no optical flow computations, making a fast video recognition. Specifically, as shown in Fig. 1-D, we can embed the S3D (Shallow 3D) CNN into an edge computing device, where multiple video frames are fed into the S3D to produce a learned 3-channel feature map (3CFM) as the output. Note that, like the SG3I, the 3CFM fits the input format of the pre-trained CNN with RGB three channels. This enables us to fine-tune any pre-trained 2D CNN for video recognition problems without resorting to a C3D neural network.

The contributions of this paper are summarized as follows.

1) We propose a Shallow 3D CNN (S3D) as Front-CNN. The S3D is trained to condense multiple video frames into a single 3-channel feature map (3CFM).
2) Treating the 3CFM as an image with RGB channels, we use the 3CFM for an input of a pre-trained 2D CNN. This naturally forms a cascade network with the Front-CNN (S3D) and the Back-CNN (2D CNN), solving a video recognition problem without 3D filters and optical flow computations.
3) We can use the S3D as a stand-alone network to condense multiple video frames, reducing the transmission cost in a client-server framework.
4) We have evaluated the real-time performance for our cascade S3D-MobileNet network via *NVIDIA* Jetson Nano Developer.

II. BACKGROUND AND RELATED WORK

A. EDGE COMPUTING

Traditional video surveillance systems [12] demand human intervention to some extent. However, as the number of IP cameras increases explosively, a fully automatic video recognition framework becomes essential, replacing a manual monitoring. Many algorithms [4], [5], [13]–[15] have been developed to handle vast amounts of data automatically. These algorithms can be used for the video recognition in a cloud server. As an example, a violence detection [16] was performed by transmitting the video data obtained from the drone camera to the cloud server. Also, by transmitting the road video obtained from the camera to the cloud server, the license plate of the vehicle was extracted [17].

In the above scenarios, the video data captured by the camera are transmitted to the cloud server to do the entire recognition processes, which may hamper real-time video recognitions due to transmission delays through the communication channel. Alternatively, to send key information only, a simple pre-processing technique can be applied to the video acquired from the camera before transmitting to the cloud server. SWEETCAM proposed in [18] has an image processing module in the camera and can perform pre-processing tasks such as background subtraction, contour detection, and object classification. Also, one can execute a function of determining occupancy in data acquired from multiple cameras using a local binary pattern with the support vector machine classifier [19].

Video pre-processing tasks can be done at the edge-computing device [20], [21], which is located in-between the camera and the cloud server (see Fig. 2). Without doing video pre-processing at the edge-side, the cloud server should take all computational loads. Therefore, the purpose of the video pre-processing is to reduce not only the burden of data transmission but also the computational load at the cloud. Since the edge computing device is inferior to the cloud in terms of computing power, we can embed only a light-weighted DNN in the edge-side for the pre-processing tasks [22], [23].

B. NEURAL NETWORKS FOR VIDEOS

Recently, a remarkable performance improvement has been achieved by applying DNNs for video recognition problems. For example, in [24], a two-path CNN model was

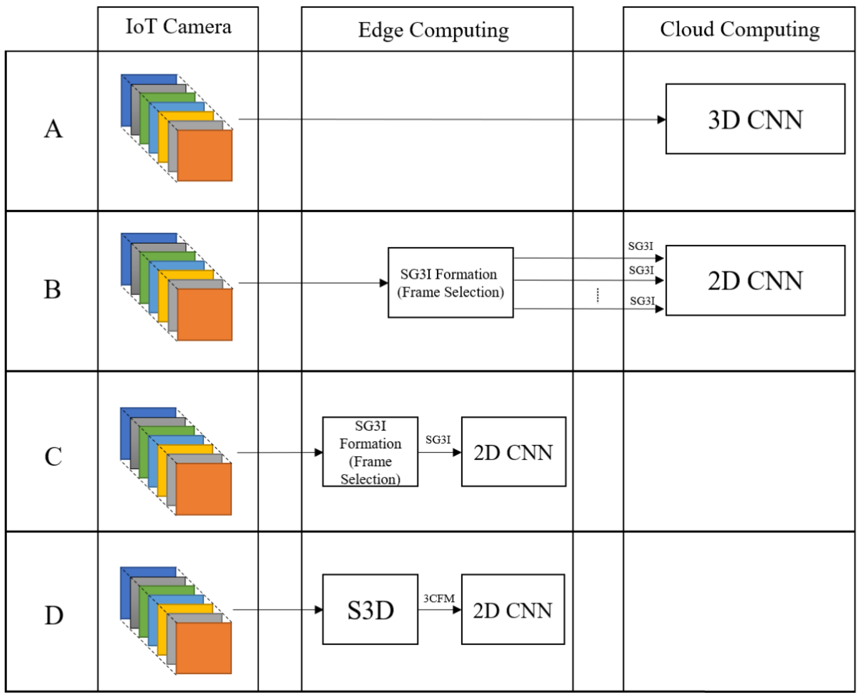


FIGURE 1. Four scenarios of incorporating neural networks in an edge computing environment. A: 3D CNN at the cloud, B: Multiple SG3Is formed by the edge-side and 2D CNN at the cloud, C: A single SG3I formation and 2D CNN at the edge-side D: A single 3CFM formed by the S3D and 2D CNN at the edge-side.

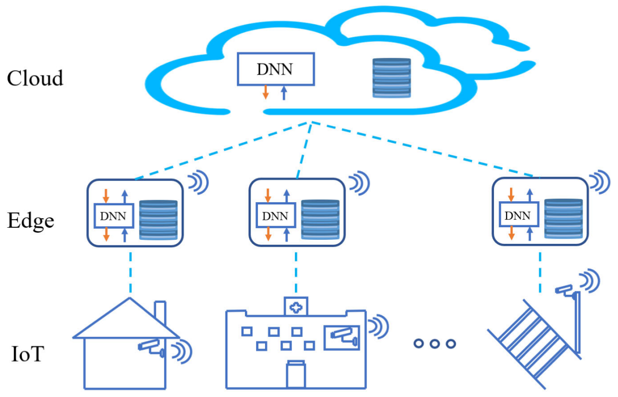


FIGURE 2. Edge computing with IoT devices.

proposed for video action recognition. The two-path CNN has a spatial-stream CNN for extracting spatial features and a temporal-stream CNN for learning temporal motions. For the input of the spatial-stream CNN, a representative frame is selected from several video frames. On the other hand, the input of the temporal stream takes a sequence of optical flows obtained from consecutive video frames. Here, the problem is that the optical flow demands a lot of computations. To avoid the optical flow computation, in [25], the dynamic image based on a rank pooling concept, which is a summarized single frame of the motion information in the multiple frames of the original video clip, was proposed. Similarly, in [10], three gray-scale images are extracted from video frames and stacked in R, G, and B channels to form an image of SG3I. The SG3I, which is compatible with the input of pre-trained 2D CNNs, requires a very small amount of computation and is suitable for real-time video recognition problems.

Previously, the optical flows were computed directly from the video frames [4]. Now, the optical flow computations have been replaced by neural networks. In [5], DBN (Dynamic Bayesian Network) was adopted to learn the fused clues of video (motion trajectory, arms and legs direction information) and audio (scream, yell). Also, in [13], LSTM (Long Short-Term Memory) was adopted to learn temporal information, while a CNN was used to extract spatial information in the video. Also, in [14], a spatio-temporal auto-encoder extracts temporal and spatial information in video by designing a 3D CNN with an AutoEncoder structure. The C3D and the ranking models were also employed in [15] for detecting video anomalies.

III. MODULAR NEURAL NETWORK

Fig. 3 shows the overall structure of our Front-CNN and Back-CNN modules. The Front-CNN is a shallow 3D convolutional neural network (S3D CNN) with only 3 layer-blocks of 3D convolutional filters, batch normalization (BN), and ReLU activation function. The Front-CNN is responsible for condensing a group of video frames into a three-channel feature map (3CFM) with the learned motion information. Then, since the 3CFM is compatible with the input of pre-trained 2D CNNs, we can adopt any pre-trained 2D CNN for the Back-CNN. That is, we can feed the 3CFM into the Back-CNN for transfer-learning. The Front-CNN of the S3D CNN enables us to use any pre-trained 2D CNN for the video recognition problem via the transfer-learning. Note that the Front-CNN can be used as a stand-alone or it can be connected with the Back-CNN as a single network for end-to-end training.

We introduce two user-scenarios for utilizing the Front-CNN as a stand-alone network. First, we can train a classifier with the output of the Front-CNN (i.e., 3CFM) to pre-screen videos. Obtaining any meaningful motion, we can feed the 3CFM into the input of the Back-CNN to have more detailed information. So, the additional work for the Back-CNN is required only when we detect a meaningful motion. In this way, we can avoid any unnecessary computations at the edge-computing side. As a second scenario, we can use the Front-CNN as a means of compressing video data. Since multiple video frames can be condensed into only 1 frame with 3 channels by the Front-CNN, we can reduce the transmission cost in an client-server computing environment. In this scenario, the video data captured at the camera are transmitted to the server for video analytics and the role of the Front-CNN is to reduce the amount of data at the client-side.

In the following sub-sections, we explain the Front-CNN, the Back-CNN, and their serial connection for the end-to-end training in more detail.

A. FRONT-CNN FOR VIDEO CONDENSATION

In this subsection, we introduce the Front-CNN that can condense a video shot with multiple frames into only three channels of feature maps using the S3D. The main element of our Front-CNN is the 3D convolution filter, which can deal with multiple video frames. As shown in Fig. 3, the Front-CNN receives video frames of $W * H * 3@T$ as an input, which are sampled from a video shot of $w * h * 3@t$ and resized such that $t \geq T$, $w \neq W$, and $h \neq H$. Passing through the three 3D convolution layers, the S3D of the Front-CNN outputs a 3CFM with $W * H * 3@1$, where T frames are now condensed into only 1 frame. The specific elements of our S3D are listed in Table 1. As shown in the table, to reduce the number of frames of the input video gradually, we set the numbers of the intermediate frames, T_{out1} and T_{out2}, such that $T > T_{out1} > T_{out2} > 1$, where T_{out1} and T_{out2} should be factors of T.

The output of the S3D (i.e., the 3CFM) has only three channels of feature maps, which are compatible with the RGB channels in a color image. This implies that the 3CFM can be used as the input of any pre-trained 2D CNN. So, we can directly connect the S3D of the Front-CNN and the pre-trained 2D CNN of the Back-CNN for the end-to-end fine-tuning. Note that the network parameters in both the S3D (i.e., the Front-CNN) and the last layers of the pre-trained CNN (i.e., the Back-CNN) are updated during the fine-tuning. That is, the network parameters in early layers of the pre-trained 2D CNN are fixed without updating. Then, since the Back-CNN has been pre-trained by general 2D images but not feature maps like the 3CFM, the 3CFM input to the Back-CNN may deteriorate the performance of the end-to-end fine-tuning. To solve this problem, we employ a skip connection in the Front-CNN, which adds the flavor of a color image selected from the video frames to the feature map of the 3CFM (see Fig. 3). Specifically, for a sequence of input frames $X = \{X_1, X_2, \ldots, X_T\}$, we have the input of Back-CNN, $Y(X_i)$, as follows

$$Y(X_i) = f(X_1, X_2, \ldots, X_T) + X_i, \tag{1}$$

where f denotes the output of the S3D and X_i, $i \in (1, \ldots, T)$, is a randomly selected frame from X. So, the input of the Back-CNN is the sum of the 3CFM and a selected RGB frame in the video shot.

Fig. 4 shows some examples of the 3CFM and $Y(X_i)$. Rows 1 through 3 are video shots of Abuse007_x264, Arrest002_x264, Shooting002_x264, respectively, in the UCF-Crime dataset. As shown in the figures, the regions of interest (RoI) for the anomalies tend to be ruddy in the 3CFM. Then, more specific regions are getting brighter by $Y(X_i)$, which are expected to help the following Back-CNN to learn the anomalies.

TABLE 1. Details of Shallow 3D convolutional neural network (S3D CNN) architecture used in the Front-CNN. In our S3D, with the strides of 4, 2, and 2 in order, we set $T = 16$, $T_{out1} = 4$, and $T_{out2} = 2$.

Layers	Number of Filters	Filter Size	Stride	Output Dimension
Convolution	16	3x3x3	$\frac{T}{T_{out1}}$x1x1	WxHx3@T_{out1}
BatchNorm				
ReLU				
Convolution	32	3x3x3	$\frac{T_{out1}}{T_{out2}}$x1x1	WxHx3@T_{out2}
BatchNorm				
ReLU				
Convolution	3	3x3x3	T_{out2}x1x1	WxHx3@1
BatchNorm				
ReLU				

B. BACK-CNN FOR VIDEO RECOGNITION

Two-stream CNN structure with spatial and temporal streams is proven to be effective for video action recognition problems. Here, the temporal stream CNN is trained by the motion information in consecutive video frames. For example, optical flow [24], dynamic images [25], and SG3I [10] can be used as the input for the temporal stream CNN. On the other hand, for the spatial stream CNN, a representative single frame of the video shot is selected as an input for the spatial stream CNN. Then, the two-stream CNN combines the features obtained in both the spatial stream and the temporal stream CNNs.

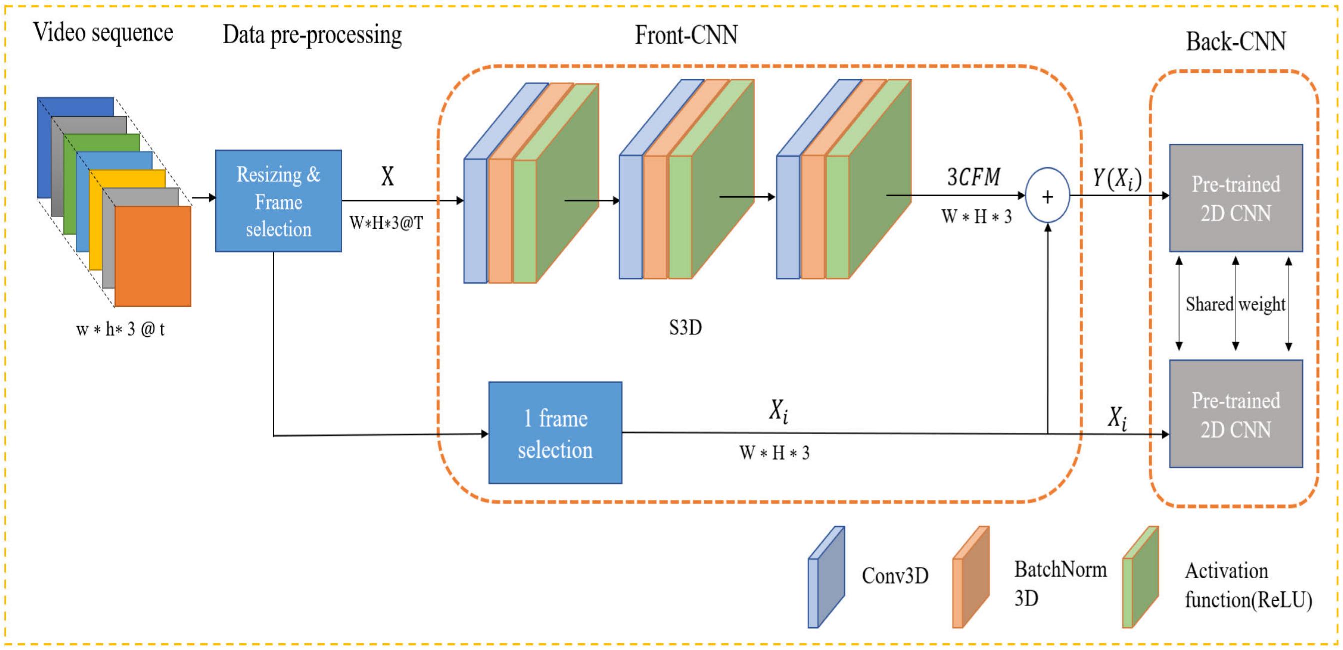


FIGURE 3. The overall structure of the serially connected Front-CNN and Back-CNN.

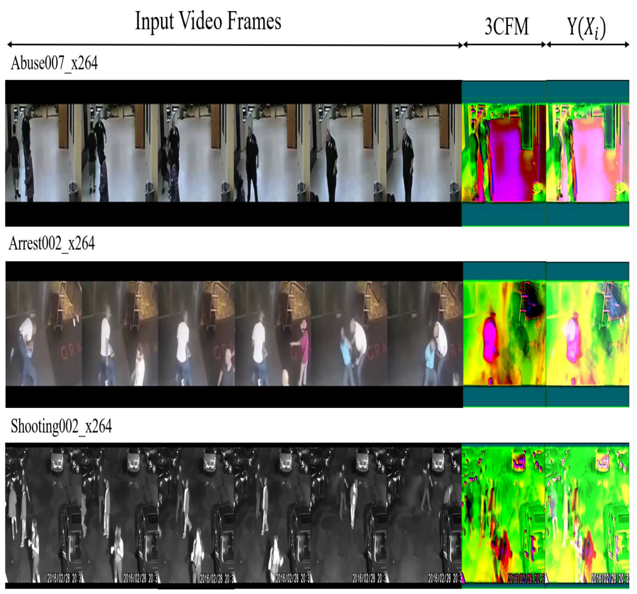


FIGURE 4. Visualization of 3CFM and $Y(X_i)$ for UCF-Crime dataset.

For our Back-CNN, we basically follow the two-stream CNN model with two pre-trained 2D CNNs in [10]. The first stream of 2D CNN is for learning the spatial information via a single frame chosen from the video shot. The second one is for learning the temporal motion by the SG3I, which is formed by stacking the three gray-scale images chosen from the video shot. On the other hand, in this paper, the input for the temporal 2D CNN is the learned $Y(X_i)$ from the Front-CNN, whereas the input for the spatial 2D CNN is the original frame X_i in the video shot. Note that the two-stream model needs to keep two 2D CNNs, which can be a burden on memory complexity in edge-computing environments. Therefore, we use only one of them and share it for learning both spatial and temporal information. Specifically, as in (2), the loss function $\mathcal{L}$ for the parameter-update is a weighted sum of two losses of $\mathcal{L}_{Y(X_i)}$ for the temporal learning and $\mathcal{L}_{X_i}$ for the spatial learning

$$\mathcal{L} = \mathcal{L}_{Y(X_i)} + \lambda \mathcal{L}_{X_i}, \tag{2}$$

where λ is a weighting factor. The parameter λ is determined experimentally and set to 0.1 for all our experiments. Now, before we make a back-propagation for the parameter-update by $\mathcal{L}$, our coordinated end-to-end training method needs to execute two forward computations to have $\mathcal{L}_{X_i}$ by X_i and $\mathcal{L}_{Y(X_i)}$ by $Y(X_i)$.

A limited computing power of edge-computing devices forces us to choose a light 2D CNN model for the Back-CNN. This leads us to consider the pre-trained MobileNet [26], [27] as the Back-CNN.

C. END-TO-END TRAINING FOR FRONT-CNN AND BACK-CNN

As shown in Fig. 3, the Front-CNN and the Back-CNN are connected in series to complete the network. Again, a shallow 3D CNN of S3D is employed for the Front-CNN and a light-weighted 2D CNN such as MobileNet-v2 [26], Mobilenet-v3(small), or MobileNet-v3(large) [27] is used for the Back-CNN. Note that the three versions of the MobileNet, which were pre-trained by the ImageNet, have relatively fewer parameters. So, they are suitable for the real-time applications.

For the training of the end-to-end network with Front-CNN and Back-CNN, all network parameters except the early layers of the Back-CNN (i.e., the pre-trained MobileNet) are updated. That is, the pre-trained parameters of the early layers of MobileNet are frozen but those of all S3D and the last layers of MobileNet are trained.

IV. EXPERIMENTS

A. DATASETS AND IMPLEMENTATION DETAILS

For the performance evaluation of the proposed network, UCF-101 [28], HMDB-51 [29], UCF-Crime [15], and UR-FALL Detection [30] video dataset were used, where UCF-101 and HMDB-51 have been widely used for performance evaluation of the video action recognition. The details of the above datasets are as follows.

- UCF-101 has 13320 video clips with 101 categories in 5 action groups (i.e., Sports, Playing Musical Instrument, Human-Object Interaction, Body-Motion, and Human-Human Interaction).
- HMDB-51 has 6766 video clips with 51 categories, which are from 5 action groups (i.e., General facial actions, Facial actions with object manipulation, General body Movements, Body Movements with object interaction, and Body movements for human interaction).
- The UCF-Crime dataset provides 800 normal and 810 anomalous video shots (video clips) for training. Also, there are 150 normal and 140 anomalous videos for testing. Although the anomalous videos contain 13 real-world anomalies, our task is a general anomaly detection. So, we consider all anomalies as one group and all normal activities as another one. The UCF-Crime training dataset has weakly supervised labels in the sense that a common label is given for each video shot but not for every individual frame. In our method, each labeled video shot is divided into multiple sub-shots, where each sub-shot is used as a basic unit for the classification. Therefore, we can just assign the same label of a video shot for all its sub-shots for training. Similarly, a sequential set of frames are grouped to form a sub-shot for testing. Fig. 5 shows two examples of the anomaly/normal situations in the UCF-Crime dataset.
- The UR Fall Detection Dataset has 40 activities of daily living (ADL) and 30 fall videos. Fall video has camera-0 version taken horizontally and camera-1 version taken vertically, and ADL video only has camera-0 version. In addition, depth information is provided using Microsoft's Kinect when recording, but only RGB images were used in our experiments. The criteria for dividing Fall/Not Fall of the data set are classified as (a) pre-fall (b) critical (c) fall (d) recovery, as in the criteria used in [31] and [32], (a) + (b) + (d) was proceeded to Not Fall and (c) to Fall. Since the criteria for dividing train/validation/test were not provided, our experiments were conducted by dividing the dataset into 5 folds. Fig. 6 shows examples of the Fall and the ADL situations of the UR-Fall Detection dataset. The main difference between the Fall and the ADL is whether a person lies down completely, as shown in Fig. 6, or not.

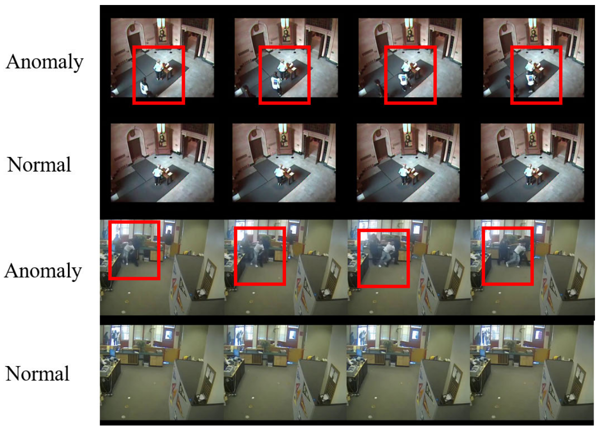


FIGURE 5. Example of the anomaly/normal situations in the UCF-Crime dataset. Red bounding boxes indicate the anomalies.

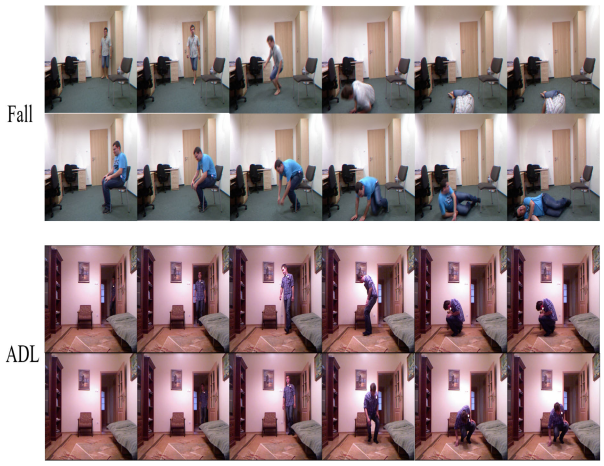


FIGURE 6. Example of the Fall/ADL situations in the UR-Fall Detection dataset.

In our experiments, since each video shot in the datasets usually has hundreds of video frames, we first divide the entire video shot into N sub-shots with $N = 10$. Then, each sub-shot will have t frames, which are variable depending on the total number of frames in the video shot. Among t frames in the sub-shot, we uniformly sub-sample T frames such that $T < t$ (see Fig. 3). In our experiments, we set $T = 16$.

In fine-tuning the network, we set the training hyper-parameters as follows: the stochastic gradient descent (SGD) optimization with momentum 0.9, minibatch size 16, and initial learning rate $1e - 3$. And, we used the average fusion [24] for the results of the spatial-stream (i.e., the result of X_i) and the temporal-stream (i.e., the result of $Y(X_i)$). To simulate the edge-computing environment, the inference for the Front and Back-CNN model was implemented on *NVIDIA* Jetson nano Kit [33].

In evaluating the anomaly detection methods, we adopted widely used performance measures such as AUC (Area Under

TABLE 2. Comparison between 3CFM and SG3I as the temporal stream CNN for action recognitions with UCF-101 and HMDB-51 datasets.

Method	2D-CNN	Dataset	mAP		
			Spatial Stream	Temporal Stream	Fusion
SG3I [10]	Xception	UCF-101	86.1	84.7	87.7
		HMDB-51	65.5	64.8	67.5
3CFM	Xception	UCF-101	86.1	79.2	**91.0**
		HMDB-51	65.5	55.4	**71.8**

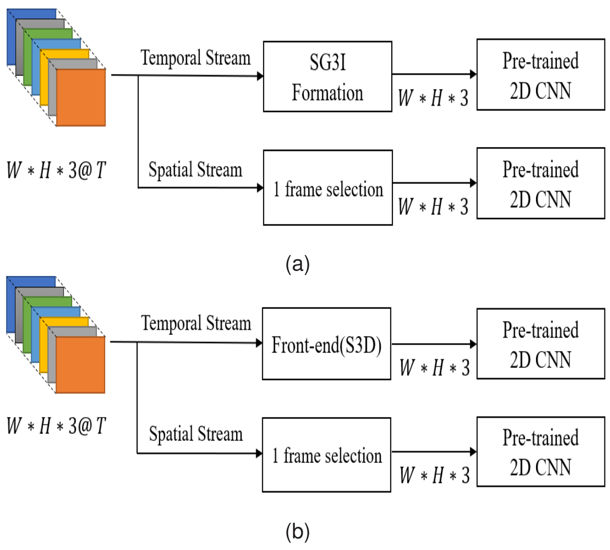


FIGURE 7. Two-stream CNNs for action recognition with: (a) SG3I and (b) S3D as temporal-stream CNN.

the Curve), Precision, Recall, and Accuracy defined as follows

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \tag{5}$$

where TP, TN, FP, and FN are True Positive, True Negative, False Positive, and False Negative, respectively.

B. PERFORMANCE COMPARISON FOR ACTION RECOGNITION

Note that the proposed method adopts a coordinated training for the spatial-stream (i.e., the result of X_i) and the temporal-stream (i.e., the result of $Y(X_i)$) with a shared 2D CNN. So, by setting $\lambda = 0$ in (2) of the loss function, only a temporal-stream network remains. This can be used to evaluate the performance of the proposed S3D for temporal motion learning. That is, for the performance comparison between the 3CFM (i.e., the output of the S3D) and the SG3I [10], we trained the temporal stream CNN with the SG3I and the 3CFM, separately, for the UCF-101 and HMDB-51 datasets (see Fig. 7). Since Xception [34] was used in [10] for the SG3I, we also trained the Back-CNN with the same model. Also, with the same spatial stream, the result of each temporal stream is fused with the result of the spatial stream. As shown in Table 2, after fusing the results of the two streams, our method with 3CFM yields higher mAP (mean AP) by 3.3% in UCF-101 and 4.3% in HMDB-51 than the SG3I.

TABLE 3. Comparison of Inference speeds in terms of fps (frames per second) with the various combinations for the end-to-end Front-CNN and Back-CNN network implemented on *NVIDIA* Jetson Nano. UCF-Crime dataset [15] with 256 × 256 × 3@16 was used for the speed comparison.

S3D Layers	Back-CNN			Channels
	MobileNet -v2	MobileNet -v3(small)	MobileNet -v3(Large)	
3 Layers	76.6	89.2	79.9	16, 32, 3
4 Layers	42.2	50.1	43.9	16, 32, 32, 3
	34.4	39.4	35.4	16, 32, 64, 3
5 Layers	35.4	40.6	36.5	16, 32, 32, 32, 3
	16.9	18.0	17.1	16, 32, 64, 128, 3

TABLE 4. Comparison of the anomaly detection performance for UCF-Crime. The speeds in fps (frame-per-second) were measured on Jetson Nano.

Method	UCF-Crime		
	Detector	AUC	fps
Sultani et al. [15]	SVM Baseline	50.0	-
Hasan et al. [38]	Deep Autoencoder	50.6	-
Sultani et al. [15]	C3D	75.4	N/A
Zhu & Newsam [37]	C3D	79.0	N/A
Ullah et al. [36]	BI-LSTM	85.5	-
Kim & Won [10]	MobileNet-v2	84.5	33.2
Kim & Won [10]	MobileNet-v3 (Small)	84.3	29.4
Kim & Won [10]	MobileNet-v3 (Large)	85.9	27.2
S3D+2DCNN	MobileNet-v2	**87.4**	76.6
S3D+2DCNN	MobileNet-v3 (Small)	85.0	**89.2**
S3D+2DCNN	MobileNet-v3 (Large)	87.1	79.9

It is interesting to note that, for the mAP performance of the temporal stream only, the results of the SG3I are better than the 3CFM in Table 2. On the other hand, the 3CFM is better than the SG3I for the final fused results. We believe that this is due to the different motion information contained in the SG3I and the 3CFM. That is, since the SG3I is formed by just stacking 3 gray-scale images in the video shot, it has a higher correlation with the input image for the spatial-stream CNN. On the other hand, the 3CFM is the output of the learnt S3D and is less correlated with the input of the spatial-stream CNN, being complementary with the spatial-stream CNN.

TABLE 5. Comparisons of the anomaly detection performance for UR Fall Detection dataset. The speed of the proposed method was measured in Jetson Nano by frame-per-second (fps).

Method	UR-Fall Detection				
	Detector	Precision	Recall	Accuracy	fps
Harrou et al. [39]	MEWMA(+SVM)	93.6	**100.0**	96.6	-
Harrou et al. [40]	GLR(+SVM)	94.0	**100.0**	96.6	-
Feng et al. [41]	BI-LSTM	94.8	91.4	-	-
Li et al. [31]	CNN	-	-	99.9	-
Kim & Won [10]	MobileNet-v2	96.9	**100.0**	97.1	33.2
Kim & Won [10]	MobileNet-v3 (Small)	96.7	99.8	96.7	29.4
Kim & Won [10]	MobileNet-v3 (Large)	**100.0**	98.1	98.3	27.2
S3D+2DCNN	MobileNet-v2	**100.0**	**100.0**	**100.0**	76.6
S3D+2DCNN	MobileNet-v3 (Small)	99.4	99.8	99.5	**89.2**
S3D+2DCNN	MobileNet-v3 (Large)	**100.0**	**100.0**	**100.0**	79.9

C. ANOMALY DETECTION IN EDGE-COMPUTING ENVIRONMENT

In this sub-section, we applied the proposed network with Front-CNN and Back-CNN to anomaly detection in an edge-computing environment. For the Front-CNN, the S3D with three layers of 3D convolution blocks is used. As mentioned already, the Back-CNN can adopt any existing 2D CNN. Considering the real-time constraints in the edge-computing environment, however, a relatively light-weighted CNN is required. Therefore, for the Back-CNN, we used MobileNet-V2 [26], MobileNet-V3(Small) [27], and MobileNet-V3(Large) [27], which were pre-trained by still image datasets such as ImageNet [35]. Table 3 shows the inference speeds for various combinations with different layers of S3D (Front-CNN) and different versions of MobileNet [26], [27] (Back-CNN). As shown in the table, our modular network of Front-CNN and Back-CNN is fast enough to support real-time applications with Jetson Nano for all versions of MobileNet with 3 layers of S3D.

The serially connected Front-CNN and Back-CNN are fine-tuned via the coordinated end-to-end training method. Note that the S3D for the Front-CNN is untrained, but the MobileNet of the Back-CNN is pre-trained. Therefore, the pre-trained weights in the early layers (say, up to 10 layers) of the MobileNet are frozen during the fine-tuning, but only the network parameters in the S3D and the last layers of the MobileNet are updated.

Table 4 shows the anomaly detection results on UCF-Crime dataset. Since most of the previous results for UCF-Crime were reported in terms of the AUC, we also used the AUC as the performance measure in Table 4. As shown in the table, our method achieved the best performance both in AUC and fps. To be specific, our method is about 2% and 6∼10% superior to BI-LSTM [36] and the methods in [15], [37], respectively. Our method is better by about 1∼3% than the SG3I [10] in the AUC and is much faster in terms of the fps performance on *NVIDIA* Jetson Nano. Note that, since the SG3I method in [10] needs to fuse all the results for 10 SG3I inputs, it may suffer from latency. In the case of C3D, it is impossible to execute the C3D inference with Jetson Nano due to insufficient memory.

Table 5 shows the performance of the proposed method for UR-Fall detection dataset, where the results are averaged for all 5 folds of train/test datasets. Since most of the previous works used the accuracy as the performance measure, we also used the accuracy as well as the precision and the recall for UR-Fall dataset. As one can see in Table 5, the proposed method achieved the state-of-the-art results in the real-time constraints with the average accuracy of 100%. Since the 'Fall' includes only the situation when the person is lying on the floor, it is much easier to detect the falls in the UR-Fall than the crime actions in the UCF-Crime dataset. That is, the crime situations include much more complex anomalies and actions. This is why the 'Fall' detection performance of the non-DNN methods such as [39] and [40] also show excellent detection performance of 96.6% and the DNN-based methods such as [31], [41], and [15], including the proposed method, make only slight performance improvements.

V. CONCLUSION

The goal of this paper is to construct a deep neural network that can handle video data in real-time constraints.

Our strategy to achieve the goal is to avoid the computationally expensive optical flow computation and to minimize the 3D convolution operations. Accordingly, the proposed solution is a modular architecture with the Front-CNN and the Back-CNN. The Front-CNN has a shallow 3D (S3D) CNN with only three layers of 3D convolution blocks, which is trained to condense multiple frames of the video into only three channels of feature maps (3CFM). Then, since the 3CFM is compatible with a 2D image with RGB channels, we can employ any pre-trained 2D CNN for the following Back-CNN. Our modular S3D-2D CNN architecture was applied for anomaly detections with Jetson Nano developer for an edge-computing environment. Experimental results confirm the real-time execution with state-of-the-art performance for the anomaly detection problem on UCF-Crime and UR Fall datasets. Although more 3D convolutional layers can be added to the S3D for more challenging video tasks, they certainly limit the real-time execution and increases the hardware cost for the edge-computing device.

REFERENCES

[1] M. P. J. Ashby, “The value of CCTV surveillance cameras as an investigative tool: An empirical analysis,” *Eur. J. Criminal Policy Res.*, vol. 23, no. 3, pp. 441–459, Sep. 2017.

[2] E. L. Piza, B. C. Welsh, D. P. Farrington, and A. L. Thomas, “CCTV surveillance for crime prevention: A 40-year systematic review with meta-analysis,” *Criminol. Public Policy*, vol. 18, no. 1, pp. 135–159, Feb. 2019.

[3] H. Heng, D. Jazayeri, L. Shaw, D. Kiegaldie, A.-M. Hill, and M. E. Morris, “Hospital falls prevention with patient education: A scoping review,” *BMC Geriatrics*, vol. 20, no. 1, pp. 1–12, Dec. 2020.

[4] R. Parvathy, S. Thilakan, M. Joy, and K. M. Sameera, “Anomaly detection using motion patterns computed from optical flow,” in *Proc. 3rd Int. Conf. Adv. Comput. Commun.*, Aug. 2013, pp. 58–61.

[5] J. F. P. Kooij, M. C. Liem, J. D. Krijnders, T. C. Andringa, and D. M. Gavrila, “Multi-modal human aggression detection,” *Comput. Vis. Image Understand.*, vol. 144, pp. 106–120, Mar. 2016.

[6] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[7] G. Varol, I. Laptev, and C. Schmid, “Long-term temporal convolutions for action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, Jun. 2017.

[8] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4489–4497.

[9] J. Carreira and A. Zisserman, “Quo vadis, action recognition? A new model and the kinetics dataset,” in *Proc. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4724–4733.

[10] J.-H. Kim and C. S. Won, “Action recognition in videos using pre-trained 2D convolutional neural networks,” *IEEE Access*, vol. 8, pp. 60179–60188, 2020.

[11] F. Lei, X. Liu, Q. Dai, and B. W.-K. Ling, “Shallow convolutional neural network for image classification,” *Social Netw. Appl. Sci.*, vol. 2, no. 1, pp. 1–8, Jan. 2020.

[12] F. F. Chamasemani and L. S. Affendey, “Systematic review and classification on video surveillance systems,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 5, no. 7, p. 87, Jun. 2013.

[13] 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.

[14] 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.

[15] 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.

[16] A. Singh, D. Patil, and S. N. Omkar, “Eye in the sky: Real-time Drone Surveillance System (DSS) for violent individuals identification using ScatterNet Hybrid Deep Learning network,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1629–1637.

[17] R. Polishetty, M. Roopaei, and P. Rad, “A next-generation secure cloud-based deep learning license plate recognition for smart cities,” in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 286–293.

[18] K. Abas, C. Porto, and K. Obraczka, “Wireless smart camera networks for the surveillance of public spaces,” *Computer*, vol. 47, no. 5, pp. 37–44, 2014.

[19] S. Vítek and P. Melniáuk, “A distributed wireless camera system for the management of parking spaces,” *Sensors*, vol. 18, no. 1, p. 69, Dec. 2017.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[21] A. E. Eshratifar, A. Esmaili, and M. Pedram, “BottleNet: A deep learning architecture for intelligent mobile cloud computing services,” in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Lausanne, Switzerland, Jul. 2019, pp. 1–6.

[22] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. R. Faughnan, “Real-time human detection as an edge service enabled by a lightweight CNN,” in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 125–129.

[23] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, “A lightweight privacy-preserving CNN feature extraction framework for mobile sensing,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1441–1455, May/Jun. 2020.

[24] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.

[25] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, “Dynamic image networks for action recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 3034–3042.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. CVPR*, Jun. 2018, pp. 4510–4520.

[27] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for MobileNetV3,” 2019, *arXiv:1905.02244*. [Online]. Available: http://arxiv.org/abs/1905.02244

[28] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” 2012, *arXiv:1212.0402*. [Online]. Available: http://arxiv.org/abs/1212.0402

[29] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: A large video database for human motion recognition,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 2556–2563.

[30] B. Kwolek and M. Kepski, “Human fall detection on embedded platform using depth maps and wireless accelerometer,” *Comput. Methods Programs Biomed.*, vol. 117, no. 3, pp. 489–501, Dec. 2014.

[31] X. Li, T. Pang, W. Liu, and T. Wang, “Fall detection for elderly person care using convolutional neural networks,” in *Proc. 10th Int. Congr. Image Signal Process., Biomed. Eng. Informat. (CISP-BMEI)*, Shanghai, China, pp. 1–6, Oct. 2017.

[32] N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. O Laighin, V. Rialle, and J. E. Lundy, “Fall detection-principles and methods,” in *Proc. 29th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2007, pp. 1663–1666.

[33] NVIDIA Developer. (2019). *NVIDIA Jetson-Hardware For Every Situation*. [Online]. Available: https://developer.nvidia.com/embedded/develop/hardware

[34] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.

[36] W. Ullah, A. Ullah, I. U. Haq, K. Muhammad, M. Sajjad, and S. W. Baik, “CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks,” *Multimedia Tools Appl.*, vol. 80, pp. 16979–16995, Aug. 2020.

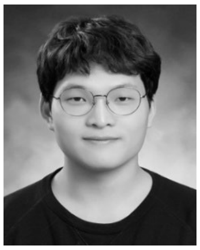
[37] Y. Zhu and S. Newsam, “Motion-aware feature for improved video anomaly detection,” 2019, *arXiv:1907.10211*. [Online]. Available: http://arxiv.org/abs/1907.10211

[38] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, ''Learning temporal regularity in video sequences,'' in *Proc. IEEE CVPR*, Jun. 2016, pp. 733–742.

[39] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, ''Vision-based fall detection system for improving safety of elderly people,'' *IEEE Instrum. Meas. Mag.*, vol. 20, no. 6, pp. 49–55, Dec. 2017.

[40] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, ''An integrated vision-based approach for efficient human fall detection in a home environment,'' *IEEE Access*, vol. 7, pp. 114966–114974, 2019.

[41] Q. Feng, C. Gao, L. Wang, Y. Zhao, T. Song, and Q. Li, ''Spatio-temporal fall event detection in complex scenes using attention guided LSTM,'' *Pattern Recognit. Lett.*, vol. 130, pp. 242–249, Feb. 2020.



JUN-HWA KIM received the B.S. and M.S. degrees in electronics engineering from Dongguk University, Seoul, South Korea, in 2019 and 2020, respectively, where he is currently pursuing the Ph.D. degree in electronics engineering. His current research interests include image/video processing, generative model, facial expression recognition, object detection/tracking, and edge computing.

NAMHO KIM received the B.S. degree in electronics engineering from Dongguk University, Seoul, South Korea, in 2021, where he is currently pursuing the master's degree in electronics engineering. His current research interests include image processing, object detection, and edge computing.

CHEE SUN WON (Member, IEEE) received the B.S. degree in electronics engineering from Korea University, Seoul, South Korea, in 1982, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts at Amherst, MA, USA, in 1986 and 1990, respectively.

From 1989 to 1992, he was a Senior Engineer with GoldStar Company Ltd. (LG Electronics), Seoul. In 1992, he joined Dongguk University, Seoul, where he is currently a Professor with the Division of Electrical and Electronics Engineering. He has been a Visiting Professor at Stanford University, Stanford, CA, USA, and at McMaster University, Hamilton, ON, Canada. His research interests include computer vision, deep neural networks, video signal processing, and image feature detection and matching.