Received September 15, 2021, accepted October 11, 2021, date of publication October 21, 2021, date of current version October 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3121093

Air-Writing Recognition Based on Deep Convolutional Neural Networks

CHAUR-HEH HSIEH[1], YOU-SHEN LO[2], JEN-YANG CHEN[2], AND SHENG-KAI TANG[2]
[1]College of Artificial Intelligence, Yango University, Fuzhou 350015, China
[2]Department of Electronic Engineering, Ming Chuan University, Taoyuan 320338, Taiwan

Corresponding author: Jen-Yang Chen (jychen@mail.mcu.edu.tw)

This work was supported in part by the Ministry of Science and Technology under Grant MOST 109–2221-E-130–009.

ABSTRACT Air-writing recognition has received wide attention due to its potential application in intelligent systems. To date, some of the fundamental problems in isolated writing have not been addressed effectively. This paper presents a simple yet effective air-writing recognition approach based on deep convolutional neural networks (CNNs). A robust and efficient hand tracking algorithm is proposed to extract air-writing trajectories collected by a single web camera. The algorithm addresses the push-to-write problem and avoids restrictions on the users' writing without using a delimiter and an imaginary box. A novel preprocessing scheme is also presented to convert the writing trajectory into appropriate forms of data, making the CNNs trained with these forms of data simpler and more effective. Experimental results indicate that the proposed approach not only obtains much higher recognition accuracy but also reduces the network complexity significantly compared to the popular image-based methods.

INDEX TERMS Air writing recognition, isolated writing, convolutional neural network (CNN), hand tracking.

I. INTRODUCTION

With the rapid growth of artificial intelligence technology, many intelligent applications have been developed such as smart TV and intelligent robots. The most natural way for humans to communicate with these intelligent systems is dynamic gestures. In recent years, air writing has become one of the most popular dynamic gestures. It is defined as writing alphanumeric with hand or finger movements in a three-dimensional (3D) free space. Air writing is particularly useful for user interfaces that do not allow the user to type on the keyboard or write on the touchpad/touch screen or for text input for intelligent system control [1].

Air-writing recognition is closely related to motion gestures or sign language recognition. Motion gesture recognition methods can be roughly divided into two categories: device-based and device-free. The device-based method requires the use of either handheld or worn devices to obtain hand (or finger) movement in three dimensions, for example, handheld pointing devices such as Wii [1], inertial sensors attached to a glove [2], [3], or motion sensors on the watch [4]. However, the requirement for handheld or worn devices and sensors are troublesome and complicated to use; thus, device-based methods are not commonly used. By contrast, in the device-free method, users do not need to hold or wear any devices; hence, this method is more convenient than the device-based method. Device-free methods can be further divided into vision-based and radio-based methods. The former utilizes 2D or 3D cameras to capture gesture input images. The latter uses radio sensors such as radar [5]–[7] or WiFi [8]–[11] to obtain gesture signals.

Air writing can be realized in three manners [1]: isolated, connected, and overlapped air writing. In isolated writing, the letters are written in an imaginary box with fixed height and width in the field of view of an image, one at a time. In connected writing, multiple letters are written from left to right, which is similar to writing on a paper. In the last manner, one can write multiple letters stacked contiguously one over another in the same imaginary box. We study the isolated writing style in this paper.

Isolated writing is the most essential and popular method. Motion characters are isolated alphanumeric letters written in a unistroke. The steps involved in air-writing recognition generally include hand/finger tracking, feature extraction and classification. The fundamental problems in isolated writing include [1], [12]:

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

(a) tracking of hand and/or fingers,
(b) segmentation of writing acts (or push-to-write),
(c) restrictions on the users' writing due to the limitation of an imaginary box, and
(d) intraclass variability of the writing patterns of a letter.

For vision-based methods, the first problem has been addressed, but different solutions must be used for 2D and 3D image sensors. 2D camera-based systems often utilize color markers on fingers to increase tracking performance since finger tracking without markers is challenging. 3D camera-based systems address the hand/finger tracking problem well simply using the depth information provided by 3D image sensors such as Kinect [12], Leap Motion Controller (LMC) [13], or Intel RealSense camera [14].

Air writing lacks a reference position on the writing plane and thus lacks the beginning and end points of a stroke. Therefore, it needs to automatically detect the start and end coordinates of the characters written in the air. This is referred to as segmentation of writing acts, or the so-called push-to-write problem. One of the possible solutions is to use a specific posture to signal the endpoint of a writing act [1], e.g., a fist posture. However, this will increase the number of gestures that users must remember. When depth information is available, the segmentation of writing acts can be done by merely using a depth threshold [15].

In summary, 3D camera-based systems address the first two problems more conveniently than 2D camera-based systems. However, 3D systems are more complex and expensive.

The imaginary box limits the range of writing. It reduces the variations of letter input such as position, scaling or rotation of the written image. This alleviates the burden of the subsequent processing. Nevertheless, from the users' perspective, this method causes inconvenience and restrictions of users on writing.

In this paper, we design a simple yet effective air-writing recognition approach based on deep convolutional neural networks (CNNs) using a single low-cost 2D web camera. Our approach solves the first three problems in a convenient manner. Furthermore, it can work in a real-time smart-TV-like environment. The major contributions of this article are:

(a) A robust air-writing trajectory acquisition algorithm based on a web camera. The algorithm combines skin and moving features to detect the moving skin region and then applies the Camshift algorithm to track the moving hand. It performs hand tracking only, thus avoiding the complicated procedures for finger tracking. In addition, the proposed algorithm solves the push-to-write problem without using a delimiter. Furthermore, it does not utilize an imaginary box; hence, users can write freely in the air without any restrictions.
(b) A novel data preprocessing scheme. The scheme normalizes the x and y coordinate sequences of the writing trajectory and then combines them into 1D and 2D arrays. The two types of data arrays are employed to train 1D-CNN and 2D-CNN. These simple data arrays make the designed CNNs simpler and more effective than the use of complex written images.
(c) A CNN-based air-writing recognition system using a low-cost web camera. It achieves real-time recognition with a high accuracy of more than 99% and very low network complexity. It outperforms the popular approaches using written images as input.

The remainder of this paper is organized as follows. Section II discusses the related prior work. Section III describes the proposed method in detail. The experimental results are presented in Section IV. Finally, the conclusions are drawn in Section V.

II. RELATED WORK

This work presents a vision-based approach; hence, only the vision-based methods that utilize 2D and 3D cameras in the literature are discussed in the following.

Many studies have been carried with 2D technology. Air-writing recognition can also be considered in parallel to hand gesture recognition. The steps involved in vision-based 2D hand gesture recognition are hand/finger detection and tracking, feature extraction and classification. An early vision-based work by Oka *et al.* [16] used a complex device with an infrared and color sensor for fingertip tracking and recognition.

To simplify the acquisition process of the writing trajectory based on generic 2D video cameras, Roy *et al.* [17] used a marker of a fixed color for writing in the air. The marker tip can be easily detected by color-based segmentation. This work also presented a velocity threshold of writing to achieve the segmentation of writing acts. The proposed method achieved 97.7%, 95.4% and 93.7% recognition rates in person-independent evaluations over English, Bengali and Devanagari numerals, respectively. To incorporate flexibility in marker choice and stable motion tracking under varying lighting conditions, Rahman *et al.* [18] improved the marker tip tracking scheme by a marker calibration mechanism. They presented a dual network configuration consisting of RNN-LSTM (recurrent neural network–long short-term memory) networks for noise elimination and digit recognition. The proposed method yielded a recognition rate of 98.75% for single-digit recognition and 85.27% for multidigit recognition. Due to the variation in writing speed of different users, we argue that it may be difficult to set an appropriate velocity threshold value. Recently, Misra *et al.* [19] developed a hand gesture recognition scheme to recognize letters, numbers, arithmetic operators and ASCII characters using a red marker placed on the finger for fingertip detection. The scheme achieved a recognition rate of 96.95% for the classification of 58 gestures. The above maker-based schemes impose behavioral constraints on the users. Therefore, a marker-free approach is a better option.

Marker-free fingertip detection/tracking is very challenging because a face that is a moving object with similar skin tone to hands is present in video frames, making hand detection and hence fingertip detection much more complicated. The preceding step of fingertip detection is hand segmentation. Numerous works for hand segmentation and fingertip detection based on 2D cameras have been performed in recent years [20]. These works can be divided into two categories: model-less and model-based approaches. The former utilizes color and motion cues, which are simple and can operate in real time but are often less robust with respect to environmental variations such as illumination changes [21], [22]. By contrast, the latter usually provides higher robustness but incurs a high computational cost and requires a large amount of training data, making it unsuitable for real-time application [22]. The air-writing recognition system in [20] proposed a new writing hand pose detection algorithm for the initialization of air writing. Furthermore, the work used a distance-weighted curvature entropy for robust fingertip detection and tracking. In addition, it also proposed a termination criterion based on the moving velocity of the fingertip to serve as a delimiter and mark the completion of the air-writing gesture. Character recognition experiments gave a mean accuracy of 96.11%.

Recently, several air-writing methods based on 3D image sensors have been developed. References [12] presented a Kinect-based online handwriting recognition system. The authors in [13] used LMC to obtain the 3D positions of fingertips, the center of the palm and the orientation of the hand. References [14] developed an air-writing recognition scheme using 3D trajectories of fingertips acquired by an Intel RealSense 3D depth camera.

The writing trajectory can be obtained after the detection/tracking of fingertips is completed. The subsequent processing is to recognize the trajectory, generally including feature representation (extraction) and classification in traditional machine learning. The vision-based representation contains 3D model-based and appearance-based approaches [23]. The appearance-based approach is more widely used than the 3D model-based approach. The appearance-based model is further categorized into color-based, silhouette geometry, deformable Garbarit, and motion-based models [23]. Based on these models, a wide variety of distinguishing features for the representation of gestures have been proposed in recent years [24].

Several classification algorithms have been developed for hand gestures with a temporal dimension. The popular algorithms are the hidden Markov model (HMM) [25], dynamic time warping (DTW) [26], finite state machine (FSM) [27], support vector machine (SVM) [28] and random forest [29].

The methods based on traditional machine learning extract features in a hand-designed manner and then train a classification model. While those methods are robust, they have some limitations in the generalization of the models for many cases. Recently, some deep learning-based approaches have been presented, such as [30]–[33]. The work in [30] mapped 3D fingertip coordinates acquired with LMC into a trajectory image that was used to train a 2D-CNN model. References [31] presented air-writing recognition based on a fusion framework that combines 2D-CNN and BLSTM to model the spatial and temporal features of gestures. The method achieved 99.25% and 99.83% for the alphabet gesture and the numeric gesture, respectively. References [32] developed dynamic hand gesture recognition with a 3D-CNN that fuses the motion volume of normalized depth and image gradient values. References [33] presented an approach for activity and gesture recognition with 3D spatiotemporal data based on a combination of a CNN and an LSTM (long short-term memory) network. The CNN was utilized to extract relevant features from 3D skeleton data, and the LSTM was applied to tackle the activity recognition. In [34], the authors proposed a gesture recognition system based on the data collected by an RGB camera and a depth sensor. By combining 3D-CNN and LSTM networks to extract spatiotemporal features of the gesture sequence, the system achieved a recognition rate of 97.8% for eight selected gestures. References [35] developed an air-writing recognition system using 3D trajectories collected by a depth camera. The LSTM recognizer of the system achieved the highest recognition rates of 99.17% and 99.32% for two different datasets.

The deep-learning-based methods stated above perform better than the conventional methods in recognition rate. However, most of these methods are very complex since they use 2D/3D networks and/or written images. This work aims to develop a simple yet effective system using a 1D or 2D network that utilizes only the writing trajectory data instead of images.

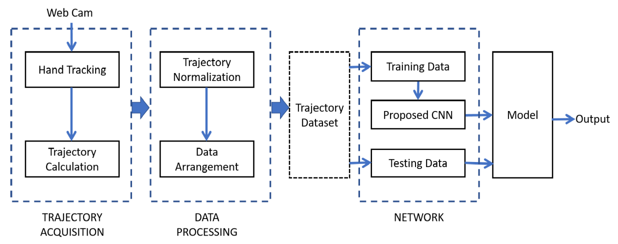

FIGURE 1. **The proposed air-writing recognition system.**

III. PROPOSED METHOD

The proposed air-writing method is shown in FIGURE 1. It includes three stages: trajectory acquisition, data processing and network. The image sequence is acquired with a web camera. Based on the image sequence, a novel hand tracking algorithm is presented to calculate the trajectory of a stroke that a user writes in the air. Then, the trajectory data are processed and converted into two kinds of forms: 1D arrays and 2D arrays. The two kinds of data are formed into trajectory datasets, which are used to learn CNN models in the offline training phase. During online prediction, the system receives real-time data from the web camera and then predicts the digit (or symbol) that the user writes using the learned models. We describe the three main stages of the proposed system as follows.

A. TRAJECTORY ACQUISITION

The purpose of this unit is twofold: to acquire the 2D image that the user writes in the air and to record the coordinate sequence of a stroke, called the trajectory of writing. The trajectory is formed by the coordinates of the center of a moving hand. Thus, detection and tracking of the moving hand from the 2D image sequence is essential in this unit.

Hand detection/tracking has been studied for a long time. However, it is still a challenging issue if both robustness and real-time execution are required. In this paper, we combine skin and moving features to detect the moving skin region and then apply the Camshift algorithm [36] to track the moving hand. The proposed algorithm is robust and can operate in real time.

Hand detection using skin features is a simple and fast method. However, it is easily prone to errors due to the interference of skin-like objects, lighting changes, and skin changes of different users. To solve the interference of skin-like still objects, a moving feature is included in our skin-pixel detection algorithm. Moreover, to adapt the skin feature variation due to the light change and user change, we extract the skin feature of the face region of the user who is writing. Specifically, we use a face detection algorithm to extract the face region of a particular user and calculate the histogram of the H channel in HSV color space. Then, we apply backprojection on the whole image to detect other regions of the image that have the same histogram. The backprojection is calculated from the histogram. It replaces every pixel by its probability to occur in the image.

The detected regions above can be hands or naked body parts. By combining moving features, we can further remove the naked body and detect the moving hand simply using a logical AND operator. Here, the simple frame differencing method is employed to detect moving pixels using adjacent frames, as shown in FIGURE 2(a) and FIGURE 2(b). The binary image after the AND operation may have few small holes and/or noise. We removed them with morphological operations, and the detected hand was clean and complete, as shown in FIGURE 2(c).

Finally, we apply the Camshift algorithm to track the moving hand region and record the center coordinates of the hand region of every frame of a gesture that corresponds to writing a character. The sequences of the coordinates form the trajectory of the character.

The procedure for obtaining the image and trajectory of air writing of a digit (symbol) is as follows. A user sits down in front of a web camera. When the system detects the face of the user, the air-writing session begins. When the user raises his or her hand and writes a character by moving his or her finger, the system will detect the moving hand. The frame immediately after the moving hand is detected is regarded as the start of a stroke. The frame immediately after the moving hand disappears is the end of the stroke. The center coordinates of the hand in all frames between the start and end of the stroke are recorded, forming the trajectory of the digit, as illustrated in FIGURE 3. In addition, the image of

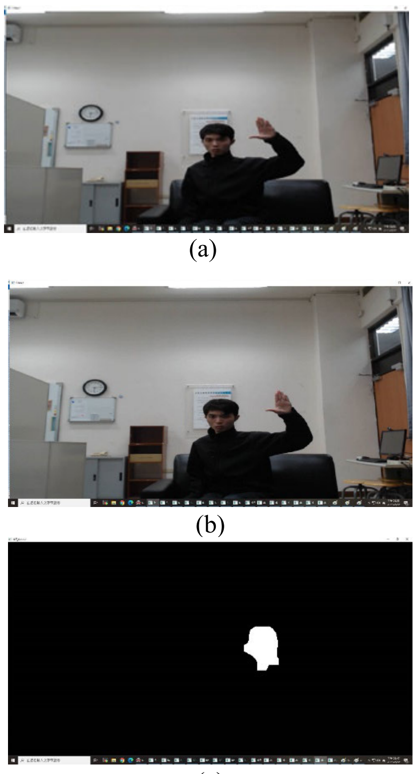
(a)
(b)
(c)

FIGURE 2. **(a) Frame k image, (b) Frame k+1 image, (c) The result after moving the skin filter.**

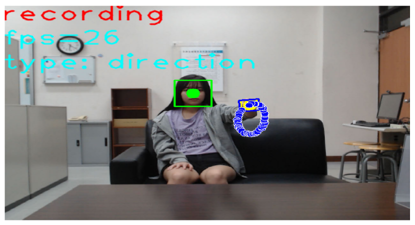

FIGURE 3. **Illustration of writing trajectory.**

the written character is also recorded. The above procedure solves the push-to-write problem without a delimiter in a simple manner.

B. DATA PROCESSING

1) 2D IMAGE

As stated before, we convert the handwritten data into a 2D image. The original size of the captured image is 640 × 480. The user most likely writes commands in different positions in air. To attack the shift variance, we transform the captured image into an image that has a size of 360 × 360 and is located in the middle of a window. The resulting image is shown in the top row of FIGURE 4.

The transformation is performed using the following equations:

$$x_i' = \frac{x_i - x_{avg}}{1.4r} * 360 + 180 \tag{1}$$

$$y_i' = \frac{y_i - y_{avg}}{1.4r} * 360 + 180. \tag{2}$$

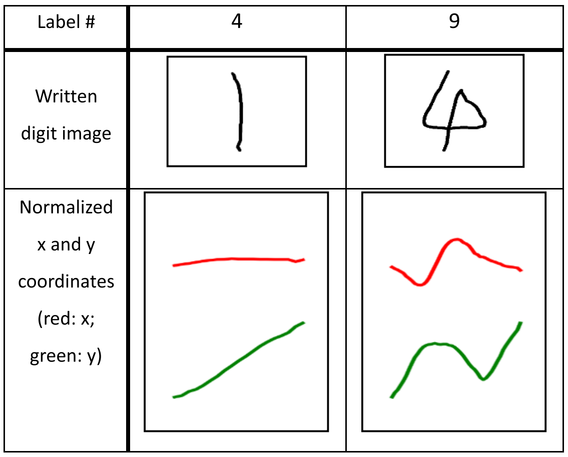

FIGURE 4. **Examples of written images and normalized coordinates of trajectory.**

where

$$x_{avg} = (x_{max} + x_{min})/2 \tag{3}$$

$$y_{avg} = (y_{max} + y_{min})/2 \tag{4}$$

$$r = \begin{cases} \Delta x = x_{max} - x_{min}, & if\ \Delta x \geq \Delta y \\ \Delta y = y_{max} - y_{min}, & if\ \Delta x < \Delta y \end{cases} \tag{5}$$

In the above equations, x_i and y_i are the original coordinates in the x-axis and y-axis, respectively, and x_i' and y_i' are the transformed coordinates. x_{max} and y_{max} are the maximal values of coordinates in the x-axis and y-axis, respectively. The purpose of $1.4 \times r$ in Eqs. (1) and (2) is to leave 0.2r margins on the left boundary and right boundary when the coordinates are plotted as an image. To reduce the computational load in training, the normalized image with a size of 360×360 is further resized to 36×36 with linear interpolation. The resized image set is used to implement the existing popular approaches for comparison.

2) 1D TRAJECTORY

To reduce network complexity, we normalize the coordinates of the original trajectory and then transform them into a 1D sequence. The normalization is performed according to

$$x'_{max} = x_{avg} + r/2 \tag{6}$$

$$x'_{min} = x_{avg} - r/2 \tag{7}$$

$$y'_{max} = y_{avg} + r/2 \tag{8}$$

$$y'_{min} = y_{avg} - r/2. \tag{9}$$

The average values $[x_{avg}, y_{avg}]$ and the range r above are defined in Eq. (3)~(5). Using the new maximal and minimal values defined in Eq. (6)~(9), we can normalize the coordinates into [-1, 1] by

$$x_i'' = \frac{2\left(x_i - x'_{min}\right) - \left(x'_{max} - x'_{min}\right)}{r} \tag{10}$$

$$y_i'' = \frac{2\left(y_i - y'_{min}\right) - \left(y'_{max} - y'_{min}\right)}{r} \tag{11}$$

It is noted that the x coordinate and y coordinate are normalized with the same value r that is the length of the long axis of the written image. Thus, the aspect ratio of the width and height of the written image can be preserved. If normalization is performed independently on the x-axis and y-axis, the aspect ratio will be lost. Our experience indicates that the above aspect-ratio preservation will significantly improve the performance. Examples of the normalized x-coordinate and y-coordinate sequences are illustrated in FIGURE 4.

The times required to complete a writing action of a digit (or character or symbol) are generally not equal for different users. Therefore, the data lengths for different writing actions are not the same. Our experiences indicate that 3 seconds is sufficient to complete the writing of a digit for general users. The frame rate per second (fps) of a camera is 30 in our system. Thus, the data length of a written digit is not greater than 90 points. Here, we set the data length to 100 to consider tolerance. The results obtained from Eqs. (10) and (11) are then upsampled by linear interpolation to obtain 100 points of data for each dimension. To further study the effect of the data arrangement on system performance, we arrange the data into two ways as follows.

a: 1D_PAD

The x-coordinate sequence is padded with 14 zeros at both ends to form a [1,128] sequence. The y-coordinate sequence is processed using the same method. The resulting x and y sequences are then concatenated into a 1-D array [1,256]. The zero-padding is used to isolate the x and y coordinates to avoid their interference with each other in the convolution operation.

b: 2D_NO-PAD

The x coordinate and y coordinate sequences without padding are arranged in a 2D array [2,100], where the x sequence is placed in the first row, and the y sequence is placed in the second row.

C. CONVOLUTIONAL NEURAL NETWORK DESIGN

A basic CNN is composed of several convolutional layers for feature extraction, each of which is usually followed by a pooling layer. The last convolutional layer is also followed by one or more fully connected (dense) layers for classification.

For the 1D and 2D trajectory data stated above, we design a 1D-CNN and 2D-CNN, respectively, to recognize the input digits (or directional symbols). The typical architectures of our proposed 1D-CNN and 2D-CNN for recognizing digits are shown in FIGURE 5(a) and FIGURE 5(b), respectively, and consist of several 1D or 2D convolutional blocks. The architectures for directional symbols are similar; hence, they are neglected here. Each convolutional block contains convolution, maximal pooling, batch normalization, and activation function. The CNN (1D or 2D) applies batch normalization after convolution and before activation because it helps to improve the performance and stability of neural networks [37]. The ReLu function is adopted as the activation function in the hidden layers to avoid the vanishing gradient

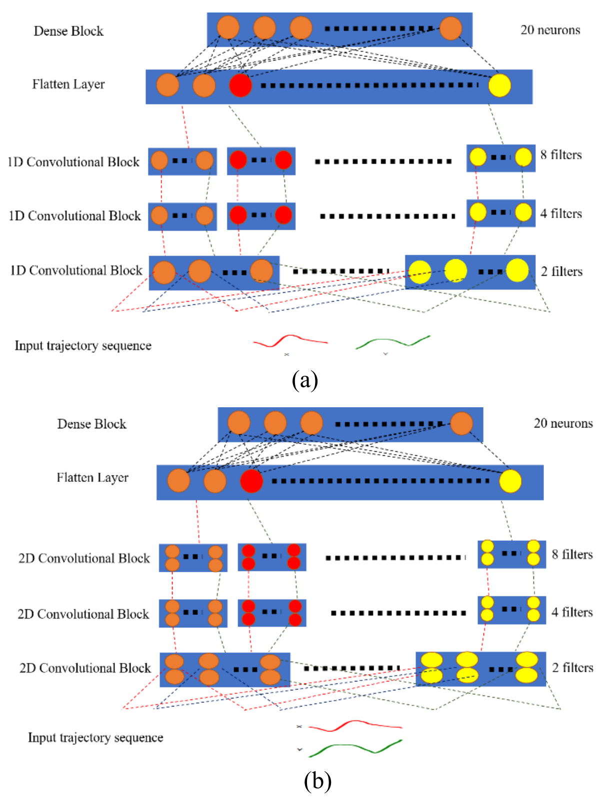

FIGURE 5. **Proposed CNN architectures for trajectory data. (a) 1D-CNN, (b) 2D-CNN.**

problem [38]. The dense block consists of more than one dense layer. The softmax activation function is employed in the output dense layer that maps the real-value input into the prediction probability in the range of [0,1]. Dropout is also employed between the two hidden layers since it is beneficial for avoiding overfitting [38].

We apply the minibatch gradient descent (MBGD) algorithm [39] to learn the CNN model. MBGD computes the gradient of the loss function l with respect to the parameter set φ for every minibatch of n training examples and then performs an update iteratively to obtain the optimal parameter set (corresponding to the minimal loss function) by

$$\varphi_t = \varphi_{t-1} - \rho \nabla_{\varphi} l(\varphi; x^{(i:x+n)}; y^{(i:y+n)}) \tag{12}$$

where x and y are the target output and the predicted output of the network, respectively, ∇_{φ} is the gradient operator, and ρ is the learning rate. MBGD utilizes the backpropagation (BP) scheme to compute the gradient of the loss function. In this work, we choose cross-entropy in Eq. (13) as the loss function.

$$l(\varphi, x, y) = -\sum\nolimits_{i=1}^{N} x_i \log y_i \tag{13}$$

In MBGD training, choosing a suitable fixed learning rate is difficult. A learning rate that is too small will lead to slow convergence, while a learning rate that is too large will hinder convergence and cause the loss function to oscillate around the minimum or even cause divergence. To solve this problem, several gradient descent optimization algorithms with different learning rate schedules have been reported such as Adagrad, Adadelta, Adam and RMSprop [39]. The Adam (adaptive moment estimation) algorithm is employed in this work since it has been experimentally proven to be effective [39]. Adam estimates the individual adaptive learning rates of different parameters according to the first and second moments of the gradients of the loss function. The update algorithm of the parameter set of the network is given by [39]

$$\varphi_t \leftarrow \varphi_{t-1} - \frac{\eta \widehat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}} \tag{14}$$

where η is a fixed learning step size, ε is a very small constant, and $\widehat{m}_t$ and $\hat{v}_t$ are the first and second moments after bias correction, respectively, that are calculated by

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{15}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{16}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{17}$$

where g_t denotes the gradient of the lost function at time t, and β_1 and β_2 are the decay constants for the first and second moments, respectively.

IV. EXPERIMENTAL RESULTS

A. DATASET CREATION

Our work aims to develop an air-writing system for smart-TV control. Since no public dataset for this purpose is available, we create two types of datasets. One is a digit dataset, including 0 to 9 with different writing directions: clockwise and anticlockwise; hence, it contains a total of 20 symbols. The other is a pure directional symbol dataset that includes 16 symbols. FIGURE 6(a) and FIGURE 6(b) show the images of the symbols in the two datasets.

Each of the two datasets contains a training set and test set. The training and test sets were obtained by 6 and 8 volunteers, respectively, with ages ranging from 20 to 30 years old. To improve the robustness of our system, the volunteers for the collection of training data and test data are completely different. For the digit dataset, the training set size and test set size are 12,000 and 1,600, respectively. For the directional symbol dataset, they are 9,600 and 1,280, respectively.

K-fold cross-validation is the most popular method in various applications of machine learning [38], [40]. We apply K-fold cross-validation for tuning the hyperparameters using the training sets. To find the best K value, we divide the training set into training and validation subsets with different size ratios and then carry out training and validation for each size ratio. The result indicates that K = 5 (size ratio = 4:1) achieves the highest recognition rate; therefore, we used 5-fold cross-validation in this work.

B. OPTIMIZATION OF CNN CONFIGURATIONS AND PERFORMANCE EVALUATION

This subsection discusses the design of optimal CNN configurations and the evaluation of performance in terms

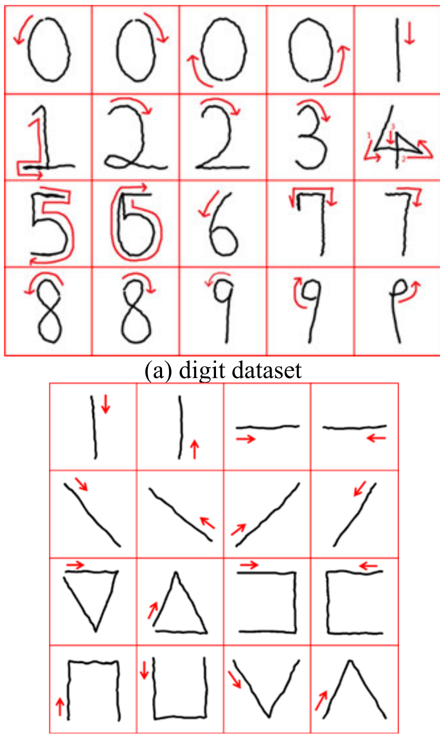

FIGURE 6. Two datasets used in our work.

of two metrics: recognition rate and network complexity presented in [8]. Here, the total number of parameters of the CNN is used to evaluate the network complexity. A highly complex network often requires a large amount of training data to avoid overfitting and involves a high computational cost. We apply the popular deep learning platform Keras to calculate the two metrics [41].

1) OPTIMIZATION OF CNN CONFIGURATIONS

The design of CNN configurations involves hyperparameter optimization, that is, to set various hyperparameters, including the number of hidden layers, nodes of every layer, batch size, and learning rate. The goal of the hyperparameter optimization in our work is to find the set of hyperparameters that obtains the highest recognition rate given network complexity. Some general strategies such as grid search and random search have been presented to find the best network that achieves the highest recognition rate [29]. However, these strategies do not consider the metric of the network complexity. Therefore, in this work, we apply the process of trial and error based on our experience to obtain improved network configuration that balances recognition accuracy and network complexity.

For 1D zero-padding data, we design two 1D-CNNs: the first has two convolutional layers and is denoted as 1D-2, and the second has three convolutional layers and is denoted as 1D-3. The leftmost columns in TABLE 1 and TABLE 2 show the best configurations of 1D-2 and 1D-3, respectively. The other two columns list the network parameters for the two datasets. It is obvious that the 3-layer CNN achieves a higher recognition rate with much lower network complexity for the digit set than the 2-layer CNN.

TABLE 1. 1D-2 network configuration and its performance for 1D_PAD data.

	Digit	Direction
Layers	Parameters	Parameters
Conv1D	Filter 4/kernel 43	Filter 4/kernel 79
Maxpooling1D	2	4
Conv1D	Filter 8/kernel 43	Filter 8/kernel 3
Maxpooling1D	2	4
Dropout	0.025	0.05
Flatten		
Dense	20	16
Dropout	0.025	0.1
Dense	20	16
Recognition rate	99.275%	99.750%
Total Num of Params	1,2240	2,760

TABLE 2. 1D-3 network configuration and its performance for 1D_PAD data.

	Digit	Direction
Layers	Parameters	Parameters
Conv1D	Filter 2/kernel 47	Filter 2/kernel 35
Maxpooling1D	2	2
Conv1D	Filter 4/kernel 27	Filter 4/kernel 27
Maxpooling1D	2	2
Conv1D	Filter 8/kernel 19	Filter 8/kernel 3
Maxpooling1D	4	4
Dropout	0.025	0.05
Flatten		
Dense	20	16
Dropout	0.025	0.1
Dense	20	16
Recognition rate	99.375%	99.766%
Total Num of Params	3932	2732

Similarly, we also train two 2D-CNN models for [2,100] data. The results are shown in TABLE 3 and TABLE 4, respectively, and indicate that the 3-layer CNN is much improved with respect to network complexity (4548 vs. 8252 on average of two datasets) at the cost of a small decrease (approximately 0.2%) in recognition rate.

2) PERFORMANCE EVALUATION

Currently, there is no standard air-writing dataset available for smart TV control. Therefore, we implement the popular networks using the written images of our datasets. The results in terms of recognition rate and total number of

TABLE 3. **2D-2 network configuration and its performance for 2D_NO-PAD data.**

	Digit	Direction
Layers	Parameters	Parameters
Conv2D	Filter 4/kernel (3, 19)	Filter 4/kernel (3, 3)
Maxpooling2D	(1, 2)	(1, 4)
Conv2D	Filter 8/kernel (3, 43)	Filter 8/kernel (3, 19)
Maxpooling2D	(1, 2)	(1, 4)
Dropout	0.025	0.05
Flatten		
Dense	20	16
Dropout	0.025	0.1
Dense	20	16
Recognition rate	99.075%	99.703%
Total Num of Params	12,808	3,696

TABLE 4. **2D-3 network configuration and its performance for 2D_NO-PAD data.**

	Digit	Direction
Layers	Parameters	Parameters
Conv2D	Filter 2/kernel (3, 31)	Filter 2/kernel (3, 99)
Maxpooling2D	(1, 2)	(1, 2)
Conv2D	Filter 4/kernel (3, 35)	Filter 4/kernel (3, 35)
Maxpooling2D	(1, 2)	(1, 3)
Conv2D	Filter 8/kernel (3, 23)	Filter 8/kernel (3, 7)
Maxpooling2D	(1, 4)	(1, 3)
Dropout	0.025	0.05
Flatten		
Dense	20	16
Dropout	0.025	0.1
Dense	20	16
Recognition rate	98.750%	99.625%
Total Num of Params	5,608	3,688

parameters are compared with those of our methods. Here, three popular networks widely used in the literature [30]–[35] are implemented for comparison: (a) pure 2D-CNN, (b) 2D-CNN plus LSTM (2DCNN-LSTM), and (c) 2D-CNN plus SVM (2DCNN-SVM). The first is an end-to-end pure CNN approach that uses 2DCNN to complete both the feature extraction and classification tasks. The last two are hybrid approaches that utilize 2D-CNN for feature extraction and then apply SVM or LSTM for classification. The results of the three methods are shown in TABLE 5-7. It is noted that 2DCNN-SVM is implemented in a simple manner, as reported in [42]. Specifically, at the output layer of the CNN, instead of the conventional softmax function with the cross entropy function, the Euclidean norm with the squared hinge loss is used [42].

TABLE 5. **2DCNN network configuration and its performance for original written image.**

	Digit	Direction
Layers	Parameters	Parameters
Conv2D	Filter 16/kernel (3, 3)	Filter 16/kernel (19, 19)
Maxpooling2D	(2, 2)	(2, 2)
Conv2D	Filter 36/kernel (3, 3)	Filter 36/kernel (3, 3)
Maxpooling2D	(2, 2)	(2, 2)
Conv2D	Filter 64/kernel (11, 11)	Filter 64/kernel (3, 3)
Maxpooling2D	(2, 2)	(2, 2)
Dropout	0.25	0.25
Flatten		
Dense	128	128
Dropout	0.5	0.5
Dense	20	16
Recognition rate	83.588%	79.656%
Total Num of Params	418,008	165,076

TABLE 6. **2DCNN-LSTM network configuration and its performance for original written image.**

	Digit	Direction
Layers	Parameters	Parameters
Conv2D	Filter 16/kernel (4, 4)	Filter 16/kernel (4, 4)
Maxpooling2D	(2, 2)	(2, 2)
Dropout	0.4	0.4
Conv2D	Filter 16/kernel (4, 4)	Filter 16/kernel (4, 4)
Maxpooling2D	(2, 2)	(2, 2)
Dropout	0.4	0.4
Conv2D	Filter 16/kernel (4, 4)	Filter 16/kernel (4, 4)
Maxpooling2D	(2, 2)	(2, 2)
Dropout	0.4	0.4
Flatten		
LSTM	Unit 10	Unit 10
Dense	20	16
Recognition rate	95%	93.75%
Total Num of Params	36,548	36,504

The best models in every case of the proposed CNNs that are trained with the 1D form and 2D form of the writing trajectory are selected and compared with the three popular models mentioned above in TABLE 8. By using the 1D form and 2D form of the writing trajectory data, we obtain the first two models in this table. It is noted that the values of the

TABLE 7. 2DCNN-SVM network configuration and its performance for original written image.

	Digit	Direction
Layers	Parameters	Parameters
Conv2D	Filter 4/kernel (4, 4)	Filter 4/kernel (4, 4)
Maxpooling2D	(2, 2)	(2, 2)
Conv2D	Filter 16/kernel (4, 4)	Filter 16/kernel (4, 4)
Maxpooling2D	(2, 2)	(2, 2)
Conv2D	Filter 16/kernel (4, 4)	Filter 16/kernel (4, 4)
Flatten		
Dense	20	16
Recognition rate	81.6%	75.34%
Total Num of Params	2,263,288	1,811,700

TABLE 8. Performance comparison of our proposed methods and the popular methods.

Models	Recognition rate (%)	Total Num of Params
Proposed 1DCNN for Trajectory 1D Data	99.571	3,332
Proposed 2DCNN for Trajectory 2D Data	99.188	4,648
2DCNN for Written Images	81.622	291,542
2DCNN-LSTM for Written Images	94.375	36,526
2DCNN-SVM for Written Images	78.450	2,037,494

recognition rate or total number of parameters in this table are calculated from the average of two datasets, i.e., digit and direction. We conclude that 1D data concatenated by the x-coordinate sequence and y-coordinate sequences achieve the best performance in terms of the recognition rate and network complexity. Moreover, the proposed approach using trajectory data in 1D or 2D form as the input of CNN is superior to the popular methods that use written images as the input.

V. CONCLUSION

In this paper, we have proposed deep CNNs for the recognition of air-writing digits and special direction symbols for smart-TV-like control. A robust air-writing trajectory acquisition algorithm based on a web camera is developed that performs hand tracking only, avoiding the use of complicated procedures for finger tracking. By preprocessing the writing trajectory, we obtain one-dimensional and two-dimensional data that are utilized to design 1D-CNN and 2D-CNN, respectively. Through careful design and optimization of hyperparameters, the proposed CNNs achieve excellent performance with a recognition rate greater than 99%. Among our proposed networks, 1D-CNN is slightly better than 2D-CNN. The two CNN models based on trajectory data significantly outperform the existing popular methods using written images. In addition, the network complexity of our proposed neural networks is much lower than those of the popular methods, and our systems can operate in real time.

ACKNOWLEDGMENT

The authors would like to express their sincere thanks to the anonymous reviewers for their invaluable comments, which truly helped towards an effective presentation of this paper. The authors would also like to thank B.H. Nien for the implementation of the popular methods.

REFERENCES

[1] M. Y. Chen, G. AlRegib, and B.-H. Juang, "Air-writing recognition—Part I: Modeling and recognition of characters, words, and connecting motions," *IEEE Trans. Human-Mach. Syst.*, vol. 46, no. 3, pp. 403–413, Jun. 2016.

[2] C. Amma, D. Gehrig, and T. Schultz, "Airwriting recognition using wearable motion sensors," in *Proc. 1st Augmented Hum. Int. Conf.*, Megeve, France, Apr. 2010, pp. 1–8.

[3] C. Amma, M. Georgi, and T. Schultz, "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors," in *Proc. Int. Symp. Wearable Comput. (ISWC)*, Newcastle, U.K., Jun. 2012, pp. 52–59.

[4] D. Moazen, S. A. Sajjadi, and A. Nahapetian, "AirDraw: Leveraging smart watch motion sensors for mobile human computer interactions," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2016, pp. 442–446.

[5] M. Arsalan and A. Santra, "Character recognition in air-writing based on network of radars for human-machine interface," *IEEE Sensors J.*, vol. 19, no. 19, pp. 8855–8864, Oct. 2019.

[6] S. K. Leem, F. Khan, and S. H. Cho, "Detecting mid-air gestures for digit writing with radio sensors and a CNN," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1066–1081, Apr. 2020.

[7] P. Wang, J. Lin, F. Wang, J. Xiu, Y. Lin, N. Yan, and H. Xu, "A gesture air-writing tracking method that uses 24 GHz SIMO radar SoC," *IEEE Access*, vol. 8, pp. 152728–152741, 2020.

[8] C.-H. Hsieh, J.-Y. Chen, and B.-H. Nien, "Deep learning-based indoor localization using received signal strength and channel state information," *IEEE Access*, vol. 7, pp. 33256–33267, 2019.

[9] Z. Fu, J. Xu, Z. Zhu, A. X. Liu, and X. Sun, "Writing in the air with WiFi signals for virtual reality devices," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 473–484, Feb. 2019.

[10] X. Cao, B. Chen, and Y. Zhao, "Wi-Wri: Fine-grained writing recognition using wi-fi signals," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, Aug. 2016, pp. 1366–1373.

[11] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "WiDraw: Enabling hands-free drawing in the air on commodity WiFi devices," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, Paris, France, Sep. 2015, pp. 77–89.

[12] X. Zhang, Z. Ye, L. Jin, Z. Feng, and S. Xu, "A new writing experience: Finger writing in the air using a Kinect sensor," *IEEE MultiMedia*, vol. 20, no. 4, pp. 85–93, Apr. 2013.

[13] P. Kumar, R. Saini, P. P. Roy, and D. P. Dogra, "Study of text segmentation and recognition using leap motion sensor," *IEEE Sensors J.*, vol. 17, no. 5, pp. 1293–1301, Mar. 2017.

[14] J. Malik, A. Elhayek, S. Guha, S. Ahmed, A. Gillani, and D. Stricker, "DeepAirSig: End-to-end deep learning based in-air signature verification," *IEEE Access*, vol. 8, pp. 195832–195843, 2020.

[15] C. Qu, "Online kinect handwritten digit recognition based on dynamic time warping and support vector machine," *J. Inf. Comput. Sci.*, vol. 12, no. 1, pp. 413–422, Jan. 2015.

[16] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Comput. Graph. Appl.*, vol. 22, no. 6, pp. 64–71, Nov. 2002.

[17] P. Roy, S. Ghosh, and U. Pal, "A CNN based framework for unistroke numeral recognition in air-writing," in *Proc. 16th Int. Conf. Frontiers Handwr. Recognit. (ICFHR)*, Aug. 2018, pp. 404–409.

[18] A. Rahman, P. Roy, and U. Pal, "Air writing: Recognizing multi-digit numeral string traced in air using RNN-LSTM architecture," *Social Netw. Comput. Sci.*, vol. 2, no. 1, pp. 1–13, Jan. 2021.

[19] S. Misra, J. Singha, and R. H. Laskar, "Vision-based hand gesture recognition of alphabets, numbers, arithmetic operators and ascii characters in order to develop a virtual text-entry interface system," *Neural Comput. Appl.*, vol. 29, no. 8, pp. 1–19, Jan. 2017.

[20] S. Mukherjee, S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, "Fingertip detection and tracking for recognition of air-writing in videos," *Expert Syst. Appl.*, vol. 136, pp. 217–229, Dec. 2019.

[21] H. Liang, J. Yuan, and D. Thalmann, "3D fingertip and palm tracking in depth image sequences," in *Proc. 20th ACM Int. Conf. Multimedia*, 2012, pp. 785–788.

[22] P. Krejov and R. Bowden, "Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima," in *Proc. 10th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, Apr. 2013, pp. 1–7.

[23] A. S. Al-Shamayleh, R. Ahmad, M. A. M. Abushariah, K. A. Alam, and N. Jomhari, "A systematic literature review on vision based gesture recognition techniques," *Multimedia Tools Appl.*, vol. 77, no. 21, pp. 28121–28184, Apr. 2018.

[24] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 1, pp. 131–153, Jan. 2019.

[25] A. Schick, D. Morlock, C. Amma, T. Schultz, and R. Stiefelhagen, "Vision-based handwriting recognition for unrestricted text input in mid-air," in *Proc. 14th ACM Int. Conf. Multimodal Interact.*, Monica, CA, USA, 2012, pp. 217–220.

[26] G. Plouffe and A.-M. Cretu, "Static and dynamic hand gesture recognition in depth data using dynamic time warping," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 2, pp. 305–316, Feb. 2016.

[27] R. Verma and A. Dev, "Vision based hand gesture recognition using finite state machines and fuzzy logic," in *Proc. Int. Conf. Ultra Mod. Telecommun. Workshops*, St. Petersburg, Russia, Oct. 2009, pp. 1–6.

[28] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2368–2377, Dec. 2014.

[29] A. Kuznetsova, L. Leal-Taixe, and B. Rosenhahn, "Real-time sign language recognition using a consumer depth camera," in *Proc. IEEE ICCV Workshops*, Sydney, NSW, Australia, Dec. 2013, pp. 83–90.

[30] J.-T. Hu, C.-X. Fan, and Y. Ming, "Trajectory image based dynamic gesture recognition with convolutional neural networks," in *Proc. 15th Int. Conf. Control, Autom. Syst. (ICCAS)*, Busan, South Korea, Oct. 2015, pp. 1885–1889.

[31] B. Yana and T. Onoye, "Air-writing recognition based on fusion network for learning spatial and temporal features," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E101.A, no. 11, pp. 1737–1744, Nov. 2018.

[32] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," in *Proc. IEEE CVPR*, Boston, MA, USA, Jun. 2015, pp. 1–7.

[33] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vélez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition," *Pattern Recognit.*, vol. 76, pp. 80–94, Apr. 2018.

[34] N. L. Hakim, T. K. Shih, S. P. Kasthuri Arachchi, W. Aditya, Y.-C. Chen, and C.-Y. Lin, "Dynamic hand gesture recognition using 3DCNN and LSTM with FSM context-aware model," *Sensors*, vol. 19, no. 24, p. 5429, Dec. 2019.

[35] M. S. Alam, K.-C. Kwon, M. A. Alam, M. Y. Abbass, S. M. Imtiaz, and N. Kim, "Trajectory-based air-writing recognition using deep neural network and depth sensor," *Sensors*, vol. 20, no. 2, p. 376, Jan. 2020.

[36] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," Intel Technol. J., Tech. Rep. Q2, 1998, pp. 1–15.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 448–456.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747

[40] G. James, *An Introduction to Statistical Learning*. New York, NY, USA: Springe, 2017.

[41] *How to Calculate the Number of Parameters for a Convolutional and Dense layer in Keras*. Accessed: 2021. [Online]. Available: https://androidkt.com/calculate-number-parameters-convolutional-dense-layer-keras/

[42] M. Abien and F. Agarap, "An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification," 2019, *arXiv:1712.03541*. [Online]. Available: https://arxiv.org/abs/1712.03541

CHAUR-HEH HSIEH received the Ph.D. degree in electrical engineering from the Chung Cheng Institute of Technology (CCIT), Taiwan, China, in 1990. In 1981, he joined the Faculty of the Department of Electrical Engineering, CCIT, where he was a Professor, in 1993. Since 1996, he has been a Full Professor with the Information Engineering Department, I-Shou University (ISU). From 1999 to 2002, he served as the Chairperson for the department. He was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, from February 2006 to July 2006. From 2007 to 2018, he was a Professor with Ming-Chuan University. He is currently a Professor with the College of Artificial Intelligence, Yango University, Fuzhou, China. His current research interests include signal and image processing, computer vision, and deep learning. He is an IET Fellow.

YOU-SHEN LO was born in Taipei, Taiwan, China, in 1958. He received the B.S. degree from National Chiao Tung University, in 1980, the M.S. degree from the National Taiwan University of Science and Technology, in 1990, and the Ph.D. degree from National Taiwan University, in 2000. He is currently an Associate Professor with the Department of Electronic Engineering, Ming-Chuan University. His current research interests include image/video processing, computer vision, machine learning, and embedded microprocessor applications.

JEN-YANG CHEN received the M.S. degree in electrical engineering from Tatung University, Taipei, Taiwan, China, in 1992, and the Ph.D. degree in electrical engineering from Tamkang University, in 2000. He is currently a Full Professor/chairperson with the Department of Electronic Engineering, Ming-Chuan University. His research interests include deep learning, intelligent control, soft computing, and embedded micro-controller application and design.

SHENG-KAI TANG was born in Taipei, Taiwan, China, in 1996. He received the master's degree in electronic engineering from Ming Chuan University, in 2020. His current research interests include machine learning, deep learning, image/video processing, computer vision, and face recognition.