IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2021, accepted June 2, 2021, date of publication June 4, 2021, date of current version June 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086668

Human Action Recognition Based on Transfer Learning Approach

YOUSRY ABDULAZEEM[1], HOSSAM MAGDY BALAHA[2], WALEED M. BAHGAT[3], AND MAHMOUD BADAWY[2]

[1]Computer Engineering Department, Misr Higher Institute for Engineering and Technology, Mansoura 35516, Egypt
[2]Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura 35511, Egypt
[3]Information Technology Department, Faculty of Computers and Information Sciences, Mansoura University, Mansoura 35511, Egypt

Corresponding author: Mahmoud Badawy (engbadawy@mans.edu.eg)

ABSTRACT Human action recognition techniques have gained significant attention among next-generation technologies due to their specific features and high capability to inspect video sequences to understand human actions. As a result, many fields have benefited from human action recognition techniques. Deep learning techniques played a primary role in many approaches to human action recognition. The new era of learning is spreading by transfer learning. Accordingly, this study's main objective is to propose a framework with three main phases for human action recognition. The phases are pre-training, preprocessing, and recognition. This framework presents a set of novel techniques that are three-fold as follows, (i) in the pre-training phase, a standard convolutional neural network is trained on a generic dataset to adjust weights; (ii) to perform the recognition process, this pre-trained model is then applied to the target dataset; and (iii) the recognition phase exploits convolutional neural network and long short-term memory to apply five different architectures. Three architectures are stand-alone and single-stream, while the other two are combinations between the first three in two-stream style. Experimental results show that the first three architectures recorded accuracies of 83.24%, 90.72%, and 90.85%, respectively. The last two architectures achieved accuracies of 93.48% and 94.87%, respectively. Moreover, The recorded results outperform other state-of-the-art models in the same field.

INDEX TERMS Convolutional neural network (CNN), human action recognition (HAR), long short-term memory (LSTM), spatiotemporal info, transfer learning (TL).

I. INTRODUCTION

Understanding human actions by inspecting video sequences has become an essential research topic. Human Action Recognition (HAR) technology enables the computer to achieve this level of understanding. HAR has a high significance in a wide range of applications. Fields like video surveillance [1], [2], virtual reality [3], [4], intelligent human-computer interface [5], and identity recognition [6] have benefited from HAR.

There are many approaches to categorize HAR techniques. From the input perspective, HAR is categorized into two types: (1) video-based HAR and (2) sensor-based HAR [7]. Video-based HAR takes videos or images as input to recognize human activity or motion. Sensor-based HAR gets the input from smart sensors such as accelerometers, gyroscopes, and sound. There are hand-crafted directions [8], [9], and deep learning methods [10] from the methodology perspective. The main difference between them is in feature learning. Hand-crafted methods learn features manually, while deep learning methods learn features automatically from videos. Recently, deep learning techniques gained wide interest after proving high-efficiency computer vision applications. Convolutional Neural Networks (CNN) was the first technique used in HAR applications [11]. CNN's superiority was due to its high capabilities in image analysis [12].

Using deep learning in HAR has been widely studied, and it involved several issues and challenges. Accordingly, various systems and architectures have been proposed. Despite the progress achieved in HAR, researchers stand short of facing several issues and challenges. A single-stream CNN structure models only a single type of information. It cannot parse both spatial and temporal information, as shown in Figure 1. A good popular solution for this issue was using two-stream

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

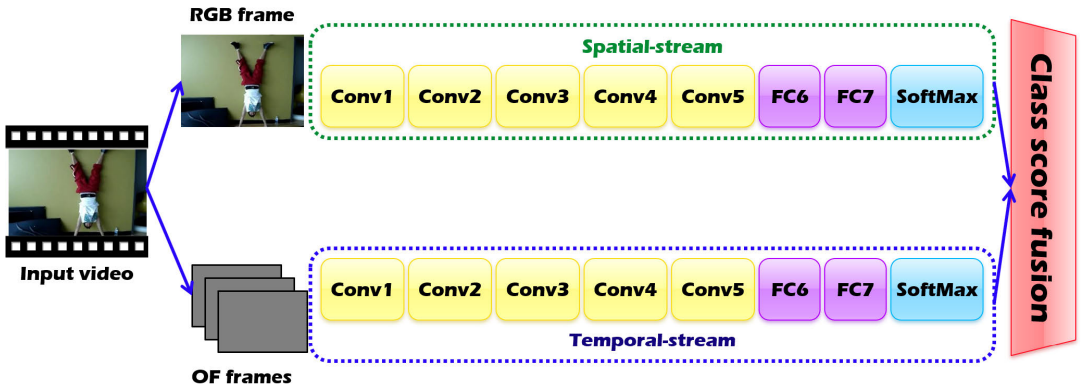


FIGURE 1. Illustration of the two-stream CNN architecture.

CNN architecture [13], [14]. The main idea was to use two CNN networks for modeling spatial and temporal information. The first network (stream) receives an RGB (Red-Green-Blue) image to model spatial information. The second stream receives stacked Optical Flow (OF) images [15] to model the temporal information. A score fusion method combines the features extracted from both streams.

Another limitation of CNN recognition methods is that it needs a large amount of training data. Although training data is crucial for network weights optimization, it is not easy to obtain a large amount of training data. The concept of Transfer Learning (TL) provides an effective solution to this problem.

Transfer learning is a machine learning approach that focuses on extracting data from a similar domain to improve learning ability or reduce the number of labeled samples required in a target domain [16]–[18]. In transfer learning, training and testing data do not need to be from the same domain, and the target domain model does not need to be trained from scratch, which can significantly reduce the training data and training time in the target domain. The concept behind transfer learning is illustrated graphically in Figure 2.

There are three main scenarios of CNN transfer learning: fixed feature extraction, fine-tuning and layers freezing, and pre-trained models [19]. In the fixed feature extraction scenario, a pre-trained final fully-connected layer is removed from the CNN model, while both the input and feature extraction layers retain their weights and structure and can be considered a fixed feature extractor. In the fine-tuning and layers freezing scenario, the pre-trained model is retained by fine-tuning the pre-trained network's weights. The fine-tuning process can be performed for all CNN network layers or only for the network's higher layers. There are many architectures are pertained for large datasets such as the ImageNet dataset [12] including Xception [20], DenseNet [21], and VGG16 [22], [23] and …etc. These pre-trained architectures are adopted to fine-tune each CNN network with a different dataset in the last scenario. Transfer learning has several benefits to improve CNN networks' performance, including speeding up the training process, improving the learning process, improving network generalization, and improving accessibility.

This paper proposes a Transfer Learning-based Human Action Recognition (TL-HAR) framework. The TL-HAR framework is based on a two-stream CNN architecture. The TL-HAR architecture applies TL techniques to overcome the previous limitations. This technique reduces the dependency on a large number of target domain data. The notable contributions of the current study can be summarized as follows:

- Provide a layered framework based on CNN architecture for efficient HAR.
- Analyze the concept of transfer learning and its impact on classification accuracy.
- Provide a stack of recognition architectures and analyze their different performance metrics.

The rest of this paper is organized as follows: In Section II, the related work is reviewed. The proposed framework is described in Section III. Section IV presents the experimental results. Finally, in Section V, the paper is concluded.

II. RELATED WORK

Recently, there has been extensive research on creating HAR systems based on deep learning approaches. The related work can be categorized into four main categories as follows: (i) 3D-Convolution Networks, (ii) Fusion-based Networks, (iii) Pooling-based Networks, and (iv) Multi-Stream-based Networks. The next subsections demonstrate a detailed description of the previous efforts for each category. The key challenges and issues for each category will also be described.

A. 3D-CONVOLUTION NETWORKS

Ji *et al.* [24] have introduced a CNN-based 3D architecture for multifunctional information channels generated from adjacent video frames. They used 3D kernels to extract spatial and temporal characteristics. Experimental findings have shown its high-performance architecture rather than its counterparts based on 2D frames.

The work done by Ji *et al.* [24] improved by Tran *et al.* They included five 3D pooling layers, with a compact descriptor called C3D, which averaged the outputs of the initial fully connected network layer. However, they produced

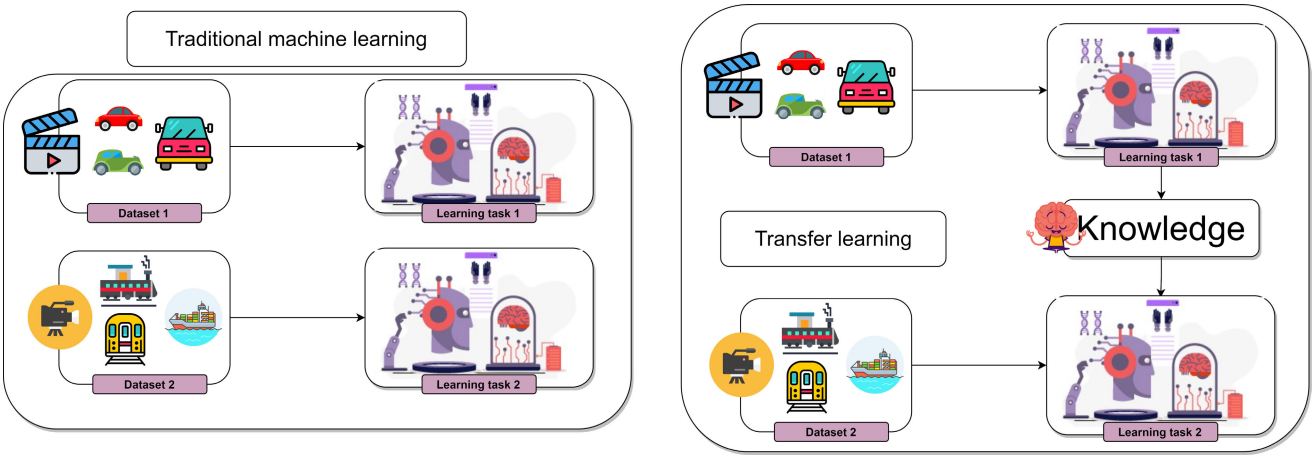


FIGURE 2. Transfer learning vs. non-transfer learning graphical illustration.

short video clips and aggregated spatial and temporal information via a late score fusion. This did not work if a long sequence of actions, such as walking or swimming, took a few seconds and took place in numerous video frames. The actions have not been modeled in their entirely temporal form.

Varol *et al.* [25] introduced the long-term temporal convolution (LTC) architecture to handle that problem. In a large number of video frames, they used space-time convolutions. The spatial resolution was reduced to track the complexity of the networks. Unfortunately, extending spatial kernels to 3D Spatio-temporal derivative led to a dramatic increase in the network parameters. They also examined the impact of various low-level representations and demonstrated the importance of accurate learning action models of high-quality optical flow estimates. Their reported results were 92.70% and 67.20% for UCF-101 and HMDB-51, respectively.

To this end, Sun *et al.* [26] proposed a deep, factional spatial-temporal networks (FstCN). The main objective was to factorize a 3D filter into a combination of 2D spatial kernels on the lower network layers and 1D temporal kernels on the upper network. The number of network parameters to be studied has been significantly reduced, which leads to mitigating high kernel complexity and a failure to train video information. The UCF-101 and HMDB-51 tested FstCN. The existing CNN methods were superior. Besides, it achieved notable performance without using auxiliary training videos to boost the overall performance.

B. FUSION-BASED NETWORKS

Karpathy *et al.* [11] proposed a slow fusion concept, where higher layers get access to progressively more global information in both spatial and temporal dimensions. Besides, they evaluated four temporal fusion methods (single frame, early, late, and slow fusion). They showed that slow fusion had better performance rather than the other fusion methods. Their best Spatio-temporal networks displayed significant performance improvements from 55.30% to 63.90%.

Feichtenhofer *et al.* [27] proposed the ConvNet architecture for the Spatio-temporal fusion of video snippets. They evaluated different ways of fusing both spatial and temporal networks to get the best performance results. They proved that the fusion at a convolution layer is better than fusion at the softmax layer. They evaluated different fusion methods such as Max, Concatenation, Bilinear, Sum, and Convolution. Convolution fusion achieved better performance. Also, they answered an important question about when to fuse the networks. They showed that fusing such networks spatially at the last convolutional layer is better than earlier and that, additionally, fusing at the class prediction layer can boost accuracy. Their approach got substantial parameter savings while holding the same performance. Finally, they answered another question, ''How to fuse the two streams temporally?''. They showed that using 3D pooling instead of 2D pooling after the fusion layer enhances the overall performance.

C. POOLING BASED NETWORK

Bilen *et al.* [28] proposed to adopt rank pooling [29], they introduced the concept of a dynamic image where the video is encoded into one dynamic set of images. They used CNN models directly on video data with fine-tuning. The end-to-end learning methods with rank pooling have also been proposed in [30]. Unfortunately, as the number of used features to describe the input frames increased and the video sequence's complexity grew, a single dynamic image-level was insufficient to have acceptable performance. Moreover, the linear ranking employed capacity was limited, and the rank pooling representation was not discriminative for the task.

For this reason, Hierarchical rank pooling [31] was proposed to support higher-order and non-linear representations rather than the work done by Fernando [29], [30]. It consisted of a network of rank pooling functions that captured the dynamics of rich convolutional neural network features within a video sequence. A high-capacity dynamic encoding mechanism was obtained to achieve action recognition by stacking non-linear feature functions and rank pooling over one another. Cherian *et al.* introduced Generalized ranking pooling [32] to improve the original method using a quadratic

ranking function that together brought a low input ranking approach to the data and maintained their temporal order in a subspace.

D. MULTI-STREAM BASED NETWORKS

Simonyan and Zisserman [13] proposed two types of networks, a spatial stream and a temporal one. The spatial network was submitted to raw video frames, while the temporal stream input was provided as the optical flow fields. Then, the two streams were fused using the SoftMax score. They trained and evaluated their architecture on the UCF-101 and HMDB-51 benchmarks.

By integrating improved trajectory, Wang *et al.* [33] extended two-stream networks. They have used trajectory-constrained sampling and pooling to deeply encode features that have been learned from deep CNN architecture. Wang *et al.* [34] have extended their work by introducing a devised network for temporal segments (TSN) using a sampling scheme to extract short clips over a long video sequence to include a long-range temporal structure using the two-stream networks. The aggregate information has been obtained through redundancies removing a segmental structure from consecutive frames.

Zhang *et al.* [35] replaced the optical flow with the compressed videos' obtained motion vector to avoid extra calculations. This led to the acceleration of the two-stream structure. Their reported experimental results showed a comparable recognition performance.

Singh and Vishwakarma [36] proposed a hybrid model for automating human activity recognition. The Inception-v3 architecture was chosen. They also processed the RGB frames with the Bi-LSTM model. To deal with view variations and occlusions in images, they used the principle of compact single dynamic motion image (DMI) instead of optical flow. To reduce the complexity of their model, they only used RGB frames to learn the features.

Singh *et al.* [37] proposed a two-stream model for activity recognition that combined residual- CNN with Transfer Learning. They used sum fusion, max fusion, weighted average, and weighted product fusion, among other fusion techniques. To build the two-stream model, they merged 2D and 3D residual networks. They used the standard UCF101 HMDB-51 benchmark dataset to test the performance of their architectures.

Chakraborty *et al.* [38] presented a two-stream network for human activity recognition. They employed transfer learning as they used many architectures such as DenseNet201, InceptionResNetV2, MobileNetV2, Xception, and InceptionV3 CNNs pre-trained on the ImageNet dataset for feature extraction. They used LSTM to model the temporal dynamics. They achieved 92% accuracy on the UCF-101 dataset.

III. THE TRANSFER LEARNING-BASED HUMAN ACTION RECOGNITION (TL-HAR) FRAMEWORK

This section provides a detailed description of the proposed Transfer learning-based Human Action Recognition (TL-HAR) framework. The TL-HAR framework consists of three main phases: pre-training, Data pre-processing and augmentation, and Recognition, as shown in Figure 3. The pre-trained features are extracted in the pre-training phase, then transfer to the recognition phase to adjust the network weights. Input data is acquired and pre-processed before entering the recognition phase. The TL-HAR framework and the different phases are discussed in detail in the following subsections.

A. PRE-TRAINING AND TRANSFER LEARNING

The concept of transfer learning depends on pre-training a network on a generic dataset for feature extraction. Afterward, network weights are adjusted for the classification task. Formally, a model M is trained on a dataset D_1. Parameters are adjusted to be prepared for training M as a refinement stage on the target dataset D_2. The last step is fine-tuning, on which M is trained on D_2. In this case, knowledge is transferred from D_1 to D_2.

In the proposed TL-HAR framework, three deep CNN architectures are adopted as feature extractors and classifiers. These architectures are: Xception [20], DenseNet [21], and VGG16 [22], [23]. The models are pre-trained on the ImageNet dataset [12]. It is a very large-scale dataset of over 15 million labeled high-resolution images with roughly 22,000 categories.

B. DATA PRE-PROCESSING AND AUGMENTATION

Most of the existing deep architectures in action recognition operate on a single frame [13], [14] or stack of consecutive frames at a fixed sampling rate [25], [27], [39]. Accordingly, these structures cannot incorporate long-range temporal information of videos into action models' learning process [40]. They also suffer in both the computational and modeling aspects. From the computational point of view, the cost of ConvNet training increased as it requires a large number of frames to capture the long-range actions. Varol *et al.* [25] used 100 frames for samples, and Yue-Hei *et al.* [39] used 120 frames. On the modeling side, the temporal coverage is limited by a fixed sampling interval. This limited coverage led to failure in visual content capturing over the whole video. The need to observe the entire video is crucial yet limited by computational costs.

The segmentation step in the proposed TL-HAR framework provides a sparse temporal sampling technique to cover the whole video. A small number of sampled frames are used to model the temporal structures in the video. This number is fixed regardless of the duration of the videos. The segmentation technique ensures fixed computational cost with long-range temporal coverage.

Algorithm 1 outlines the steps for generating both the RGB and OF frames.

Formally, given a video V with total m frames F, V is divided into n equal duration portions P as shown in Figure 4, where V and n are given as inputs to the Algorithm. One frame is sampled from each portion, resulting in n sampled frames T. To ensure the equal distance between sampled

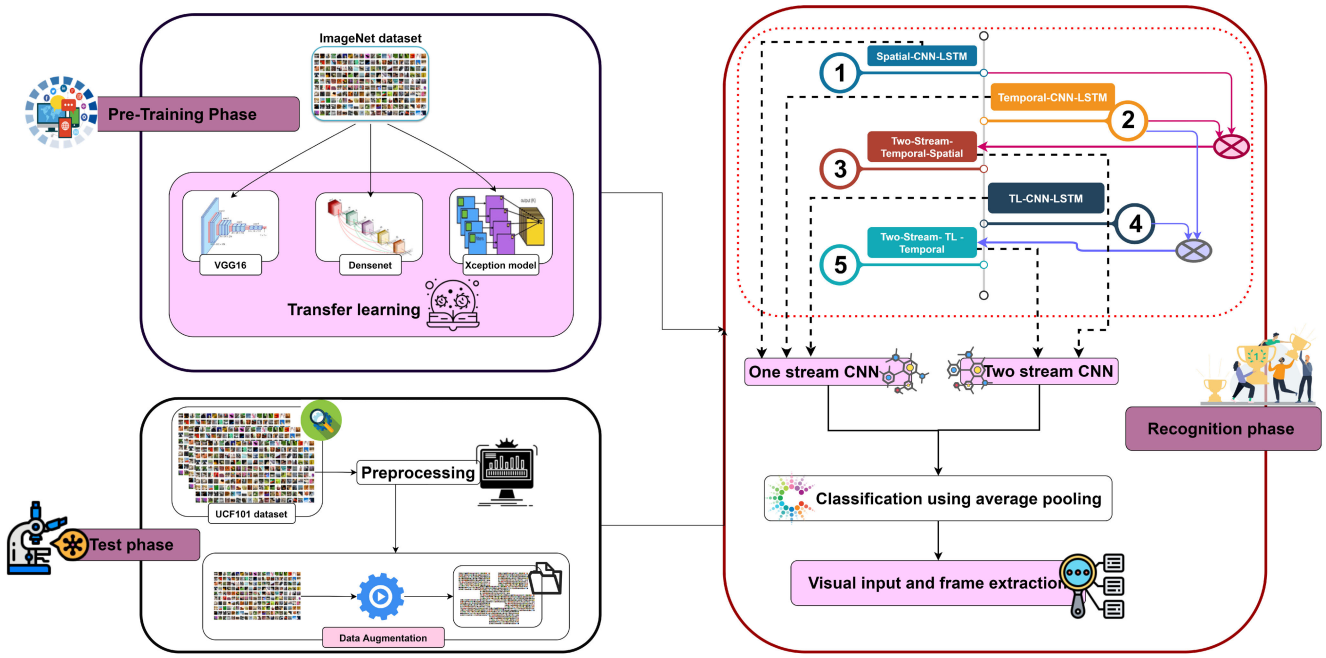


FIGURE 3. The transfer learning-based human action recognition (TL-HAR) framework.

Algorithm 1: Video Pre-Processing for Generating Both the RGB and of Frames

Input: V, t, n // Video file, Initial step, Number of Portions
Output: T, O // Sampled frames, OF-frames

```
m ← get_number_of_frames(V)
P ← divide_into_portions(V)
s ← ⌈m/n⌉
foreach i ∈ n do
    T_i ← get_frame(V, s × (i − 1) + t)
    T ← push_to_list(T, T_i)
    O ← push_to_list(O, TVL1(T_i))
end
return T, O
```

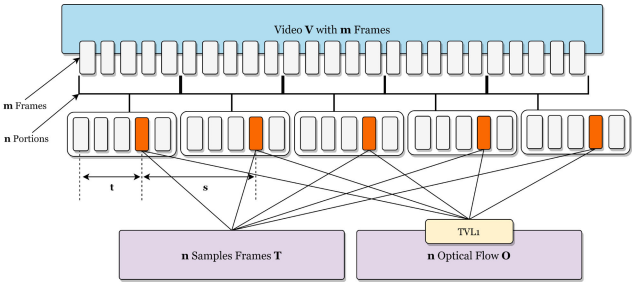


FIGURE 4. Graphical illustration of the video pre-processing algorithm.

frames, a step parameter s is calculated as $s = \lceil \frac{m}{n} \rceil$. Target frames are sampled from each portion starting with a specific number t and with step s such that:

$$Location_i = s \times (i-1) + t \tag{1}$$

where $i = 1, 2, .., n$ and $0 \leq t < s$.

The set of sampled frames T represents the new raw RGB frames fed to the CNN's spatial stream. It is also used to generate OF frames fed to the temporal stream of the CNN. Ma *et al.* [41] showed that extracting OF images using TV-L1 [42] is better than using Brox [43]. TV-L1 is used to generate OF frames for each frame T_i in the set T. Ten two-channel OF frames are stacked into a new frame with 20-channels. This method is similar to the method used in [34], [40], [44]–[46]. Frames are normalized using the min-max normalization to guarantee same-size frames (255). This step is important for the fusion process; ignoring it may cause overfitting.

The performance of CNN decreases with small datasets due to overfitting. Overfitting means that, while training, the network performs very well, then the performance drops on test data. The common solution is applying the data augmentation technique [12], [47]. It helps in increasing the dataset by applying geometric and color transformations to the sampled frames. In this regard, the dataset is increased by shearing, flipping, width and height shifting, rotation, zooming, and brightness changing. Figure 5 presents a graphical illustration of the data augmentation alternatives.

C. RECOGNITION

Deep learning techniques play a principal role in this phase. Techniques such as CNN and Long Short-Term Memory (LSTM) [48] are the cornerstone of this phase. Using deeper ConvNets improves the performance of the two-stream method [34], [39], [41]. Using pre-trained models effectively

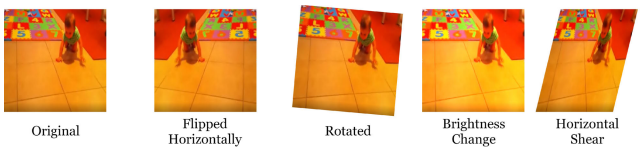


FIGURE 5. Graphical illustration of the data augmentation alternatives.

helps ConvNet learn and extract basic image features, which works well on datasets that do not have enough training samples. This phase presents five different alternative architectures to obtain the most benefit from these techniques. The five architectures' design is obtained after a set of trials in changing the hierarchy, layers types, layers sizes, and overall complexity.

The first two architectures are spatial and temporal single-streamed models, respectively. The third architecture incorporates the first two in a two-stream network model. The fourth architecture presents the TL concept in a spatial single-stream model. The final architecture utilizes the same model as the third, thus taking TL into account as in the fourth.

1) FIRST ARCHITECTURE: SPATIAL-CNN-LSTM

The first architecture depends mainly on spatial frames in a stacked manner. The input layer accepts a ten-stacked-spatial-frames input and passes it to the following layers. The time domain is added through the time distribution layer. The input layer is connected to five consequent Time Distributed Blocks (TDB). Figure 6 shows the structure of a TDB. Each TDB block consists of two convolutional layers, two batch normalization layers [49], two activation layers, and a max-pooling layer. The convolutional layers use the $L2$ regularization method [50] with a value of 0.001 and Glorot Uniform weight initializer. The max-pooling layers use (3×3) strides. The stride controls how the kernel convolves around the given input.

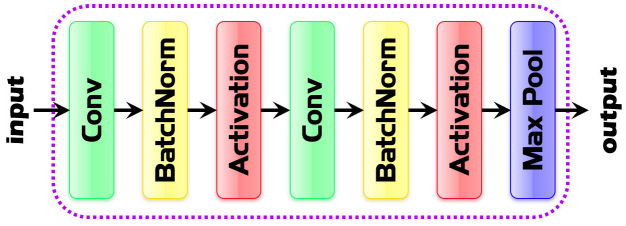


FIGURE 6. Time distributed block (TDB) internal structure.

The convolutional layers of the five TDBs have kernels of sizes 32, 64, 128, 256, and 512. After that, a flatten and LSTM layers are added. The LSTM layer has a size of 256 and a dropout with a ratio of 0.5. Dropout [51] with a ratio of 0.5 is applied after each dense layer. Dropout is sitting the output of hidden neurons with a certain probability to zero [52], [53]. The dropped-out neurons do not contribute to the forward pass nor the backpropagation. The dropout probability used is 0.5, which leads to maximum regulation.

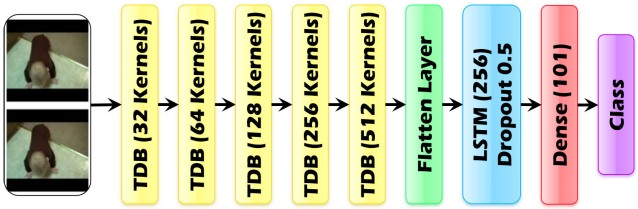


FIGURE 7. First architecture: Spatial-CNN-LSTM.

The number of UCF-101 dataset categories is 101 (as described in the experiments section), and hence a last dense layer of 101 neurons is added as the output layer. The batch size is 32, and the number of epochs is 12. The architecture uses AdaMax for the optimization process. In the training, testing, and validation processes, all available ten-stacked-spatial-frames are extracted from each video. The frame shape is (100×100) in the colored (RGB) mode. Figure 7 shows the structure design of the first architecture.

2) SECOND ARCHITECTURE: TEMPORAL-CNN-LSTM

The second architecture depends mainly on the temporal frames in a stacked manner. The input layer accepts a twenty-stacked-temporal-frames input and passes it to the following layers. The twenty constructing frames are combined from ten *Us* and *Vs* temporal frames. The *Us* and *Vs* temporal frames are extracted from the TVL1 function. The time domain is added using the time distribution layer. The input layer is connected to five TDBs with the same structure as in Figure 6.

The convolutional layers of the five TDBs have kernels of sizes 32, 64, 128, 256, and 512. After the flatten layer, an LSTM layer with size 256 and a dropout with a ratio of 0.5 is used. A dense layer with 101 neurons is added as the output layer. The batch size is set to 4, and the number of epochs is 152. The architecture uses AdaMax for the optimization process [54]. In training, testing, and validation processes, twenty stacked-temporal frames are extracted from each video. The frame shape is (100×100) in the colored (RGB) mode. Figure 8 shows the structural design of the second architecture.

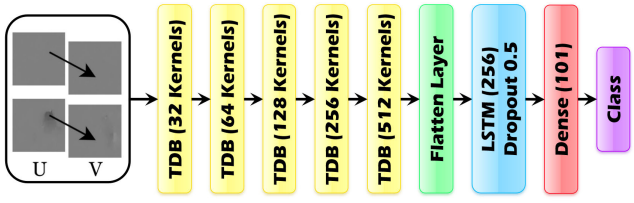


FIGURE 8. Second architecture: Temporal-CNN-LSTM.

3) THIRD ARCHITECTURE: TWO-STREAM-SPATIAL-TEMPORAL

This architecture applies the two-stream model (spatial and temporal streams). The spatial-stream applies the Spatial-CNN-LSTM architecture. This stream's input is a

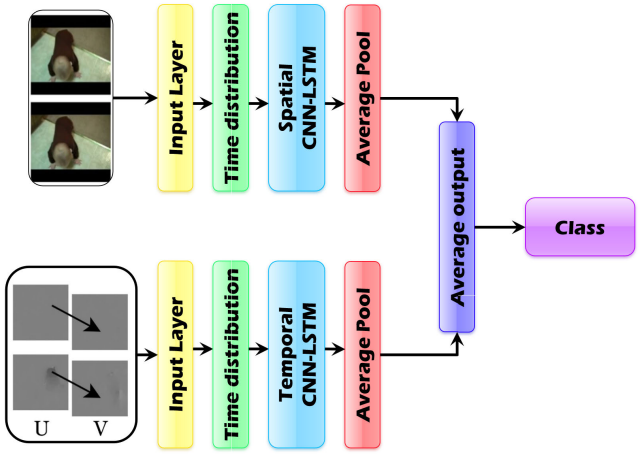


FIGURE 9. Third architecture: Two-stream-temporal-spatial.

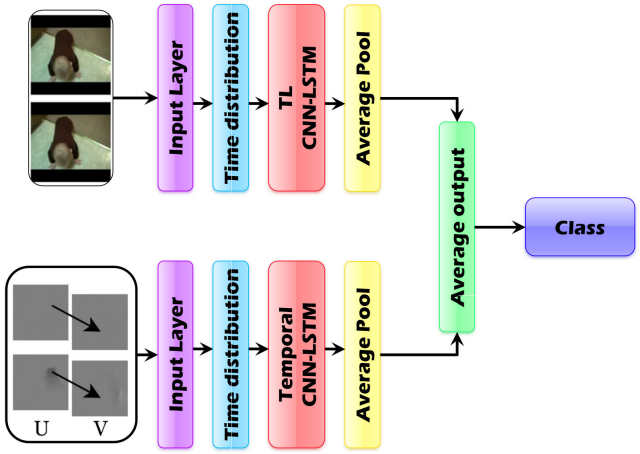


FIGURE 11. Fifth architecture: Two-stream-TL-temporal.

stack of chronologically ordered RGB-frames that have to pass through a time distribution layer. This layer has a great role in detecting movements, actions, and directions. This detection has a great impact on the action recognition process. The temporal-stream applies the Temporal-CNN-LSTM architecture. The input of this stream is a stack of temporal frames. These frames also pass through a time distribution layer. A global average-pooling layer is added after each architecture. Global average pooling avoids overfitting and sums up the spatial information [55]. The average is taken after the two average layers. The frame shape is (100×100) in the colored (RGB) mode. Figure 9 shows the structural design of the third architecture.

4) FOURTH ARCHITECTURE: TL-CNN-LSTM

In this architecture, the influence of the TL concept is investigated. The input layer receives the RGB frames and forwards them to the pre-trained network. Using TL, a model is trained on a large fully-labeled dataset to adjust the network weights. Subsequently, the architecture employs the pre-trained model. In this context, three state-of-the-art models (Xception, DenseNet, and VGG16) are pre-trained on the ImageNet dataset. Afterward, the model is set to be non-retrainable. Figure 10 shows the structural design of the fourth architecture.

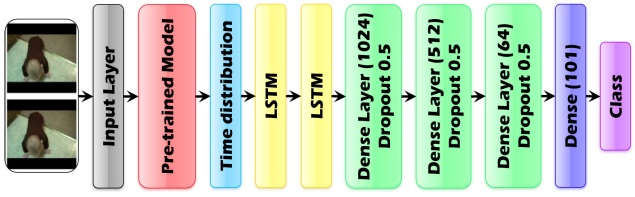


FIGURE 10. Fourth architecture: TL-CNN-LSTM.

The architecture starts with the time distribution layer to add the time domain. Two LSTM layers follow the time distribution layer. After these layers, three dense layers with a dropout of 0.5 come with several neurons 1024, 512, and 64, respectively. A dense layer of 101 neurons is added as the output layer. The batch size is 8, and the number of epochs is 128. The dense layers use the Glorot Uniform weight initializer [56]. It allows network weights to be initialized so that neuron activation does not start in saturated or dead regions. This leads to faster convergence and higher accuracy. The architecture uses NAdam for the optimization process [57]. In the training, testing, and validation processes, ten three-stacked-spatial frames are extracted from each video. The frame shape is resized to be (100×100) in the colored (RGB) mode.

5) FIFTH ARCHITECTURE: TWO-STREAM-TL-TEMPORAL

This architecture also applies the two-stream model. The spatial-stream applies the TL-CNN-LSTM architecture. This architecture pre-trains the model on the ImageNet dataset before receiving the RGB-frames. These frames pass through a time distribution layer. The Temporal-CNN-LSTM architecture is used in the temporal-stream, just as it is in the third architecture. A global average-pooling layer is added after each architecture. The average is taken after the two average layers. The frame shape is (100×100) in the colored (RGB) mode. With the adoption of TL, this architecture is a more enhanced version of the third. This update is expected to improve overall performance results. Figure 11 shows the structural design of the fifth architecture.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Several experiments are conducted to ensure the efficiency and effectiveness of the proposed framework. The performance of these experiments is described by significant measures used in existing research.

The experiments are performed on a Toshiba Qosmio X70-A device with Windows 10 operating system, Intel Core i7 processor, 32 GB RAM, and Nividia GTX with 4 GB GPU graphics card. Python 3 was the used programming language. The used packages were TensorFlow 2.1,

NumPy v.2.1.0 [58], Matplotlib v.3.2.1 [59], and OpenCV v.4.2.0 [60].

A. DATASETS

All the conducted experiments are performed on the UCF-101 dataset [61]. It consists of 13,320 videos of 101 human action categories. These 101 categories can be divided into five types: (1) Human-Human interaction, (2) Human-Object interaction, (3) Body-motion only, (4) Playing musical instruments, and (5) Sports. All videos are realistic and collected from YouTube. In the pre-trained part of the framework (i.e., the first architecture), ImageNet is used. Videos are not equal in size nor duration (i.e., each video has a different time duration and hence a different number of frames).

The videos are divided into three portions: training, validation, and testing with a ratio of 70%, 15%, and 15%, respectively. Since the number of videos in each category of the UCF-101 dataset is not equal, each category is split into this ratio. This approach ensures that every category is represented in each of the three portions. The actual numbers of videos for training, validation, and testing portions are "10,109", "2,525", and "686", respectively. Figure 12 shows samples from the UFC-101 dataset.

FIGURE 12. Samples from the UCF-101 dataset.

B. PERFORMANCE METRICS

The experiments evaluate several performance metrics, including Accuracy, Recall, Precision, F1-score, and Loss [62]. Among these metrics, accuracy has the most attention. It is the fraction of predictions the model classified correct to all the predictions of the model as in equation 2,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

where TP refers to true positive, TN is true negative, FP is false positive, and FN is false negatives. Recall is the fraction of actual positive predictions classified correctly, often referred to as sensitivity or true positive rate as in equation 3,

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Precision is the fraction of positive predictions to the total predicted positive samples, as in equation 4,

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

F1-score combines both the precision and recall into a single parameter. It is twice the ratio between the multiplication to the summation of precision and recall metrics as in equation 5,

$$F1_{score} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5}$$

Loss is the number indicating how bad the model classification was. It is the distance between the true values of the problem and the values predicted by the model as in equation 6,

$$l(y, p) = \sum_{c=1}^{M} y_{o,c} . \log p_{o,c} \tag{6}$$

C. PRE-TRAINING THE MODELS

In the fourth architecture, three models are pre-trained on the ImageNet dataset and examined. The first experiment is determining the best model with higher performance metrics. Table 1 shows the measured parameters for each used model.

TABLE 1. The experiments' performance metrics results.

Model	Accuracy	Recall	Precision	F1 Score	Loss
Xception	90.25%	88.90%	92.09%	0.905	0.674
DenseNet	83.29%	79.96%	88.36%	0.839	0.837
VGG16	93.48%	93.25%	94.33%	0.938	1.476

Although the VGG16 model has the highest loss readings, it has the highest records in all other parameters. VGG16 is chosen as the target domain for TL in the first architecture. Figure 13 shows a graphical representation of the reported results.

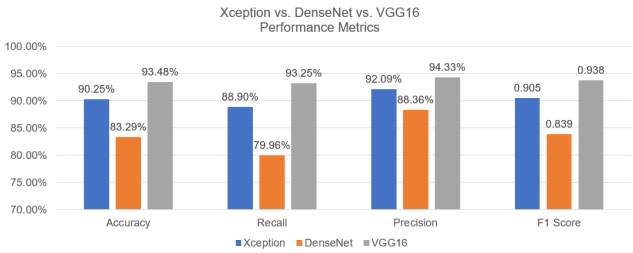


FIGURE 13. Graphical representation of the reported performance metrics results in Table 1.

D. EXPERIMENTS RESULTS

As mentioned earlier, for all architectures, the dataset is split into 70% for training, 15% for validation, and 15% for testing. ReLU and SoftMax are used, as the hidden activation function and output activation function respectively. Table 2 shows

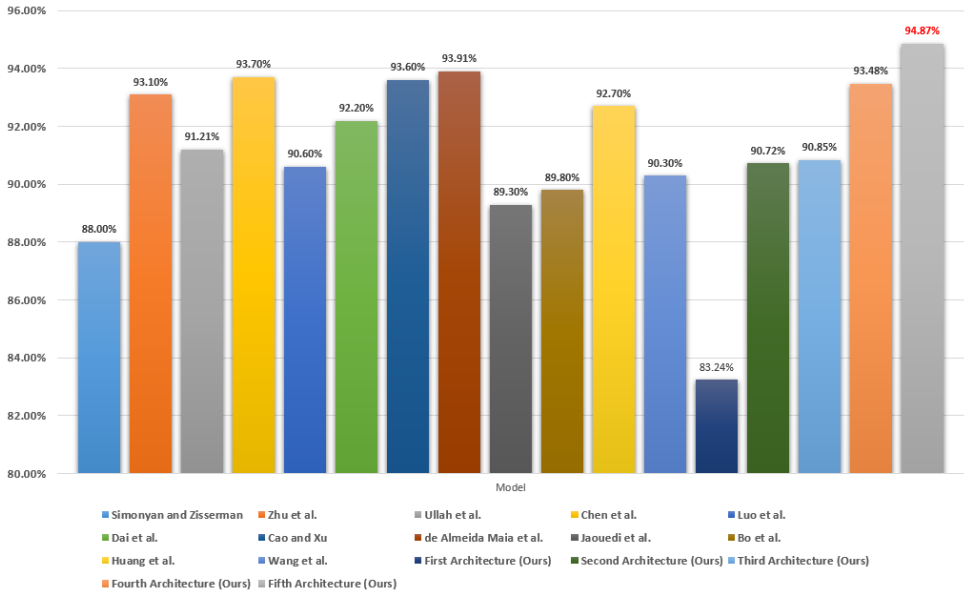


FIGURE 14. Graphical representation of the reported performance comparison in Table 3.

TABLE 2. The experiments' performance metrics results for VGG-16.

Architecture	Accuracy	Recall	Precision	F1 Score	Loss
First	83.24%	81.24%	89.40%	0.894	1.766
Second	90.72%	88.62%	94.14%	0.909	0.617
Third	90.85%	88.89%	94.23%	0.914	0.607
Fourth	93.48%	**93.25%**	94.33%	0.938	1.476
Fifth	**94.87%**	93.09%	**97.09%**	**0.950**	**0.463**

TABLE 3. Performance comparison with the state-of-the-art models.

Model	Year	Accuracy
Simonyan and Zisserman [13]	2014	88.00%
Zhu et al. [63]	2016	93.10%
Ullah et al. [64]	2017	91.21%
Chen et al. [65]	2017	93.70%
Luo et al. [66]	2019	90.60%
Dai et al. [67]	2019	92.20%
Cao and Xu [68]	2019	93.60%
de Almeida Maia et al. [69]	2020	93.91%
Jaouedi et al. [70]	2020	89.30%
Bo et al. [71]	2020	89.80%
Huang et al. [72]	2020	92.70%
Wang et al. [73]	2020	90.30%
First Architecture (Ours)	2021	83.24%
Second Architecture (Ours)	2021	90.72%
Third Architecture (Ours)	2021	90.85%
Fourth Architecture (Ours)	2021	93.48%
Fifth Architecture (Ours)	2021	94.87%

the values of the measured parameters. All values are calculated on the overall dataset after the training and validation processes.

The first proposed architecture achieved an overall accuracy of 83.24%, a recall of 81.24%, a precision of 89.40%, and an F1 score of 0.894. The second proposed architecture achieved an overall accuracy of 90.72%, a recall of 88.62%, a precision of 94.14%, and an F1 score of 0.894. The third architecture has slightly higher performance metrics' values. It achieved 90.85% accuracy, 88.89% recall, 94.23% precision, and F1 score of 0.914. The impact of using the two-stream model is clear as the performance metrics increased in the third architecture.

The fourth proposed architecture achieved an overall accuracy of 93.48%, a recall of 93.25%, a precision of 94.33%, and an F1 score of 0.938. Despite the fact that it is a single-stream model, it outperforms the previous architectures, including the two-stream model. This is obviously due to the impact of TL on performance enhancement.

Finally, the fifth architecture achieved the best performance among the other architectures. It achieved 94.87% accuracy, 93.09% recall, 97.09% precision, and F1 score value of 0.95. This architecture benefits from both the two-stream model and the TL concept. It is clear that the use of TL improves the performance of the recognition and classification processes.

Table 3 shows a comparison between the results of both the proposed architectures and the other state-of-the-art models. The models are ordered according to the year of publication. The recorded results show that most of the proposed architectures outperform the state-of-the-art models. Furthermore, it is clear that the fifth architecture,

Two-Stream-TL-Temporal outperforms all other models. Figure 14 shows a graphical representation of the reported results in ascending order.

V. CONCLUSION

This paper proposed a TL-HAR framework based on transfer learning techniques. TL-HAR consists of three main phases, namely, pre-training, preprocessing, and recognition. In the pre-training, three models are trained on a generic dataset to adjust network weights. This pre-trained network is used to recognize human activities in a realistic dataset. In pre-processing, a certain number of frames are extracted from the whole video. The segmentation technique ensures fixed computational cost with long-range temporal coverage. The extracted frames are used to feed spatial-streams in the proposed architectures. TV-L1 is used to generate OF frames. Stacked OF frames are used to feed the temporal-streams in the proposed architectures. Data augmentation techniques are applied to the training and validation of the model.

The recognition phase proposes five different architectures. The first two architectures are spatial and temporal single-streamed models, respectively. The third architecture incorporates the first two in a two-stream network model. The fourth architecture presents the TL concept in a spatial single-stream model. The final architecture utilizes the same model as the third, thus taking TL into account as in the fourth.

Different experiments are performed such that: (1) VGG16 outperforms Xception and DenseNet as it achieved 93.48% accuracy, (2) VGG16 is selected to be tested on the five architectures, and (3) The fifth proposed architecture has the highest accuracy value (94.87%).

Experimental results show that the combined architectures achieved higher accuracy than the self-paced architectures. The first architecture, which benefits from transfer learning techniques, achieved better results than other architectures. The superiority of the proposed architectures is clear through the comparison with state-of-the-art models.

REFERENCES

[1] M. Babiker, O. O. Khalifa, K. K. Htike, A. Hassan, and M. Zaharadeen, "Automated daily human activity recognition for video surveillance using neural network," in *Proc. IEEE 4th Int. Conf. Smart Instrum., Meas. Appl. (ICSIMA)*, Nov. 2017, pp. 1–5.

[2] R. M. Raval, H. B. Prajapati, and V. K. Dabhi, "Survey and analysis of human activity recognition in surveillance videos," *Intell. Decis. Technol.*, vol. 13, no. 2, pp. 271–294, May 2019.

[3] A. S. Fangbemi, B. Liu, N. H. Yu, and Y. Zhang, "Efficient human action recognition interface for augmented and virtual reality applications based on binary descriptor," in *Proc. Int. Conf. Augmented Reality, Virtual Reality Comput. Graph.* Springer, 2018, pp. 252–260.

[4] B. Kwon, J. Kim, K. Lee, Y. K. Lee, S. Park, and S. Lee, "Implementation of a virtual training simulator based on 360° multi-view human action recognition," *IEEE Access*, vol. 5, pp. 12496–12511, 2017.

[5] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey," *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 1–54, Jan. 2015.

[6] S. N. Paul and Y. J. Singh, "Survey on video analysis of human walking motion," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 7, no. 3, pp. 99–122, Jun. 2014.

[7] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognit. Lett.*, vol. 119, pp. 3–11, Mar. 2019.

[8] M. F. Aslan, A. Durdu, and K. Sabanci, "Human action recognition with bag of visual words using different machine learning methods and hyperparameter optimization," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 8585–8597, Jun. 2020.

[9] S. Nazir, M. H. Yousaf, J.-C. Nebel, and S. A. Velastin, "A bag of expression framework for improved human action recognition," *Pattern Recognit. Lett.*, vol. 103, pp. 39–45, Feb. 2018.

[10] M. Al-Faris, J. Chiverton, D. Ndzi, and A. I. Ahmed, "A review on computer vision-based methods for human action recognition," *J. Imag.*, vol. 6, no. 6, p. 46, Jun. 2020.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[13] 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.

[14] 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. (ICCV)*, Dec. 2015, pp. 4489–4497.

[15] Z. Tu, W. Xie, D. Zhang, R. Poppe, R. C. Veltkamp, B. Li, and J. Yuan, "A survey of variational and CNN-based optical flow techniques," *Signal Process., Image Commun.*, vol. 72, pp. 9–24, Mar. 2019.

[16] D. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowl. Inf. Syst.*, vol. 36, no. 3, pp. 537–556, Sep. 2013.

[17] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[18] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.* Bratislava, Slovakia: Springer, 2018, pp. 270–279.

[19] A. G. P. Goel, "A survey on deep transfer learning for convolution neural networks," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 6, pp. 8399–8410, 2020.

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

[21] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," 2014, *arXiv:1404.1869*. [Online]. Available: https://arxiv.org/abs/1404.1869

[22] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 169–175.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[24] 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.

[25] 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. 2018.

[26] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4597–4605.

[27] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1933–1941.

[28] 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. (CVPR)*, Jun. 2016, pp. 3034–3042.

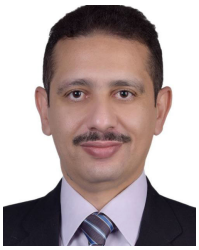
[29] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 773–787, Apr. 2017.

[30] B. Fernando and S. Gould, "Learning end-to-end video classification with rank-pooling," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1187–1196.

[31] B. Fernando, P. Anderson, M. Hutter, and S. Gould, "Discriminative hierarchical rank pooling for activity recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1924–1932.
[32] A. Cherian, B. Fernando, M. Harandi, and S. Gould, "Generalized rank pooling for activity recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3222–3231.
[33] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4305–4314.
[34] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019.
[35] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2718–2726.
[36] T. Singh and D. K. Vishwakarma, "A deeply coupled ConvNet for human activity recognition using dynamic and RGB images," *Neural Comput. Appl.*, vol. 33, no. 1, pp. 469–485, Jan. 2021.
[37] R. Singh, R. Khurana, A. K. S. Kushwaha, and R. Srivastava, "A dual stream model for activity recognition: Exploiting residual-cnn with transfer learning," *Comput. Methods Biomech. Biomed. Eng., Imag. Visualizat.*, vol. 9, pp. 1–11, Aug. 2020.
[38] M. Chakraborty, A. Pramanick, and S. V. Dhavale, "Two-stream mid-level fusion network for human activity detection," in *Proc. Int. Conf. Innov. Comput. Commun.* Singapore: Springer, 2021, pp. 331–343.
[39] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4694–4702.
[40] Y. Zhao, K. L. Man, J. Smith, K. Siddique, and S.-U. Guan, "Improved two-stream model for human action recognition," *EURASIP J. Image Video Process.*, vol. 2020, no. 1, pp. 1–9, Dec. 2020.
[41] C.-Y. Ma, M.-H. Chen, Z. Kira, and G. AlRegib, "TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition," *Signal Process., Image Commun.*, vol. 71, pp. 76–87, Feb. 2019.
[42] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L^1 optical flow," in *Proc. Joint pattern Recognit. Symp.* Tübingen, Germany: Springer, 2007, pp. 214–223.
[43] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2004, pp. 25–36.
[44] D. Torpey and T. Celik, "Human action recognition using local two-stream convolution neural network features and support vector machines," 2020, *arXiv:2002.09423*. [Online]. Available: https://arxiv.org/abs/2002.09423
[45] Q. Xiong, J. Zhang, P. Wang, D. Liu, and R. X. Gao, "Transferable two-stream convolutional neural network for human action recognition," *J. Manuf. Syst.*, vol. 56, pp. 605–614, Jul. 2020.
[46] C. Dai, X. Liu, and J. Lai, "Human action recognition using two-stream attention based LSTM networks," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105820.
[47] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3642–3649.
[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[49] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7694–7705.
[50] J. Kukačka, V. Golkov, and D. Cremers, "Regularization for deep learning: A taxonomy," 2017, *arXiv:1710.10686*. [Online]. Available: https://arxiv.org/abs/1710.10686
[51] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Netw.*, vol. 71, pp. 1–10, Nov. 2015.
[52] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8609–8613.
[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[55] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2921–2929.
[56] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, Mar. 2010, pp. 249–256.
[57] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. ICLR*. Stanford, CA, USA: Stanford Univ., 2016.
[58] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and R. Kern, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
[59] G. Moruzzi, "Plotting with matplotlib," in *Essential Python for the Physicist*. New York, NY, USA: Springer, 2020, pp. 53–69.
[60] S. Gollapudi, *Learn Computer Vision Using OpenCV*. New York, NY, USA: Springer, 2019.
[61] 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: https://arxiv.org/abs/1212.0402
[62] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 237–242.
[63] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao, "A key volume mining deep framework for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1991–1999.
[64] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.
[65] H. Chen, J. Chen, R. Hu, C. Chen, and Z. Wang, "Action recognition with temporal scale-invariant deep learning framework," *China Commun.*, vol. 14, no. 2, pp. 163–172, Feb. 2017.
[66] X. Luo, O. Ye, and B. Zhou, "An modified video stream classification method which fuses three-dimensional convolutional neural network," in *Proc. Int. Conf. Mach. Learn., Big Data Bus. Intell. (MLBDBI)*, Nov. 2019, pp. 105–108.
[67] W. Dai, Y. Chen, C. Huang, M.-K. Gao, and X. Zhang, "Two-stream convolution neural network with video-stream for action recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[68] D. Cao, L. Xu, and D. Zhang, "Cross-enhancement transform two-stream 3D ConvNets for action recognition," 2019, *arXiv:1908.08916*. [Online]. Available: https://arxiv.org/abs/1908.08916
[69] H. de Almeida Maia, D. T. Concha, H. Pedrini, H. Tacon, A. de Souza Brito, H. de Lima Chaves, M. B. Vieira, and S. M. Villela, "Action recognition in videos using multi-stream convolutional neural networks," in *Deep Learning Applications*. Springer, 2020, pp. 95–111.
[70] N. Jaouedi, N. Boujnah, and M. S. Bouhlel, "A new hybrid deep learning model for human action recognition," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 4, pp. 447–453, May 2020.
[71] Y. Bo, Y. Lu, and W. He, "Few-shot learning of video action recognition only based on video contents," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 595–604.
[72] Y. Huang, Y. Guo, and C. Gao, "Efficient parallel inflated 3D convolution architecture for action recognition," *IEEE Access*, vol. 8, pp. 45753–45765, 2020.
[73] H. Wang and J. Li, "Human action recognition algorithm based on multi-feature map fusion," *IEEE Access*, vol. 8, pp. 150945–150954, 2020.

YOUSRY ABDULAZEEM received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the Faculty of Engineering, Mansoura University, Egypt, in 2004, 2009, and 2014, respectively. His Ph.D. subject was ranking distributed uncertain databases and M.Sc. subject was in software performance engineering. In 2005, he joined the Computer Science Department, Misr Higher Institute for Commerce and Computers, Mansoura, Egypt, as a Teaching Assistant, and became a Lecturer Assistant, in 2009. In 2014, he joined the Computer Engineering Department, Misr Higher Institute for Engineering and Technology, Mansoura. His research interests include (but are not limited to) big data, deep learning, probabilistic database, query processing, distributed databases, software engineering, and software models.

HOSSAM MAGDY BALAHA was born in Egypt, in 1993. He received the B.Sc. and M.Sc. degrees from the Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Egypt. He is currently an Assistant Lecturer with the Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, a Senior Full Stack Laravel Web Developer, and a Freelancer. He made different courses on YouTube, including web development and the IoT. He built different systems, including web and mobile applications. He had a part in different projects and competitions related to his interests. His major research interests include Web development, the Internet of Things (IoT), deep learning (DL), computer vision (CV), soft computing, embedded systems, and robotics. He has served as a Reviewer in different journals, such as IEEE Access, IEEE Transactions on Cybernetics, *Journal of Ambient Intelligence and Humanized Computing (JAIHC)*, and *International Journal on Intelligent Systems (INT2)*.



WALEED M. BAHGAT received the B.Sc., M.Sc., and Ph.D. degrees in communication engineering from the Faculty of Engineering, Mansoura University, Egypt. He currently works as an Assistant Professor with the Information Technology Department, Faculty of Computer and Information Sciences, Mansoura University. His research interests include (but are not limited to) computer networks, big data, deep learning, and distributed databases.

MAHMOUD BADAWY received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the Faculty of Engineering, Mansoura University, Egypt, in 2004, 2009, and 2014, respectively. He currently works as an Associate Professor with the Computer Engineering and System Department, Faculty of Engineering, Mansoura University. His research interests include (but are not limited to) computer networks, big data, deep learning, QoS, WSN, and distributed databases.