Hindawi
Journal of Electrical and Computer Engineering
Volume 2021, Article ID 6610461, 18 pages
https://doi.org/10.1155/2021/6610461

Research Article

Proposing a Recognition System of Gestures Using MobilenetV2 Combining Single Shot Detector Network for Smart-Home Applications

Phat Nguyen Huu,[1] **Huong Nguyen Thi Thu**,[1] **and Quang Tran Minh**[2,3]

[1]*School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam*
[2]*Department of Information Systems, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet, District 10, Ho Chi Minh, Vietnam*
[3]*Vietnam National University Ho Chi Minh City (VNU-HCM), Linh Trung Ward, Thu Duc District, Ho Chi Minh, Vietnam*

Correspondence should be addressed to Phat Nguyen Huu; phat.nguyenhuu@hust.edu.vn

Received 21 December 2020; Revised 26 January 2021; Accepted 1 February 2021; Published 12 February 2021

Academic Editor: Yang Li

The paper proposes a system for identifying gestures and actions in smart homes. The proposed method is based on MobilenetV2 feature extraction combining with single shot detector (SSD) network. We used eleven types of gestures of walking, sitting down, falling back, wearing shoes, waving hands, falling down, smoking, baby crawling, standing up, reading, and typing for recognizing the gestures. In this system, the data are captured from the camera of mobile devices that are used to detect the object. The results are obtained objects on the frame by a bounding box. The results show that the system meets the requirements with an accuracy of over 90% that is suitable for real application.

1. Introduction

For identifying gestures, actions from still images and video sequences are challenging due to issues such as background image and lighting ratio. Many interactive applications between humans and computers or humans and robots or recently control electronic devices are widely studied. It allows computer systems to assist users for improving their lives and healthcare [1–3]. Two main methods for deploying the system are surveillance and wearable devices [4]. Surveillance equipment is usually fixed to the camera for user interaction or wearable devices such as smart watches that use voice to control or touch the automatic systems. In the content of this paper, we focus on the method of using fixed monitoring equipment.

ImageNet database inherits the combination of two-threaded ConvNet and recurrent neural network (RNN). In the paper, the authors receive not only information about time but also the space taken as input to the RNN [5]. Using a large number of parameters and computational complexity will not achieve high performance in terms of processing time or memory if used for feature extraction. In [6], the authors proved that the parameter number of Mobilenet is much smaller than the networks for extracting the characteristics, while the accuracy of the two models is almost the same. From there, we propose a method using Mobilenet networks which significantly reduces the number of parameters that are easy to use in weak configuration equipment.

When trying to accept human action gestures, we need to identify characteristics since computers can identify them effectively. Gestures, actions such as walking, sitting down, waving, and tying shoelaces, are very natural gestures in human life that are given priority. However, in machine learning, especially deep learning, when a large amount of computation is required, a computer with a strong configuration is required. In the paper, we reduce the MobilenetV2 network parameters by removing the full connected

layer to extract the image feature. We then used the MobilenetV2 network output as the input to the SSD network to identify the action.

In this paper, there are three main points that we propose as follows. Firstly, we propose the gesture recognition system combining Mobilenet V2 and SSD. Secondly, we propose building our set of gestures that are suitable for smart-home applications. Third, we build algorithm applications running on mobile devices with real data with an accuracy of over 90%.

Image recognition is comparable to human visual perception. It has come into everyday life and serves various demands. Facebook and media platforms use the technology to enhance searching image and assist visually impaired users. Businesses use image reception to scan large databases that satisfy customer demands and improve the customer experience in their stores and online shopping. In the healthcare system, medical image recognition and processing systems help professionals predict health risks and detect disease early which provide more services to patients. The goal of action identification is to create a system that can be used to control smart-home devices. It could be applied to control digital devices in the future. This is an advanced technology in the smart-home application that allows controlling the screen without touching the device using AI technology.

The rest of the paper is presented as follows. In Section 2, we will present related work. In Sections 3 and 4, we present and evaluate the effectiveness of the proposed model, respectively. Finally, we give the conclusion in Section 5.

2. Related Work

Identifying action is one of the applications in the control of digital devices in the future. This is an advanced technology that is being widely used in smart homes. Currently, many companies and research centers are actively testing high-tech models that allow screen control without touching the device by artificial intelligence (AI) technology. This is the area that is more concerned with action identification.

There are many studies to identify actions [2–4]. In [2], the authors perform 3D skeleton identification based on datasets of NTU-RGB + D and Kinetic. The authors [3] perform noron-based identity and joint trajectory maps (JTM). Khowaja and Lee [4] propose the solution to follow which is a sequential combination of Inception-ResNetv2 and long short-term memory (LSTM) network to take advantage of time variance to improve recognition performance. In this paper, the identification accuracy is 95.9 and 73.5 % based on UCF101 and HMDB51 datasets, respectively.

Besides, there are machine learning algorithms such as local orientation chart and support vector machine [7–13]. Thanks to the ability to learn, neural networks do not need to be manually established during the simulation process of human learning and can conduct training of gesture patterns and actions to create classification map network. The deep learning model is inspired by communication and information processing models developed from biological nervous systems including neural networks with more than one hidden layer. They can acquire the characteristics of learning subjects easily and accurately.

For complex subjects, it exhibits superior performance in computer vision and natural language processing (NLP) in [8, 9]. Modern object detection systems are variants of Backpropagation Neural Network (BPNN) and Faster RCNN in [10, 14]. In [14], the authors compared AI networks and concluded that BPNN achieved the highest efficiency. In [11], the author presents a SSD that optimizes object detection. Compared to Faster RCNN, SSDs are simpler and more efficient since it completely eliminates the stages of pixel creation and subsequent proposed reproduction. It also encapsulates all calculations in a network that makes the SSD easy to train and easy to integrate into systems. Besides, it works in conjunction with the MobilenetV2 network to operate on embedded and mobile devices quickly and efficiently.

However, there are several challenges with identifying action as follows:

Developing training sample sets: identification using machine learning requires an appropriate set of sample data, so it takes time to collect data to create standard samples.

Processing time: we need to process large amounts of data. If a network has to handle too many parameters with a weakly configured machine, it will slow down affecting the results in real time.

Accuracy evaluation methods: for conventional cameras (webcams), accuracy is affected by other conditions such as light, background, and hand movement speed, so we have to make some assumptions for the application.

As analysis above, we propose an action identification system based on the combination of MobilenetV2 network with SSD network for easy use on embedded devices with weaker hardware configurations.

3. Proposal System

3.1. Overview of Proposal System. We propose the system based on [6, 15]. In [15], they use Resnet-101 model for object detection. Although the accuracy is high, size of network is large. Mobilenet that published later than Resnet-101 is proposed by authors from Google in 2017. In this network, the authors used a calculus convolution method called depthwise separable convolution to reduce size model and calculation complexity. As a result, the model is useful when implemented in mobile and embedded devices since we proposed to use Mobilenet and SSD to apply for our system. Metrics of convolution networks are shown in Table 1.

The proposed system is based on [12, 20] for application in smart-home models, as shown in Figures 1 and 2. With the proposed network, we first expand the number of channels by deep convolution with a kernel size of 3×3 over the expanded space and finally through the bottleneck filter

Table 1: Comparing metrics of convolution networks for ImageNet dataset.

No	Network	Accuracy on ImageNet	Number of parameter	Size of network (megabyte)	Depth
1	VGG-16 [5]	0.901	138,357,544	528	23
2	VGG-19 [16]	0.90	143,667,240	549	26
3	Inception-V3 [17]	0.937	23,581,784	92	159
4	Resnet-101 [15]	0.938	44,675,560	171	101
5	**MobilenetV2** [6]	**0.901**	**3,538,984**	**14**	**88**
6	Densenet201 [18]	0.936	20,242,984	80	201
7	Xception [19]	0.945	22,910,480	88	126

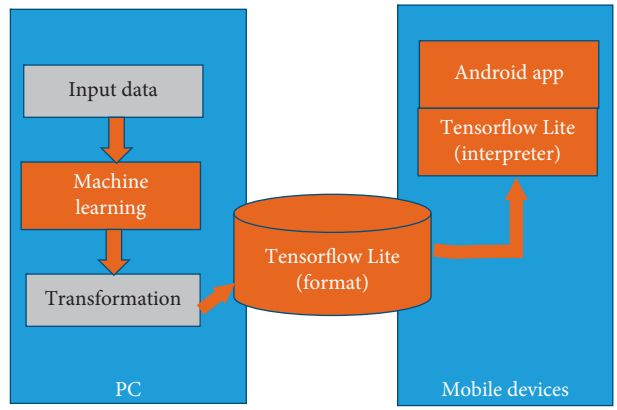


Figure 1: The proposed system works on workstations and mobile devices.

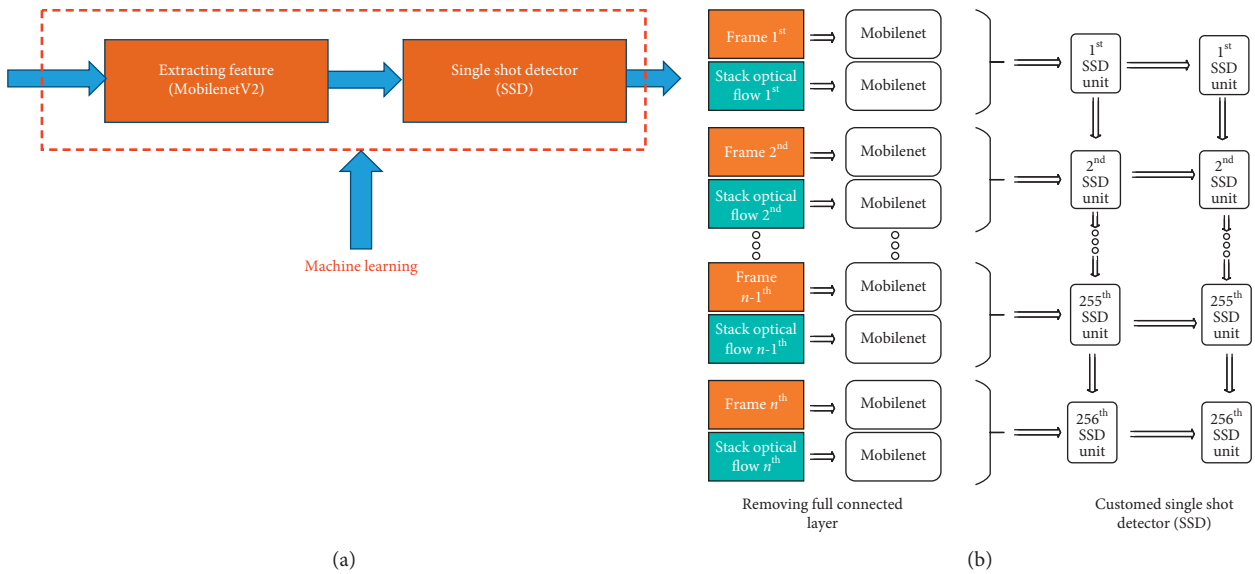


Figure 2: Proposal recognition module of gesture using CNN: (a) overview and (b) detail of the model.

back several smaller channels combining with a residual connection. They are used in gradient calculations to improve performance. Besides, we also reduce the MobilenetV2 network parameters by removing the full connected layer to extract the image feature, as shown in Figure 2.

The goal of this system is to build and process datasets from simple to complex actions. The proposed gestures include eleven actions, namely, walking, sitting down, falling back, putting on shoes, waving hands, falling down, smoking, baby crawling, standing up, reading, and typing.

First, the system extracts the characteristics of the data input using the mobilenetV2 network and then enters the SSD network to predict the results. The results obtained after the train process are converted to Tensorflow Lite (.tflite) format for performing on mobile devices.

The tensorflow model obtained Graphdef and checkpoint graphs after performing the training. These graphs are converted to Tensorflow Lite (tflite) format and then added to its interpreter. The interpreter executes the model using a set of operators. Details of the steps will be presented below.

3.2. Processing Steps. Tensorflow is used for creating models, training, manipulating data, and making predictions based on [12, 20, 21]. However, machine learning especially deep learning needs great computational power. Although training in mobile and embedded devices are possible, it will take a lot of time. To solve this problem, we will use Tensorflow for the training phase and Tensorflow Lite for the inference phase, as shown in Figure 1.

Proposal methods include the following steps:

(i) Step 1: preparing data

(ii) Step 2: assigning labels to data

(iii) Step 3: using the MobilemetV2 network to extract features

(iv) Step 4: using the output of MobilenetV2 network as input of SSD network to detect the object.

(v) Step 5: converting to Tensorflow Lite format

(vi) Step 6: creating an Android app to perform the Tensorflow Lite model.

Details of the steps are shown below.

3.2.1. Preparing Data. Firstly, we need to prepare the data including the self-built data source and the online source via Google and a part of UCF101 [22] and BU203 [23] with eight actions, namely, walking, sitting down, falling back, putting on shoes, waving hand, falling down, smoking, and baby crawling, as shown in Figure 3 [22, 23], and three actions (standing up, reading, and typing) designing by ourselves.

Number of labels and images are shown in Table 2.

3.2.2. Labeling Data. In this step, we perform the ROI determination of each action based on manual labeling. In this paper, we use a built-in labeling tool. This process basically draws boxes around objects in the image. Figure 4 is an example using the LabelImg tool that automatically creates an XML file describing the location of the object in the image.

The values obtained are shown in Figure 5 based on [24]. After labeling the data, we divide them into train/test files. Next, we convert the XML files into CSV files and then create TFRecords from these files. This TFRecords train file is given for model training. Finally, values are included in the model for evaluation.

3.2.3. Extracting Feature. The input image after being assigned will be saved in the csv format and converted into the record format in Tensorflow. We use a combination of two MobilenetV2 + SSD networks in Tensorflow to perform action identification to increase system accuracy.

In the feature extraction, we will use the MobilenetV2 network based on [6, 26–29]. The MobilenetV2 network uses convolution depth separators. The blocks are constructed, as shown in Figure 6.

First, the mobilenetV2 network uses 1×1 point convolution to expand the input channels. It then uses the deep convolution to extract the input feature and the convolution integrator linearly to combine the output features while reducing the network size. After reducing the size, it replaces the ReLU6 with a linear function to activate the output channel size to match the input, as shown in Figure 7.

The MobilenetV2 network also uses the reverse block to combine features over short-circuiting networks and features when traversing convolution to gain more functionality for output as follows. Depth convolution splits the input channels and filters into separate channels and then combines the output using 1×1 convolution. We have the network input $D_F \times D_F \times M$, where the kernel size is D_K, and the output with the number of N channels. Depth convolution will map only on each individual input channel. Therefore, the number of output channels and input channels is the same. Its computational cost function is $D_F \times D_F \times D_K \times D_K \times M$, as shown in Figure 8 [7].

The end result is a convolution. It is an assembly with a 1×1 kernel size that incorporates features created by depth convolution. Its calculated cost is $M \times N \times D_F \times D_F$, as shown in Figure 9, based on [7]. The cost calculated on depth convolution is

$$C = D_F \times D_F \times D_K \times D_K \times M + M \times N \times D_F \times D_F. \quad (1)$$

Performing calculations on each filter, we average the weights on an input filter. We then infer the output feature map calculated by the formula:

$$\widehat{G}_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} \times F_{k+i-1,l+j-1,m}, \quad (2)$$

where $\widehat{K}$ is a kernel of size $D_K \times D_K \times M$, F is input, and $\widehat{G}$ is output feature map.

3.2.4. Using SSD to Detect Objects. SSD [8, 21] is a good choice for object detection due to its greater accuracy than YOLO [9] and faster speeds than Fast-RCNN [10]. SSD uses VGG-16 base network with several additional layers such as extracting feature map. However, purpose of our paper is to perform on weaker devices such as mobile devices to reduce server-side bandwidth, reduce latency, and improve speed. As a result, the system reduces the cost of mobile traffic for users due to not having to download large amounts of raw data on computer. Therefore, we propose to use MobilenetV2 network instead of VGG16 base network to extract feature map. The SSD adds additional auxiliary bits after MobilenetV2 to predict the object.

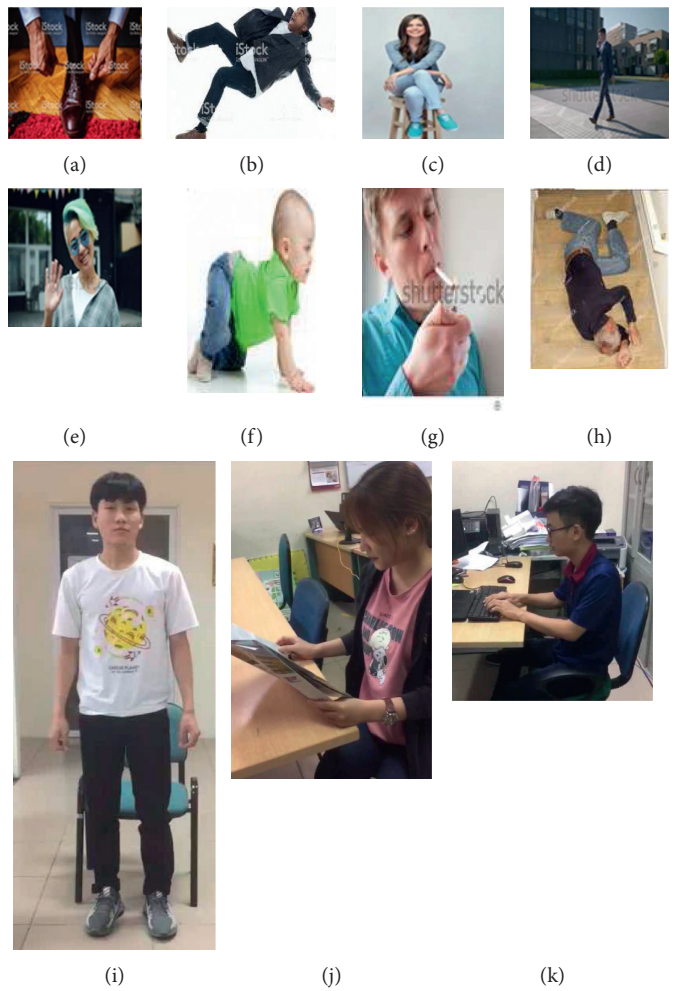


Figure 3: Prepare data for implementation: (a) putting on shoes, (b) falling back, (c) sitting down, (d) walking, (e) waving hand, (f) baby crawling, (g) smoking, (h) falling down, (i) standing up, (j) reading, and (k) typing.

Table 2: Number of images and labels are prepared for implementation.

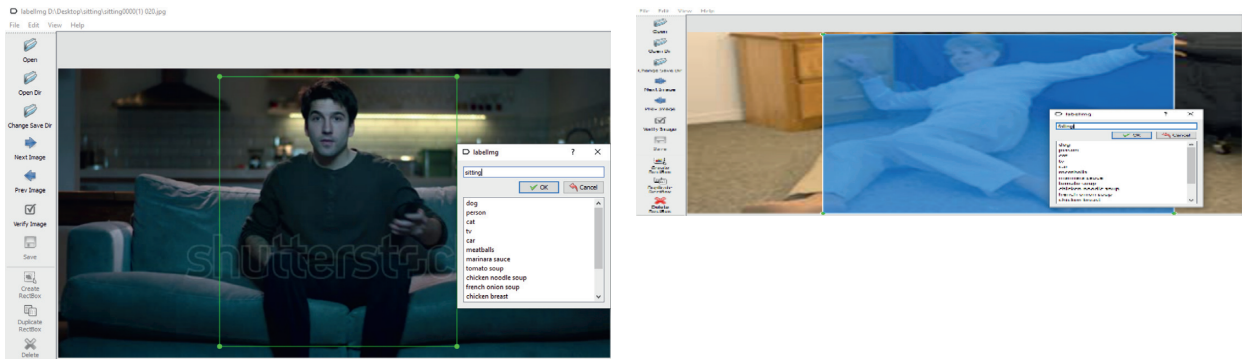
Labels	Number of training images	Number of testing images
Walking	400	100
Sitting down	400	100
Falling back	400	100
Putting on shoes	400	100
Waving hand	400	100
Baby crawling	400	100
Smoking	400	100
Falling down	400	100
Standing up	400	100
Reading	400	100
Typing	400	100

SSD model creates a vector of probability of occurrence of $c + 1$ object, where c is the number of layers and a background layer indicates that there is no object. A vector with four elements (x, y, h, w) represents the position of object of frame.

After each training step, we calculate the loss function until they are reduced and adjust them to be closer to the real object. The model converges when the difference between facts and predictions is close to zero, as shown in Figure 10, based on [8, 21].

The loss function is calculated as follows [8]:

$$(x, c, l, g) = (x, c) + \alpha. \tag{3}$$



(a) (b)

Figure 4: An example of data labeling: (a) sitting down and (b) falling down.

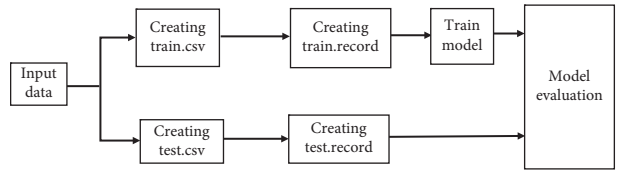


Figure 5: Detailed of the labeling model.

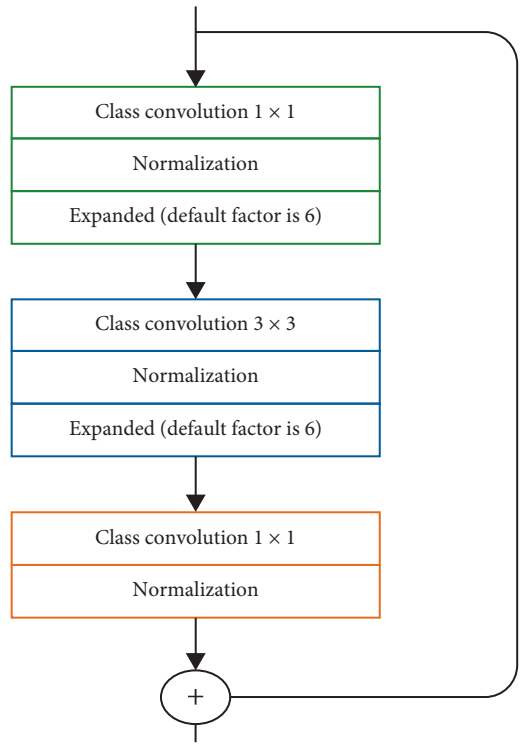


Figure 6: Model of MobilenetV2 [25].

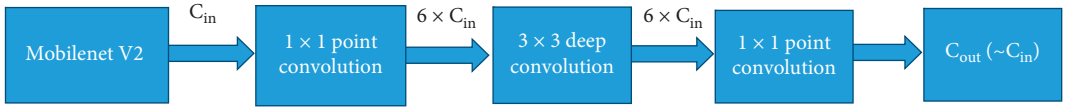


Figure 7: Operation process of MobilenetV2.

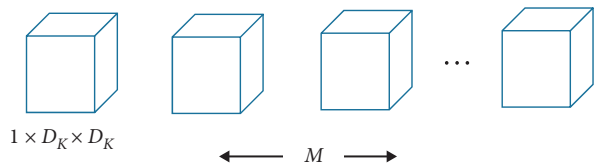


Figure 8: Filters by convolution based on [7].

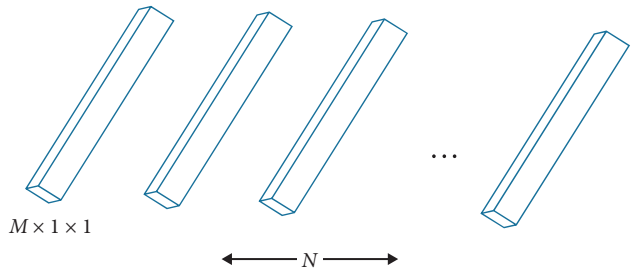


Figure 9: Combination of convolution equals accumulation convolution.

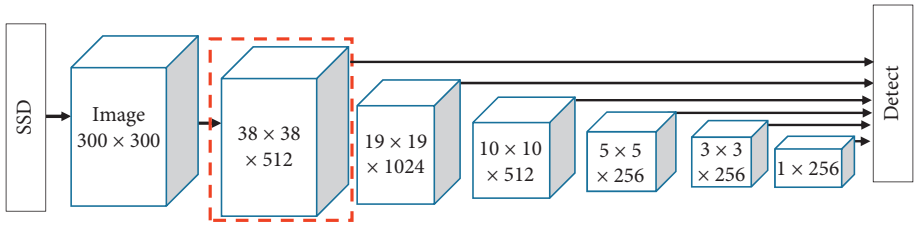


Figure 10: Calculation of loss function in MobilenetV2.

The loss function consists of two terms: L_{conf} and L_{loc}, where N is the appropriate default boxes calculated as follows:

$$L_{loc}(x, l, g) = \sum_{i \in Pos, m \in \{cx, cy, w, h\}}^{N} \sum x_{ij}^{k} \text{smooth}_{L1}\left(l_i^m - g_j^m\right), \tag{4}$$

where L_{conf} is the confidence loss that is the softmax loss over multiple classes confidences (c) (α is set to 1 by cross validation). $x_{ij}^{p} = 1, 0$ is an indicator for matching ith default box to the jth ground truth box of category p.

From the training process, we have an algorithm diagram with the input of images through mobilenetV2 network to obtain their weight. The data is then put into the SSD network to determine the coordinates and probability of the object's appearance as well as the loss function value, as shown in Figure 11.

3.2.5. Converting objects to Tensorflow Lite (TSL) Format. TSL is a lightweight Tensorflow solution for mobile and embedded devices. It allows running machine learning models on mobile devices. The process for this model is shown in Figure 12 based on [7].

The main components of Tensorflow Lite are the model file format, the interpreter for graph processing, a set of kernels to work with, and finally the interface for the hardware acceleration layer.

(i) Model file format: Tensorflow Lite is a special model file format. It is very light and less dependent on the hardware configuration. Most graph calculations are done using a 32 bit float

(ii) The interpreter: it is designed to operate at a low cost and on simple configuration devices. Tensorflow Lite has very few dependencies and is easy to build on simple devices. It uses FlatBuffers. Therefore, it can download at a fast rate with flexible costs.

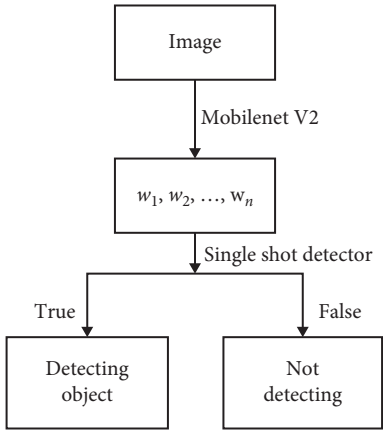


Figure 11: Determining of the object in SSD.

Figure 12: Model of Tensorflow Lite.

(iii) Ops/kernel: it is a smaller set of operators. However, all models will not be supporting them. Tensorflow Lite provides an integrated core ops and is optimized for CPU using neon. They operate in both float and quantization.

(iv) Proceed to increase hardware speed: it targets custom hardware. That is the neural network API Tensorflow Lite comes preloaded with links for the neural network API (API NN). If your device supports API NN, the data flow will delegate these operators to the API. Otherwise, it will execute directly on the CPU.

3.2.6. Applying Proposal Algorithm for Mobile. TensorFlow mobile is used for a mobile platform such as iOS and Android. This helps developers who have a successful TensorFlow model and want to integrate their model into a mobile environment. However, the fundamental challenges that one can find in integrating their model into the mobile environment are

(i) Using the TensorFlow mobile phone

(ii) Building model for a mobile platform

(iii) Adding TensorFlow libraries to the mobile application

(iv) Preparing model file

(v) Optimization of binary size, file size, RAM usage, etc.

Tensorflow Lite is a follow-up to Tensorflow mobile. It can be done on most Tensorflow mobile devices at a very fast speed. Tensorflow Lite is a set of tools to help developers run Tensorflow models on mobile and embedded and IoT devices. It allows for machine learning inference on the device with low latency and small binary size.

Tensorflow Lite includes two main components:

(i) Tensorflow Lite interpreter runs specially optimized models on a variety of hardware such as mobile phones, embedded Linux devices, and microcontrollers.

(ii) Tensorflow Lite Converter converts Tensorflow models into an efficient form for use by translators and can offer optimizations for improved binary size and performance.

Tensorflow Lite is functionally different from Tensorflow Mobile in a degree that is optimized to support system transition and deployment. Tensorflow Lite has been leveraged at every level from model compilation to hardware utilization to increase the viability of inference on the device while maintaining model integrity as follows:

(i) Model Transformation: the Tensorflow Lite (TOCO) converter takes a trained Tensorflow model as input and outputs a FlatBuffer-based TFLite (.tflite) file containing the binary representation of the original model.

(ii) The interpreter core is responsible for executing Lite models in client applications using a Tensorflow set of operators. By limiting the default operators, libraries, and tools needed to run Lite models, Interpreter Core has been reduced to 100kb or 300kb.

(iii) Hardware speed up by optimizing Tensorflow Lite reaches all hardware based on the operation of mobile and embedded devices.

(iv) The amount of automation is an important component of the neural networks that are related to Tensorflow Lite. Post-training quantization is recommended in Tensorflow Lite and provided as an attribute of the TOCO converter. The results have shown that the compression model inference delay can be reduced by up to 3 times while maintaining accuracy.

To deploy the Tensorflow Lite model file on our application, we build the system of three main components, as shown in Figure 13:

(i) Java API : includes *C++* API functions on Android

(ii) API *C++*: downloads Lite model and call interpreter

(iii) Interpreter: uses selective kernel loading, a unique Lite feature in Tensorflow

4. Simulation and Result

4.1. Setup. During implementation to increase accuracy, input data is passed through a preprocessing step to improve quality. Through this step, the data is transferred to Mobilenet and parameters are changed such as batch size, learning rate, and multibox detection match the input data as well as computer configuration to improve the accuracy and speed up the training process.

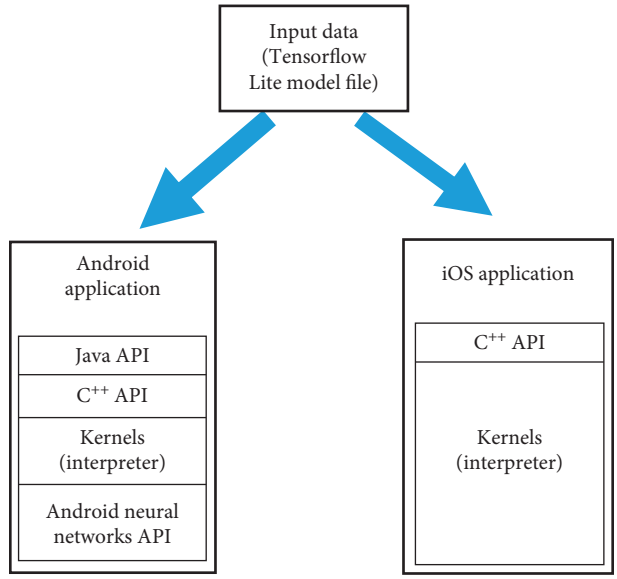


Figure 13: Applications on mobile.

In our simulation, we setup the parameters as follows [21].

The batch size is changed from 1 to 8. The learning rate decay policy is slightly different for each dataset and object. The initial learning rate is setup as 10^{-4}. In three parameters, multibox detection is the most important. For each location, we have k bounding boxes. They have different sizes and aspect ratios. In our paper, we have 8732 bounding boxes with different aspect ratios 1, 2, 3, 1/2, and 1/3.

Each training image is randomly sampled by entering the original input image.

Object is 0.1, 0.3, 0.5, 0.7, or 0.9.

The size of sampled patch is selected by [0.1,1] or original image, and the aspect ratio is from 1/2 to 2.

4.2. Result. To perform the training with the action sets mentioned above, we get the results shown in Figures 14 and 15. Figures 14 and 15 show the training on CPU with Ram 8G and core I_5. We perform for six days with the steps to reduce the loss function from 29 to 2.

We perform on models, namely, Tensorflow (RCNN + InceptionV2), Tensorflow (RFCN + Resnet101), Tensorflow Lite, and proposal model (SSD + MobilenetV2). The results are shown in Figures 16–19.

We perform identification of the above set of actions with Tensorflow. The results of the operation are shown in Figures 16 and 20.

We continue to implement identification of the set of actions with Tensorflow (RFCN + Resnet101). The results of the operation are as shown in Figure 18.

We continue to implement identification of the set of actions with Tensorflow Lite. The results of the operation are as shown in Figure 17.

We perform identification of the set of actions with proposal model. The results of the operation are shown in Figure 19.

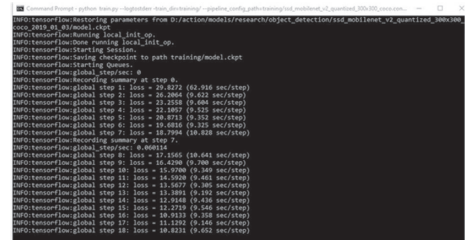


Figure 14: Simulation scenario starts for 20 steps.

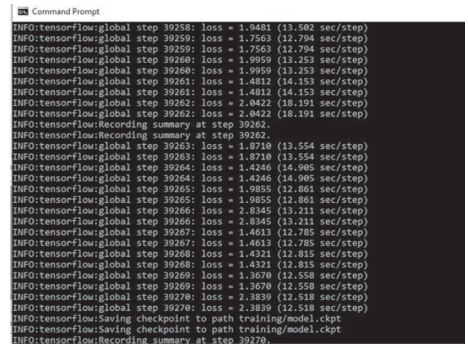


Figure 15: Simulation scenario finishes for 40000 steps.

We also perform actions to check on the image background. The results are shown in Tables 3 and 4.

From the above results, we see that the system meets the requirements set out with an accuracy of over 90%. Especially, with the use of Tensorflow and Tensorflow Lite, the system achieved an accuracy of up to 99% with an execution time of 14 seconds. This is an acceptable time for an intelligent control system.

The system of gesture recognition and action is built by SSD + MobilenetV2 algorithm and trained over 2500 images. We then use each action 10 different images for each gesture

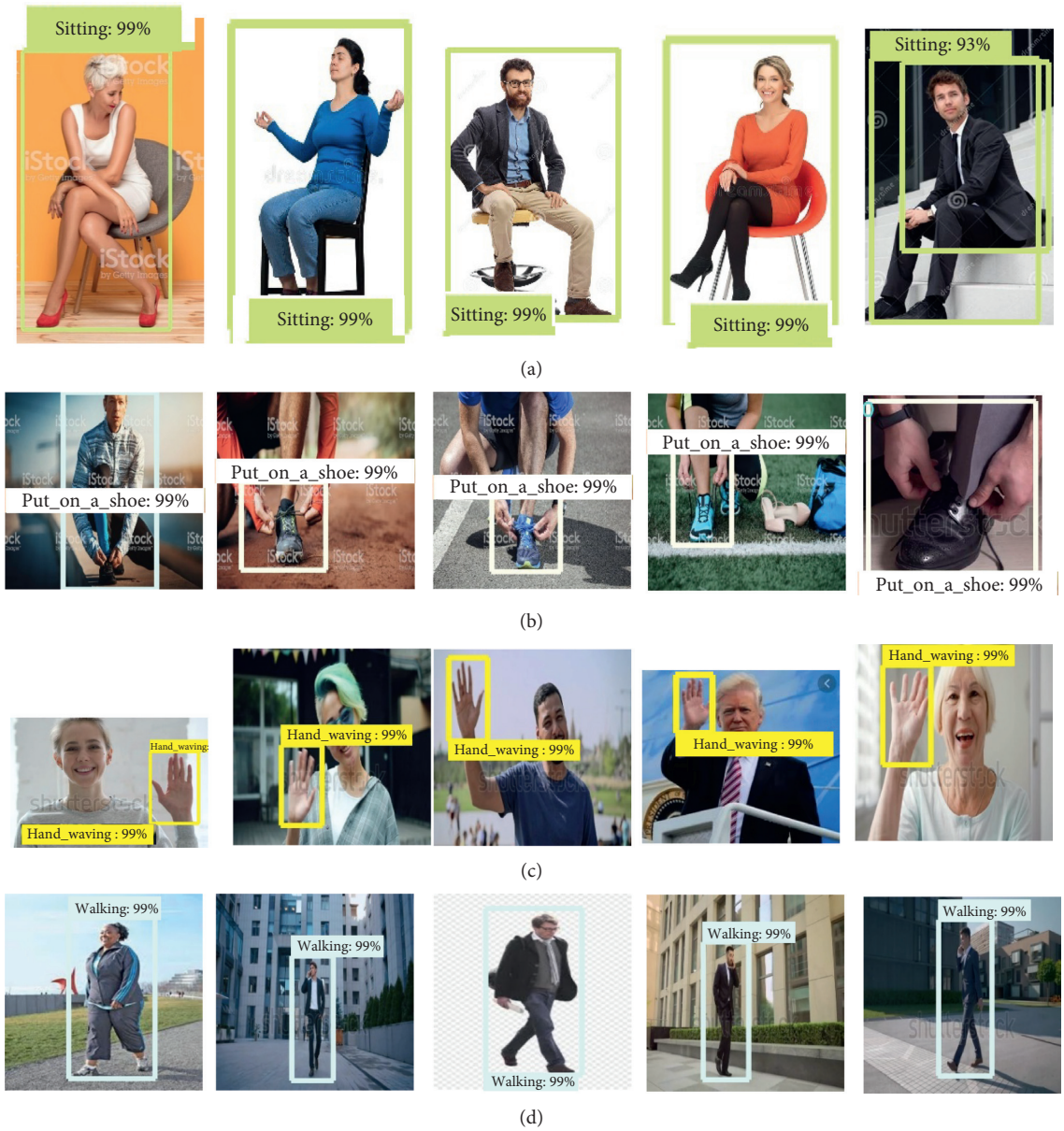


Figure 16: Training results with **Tensorflow** (RCNN + InceptionV2) for (a) sitting down, (b) wearing shoes, (c) waving hands, and (d) walking.

and action. The results show that the system is feasible with an accuracy of over 98%. The graph freezes when converting to Tensorflow with 82% precision and 82% with Tensorflow Lite.

We made video with Tensorflow and Tensorflow Lite models on the i5 computer and 8G RAM. Memory and CPU results used for each model with the proposed dataset are shown in Table 5. The results showed that although the proposed method (SSD + MobilenetV2) has low accuracy, the processing speed is 23.6 times faster than RCNN + InceptionV2 and 37.8 times faster than RFCN + Resnet101.

After moving to Tensorflow Lite format, we created an application to evaluate the real time of the system. To estimate proposal algorithm with real video, we use the input data that includes 30 frames/second with resolution 1920 × 1080 and bit rate of 82 kbps is suitable for real-time applications. The result is shown in Figures 21 and 22. The result shows that proposal model is suitable to apply for real device with accuracy up to 99%.

We perform to compare our model with [30–32]. The result shows that the accuracy of the proposal method is better than that of [30–32], as shown in Figure 23. As a result, the accuracy of the proposal system gains 98% with the Tensorflow model.

The training process is difficult for the computer when the amount of calculation is huge. A simple convolution 2D lattice for classifying 101 layers has about 5 million parameters while the same architecture when structured in 3D leads to 33 million parameters. However, it takes us 3 to 4

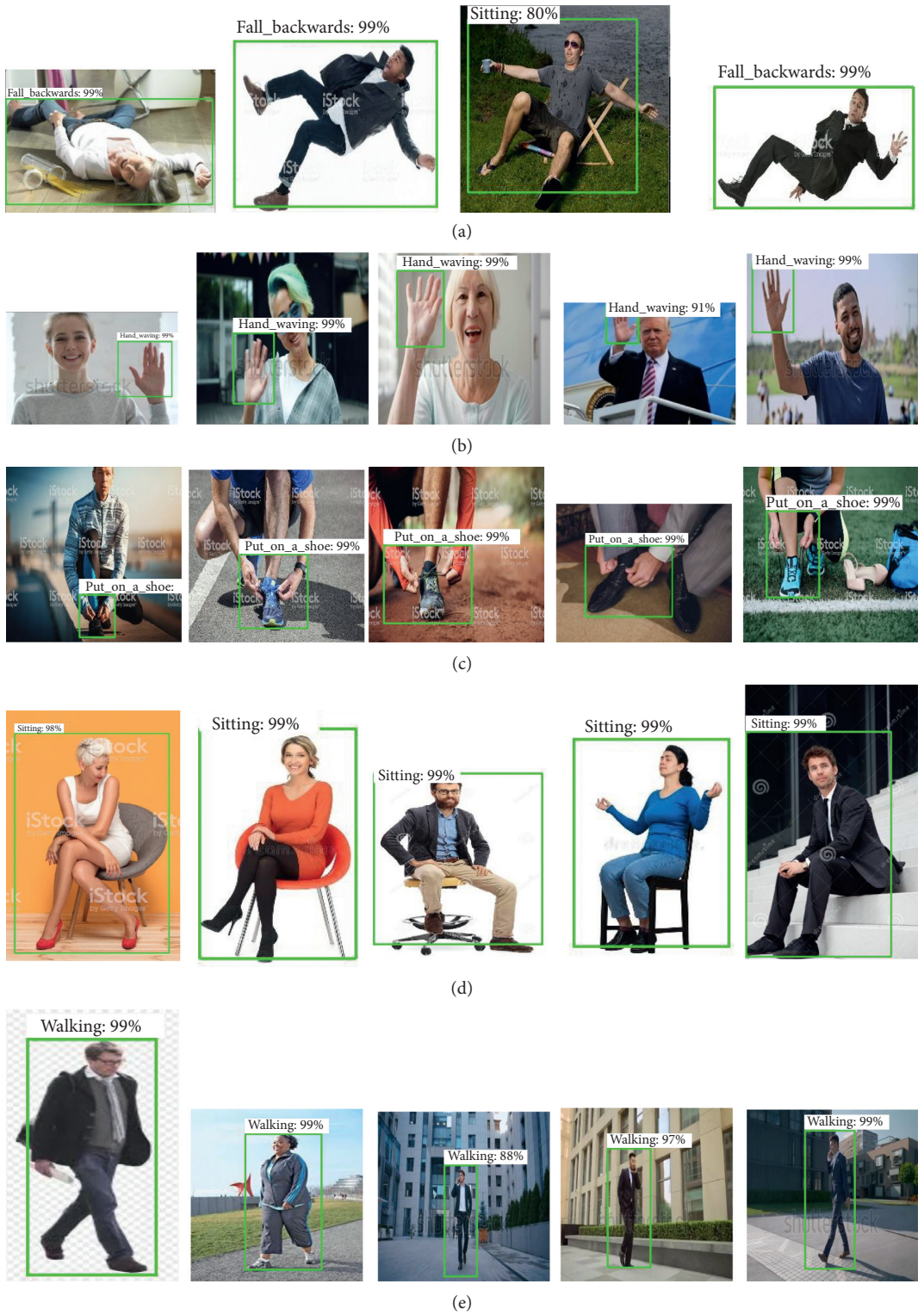


Figure 17: Training results with Tensorflow Lite for (a) falling back, (b) waving hands, (c) wearing shoes, (d) sitting down, and (e) walking.

days to train 3DConvNet on UCF101 and about two months on Sports-1M [33]. This makes finding the extended architecture difficult when used with an i5 configuration with an inadequate 8G ram CPU which is time-consuming. The results comparing the model of computational efficiency with other networks are shown in Table 6.

In Table 6, accuracy of our proposal is not high (about 82%). However, the time execution and size of the model

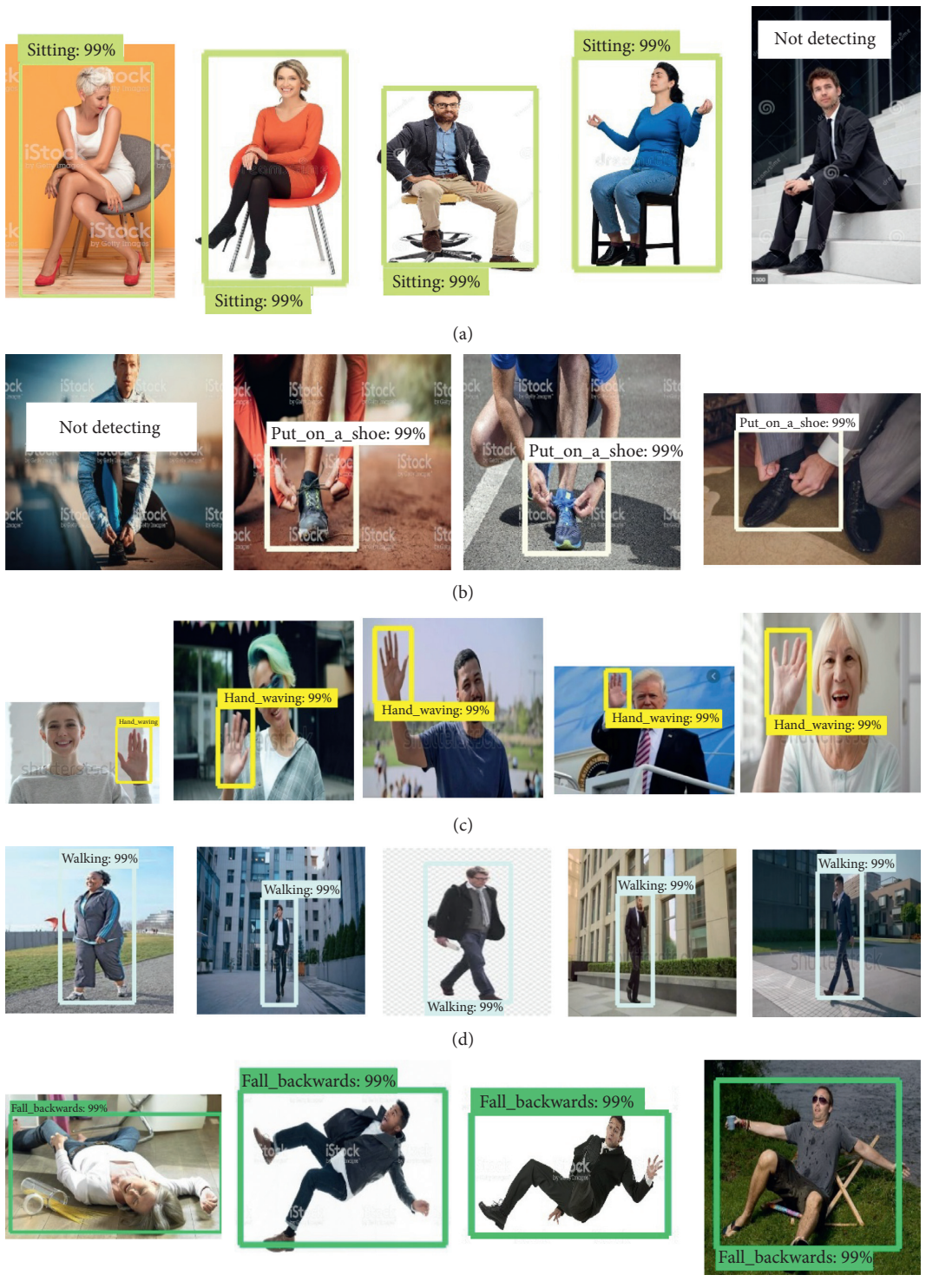


FIGURE 18: Training results with **Tensorflow** (RFCN + Resnet101) for (a) sitting down, (b) wearing shoes, (c) waving hands, (d) walking, and (e) falling back.

using Resnet-101 are 931 MB (Megabyte) and 2.791 (seconds/gesture), while our proposal uses only 19 MB (size of network) and 0.07 (seconds/gesture). Therefore, size of our proposed model has less than about 10 times, and execution speed is less than 40 times from 28 to 36 frames per gesture comparing with Resnet-101.The execution speed of a model usually depends on the number of parameters of the model. However, it also depends on the computational complexity

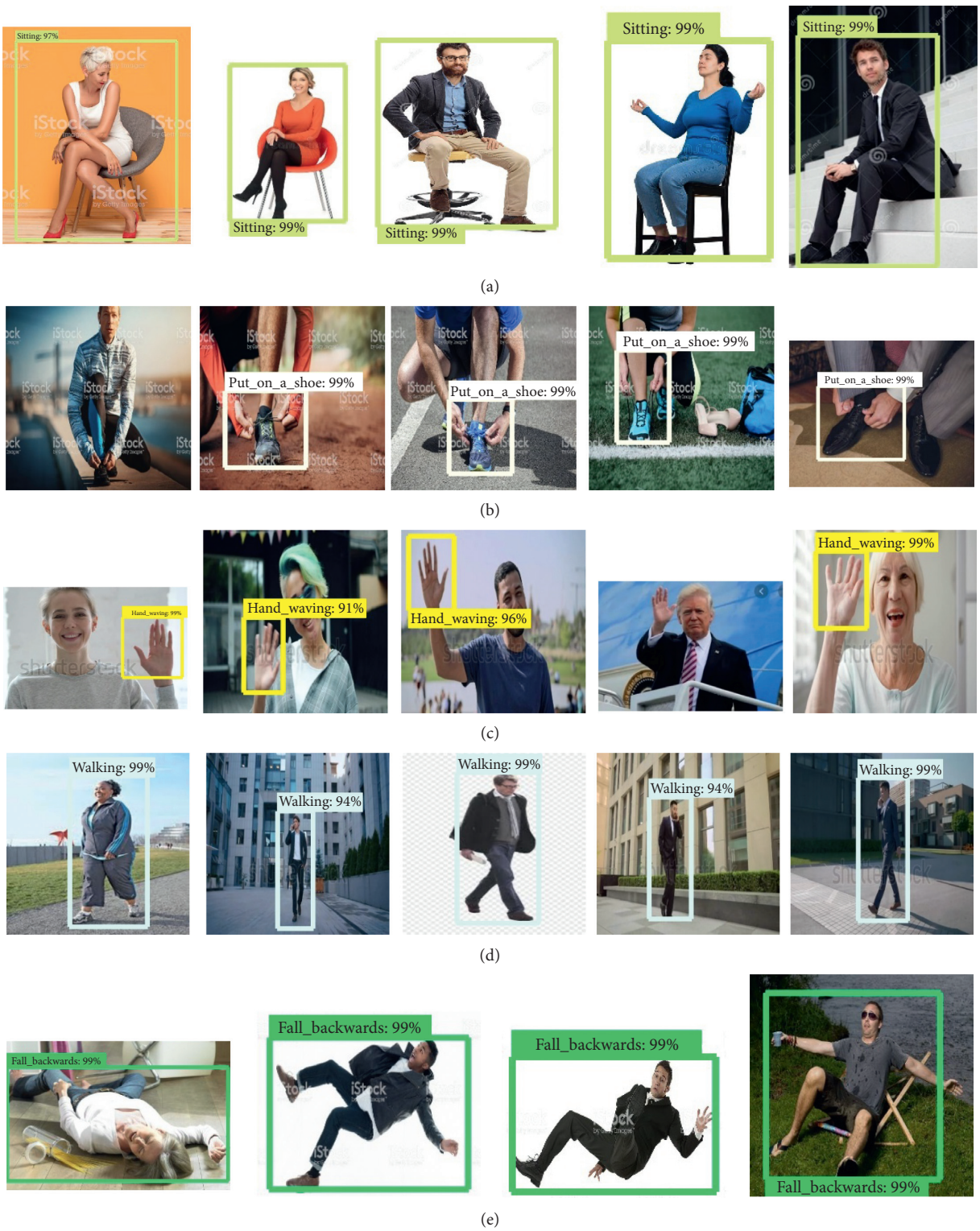


Figure 19: Training results with the proposal model (SSD + MobilenetV2) for (a) sitting down, (b) wearing shoes, (c) waving hands, (d) walking, and (e) falling back.

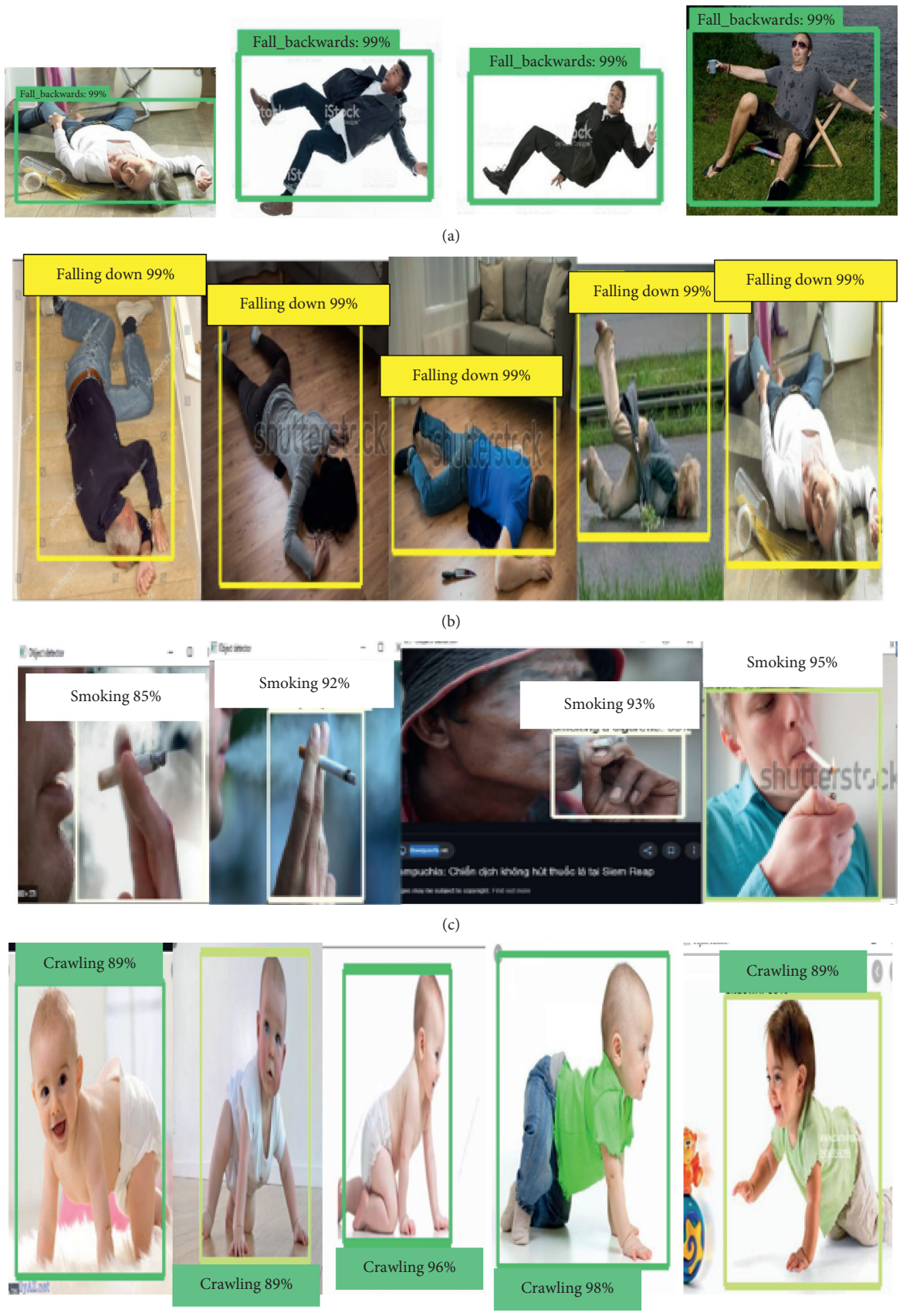


Figure 20: Training results with **Tensorflow** (RCNN + InceptionV2) for (a) falling back, (b) falling down, (c) smoking, and (d) crawling baby.

Table 3: Tensorflow results.

No.	Labels	Accuracy (%)
1	Walking	100
2	Sitting down	100
3	Falling back	100
4	Putting on shoes	100
5	Waving hand	90
6	Baby crawling	90
7	Smoking	90
8	Falling down	100
9	Standing up	100
10	Reading	100
11	Typing	90

Table 4: Tensorflow Lite results.

No.	Labels	Accuracy (%)
1	Walking	90
2	Sitting down	80
3	Falling back	90
4	Putting on shoes	80
5	Waving hand	70
6	Baby crawling	60
7	Smoking	60
8	Falling down	80
9	Standing up	90
10	Reading	90
11	Typing	70

Table 5: Performance evaluation of two models.

Model	Accuracy (%)	Memory (MB)	CPU (%)
Tensorflow	98	317.9	76.7
Tensorflow Lite	82	121.8	30.1

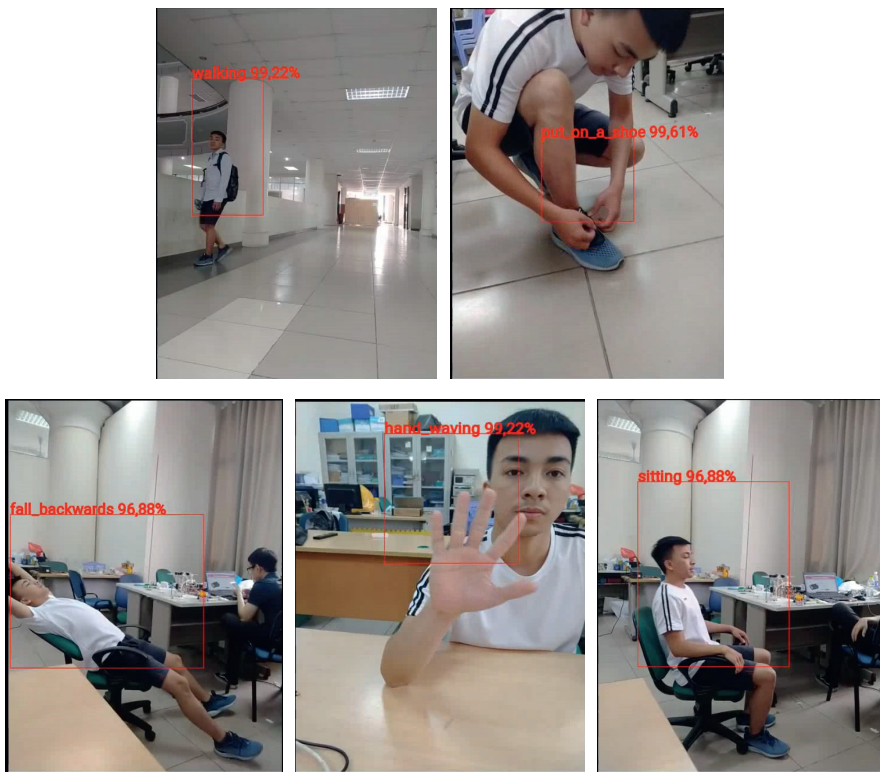


Figure 21: Result of the model with our data.

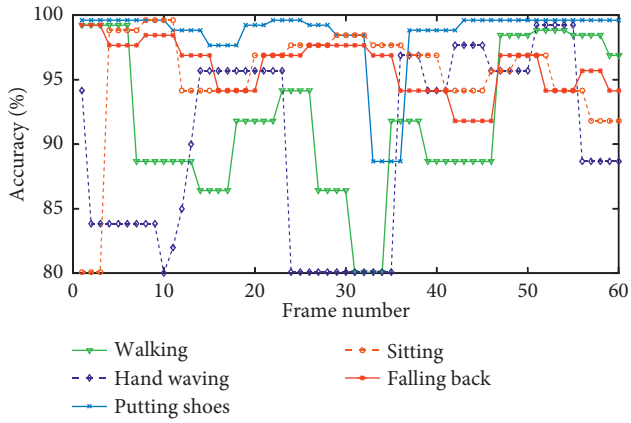


Figure 22: Result of proposal algorithm with real video.

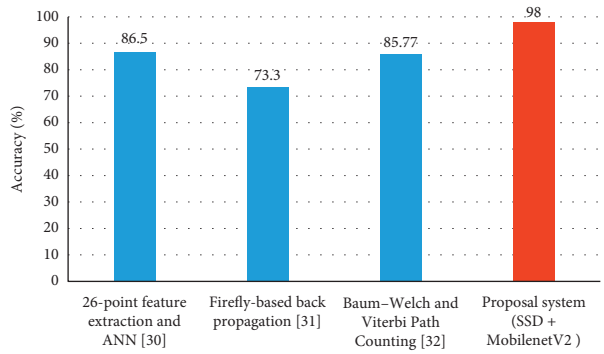


Figure 23: Result of comparing with other methods [30–32].

Table 6: Comparison of results among other methods for gestures.

Method	Number of parameters (million)	Size of network (megabyte)	Accuracy (%)	Average execution time (second/gesture)
MobileNet-V1 [6]	22.4	196	90	0.873
Our proposal	**16.8**	**18.7**	**82**	**0.075**
VGG-16 [5]	138.3	528	90	2.512
Inception-V3 [17]	23.5	92	93	0.890
Xception [19]	22.9	88	94	1.625
Resnet-101 [15]	44.6	171	93	2.791

that is determined by its architecture. By improving the architecture of the model, we will reduce its computational complexity and execution speed.

5. Conclusion

The paper focuses on the use of neural networks in identifying human actions. In this paper, we have identified actions with an accuracy of over 90%. However, the system still has disadvantages such as the result of recognizing the action is not high and the frame rate per second is still low. Therefore, we will perform the steps to increse the frame rate per second, to improve accuracy by increasing the resolution of the input image or using the pretreatment method used in the previous paper [34, 35], and to combine neural networks with other networks to increase the efficiency of calculations and performance with any object.

Data Availability

The data used to support the findings of the study include the self-built data source and the online source via Google and a part of UCF101 [14], BU203 [15], and HMDB51 with eight actions, namely, walking, sitting down, falling back, putting on shoes, waving hand, falling down, smoking, and baby crawling.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was carried out in the framework of the project funded by the Ministry of Education and Training (MOET), Vietnam under the Grant B2020-BKA-06. The authors would like to thank the MOET for their financial support.

References

[1] P. N. Huu and H. N. T. Thu, "Proposal gesture recognition algorithm combining cnn for health monitoring," in *Proceedings of the 2019 6th NAFOSTED Conference On Information And Computer Science (NICS)*, pp. 209–213, Hanoi, Vietnam, 2019.

[2] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction," Computing Research Repository (CoRR), pp. 1–19, 2019.

[3] P. Wang, W. Li, C. Li, and Y. Hou, "Action recognition based on joint trajectory maps with convolutional neural networks," *Knowledge-Based Systems*, vol. 158, pp. 43–53, 2018.

[4] S. A. Khowaja and S.-L. Lee, "Semantic image networks for human action recognition," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 393–419, 2019.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conf. On Learning Representations (ICLR2015)*, pp. 1–14, San Diego, CA, USA, 2015.

[6] A. G. Howard, M. Zhu, B. Chen et al., "Efficient convolutional neural networks for mobile vision applications," pp. 1–10, 2017.

[7] C.-C. Hsieh and D.-H. Liou, "Novel haar features for real-time hand gesture recognition using svm," *Journal of Real-Time Image Processing*, vol. 10, no. 2, pp. 357–370, 2012.

[8] J. Chen, Q. Ou, Z. Chi, and H. Fu, "Smile detection in the wild with deep convolutional neural networks," *Machine Vision and Applications*, vol. 28, no. 1-2, pp. 173–183, 2016.

[9] P. Barros, G. I. Parisi, C. Weber, and S. Wermter, "Emotion-modulated attention improves expression recognition: a deep learning model," *Neurocomputing*, vol. 253, pp. 104–114, 2017.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[11] J. Yu, J. Li, B. Sun, J. Chen, and C. Li, "Multiclass radio frequency interference detection and suppression for sar based on the single shot multibox detector," *Sensors*, vol. 18, no. 11, p. 4034, 2018.

[12] O. Alsing, "Mobile object detection using tensorflow lite and transfer learning," *Computer Science and Communication*, vol. 18, pp. 1–78, 2018.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, CVPR), Las Vegas, NV, USA, 2016.

[14] "Comparison of algorithms for an electronic nose in identifying liquors," *Journal of Bionic Engineering*, vol. 5, no. 3, pp. 253–257, 2008.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, 2016.

[16] L. Wang, Y. Xiong, Z. Wang et al., "Temporal segment networks: towards good practices for deep action recognition," in *Proceedings of the 14th European Conference*, pp. 20–36, Amsterdam, Netherlands, 2016.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, CVPR), Las Vegas, NV, USA, 2016.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2261–2269, CVPR), Honolulu, HI, USA, 2017.

[19] F. Chollet, *Xception: deep learning with depthwise separable convolutions*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 1800–1807, 2017.

[20] T. Zebin, P. J. Scully, N. Peek, A. J. Casson, and K. B. Ozanyan, "Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition," *IEEE Access*, vol. 7, pp. 133 509–133 520, 2019.

[21] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot MultiBox detector," in *Proceedings of the Computer Vision - ECCV 2016*, pp. 21–37, Amsterdam, The Netherlands, 2016.

[22] K. Soomro, A. Zamir, and M. Shah, "Ucf101: a dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. 12, pp. 1–7, 2012.

[23] S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff, "Do less and achieve more: training cnns for action recognition utilizing action images from the web," *Pattern Recognition*, vol. 68, pp. 334–345, 2017.

[24] R. Phadnis, J. Mishra, and S. Bendale, "Objects talk - object detection and pattern tracking using tensorflow," in *Proceedings of the 2018 Second International Conference On Inventive Communication And Computational Technologies (ICICCT)*, pp. 1216–1219, Coimbatore, India, 2018.

[25] K. Muhammad, S. Khan, M. Elhoseny, S. Hassan Ahmed, and S. Wook Baik, "Efficient fire detection for uncertain surveillance environment," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3113–3122, 2019.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the 2018 IEEE/CVF Conference On Computer Vision And Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, 2018.

[27] H. Kang, "Real-time object detection on 640x480 image with vgg16+ssd," in *Proceedings of the 2019 International Conference On Field-Programmable Technology (ICFPT)*, pp. 419–422, Tianjin, China, 2019.

[28] K. Duarte, Y. S. Rawat, and M. Shah, *Videocapsulenet: a simplified network for action detection*, https://arxiv.org/abs/1805.08162, 2018.

[29] D. Sinha and M. El-Sharkawy, "Thin mobilenet: an enhanced mobilenet architecture," in *Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pp. 0280–0285, New York, NY, USA, 2019.

[30] S. Kundu, H. Chhabra, H. S. Chhabra, S. S. Ara, and R. P. Mishra, "Optical character recognition using 26-point feature extraction and ann," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, no. 5, pp. 156–162, 2017.

[31] M. K. Sahoo, J. Nayak, S. Mohapatra, B. K. Nayak, and H. S. Behera, "Character recognition using firefly based back propagation neural network," *Computational Intelligence in Data Mining - Volume 2*, vol. 32, pp. 151–164, 2015.

[32] N. Liu, B. C. Lovell, P. J. Kootsookos, and R. I. A. Davis, "Model structure selection training algorithms for an hmm gesture recognition system," in *Proceedings of the Ninth International Workshop On Frontiers In Handwriting Recognition*, pp. 100–105, Tokyo, Japan, 2004.

[33] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, *Convnet architecture search for spatiotemporal feature learning*, https://arxiv.org/abs/1708.05038, 2017.

[34] N. H. Phat, T. Q. Vinh, and T. Miyoshi, "Video compression schemes using edge feature on wireless video sensor networks," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID 421307, 20 pages, 2012.

[35] P. N. Huu, V. Tran-Quang, and T. Miyoshi, "Image compression algorithm considering energy balance on wireless sensor networks," in *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN 2010)*, pp. 1005–1010, Beijing, China, July 2010.