Deep Learning Based Hand Gesture Recognition and UAV Flight Controls

Bin Hu Jiacun Wang

Monmouth University, West Long Branch, NJ 07764, USA

Abstract: Dynamic hand gesture recognition is a desired alternative means for human-computer interactions. This paper presents a hand gesture recognition system that is designed for the control of flights of unmanned aerial vehicles (UAV). A data representation model that represents a dynamic gesture sequence by converting the 4-D spatiotemporal data to 2-D matrix and a 1-D array is introduced. To train the system to recognize designed gestures, skeleton data collected from a Leap Motion Controller are converted to two different data models. As many as 9 124 samples of the training dataset, 1 938 samples of the testing dataset are created to train and test the proposed three deep learning neural networks, which are a 2-layer fully connected neural network, a 5-layer fully connected neural network and an 8-layer convolutional neural network. The static testing results show that the 2-layer fully connected neural network achieves an average accuracy of 96.7% on scaled datasets and 12.3% on non-scaled datasets. The 5-layer fully connected neural network achieves an average accuracy of 98.0% on scaled datasets and 89.1% on non-scaled datasets. The 8-layer convolutional neural network achieves an average accuracy of 89.6% on scaled datasets and 96.9% on non-scaled datasets. Testing on a drone-kit simulator and a real drone shows that this system is feasible for drone flight controls.

Keywords: Deep learning, neural networks, hand gesture recognition, Leap Motion Controllers, drones.

1 Introduction

Hand gesture recognition research has been gaining more and more attention from researchers worldwide. In addition to ordinary application in daily life, gesture recognition starts entering into virtual reality, medical systems, education, communication systems, games, mobile devices, automotive, etc.

There are basically three kinds of hand gesture recognition technologies: data glove based[1], vision based[2] and radar based[3]. A data glove is an interactive device, resembling a glove worn on the hand, which facilitates tactile sensing and fine-motion control in robotics and virtual reality[4]. The output of the sensors can be used to control video games, presentations, music and visual entertainment. The glove based approach is inconvenient because gloves are bulky to carry. One advantage of using data gloves is that it does not need to extract the human gestures from background[5]. However, due to their high cost and calibration requirements, data gloves don't have the same wide range of applications as vision-based gesture recognition systems[6].

The radar based approach is a technology which transmits a radio wave towards a target, and then the receiver of the radar intercepts the reflected energy from that target. The radar waves bounce off your hand and back to the receiver, allowing it to interpret changes in the shape or movement of your hand[7, 8]. This technology is still under research. The most promising one is Google Soli, which was approved by the U.S. government in January 2019. On the other hand, the vision based approach is gaining momentum because the user does not need to carry devices but can perform a gesture in a much more natural way. The early studies widely used color cameras for the development of gesture recognition systems[9], while today's recognition systems such as Microsoft Kinect, Leap Motion Controller[10, 11] and Intel RealSense usually use depth images as a modality. The Leap Motion Controller is a small USB powered device that uses two monochromatic infrared cameras and three infrared LEDs to track movements and motion made by hands and fingers in a roughly 1 m hemispherical 3D space. The Leap Motion Controller has always been one of the most widely used cameras for gesture recognition, because it allows users to act as freely as they do in real life. Its low-cost and depth sensors can capture video in real-time under any ambient lighting and outputs the skeletal data. Furthermore, hand gestures can be any simple hand movement or complex shape for Leap Motion Controller, making it an obvious choice for this study.

Hand gestures can be defined as either the static postures[12] or dynamic gestures[13]. In this paper, we consider dynamic gestures only.

Research Article
Special Issue on Improving Productivity Through Automation and Computing
Manuscript received March 26, 2019; accepted July 4, 2019; published online September 30, 2019
Recommended by Associate Editor Xian-Dong Ma

Most of the research of static gesture recognition focuses on neural-network-centered approaches[14–16]. For recognition of dynamic gestures, one of the most common methodologies is to represent gestures with spatiotemporal sequences[17, 18]. Since Starner and Pentland[9] started to use hidden Markov models (HMM) to recognize gestures, HMM had become a common method for gesture recognition[19, 20]. Other approaches, such as hidden conditional random fields[21], autoregressive models[22], fuzzy-logic, Kalman-filtering[23–25], support vector machines[26] and recurrent neural networks[27, 28] are used in some studies.

As a branch of the study of machine learning, deep learning models have drawn interests from both research society and industry rapidly because it is so powerful in learning and classification. Many research fields such as speech recognition, computer vision and natural language processing, etc. applied this technology[29]. In recent years, one of the most important neural networks, convolutional neural network (CNN) has achieved the best performance in gesture recognition fields[30–33].

This paper is an extension of the conference paper[34]. It attempts to apply the deep learning approach in dynamic hand gesture recognition. The engineering target of the study is the control of unmanned aerial vehicles (UAV). A data model that represents a dynamic gesture sequence by converting the 4-D spatiotemporal data to a 2-D matrix and a 1-D array is introduced. We designed two fully connected neural networks and one convolutional neural network in order to find the one with the best performance. We created two data models for neural network training and testing. We also implemented the software system based on deep learning neural networks. It is our understanding that this is the first work reported that uses Leap Motion Controllers as input devices in deep learning network based hand gesture recognition.

The rest of the paper is organized as follows: Section 2 introduces basic concepts of deep learning. Section 3 gives an overview of the hand gesture recognition system, Leap Motion Controllers and UAVs. Section 4 introduces hand gestures and datasets that are used in the system. Section 5 presents the deep learning networks, the core of the system. Section 6 discusses neural network training and testing results. Section 7 concludes the paper with some future work suggestions.

2 Deep learning fundamentals

Machine learning is a branch of artificial intelligence. It involves methods that identify algorithms and implement systems by which a computer can learn based on examples given as input. Such a system generates a common model based on the learning data so that it can predict the results of new data sets. Machine learning algorithms have been successfully applied in many research fields such as face recognition, hand gesture recognition and image recognition, etc.

2.1 Artificial neural network (ANN)

The idea of artificial neural networks comes from the way the human brain works. Researchers performed specific tasks such as classification, pattern recognition, clustering, etc. on computers to simulate the work process of the human brain. A neural network is based on connections of nodes, i.e., artificial neurons, which is illustrated in Fig. 1. Each neuron has one or more incoming connections whose task is to collect digital signals from other neurons; each connection has a weight that applies to each signal "propagating" over the connection. Each neuron has one or more output connections that transmit signals to other neurons. An activation function is used to compute the output signal value. It receives signal from input connections with other neurons. The output signal is calculated by applying the activation function to the input weighted sum. These functions change dynamically between –1 and 1 or 0 and 1.

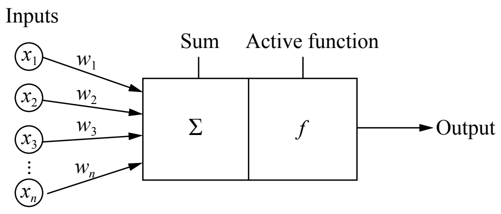


Fig. 1 Artificial neurons

There are three widely used activation functions with different complexity and output:

1) Step function: For a fixed threshold x, if the mathematical input is above or below the threshold, then the function returns 0 or 1, respectively.

2) Linear combination: The weighted sum of the input values is subtracted from a default value.

3) Sigmoid: It produces a sigmoid curve.

2.2 Deep learning

A traditional neural network consists of up to two layers and is not suitable for large-scale network computing[30]. Therefore, researchers designed a structure of neural networks with more than 3 layers. The layers which are between the input layer and output layer are called "hidden layers". These are deep learning neural networks. Deep learning is a research field of machine learning based on specific types of learning mechanisms. This learning concept is inspired by the way that the brain processes information, learns and responds to external stimuli. A deep learning neural network algorithm predicts the output by processing the raw input data which go through non-linear transformation in hidden layers.

In a deep learning neural network, the input value of each neuron in each hidden layer is calculated from the previous layer depending upon the weights, bias and activation function which performs a non-linear transformation. As the data go through the input layer and each hidden layer, the data become more and more complex and abstract. Thus, the deep learning neural networks can solve more complex problems than the simple neural networks because of multiple non-linear transformation layers.

2.3 Learning process

The transfer of copyright form should be properly completed and signed after a paper has been accepted. The learning process of a neural network optimizes the weights iteratively. The deep learning algorithm modifies the weights based on a set of labeled examples of the training set to achieve the optimal network performance.

The objective is to minimize the loss function, which indicates the degree to which network behavior deviates from expectations. Then, the performance of the network is validated on a test set consisting of objects other than the training set.

The backpropagation algorithm

The following shows the basic steps of the back-propagation algorithm training procedure[35]:

1) Initialize the network with random weights.

2) Forward pass: Calculates the difference between the desired output and the actual output of the network.

3) Backward pass: Adapt weights in the current layer to minimize the error function, starting from the output layer to the input layer.

Weight optimization

The process of weights optimization uses the gradient descent (GD) algorithm to optimize the weights. GD proceeds in the following steps:

1) Initialize values of model parameters randomly.

2) Calculate the gradient G of the error function according to each parameter of the model.

3) Change the parameters of the model to make them move in the direction of reducing the error.

4) Repeat Steps 2) and 3) until the value of G is close to zero.

2.4 Neural network architectures

There are various types of architecture in neural networks. The difference among them are the number of layers, the number of neurons in each layer and the way nodes are connected in each layer. This section only shows the two architectures used in this paper.

Multilayer network

In multi-layer networks, input and output layers define input and output, and there are hidden layers whose complexity implements different behaviors of the network. Fig. 2 shows the multi-layer network architecture:

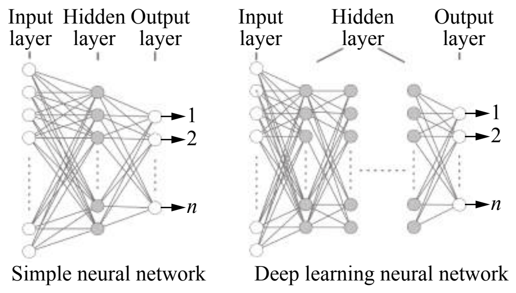


Fig. 2 Multilayer network architecture

Convolutional neural networks

Convolutional neural networks (CNNs) are specially designed for image recognition. A convolutional neural network consists of an input layer, multiple hidden layers and an output layer. Typically, the hidden layers of a CNN consist of convolutional layers, pooling layers, and fully connected layers.

In CNN models, each input image is processed by a series of convolutional layers with filters and pooling happening first. Then the processed image passes through the fully connected layers. At last, the output layer classifies an object with probabilistic values between 0 and 1 by using the Softmax function[36, 37]. Fig. 3 shows an example of CNN architecture.

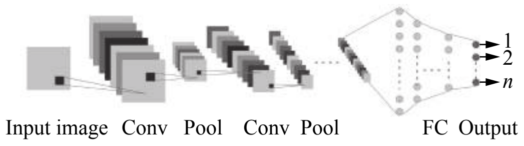


Fig. 3 CNN network architecture

3 System overview

There are three subsystems in our system: the gesture input component, the deep learning neural network component, and the UAV control component, as illustrated in Fig. 4.

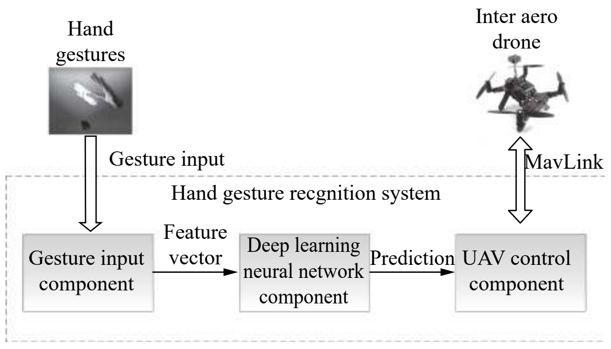


Fig. 4 System architecture

A dynamic gesture that is generated by the user is processed and used as a command to control the drone movement. The Leap Motion Controller tracks and recognizes hand gesture information and then delivers the skeleton data to the data preprocessing module, which does the feature selection and feature scaling work on the Leap Motion skeleton data and then composes the feature vectors. The feature vectors are sent to the deep learning neural network module. The gesture is recognized and the prediction is delivered to the UAV control module. Finally, the UAV control module extracts the prediction of the movement command and controls the behavior of the Inter Aero drone through WiFi in the MavLink protocol.

3.1 Gesture input

As is seen from Fig. 5, this component is made up of two modules: The Leap Motion Controller and data preprocessing module.

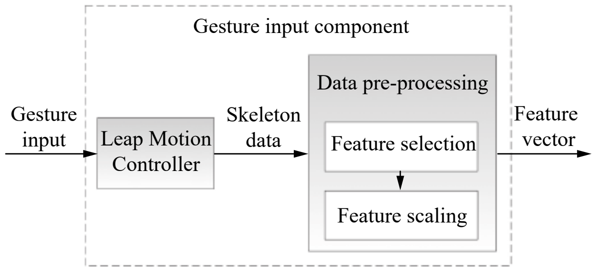


Fig. 5 Gesture input

The gesture input component is responsible for receiving hand gestures with the transducers function provided by the Leap Motion Controller. It turns skeleton data which is obtained in the form of frames into a 2-D feature vector and then delivers it to the deep learning neural network subsystem.

The Leap Motion Controller delivers the discrete position and motion. It uses 2 optical sensors and 3 infrared lights. The information that Leap Motion Controller delivers is the hand skeleton model it sees in form of frames. The skeleton model consists of objects (like fingers or tools) positions relative to the Leap Motion Controller's origin point.

The Leap Motion Controller provides a set of frames of data when it tracks hands and fingers. In object oriented programming terminology, the frames are objects containing all the tracked information regarding hands, fingers and tools within the leap motion field of view, therefore posterior data processing is required to provide the algorithms with a suitable feature vector.

3.2 The UAV

This UAV control component is responsible for transferring the classes that are recognized from the deep learning network into commands that modify the drone's behavior. Fig. 6 provides an overall scheme of the elements that play a role. There are two modules in this component: the UAV management module and the command conversion module.

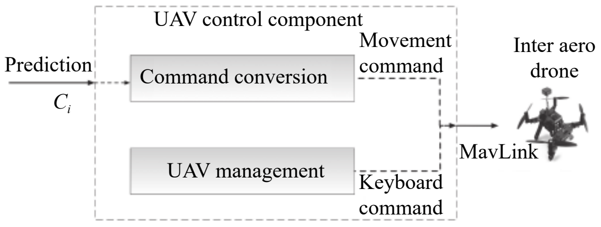


Fig. 6 UAV controlling

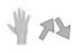
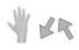
The UAV management module is to manage the UAV status. It is responsible for establishing communication with the Mavros node checking the UAV status, requesting services to Mavros and translating the recognized commands into understandable orders for the UAV movement.

4 Gestures and datasets

4.1 Gestures

In this study, we defined 10 dynamic gestures for drone motion controls, which are listed in Table 1. The name of each gesture suggests the motion pattern of the controlled target.

Table 1 List of hand gestures

Target class C_i	Gesture description	Illustration
0	Move forward	
1	Move backward	
2	Turn left	
3	Turn right	
4	Move up	
5	Move down	
6	Turn clockwise	
7	Turn anticlockwise	
8	Special movement 1	
9	Special movement 2	

4.2 Raw data

Each frame of raw data we extract from the Leap Motion Controller include:

x_1: pitch angle of hand
x_2: yaw angle of hand
x_3: roll angle of hand
x_4: X-coordinate of palm center
x_5: Y-coordinate of palm center
x_6: Z-coordinate of palm center
x_7: hand grab strength
x_8: yaw angle of thumb tip
x_9: yaw angle of middle finger tip
x_{10}: X-coordinate of thumb tip
x_{11}: Y-coordinate of thumb tip
x_{12}: Z-coordinate of thumb tip
x_{13}: X-coordinate of middle finger tip
x_{14}: Y-coordinate of middle finger tip
x_{15}: Z-coordinate of middle finger tip

In this study, one dynamic gesture is composed of 45 frames tracking gesture information. Therefore, for each gesture, the input data to our system is a 45×15 matrix.

4.3 Feature scaling

Feature scaling is a widely used approach to improve learning and recognition accuracy when the range of independent features of data are far apart. It can be done with the following simple normalization:

$$scaled\_value = \frac{value - \min\_value}{\max\_value - \min\_value}. \quad (1)$$

So now we have two data models: original data and scaled data. Datasets of these two models will be fed into the deep learning network separately to verify its performance.

5 Deep learning model

In this study we designed, trained and tested three different neural networks so as to find the one with the best performance, as shown in Fig. 7. They are a 2-layer fully connected network, a 5-layer fully connected network, and an 8-layer convolutional network. Figs. 8–10 illustrate the architectures of the three types of networks.

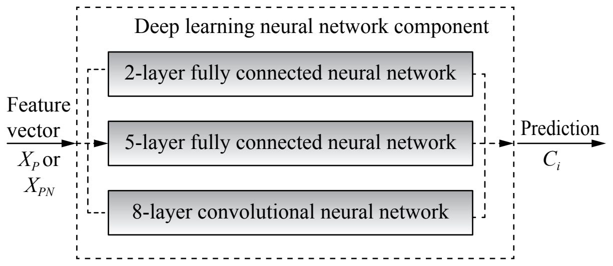

Deep learning neural network component
2-layer fully connected neural network
5-layer fully connected neural network
8-layer convolutional neural network
Feature vector
X_P or X_{PN}
Prediction
C_i

Fig. 7 The deep learning model

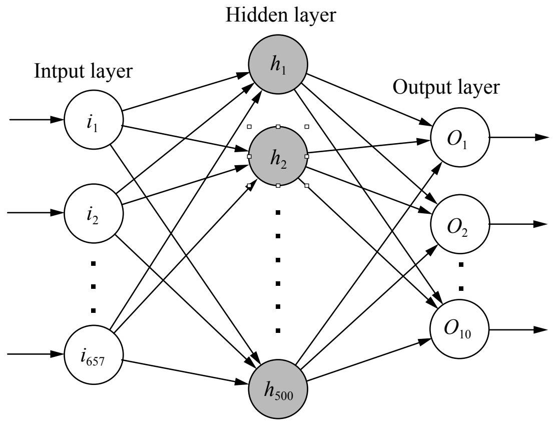


Fig. 8 Architecture of 2-layer fully connected neural network

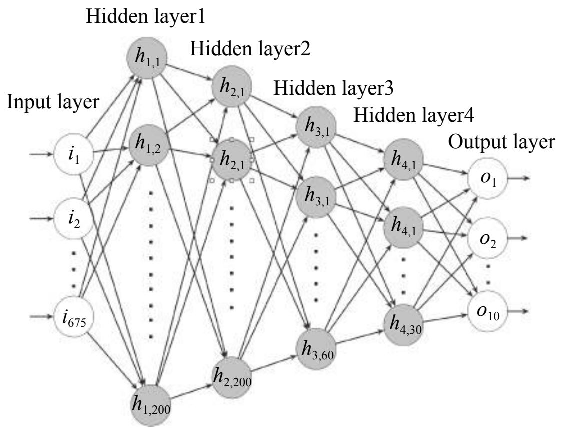


Fig. 9 Architecture of 5-layer fully connected neural network

The high-level designs of these three networks are described in Tables 2–4.

As shown in Table 4, there are three convolutional layers in the 8-layer CNN model. The first convolutional layer has a size of 6 × 6 and a depth of 6 filters. It uses a stride of 1. With an input of 45 × 15 elements, the output of the first convolutional layer would have a size of 45 × 15 × 6 elements. The first pooling layer have a stride value of 2, the resulting output of the first pooling layer would have a size of 23 × 8 × 6 elements.

After the three convolutional layers is a first fully connected layer that contains 200 neurons and receives 2 × 1 × 24 elements from the previous layer. This is followed by a ReLU and a dropout layer.

Finally, the output of the first fully connected layer is fed to the output layer that assigns a probability for each class. The output layer receives an output of 200 elements from first fully connected layer and contains 10 neurons. This layer is to assign a probability for each gesture class.

6 Training, testing and results

The experiment was conducted in a software environ-

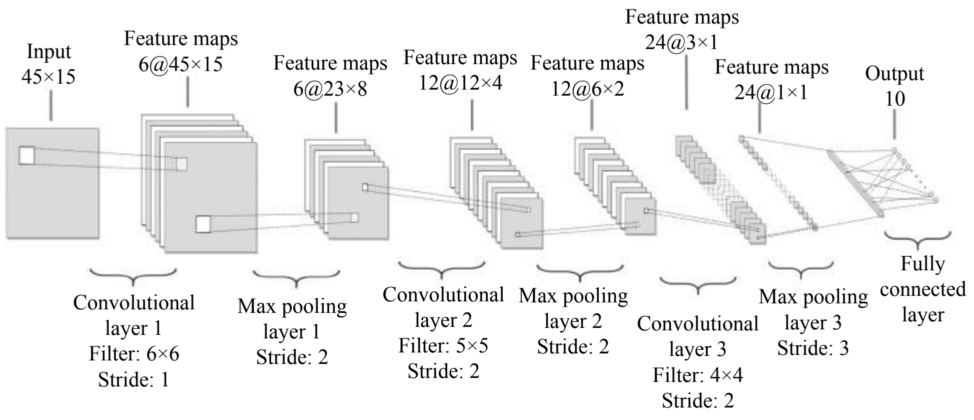


Fig. 10 Architecture of 8-layer convolutional neural network

Table 2 A 2-layer fully connected network

Layer	Number of nodes
Input layer	675
Hidden layer	500
Output layer	10

Table 3 A 2-layer fully connected network

Layer	Number of nodes
Input layer	675
Hidden layer 1	200
Hidden layer 2	100
Hidden layer 3	60
Hidden layer 4	30
Output layer	10

Table 4 An 8-layer convolutional network

Layer	Number of nodes
Input layer	45×15
Convolutional layer 1	Filter: 6×6, Depth: 6, Stride: 1
Max pooling layer 1	Stride: 2
Convolutional layer 2	Filter: 5×5, Depth: 12, Stride: 2
Max pooling layer 2	Stride: 2
Convolutional layer 3	Filter: 4×4, Depth: 24, Stride: 2
Max pooling layer 3	Stride: 2
Fully connected layer	200
Output layer	10

ment as follows: macOS High Sierra, Python 2.6, Tensorflow 1.3.0, Dronekit 3.0, and Leapmotion SDK: 2.6.5.

To examine the raw data model and normalized data model separately, two training datasets, two evaluation datasets and two testing datasets are created. Each of the two training datasets contains a total of 9 124 gestures of data. Each of the two testing datasets contains a total of 1 938 gestures of data. Seven graduate students at Monmouth University participated in the recording of these hand gestures.

6.1 Training

During the training process, the neural network performance such as accuracy and cross entropy can be evaluated to determine the network parameters at the same time with the evaluation dataset. In this study, one evaluation sample from the evaluation dataset is fed to the neural network every ten training steps.

In the training process, for each dataset, the network was evaluated at four different training batch sizes: 25, 50, 75 and 100. So, a total of 8 neural network models were trained for each of three types of neural networks, which translates to a total of $8 \times 3 = 24$ training cycles.

We followed general rules to select the deep learning network parameters before we started the training. Table 5 shows the training parameters of the 2-layer fully connected neural network.

During the training, the Tensor board collects the training accuracy and cross entropy loss and adjusts the

Table 5 Training parameters of the 2-layer fully connected neural network

Parameters name	Value
Training dataset size	9 124
Training batch size	25, 50, 75, 100
Training learning rate base	0.8
Training learning rate decay	0.99
Training regularization rate	0.000 1
Layer 1 weights	mean = 0
	std = 0.01
Layer 1 bias	0
Layer 2 weights	mean = 0
	std = 0.01
Layer 2 bias	0

network parameters accordingly. The target is to eliminate the cross entropy.

Figs. 11 and 12 show the change the accuracy and the cross entropy over training samples applied to the 2-layer network model with raw data input accuracy on batch sizes of 25, 50, 75 and 100.

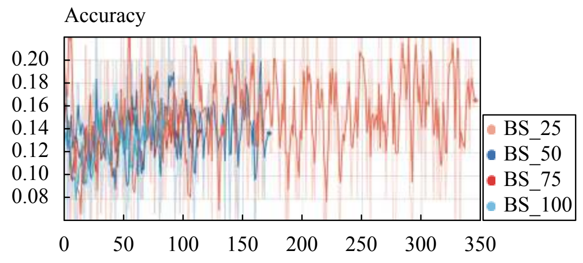


Fig. 11 Accuracy in training the 2-layer fully connected neural network on raw data

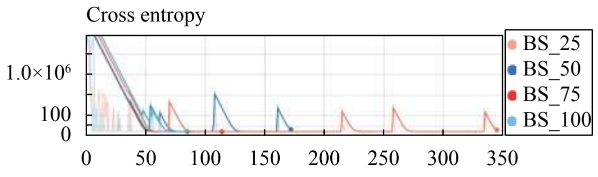


Fig. 12 Cross entropy in training the 2-layer fully connected neural network on raw data

In the accuracy diagram, the x-axis represents the training steps, and the y-axis represents the percentage of accuracy. In the cross entropy loss diagram, x-axis represents the training steps, and the y-axis is the cross entropy loss. Each color in the both diagrams represents a training batch size. For example, BS_50 is the case that the red curve's batch size is 50.

Figs. 13 and 14 show the change of accuracy and the change of cross entropy over training samples applied to the 2-layer network model with scaled data as input and batch sizes of 25, 50, 75 and 100.

It can be seen that when the input is raw data, the accuracy for each training batch size is very low and does not converge. The loss for each training batch size is big. This means the neural net learns nothing. The dataset or neural net model needs to be improved.

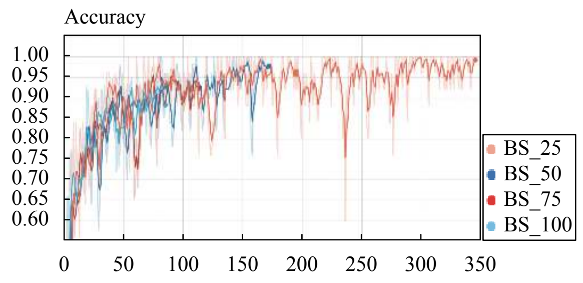


Fig. 13 Accuracy in training the 2-layer fully connected neural network on scaled data

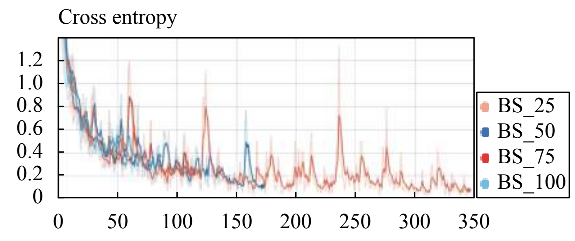


Fig. 14 Cross entropy in training the 2-layer fully connected neural network on scaled data

When the input is scaled data, however, the accuracy for each training batch size is increased sharply with the increasing of training steps. At the end of training, the accuracy is close to 1 when the batch size is 50. The cross entropy decreases with the increasing of training steps. At the end, four curves are below 0.10. This shows this neural network model works well on these training datasets in each training batch size.

As we know, the essence of the gradient descent algorithm is to find the most appropriate weight percentage between features to adapt to data. Therefore, when the input of the algorithm is not scaled, large-scale data has a greater impact on the weight and thus it will be difficult to move the weight vector to a good solution. In addition, the feature scaling only attempts to assume that all features have equal opportunities to influence the weight, which more truly reflects information about the input data. Usually it also leads to better accuracy.

In this study, the units and range of the 15 features are different. This is the reason why the scaled data achieved higher accuracy then the raw data.

Table 6 shows the training parameters of the 5-layer fully connected neural network.

Figs. 15 and 16 show the change of the accuracy and the change of the cross entropy over training samples applied to the 5-layer network model with raw data input and batch sizes of 25, 50, 75 and 100. Notice that the entropy value has been multiplied by 100 for ease of data processing. This is done for other models as well.

Figs. 17 and 18 show the change in the accuracy and cross entropy over training samples applied to the 5-layer network model with scaled data input accuracy on batch size 25, 50, 75 and 100.

It can be seen that when the input is raw data, the accuracy and cross entropy for each training batch size vary in a widely range. This means the neural network learns something, but, the dataset or neural network model still needs to be improved.

When the input is scaled data, the accuracy for each training batch size increases sharply with the training steps. At the end of training, it reaches up to 0.96. The cross entropy decreases with the increase of training steps. At the end, four curves are below 0.10. This shows that this neural network model works well on scaled data on each training batch size.

Table 6 Training parameters of the 5-layer fully connected neural network

Parameters name	Value
Training dataset size	9 124
Training batch size	25, 50, 75, 100
Training learning rate base	0.8
Training learning rate decay	0.99
Training regularization rate	0.000 1
Layer 1 weights	mean = 0
	std = 0.01
Layer 1 bias	1
Layer 2 weights	mean = 0
	std = 0.01
Layer 2 bias	1
Layer 3 weights	mean = 0
	std = 0.01
Layer 3 bias	1
Layer 4 weights	mean = 0
	std = 0.01
Layer 4 bias	1
Layer 5 weights	mean = 0
	std = 0.01
Layer 5 bias	1

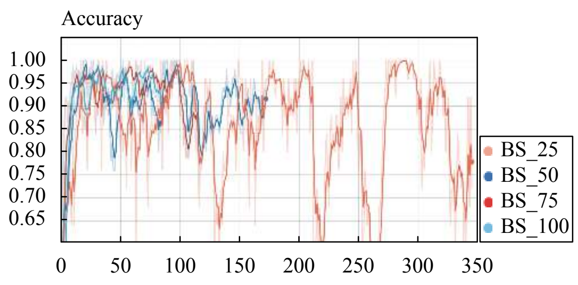


Fig. 15 Accuracy in training the 5-layer fully connected neural network on raw data

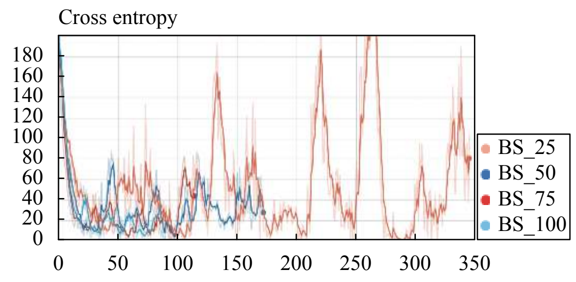


Fig. 16 Cross entropy in training the 5-layer fully connected neural network on raw data

Table 7 shows the training parameters of the 8-layer convolutional neural network.

Figs. 19 and 20 show the change of the accuracy and the change of the cross entropy over training samples applied to the 8-layer convolutional network model with raw data input on batch sizes of 25, 50, 75 and 100.

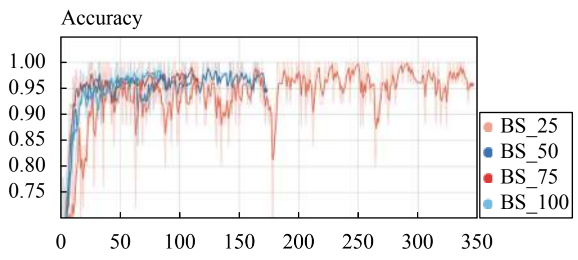


Fig. 17 Accuracy in training the 5-layer fully connected neural network on scaled data

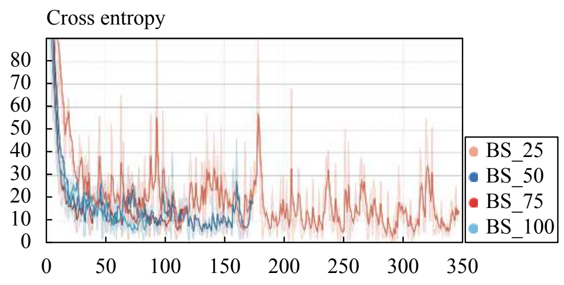


Fig. 18 Cross entropy in training the 5-layer fully connected neural network on scaled data

Figs. 21 and 22 show the change of the accuracy and the change of the cross entropy over training samples applied to the 8-layer convolutional model with raw data input and on batch sizes of 25, 50, 75 and 100.

It can be seen that this neural network gets high accuracy and low cross entropy quickly and smoothly on both the raw dataset and scaled data.

6.2 Testing

After the deep learning network is trained, the static testing and real-time testing are performed to evaluate the neural network in this study.

In the static testing stage, the accuracy performance is measured. As mentioned earlier, the testing dataset has 1 938 samples which are separated from training dataset. This ensures the deep learning network to be evaluated by sufficient unknown information. Like the training process, the four testing datasets are applied to 48 trained neural network models to evaluate its performance.

In total, there are $8 \times 3 = 24$ testing cycles in testing the three different neural networks.

In the real-time testing stage, the data was provided by the real-time input from Leap Motion Controller, deep learning network accuracy and system performance are measured manually.

6.3 Results and analysis

In this study, confusion matrix is used to evaluate the

Table 7 Training parameters of the 8-layer convolutional neural network

Parameters name	Value
Training dataset size	9124
Training batch size	25, 50, 75, 100
Training learning rate base	0.8
Training learning rate decay	0.99
Training regularization rate	0.0001
Layer 1 weights	mean = 0
	std = 0.1
Layer 1 bias	1
Layer 2 weights	mean = 0
	std = 0.1
Layer 2 bias	1
Layer 3 weights	mean = 0
	std = 0.1
Layer 3 bias	1
Layer 4 weights	mean = 0
	std = 0.1
Layer 4 bias	1
Layer 5 weights	mean = 0
	std = 0.1
Layer 5 bias	1
Layer 6 weights	mean = 0
	std = 0.1
Layer 6 bias	1
Layer 7 weights	mean = 0
	std = 0.1
Layer 7 bias	1
Layer 8 weights	mean = 0
	std = 0.1
Layer 8 bias	1

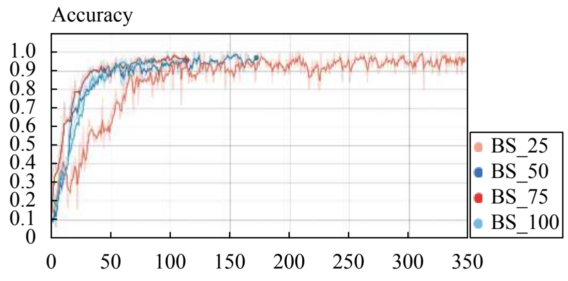


Fig. 19 Accuracy in training the 8-layer convolutional neural network on raw data

testing results. A confusion matrix is a very common technique used for the evaluation of errors in predictions. It consists of a matrix where columns represent the predicted classes and rows represent the actual classes. From confusion matrix, we can calculate the recognition accuracy.

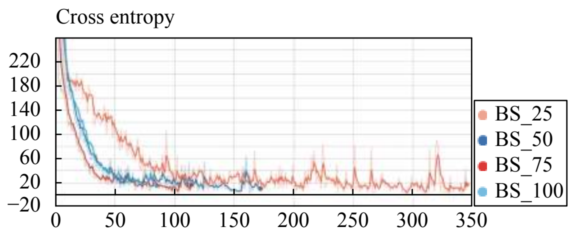


Fig. 20 Cross entropy in training the 8-layer convolutional neural network on raw data

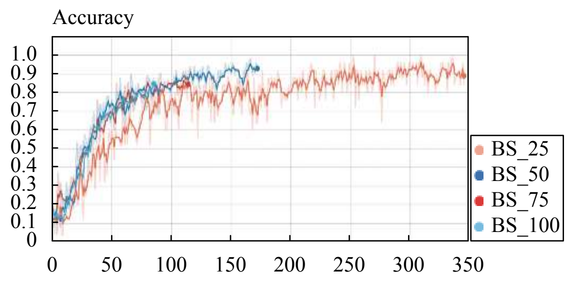


Fig. 21 Accuracy in training the 8-layer convolutional neural network on scaled data

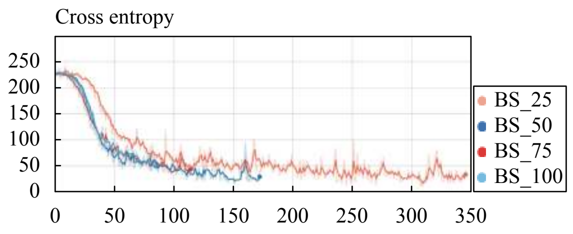


Fig. 22 Cross entropy in training the 8-layer convolutional neural network on scaled data

As an example, Tables 8 and 9 list the confusion matrices for the 2-layer fully connected neural network with raw data input and scaled data input, respectively. The batch size is 25.

Tables 8 and 9 show that the 2-layer fully connected network only get approximately 12.693% average accuracy on the raw data, approximately 97.471% average accuracy on the scaled data.

Tables 10–12 list the testing results of the three networks with raw data input and scaled data input separately, and at all different training chunks.

Table 10 shows that the 2-layer fully connected network only gets approximately 12% average accuracy on the raw dataset, which is unacceptable, and approximately 97% average accuracy on scaled (normalized) dataset. Table 11 shows that the 5-layer fully connected network only gets approximately 90% average accuracy on the raw dataset, which is not acceptable, and approximately 98% average accuracy on scaled (normalized) dataset. Table 12 shows that the 8-layer convolutional network gets approximately 97% average accuracy on the raw dataset.

Table 8 Confusion matrix of the 2-layer fully connected neural network on raw data

2-layer fully connected neural network										
Dataset raw data Training batch size 25 Average accuracy (%) 12.693										
Class	0	1	2	3	4	5	6	7	8	9
0	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	98.598	0.000	0.000	1.402	0.000	0.000	0.000	0.000	0.000	0.000
4	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
9	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9 Confusion matrix of the 2-layer fully connected neural network on scaled data

2-layer fully connected neural network										
Dataset scaled data Training batch size 25 Average accuracy (%) 94.845										
Class	0	1	2	3	4	5	6	7	8	9
0	98.276	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	100.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	0.000	96.154	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	96.667	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	93.182	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	100.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	96.078	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	69.231	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	100.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	100.000

Table 10 Recognition rate of the 2-layer fully connected neural network

Dataset\Batch size	25	50	75	100
Raw data	12.693	11.816	12.538	12.538
Scaled data	97.162	97.471	96.284	96.233

Table 11 Recognition rate of the 5-layer fully connected neural network

Dataset\Batch size	25	50	75	100
Raw data	82.456	91.279	88.905	93.859
Scaled data	98.555	97.316	98.194	98.090

Table 12 Recognition rate of the 8-layer convolutional neural network

Dataset\Batch size	25	50	75	100
Raw data	95.356	97.626	97.626	97.265
Scaled data	93.498	93.446	87.926	83.900

Table 12 also shows that with the CNN model, the recognition rate of the model trained by scaled data is worse than that trained by raw data. This result is kind of opposite to the first two networks. A CNN is quite different from fully connected neural network. It is designed for image recognition. After the raw data is scaled, the CNN can only learn the information from the image of similar size and scale. Obviously, using the images of different scales and sizes during training can achieve better performance than just choosing the "proper" size, because the CNN can learn from more diverse samples.

From the scaled dataset, when the training batch sizes are low (25 and 50), the recognition accuracy is about 93% on average. When the batch sizes are bigger (75 and 100), the accuracy goes down to below 90%.

6.4 Testing of flight controls of UAV

The experiment of real-time drone control with our hand gesture recognition system has been conducted several times in front of the Pozycki Hall, in the Great Lawn, and near the north end of Edison Building of Monmouth University campus. All designed gestures are tested. The result indicates the system can control the flight of the drone as expected. It also shows the testing environment has a big impact on the testing result. Wind speed, GPS signal, and even brightness where the Leap Motion Controller is placed are all critical to the system performance. Nevertheless, the drone control experiment proved our assertion that the hand gesture recognition system can be used to control the real engineering targets.

7 Conclusions

A real time dynamic hand gesture recognition system was designed for UAV flight controls. The core component of the system is a deep learning neural network. For comparison purpose, three neural networks were designed and trained with large gesture sample datasets which were created by this research group. The three networks are a 2-layer fully connected network, a 5-layer fully connected network, and an 8-layer convolutional network. The hand gesture data input device in this study is Leap Motion Controllers. The lab testing results show that the 2-layer and 5-layer networks achieved a 97% recognition rate on normalized input data, while the 8-layer network worked better with raw data input, at a recognition rate of 97%. It is proved by real-time tests that this system is able to control the flight of UAVs reliably.

It is our understanding that this is the first work reported that uses Leap Motion Controllers as input devices in deep leaning network based hand gesture recognition.

Future work of this study includes:

1) The current study only used the Leap Motion Controller as input sensors. The skeleton data is generated from Leap Motion Controller services. The Leap Motion Controller provides the raw image of gesture of each frame. We should consider using the raw image as the input as well, and then this system can be applied to more different devices which provide the raw image.

2) Only 10 simple dynamic gestures are defined in this study. It would be more powerful and flexible if the system can handle more complex static and dynamic gestures.

3) Currently, the training dataset holds about 12 000 samples which are recorded inside a room. It could get more recognition accuracy if more samples from different background are connected.

References

[1] S. Mitra, T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 3, pp. 311–324, 2007. DOI: 10.1109/TSMCC.2007.893280.

[2] V. I. Pavlovic, R. Sharma, T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, 1997. DOI: 10.1109/34.598226.

[3] B. Raj, K. Kalgaonkar, C. Harrison, P. Dietz. Ultrasonic Doppler sensing in HCI. *IEEE Pervasive Computing*, vol. 11, no. 2, pp. 24–29, 2012. DOI: 10.1109/MPRV.2012.17.

[4] C. Oz, M. C. Leu. Human-computer interaction system with artificial neural network using motion tracker and data glove. In *Proceedings of the 1st International Conference on Pattern Recognition and Machine Intelligence*, Springer, Kolkata, India, pp. 280–286, 2005. DOI: 10.1007/11590316_40.

[5] O. Aran. Vision Based Sign Language Recognition: Modeling and Recognizing Isolated Signs with Manual and Non-manual Components, Ph. D. dissertation, Bogazici University, Turkey, 2008.

[6] S. Mitra, T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 3, pp. 311–324, 2007. DOI: 10.1109/TSMCC.2007.893280.

[7] C. Z. Li, V. M. Lubecke, O. Boric-Lubecke, J. Lin. A review on recent advances in Doppler radar sensors for noncontact healthcare monitoring. *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 5, pp. 2046–2060, 2013. DOI: 10.1109/TMTT.2013.2256924.

[8] C. Z. Gu, C. Z. Li, J. Lin, J. Long, J. T. Huangfu, L. X. Ran. Instrument-based noncontact Doppler radar vital sign detection system using heterodyne digital quadrature demodulation architecture. *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 6, pp. 1580–1588, 2010. DOI: 10.1109/TIM.2009.2028208.

[9] T. Starner, A. Pentland. Real-time American Sign Language recognition from video using hidden Markov models. *Motion-based Recognition*, M. Shah, R. Jain, Eds., Dordrecht, Netherlands: Springer, pp. 227–243, 1997. DOI: 10.1007/978-94-015-8935-2_10.

[10] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler. Analysis of the accuracy and robustness of the Leap Motion Controller. *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013. DOI: 10.3390/s130506380.

[11] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, J. Sodnik. An analysis of the precision and reliability of the Leap Motion. *Sensors*, vol. 14, no. 2, pp. 3702–3720, 2014. DOI: 10.3390/s140203702.

[12] D. Y. Huang, W. C. Hu, S. H. Chang. Gabor filter-based hand-pose angle estimation for hand gesture recognition under varying illumination. *Expert Systems with Applications*, vol. 38, no. 5, pp. 6031–6042, 2011. DOI: 10.1016/j.2010.11.016.

[13] G. Rigoll, A. Kosmala, S. Eickeler. High performance real-time gesture recognition using hidden Markov models. In *Proceedings of International Gesture Workshop on Gesture and Sign Language in Human-computer Interaction*, Springer, Berlin, Germany, pp. 69–80, 1998. DOI: 10.1007/BFb0052990.

[14] C. Nolker, H. Ritter. Visual recognition of continuous hand postures. *IEEE Transactions on Neural Networks*, vol. 13, no. 4, pp. 983–994, 2002. DOI: 10.1109/TNN.2002.1021898.

[15] Z. Yang, Y. Li, W. D. Chen, Y. Zheng. Dynamic hand ges-

ture recognition using hidden Markov models. In *Proceedings of the 7th International Conference on Computer Science & Education*, IEEE, Melbourne, Australia, pp. 360–365, 2012. DOI: 10.1109/ICCSE.2012.6295092.

[16] D. J. Li, Y. Y. Li, J. X. Li, Y. Fu. Gesture recognition based on BP neural network improved by chaotic genetic algorithm. *International Journal of Automation and Computing*, vol. 15, no. 3, pp. 267–276, 2018. DOI: 10.1007/s11633-017-1107-6.

[17] O. Koller, S. Zargaran, H. Ney, R. Bowden. Deep sign: Hybrid CNN-HMM for continuous sign language recognition. In *Proceeding of British Machine Vision Conference*, BMVA Press, York, UK, pp. 1–12, 2016.

[18] H. Cooper, E. J. Ong, N. Pugeault, R. Bowden. Sign Language recognition using sub-units. *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2205–2231, 2012.

[19] R. D. Yang, S. Sarkar. Gesture recognition using hidden Markov models from fragmented observations. In *Proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, New York, USA, pp. 766–773, 2006. DOI: 10.1109/CVPR.2006.126.

[20] C. Keskin, A. Erkan, L. Akarun. Real time gestural interface for generic applications. In *Proceedings of the 13th European Signal Processing Conference*, IEEE, Antalya, Turkey, pp. 1–4, 2005.

[21] S. B. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, T. Darrell. Hidden conditional random fields for gesture recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, New York, USA, pp. 1521–1527, 2006. DOI: 10.1109/CVPR.2006.132.

[22] T. Ishihara, N. Otsu. Gesture recognition using auto-regressive coefficients of higher-order local auto-correlation features. In *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, IEEE, Seoul, South Korea, pp. 583–588, 2004. DOI: 10.1109/AFGR.2004.1301596.

[23] A. Ghotkar, P. Vidap, K. Deo. Dynamic hand gesture recognition using hidden Markov model by Microsoft kinect sensor. *International Journal of Computer Applications*, vol. 150, no. 5, pp. 5–9, 2016. DOI: 10.5120/ijca2016911498.

[24] O. Bimber. Continuous DOF gesture recognition: A fuzzy logic approach. In *Proceedings of the 7th International Conference in Central Europe on Computer Graphics and Visualization and Digital Interactive Media*, University of West Bohemia, Plzen, Czech Republic, pp. 24–30, 1999.

[25] A. Ramamoorthy, N. Vaswani, S. Chaudhury, S. Banerjee. Recognition of dynamic hand gestures. *Pattern Recognition*, vol. 36, no. 9, pp. 2069–2081, 2003. DOI: 10.1016/S0031-3203(03)00042-6.

[26] N. H. Dardas, N. D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 11, pp. 3592–3607, 2011. DOI: 10.1109/TIM.2011.2161140.

[27] L. Pigou, A. van den Oord, S. Dieleman, M. van Herreweghe, J. Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *International Journal of Computer Vision*, vol. 126, no. 2–4, pp. 430–439, 2018. DOI: 10.1007/s11263-016-0957-7.

[28] X. J. Chai, Z. P. Liu, F. Yin, Z. Liu, X. L. Chen. Two streams recurrent neural networks for large-scale continuous gesture recognition. In *Proceedings of the 23rd International Conference on Pattern Recognition*, IEEE, Cancun, Mexico, pp. 31–36, 2016. DOI: 10.1109/ICPR.2016.7899603.

[29] R. M. Tan, Y. Cao. Multi-layer contribution propagation analysis for fault diagnosis. *International Journal of Automation and Computing*, vol. 16, no. 1, pp. 40–51, 2019. DOI: 10.1007/s11633-018-1142-y.

[30] N. Neverova, C. Wolf, G. Taylor, F. Nebout. ModDrop: Adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1692–1706, 2016. DOI: 10.1109/TPAMI.2015.2461544.

[31] P. Molchanov, S. Gupta, K. Kim, J. Kautz. Hand gesture recognition with 3D convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, Boston, USA, pp. 1–7, 2015. DOI: 10.1109/CVPRW.2015.7301342.

[32] A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, ACM, Lake Tahoe, USA, pp. 1097–1105, 2012.

[33] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, D. Henderson. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed., San Francisco, USA: Morgan Kaufmann Publishers, pp. 396–404, 1989.

[34] B. Hu, J. C. Wang. Deep learning based hand gesture recognition and UAV flight controls. In *Proceedings of the 24th International Conference on Automation and Computing*, IEEE, Newcastle upon Tyne, UK, 2018. DOI: 10.23919/IConAC.2018.8748953.

[35] G. Zaccone, R. Karim, A. Menshawy. *Deep Learning with TensorFlow*, Birmingham, UK: Packt Publishing, pp. 8–28, 2017.

[36] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Doha, Qata, pp. 1746–1751, 2014.

[37] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014. DOI: 10.1109/TASLP.2014.2339736.

Bin Hu received the B.Sc. degree in mechanical engineering from Xi′an Jiaotong University, China in 2000. He received the two M.Sc. degrees in software engineering from Xi′an Jiaotong University, China in 2005 and Monmouth University, USA in 2018, respectively. From 2006 to 2016, he was an assistant professor at Xi′an University of Posts and Telecommunications, China. Currently, he is an adjunct professor in Department of Computer Science at New Jersey City University, USA. He published about 10 research papers in journals and conferences.

His research interests include software engineering, robotics, and wireless networking.

E-mail: binhu.philip@gmail.com

ORCID iD: 0000-0002-8009-374X

Jiacun Wang received the Ph. D. degree in computer engineering from Nanjing University of Science and Technology (NUST), China in 1991. He is currently a professor of software engineering at Monmouth University, USA. From 2001 to 2004, he was a member of scientific staff with Nortel Networks in Richardson, USA. Prior to joining Nortel, he was a research associate of the School of Computer Science, Florida International University (FIU) at Miami, USA. Prior to joining FIU, he was an associate professor at NUST, China. He authored *Timed Petri Nets: Theory and Application* (Kluwer, 1998), *Real-time Embedded Systems* (Wiley, 2018) and *Formal Methods in Computer Science* (CRC Press, 2019), edited *Handbook of Finite Stat Based Models and Applications* (CRC, 2012), and published about 90 research papers in journals and conferences. He was an Associate Editor of *IEEE Transactions on Systems, Man and Cybernetics*, Part C. He has served as general chair, program chair, and special sessions chair or program committee member for many international conferences. He is a senior member of IEEE.

His research interests include software engineering, discrete event systems, formal methods, wireless networking, and real-time distributed systems.

E-mail: jwang@monmouth.edu (Corresponding author)
ORCID iD: 0000-0003-4176-3947