



Article

Dynamic Hand Gesture Recognition Based on a Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network

Linchu Yang ¹, Ji'an Chen ^{1,*}  and Weihang Zhu ² 

¹ Department of Mechanical Engineering; Jiangsu University of Science and Technology, Zhenjiang 212003, China; yanglinchu@just.edu.cn

² Department of Engineering Technology; University of Houston, Houston, TX 77204, USA; wzhu21@central.uh.edu

* Correspondence: chen_ji_an@stu.just.edu.cn

Received: 28 February 2020; Accepted: 6 April 2020; Published: 8 April 2020



Abstract: Dynamic hand gesture recognition is one of the most significant tools for human–computer interaction. In order to improve the accuracy of the dynamic hand gesture recognition, in this paper, a two-layer Bidirectional Recurrent Neural Network for the recognition of dynamic hand gestures from a Leap Motion Controller (LMC) is proposed. In addition, based on LMC, an efficient way to capture the dynamic hand gestures is identified. Dynamic hand gestures are represented by sets of feature vectors from the LMC. The proposed system has been tested on the American Sign Language (ASL) datasets with 360 samples and 480 samples, and the Handicraft-Gesture dataset, respectively. On the ASL dataset with 360 samples, the system achieves accuracies of 100% and 96.3% on the training and testing sets. On the ASL dataset with 480 samples, the system achieves accuracies of 100% and 95.2%. On the Handicraft-Gesture dataset, the system achieves accuracies of 100% and 96.7%. In addition, 5-fold, 10-fold, and Leave-One-Out cross-validation are performed on these datasets. The accuracies are 93.33%, 94.1%, and 98.33% (360 samples), 93.75%, 93.5%, and 98.13% (480 samples), and 88.66%, 90%, and 92% on ASL and Handicraft-Gesture datasets, respectively. The developed system demonstrates similar or better performance compared to other approaches in the literature.

Keywords: hand gesture recognition; leap motion controller (LMC); recurrent neural network (RNN)

1. Introduction

The main purpose of human–computer interaction is to allow users to freely control the device with some simple operations [1]. The human–computer interaction techniques include face recognition, language recognition, text recognition, and so on. As one of the important and powerful interaction methods, dynamic hand gesture recognition has attracted wide attention and been used in various fields, such as the video game industry, food industry, and machinery industry [2–4].

Dynamic gesture recognition can be used in many applications, such as virtual reality [5,6], the remote operation of robot [7,8], video games [9,10], sign language interpretation [11,12], and so on. In general, dynamic hand gesture recognition can be mainly divided into vision-based gesture recognition and wearable device-based gesture recognition. In the past, many researches on dynamic gesture recognition basically used a monocular camera [13] to capture images of dynamic gestures, segmented images, and extracted the gesture model from a series of frames. One disadvantage of this method is that a large amount of computation is required to segment the hand information. The other method adopted a data glove [14] to collect data that focuses on the coordinates of palms and fingertips and the bending degrees of finger joints. It can intuitively obtain hand information without a lot of calculation. However, a user must wear gloves, and people using the same device may get

certain health issues. The recent years have seen some novel sensors, such as Leap Motion Controllers (LMC) [15] and Microsoft Kinect [16]. These sensors do not need to be worn and a user only need to put the hand in the appropriate acquisition space above the sensor to allow the three-dimensional information of the hand to be captured in its three-dimensional coordinate system. This is a significant contribution to the dynamic gesture recognition.

Currently, the majority of studies on the dynamic gesture recognition are based on the Support Vector Machine, Hidden Markov Model, Common Neural Network, Rule-based Modeling techniques and so on.

In the previous works, Support Vector Machine, Dynamic Time Warping, and Hidden Markov Model were commonly adopted. For example, Xu et al. displayed a method of feature extraction for dynamic hand gesture recognition based on the LMC, which ensured that the lengths of different dynamic hand gesture samples could be the same [17]. Next, a classifier based on the Support Vector Machine was used to recognize dynamic gestures. At last, the recognition accuracies for Arabic numbers (0-9), based on two different ranging criterions, were 90.75% and 91.375%, respectively. Karthick et al. presented a system which can transform Indian Sign Language to text [18]. In addition, the hand gestures were also collected by the LMC. Afterwards, they translated dynamic hand gestures into text by combining an Intelligent Sensing algorithm with Dynamic Time Warping. Moreover, Schramm obtained many patterns by tracking hand position through an RGB-D camera [19]. After that, researchers used a probabilistic model based on Dynamic Time Warping to recognize and evaluate patterns. The LMC had also been used in [20] and a method was proposed for segmenting and recognizing texts that are drawn by fingers in the 3D coordinate system. By using a heuristic analysis of stroke length between two consecutive words, the successive text was segmented into words. Then, the Hidden Markov Model was used to identify each segmented word. As a result, an accuracy of 81.25% was obtained on word recognition.

In some other works, different technologies were extended. Lu et al. proposed a special feature vector which represents the gesture and presented a Hidden Conditional Neural Field that incorporates the gate function of neural networks to recognize dynamic hand gestures with an LMC [21]. With hand information, the vector of features was collected and input into the proposed model to recognize dynamic gestures. Finally, the method was tested with two dynamic hand gesture datasets, and the accuracy was 89.5% on a subset of American Sign Language and 95% on the Handicraft-Gesture dataset. Zhang et al. presented a gesture recognition system that combines the modeling ability of the Hidden Markov Model on time-based sequences with the interpretability of the Classification and Regression Tree to achieve fast classification and regression [22]. Features were divided into four types, namely finger shape, palm ball radius, palm normal vector, and palm displacement vector. In the first part, a Hidden Markov Model was built for each type. Next, the output of the first part was regarded as the input of the Classification and Regression Tree model in the second part. The output of the second part was the recognition result. Finally, the experiments showed that the accuracy was higher than that of traditional single channel Hidden Markov Model.

In recent years, more and more researchers have used the LMC and some advanced algorithms to recognize dynamic gestures. Avola et al. presented a deep Recurrent Neural Network model to identify dynamic hand gestures from American Sign Language [23]. The proposed method was tested by a challenging dataset which consists of 12 dynamic hand gestures from the American Sign Language and the final accuracy was 96%. Zeng et al. developed a new approach to recognize dynamic hand gestures for Arabic numbers (0–9) and capital English alphabets (A–Z) [24]. In the first step, dynamic hand gesture features were derived from the LMC. Secondly, the authors used the radial basis function neural network to approximate the hand motion dynamics underlying the motion patterns of different hand gestures. The difference between motion dynamics was used to distinguish between different hand gestures. Finally, by using the leave-one-person-out, 2-fold, and 10-fold cross-validation, the recognition accuracy results for 0–9 are 94.2%, 95.1%, and 90.2%, respectively, and for A–Z are 89.2%, 92.9%, and 86.4%, respectively. Mittal et al. proposed a recognition system for continuous

sign language by using LMC [25]. In the system, the authors presented a modified Long Short-Term Memory model which has three input gates and an output gate, adding a 2D-Convolutional Neural Network to the input gate that receives feature inputs. At last, the system had been tested with Indian Sign Language and got the average accuracies of 89.5% and 72.3% on isolated sign words and signed sentences, respectively. The Arabic Sign Language recognition system with two LMCs was shown in [26]. Deriche et al. adopted the Linear Discriminant Analysis approach and a Bayesian approach that contains the Gaussian Mixture Model. As a result, an accuracy of nearly 92% was obtained.

In order to improve the performance of the dynamic hand gesture recognition on American Sign Language (ASL) and Handicraft Gesture datasets from several works, we present a gesture recognition system in this paper, which consists of the LMC and the two-layer Bidirectional Recurrent Neural Network (BRNN). In the first stage, the proposed algorithm accurately determines the start and end of dynamic hand gestures by calculating the changes in hand rotation angle and palm speed between two adjacent frames, which can ensure the validity of features. Then, to obtain a better model, the features of a single finger and adjacent fingers are introduced into the input vector of the model. In the next stage, we compare the effects of changes in the hyper-parameter on the accuracy of the classifier and improve the performance of the model. In the third stage, the validation and comparison are performed on the proposed system. Here, the challenges are how to collect real-time dynamic hand gestures efficiently and how to search for a better model with hyper-parameters. The first problem is solved, to a large extent, by using an algorithm to discriminate hand movement, and then the second problem is tackled by using a two-layer BRNN. In summary, the contributions of the proposed system are shown:

Based on the LMC, we propose a method to accurately determine when to start and end collecting dynamic gestures through its C++ function library.

The selection of features, based on fingers and palm, are highly discriminative for recognizing some gestures from the ASL and Handicraft-Gesture datasets.

For the first time, we combine a two-layer BRNN with the special features extracted by the LMC in the field of the dynamic gesture recognition. Moreover, the accuracy is higher than that reported in similar works found in the literature.

2. Materials and Methods

In this section, we present the architecture of our gesture recognition system. The systematic framework includes four parts, which are feature extraction, data collection and processing, two-layer BRNN, and model details. The architecture of gesture recognition system is shown in Figure 1, where the LMC is used to acquire the dynamic hand sequences, and data are processed by specific methods and sent into the two-layer BRNN model. Finally, the BRNN model outputs the recognition result.

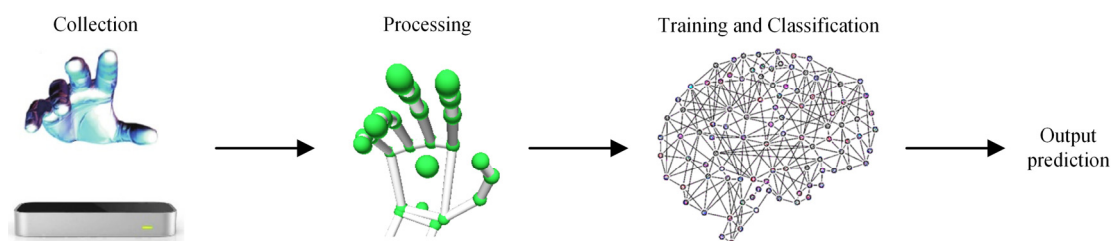


Figure 1. The architecture of the recognition system.

2.1. Feature Extraction

The LMC can detect and track the human hand and wrist which appears 25 mm to 600 mm above the device plane, returning information on coordinate, velocity, acceleration, and others of five fingers, palm, and wrist. Each dynamic motion can be taken as a sequence of consecutive frames. Each frame

can be considered as the combination of different hand features. In this part, we see a frame as a vector which consists of following features:

Fingertip distance

$$D_i = \|\vec{OF}_i - \vec{OC}\| (i = 1, 2, 3, 4, 5) \quad (1)$$

where, D_i is the Euclidean distance between a fingertip coordinate and the palm coordinate, O is the origin of the three-dimensional coordinate system of Leap Motion, F_i is the fingertip coordinate in the three-dimensional coordinate system of Leap Motion, C is the palm coordinate in the three-dimensional coordinate system of Leap Motion.

Fingertip angle

$$\theta_i = \angle(\vec{OF}_i^p - \vec{OC}, \vec{OF}_i - \vec{OC}) (i = 1, 2, 3, 4, 5) \quad (2)$$

where, θ_i is the angle between vector \vec{CF}_i^p and vector \vec{CF}_i , F_i^p is the projection of F_i in the palm plane.

Fingertip height

$$H_i = \text{sgn}(\vec{F}_i^p \cdot \vec{F}_i \cdot \vec{N}) * \|\vec{F}_i^p\| (i = 1, 2, 3, 4, 5) \quad (3)$$

where, H_i is the vertical distance from point F_i to palm plane, \vec{N} is the palm normal (Figure 2), \cdot is the dot product, $*$ is the multiplication sign.

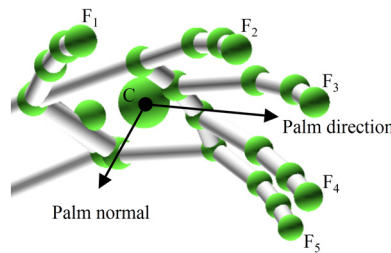


Figure 2. The hand model in the LMC.

The distance of adjacent fingertips

$$DD_i = \|\vec{F}_i \vec{F}_{i+1}\| (i = 1, 2, 3, 4) \quad (4)$$

The angle of adjacent fingertips

$$\Theta_i = \angle(\vec{CF}_i, \vec{CF}_{i+1}) (i = 1, 2, 3, 4) \quad (5)$$

where, Θ_i is the angle between vector \vec{CF}_i and vector \vec{CF}_{i+1} .

The coordinate of the palm

$$(x_{\text{palm}}, y_{\text{palm}}, z_{\text{palm}}) \quad (6)$$

The above features are derived from the work [21,27,28] and they are extended and modified. Note that they are independent of each other and the final feature vector at time t is:

$$X_t = \{D_1, \dots, D_5, \theta_1, \dots, \theta_5, H_1, \dots, H_5, DD_1, \dots, DD_4, \Theta_1, \dots, \Theta_4, x, y, z\} \quad (7)$$

This proposed vector not only solves the mislabeling problem caused by making dynamic hand gestures in different positions but also helps to recognize the difference which exists between adjacent fingers when people show different dynamic hand gestures.

A full dynamic gesture vector can be represented by:

$$X = \{X_1, X_2, \dots, X_j, \dots, X_T\} \quad (8)$$

where, T is the number of frames required to detect a full dynamic gesture and varies according to different dynamic gestures.

2.2. Data Collection and Processing

Dynamic gesture recognition relies on real-time and accurate gesture tracking. LMC uses binocular RGB high-definition cameras to improve gesture positioning accuracy and reduce the problems caused by occlusions between fingers. The infrared camera is used to filter images, which greatly reduces the impact of the background environment. Finally, a convolutional neural network is used to perform multi-layer convolution filtering on the image to extract feature data and provide it to the user. In terms of real-time and accuracy, LMC can provide gesture data more stably and accurately, providing a stable data guarantee for applications in some fields.

However, a challenge in dynamic hand gesture data collection is how to determine the start and end of a dynamic hand gesture. When LMC performs gesture acquisition, it obtains time-based dynamic sequences. Therefore, during the gesture collection, it is necessary to determine when the gesture execution starts and stops according to the threshold. This work uses the palm rotation threshold in a three-dimensional coordinate system and finger speed threshold to determine the start and end points of the dynamic hand gesture. The rotation of the palm needs to be obtained through comparison and calculation of the current frame and the historical frames, and the change in finger speed can be obtained through the library function that comes with LMC. The Algorithm 1 is shown below:

Algorithm 1

```

If(Hand is not empty)
{
  If (
    sqrt (pow (frame. rotation Angle (last frame, x), 2)+
      pow (frame. rotation Angle (last frame, y), 2)+
      pow (frame. rotation Angle (last frame, z), 2))
      >threshold1
    or
    fingertip. velocity > threshold2
  )
  Start collecting dynamic hand gestures
Else
  Stop collecting dynamic hand gestures
}
Else
  Stop collecting dynamic hand gestures

```

LMC allows access to a wide variety of features contained in the original dataset. After extraction and calculation, and obtaining specific features, the high-dimensional hand gesture sequences will make the model too computationally intensive in learning. In addition, every gesture in the acquisition process will have problems such as inconsistent shapes and inconsistent range sizes. Therefore, data need to be pre-processed, so that it is standardized to eliminate noise interference and reduce model learning costs. The same attribute of all samples is normalized separately, and all the samples become a series of data with a mean of 0 and a variance of 1.

2.3. Two-Layer Bidirectional Recurrent Neural Network

2.3.1. The Basic Long Short-Term Memory Unit

The Long Short-Term Memory unit, as shown in Figure 3, is seen as the memory block and consists of three multiplicative gates which are input gate (update gate), output gate, and forget gate in the network. Additionally, the memory block also includes memory cells that are self-connected and save the temporal information of the network. The input gate is responsible for processing the input of the information. The output gate generates the flow of the output of cell activations to the rest of the units. The forget gate adaptively forgets some information in the memory cell. At time t , the formulas in the unit are defined as follow:

$$\tilde{c}^t = \tanh(W_c \cdot [a^{t-1}, x^t] + b_c) = \tanh(W_{ca} \cdot a^{t-1} + W_{cx} \cdot x^t + b_c) \quad (9)$$

$$\Gamma_u = \sigma(W_u \cdot [a^{t-1}, x^t, c^{t-1}] + b_u) = \sigma(W_{ua} \cdot a^{t-1} + W_{ux} \cdot x^t + W_{uc} \cdot c^{t-1} + b_u) \quad (10)$$

$$\Gamma_f = \sigma(W_f \cdot [a^{t-1}, x^t, c^{t-1}] + b_f) = \sigma(W_{fa} \cdot a^{t-1} + W_{fx} \cdot x^t + W_{fc} \cdot c^{t-1} + b_f) \quad (11)$$

$$c^t = \Gamma_u \odot \tilde{c}^t + \Gamma_f \odot c^{t-1} \quad (12)$$

$$\Gamma_o = \sigma(W_o \cdot [a^{t-1}, x^t, c^t] + b_o) = \sigma(W_{oa} \cdot a^{t-1} + W_{ox} \cdot x^t + W_{oc} \cdot c^t + b_o) \quad (13)$$

$$a^t = \Gamma_o \odot \tanh c^t \quad (14)$$

where, Γ_u , Γ_o , Γ_f , and \tilde{c}^t are the input gate, the output gate, the forget gate, and the new memory cell, respectively. The symbols b_u , b_f , b_o and b_c are bias vectors, respectively. Then a^t , c^t , a^{t-1} and c^{t-1} are the output of current block, final memory cell, output of previous block and previous memory cell. These vectors have the same length. W_u , W_o , W_f and W_c are the weights of the input gate, the output gate, the forget gate, the new memory cell, and the prediction, respectively. \tanh is the hyperbolic tangent function. g is the softmax function. σ is the logistic sigmoid activation function. \odot is the element-wise product of the vectors.

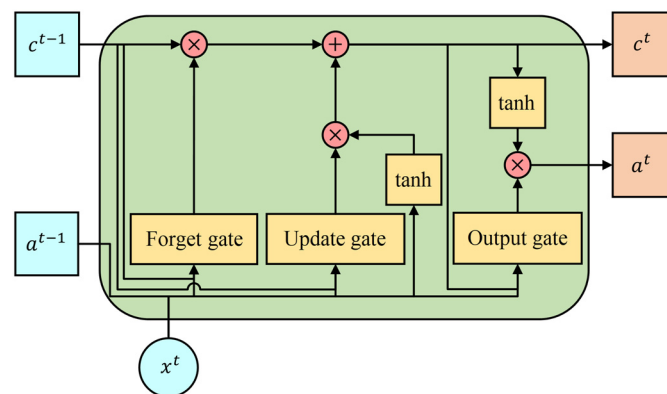


Figure 3. A basic Long Short-Term Memory unit with peephole.

2.3.2. Bidirectional Recurrent Neural Network

The common Recurrent Neural Network provides an extremely useful method to handle time-based sequences which shows correlations between closely linked data elements in the sequence. Figure 4 shows a basic Recurrent Neural Network architecture with three units. Meanwhile, the previous Long Short-Term Memory unit provides its memory cell and output for the current unit as time goes on. In this figure, the input vector x which contains T frames is input into the network frame by frame. Compared with some other time-based models like Multilayer perceptron and time delay

neural networks [29], this structure utilizes most of the available input information, from the start frame to current frame, to output vector y without the limitation of using a fixed-format input.

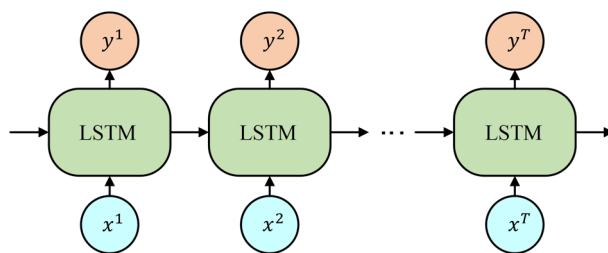


Figure 4. Example of connection between single-layer LSTM network.

However, in the one-way Recurrent Neural Network, the current unit can only output the outcome based on the information of previous units. Especially, in some problems, the output of the current unit is not only related to previous units, but also related to the future units. In this case, it is possible to use two separate recurrent neural networks and then somehow merge outputs. In the single layer Bidirectional Recurrent Neural Network, one Recurrent Neural Network goes in a forward direction. On the contrary, another one goes in backward direction. At each time point, the input is provided to two independent Long Short-Term Memory units in opposite directions and they combine their outcomes based on the hidden state. In our work, we adopt this structure to build a two-layer BRNN and combine outcomes of the final Long Short-Term Memory unit of both networks (Figure 5). There are many other merging procedures to combine outcomes and it is generally not clear how to merge network outcomes in an optimal way since different networks trained on the same data can no longer be regarded as independent [29]. The formulas of the basic BRNN unit at time t are given by the following equations:

$$\vec{c}^t = \tanh(\vec{W}_{ca} \cdot \vec{a}^{t-1} + \vec{W}_{cx} \cdot x^t + \vec{b}_c) \tag{15}$$

$$\overleftarrow{c}^t = \tanh(\overleftarrow{W}_{ca} \cdot \overleftarrow{a}^{t-1} + \overleftarrow{W}_{cx} \cdot x^t + \overleftarrow{b}_c) \tag{16}$$

$$\vec{\Gamma}_u = \sigma(\vec{W}_{ua} \cdot \vec{a}^{t-1} + \vec{W}_{ux} \cdot x^t + \vec{W}_{uc} \cdot \vec{c}^{t-1} + \vec{b}_u) \tag{17}$$

$$\overleftarrow{\Gamma}_u = \sigma(\overleftarrow{W}_{ua} \cdot \overleftarrow{a}^{t-1} + \overleftarrow{W}_{ux} \cdot x^t + \overleftarrow{W}_{uc} \cdot \overleftarrow{c}^{t-1} + \overleftarrow{b}_u) \tag{18}$$

$$\vec{\Gamma}_f = \sigma(\vec{W}_{fa} \cdot \vec{a}^{t-1} + \vec{W}_{fx} \cdot x^t + \vec{W}_{fc} \cdot \vec{c}^{t-1} + \vec{b}_f) \tag{19}$$

$$\overleftarrow{\Gamma}_f = \sigma(\overleftarrow{W}_{fa} \cdot \overleftarrow{a}^{t-1} + \overleftarrow{W}_{fx} \cdot x^t + \overleftarrow{W}_{fc} \cdot \overleftarrow{c}^{t-1} + \overleftarrow{b}_f) \tag{20}$$

$$\vec{c}^t = \vec{\Gamma}_u \odot \vec{c}^t + \vec{\Gamma}_f \odot \vec{c}^{t-1} \tag{21}$$

$$\overleftarrow{c}^t = \overleftarrow{\Gamma}_u \odot \overleftarrow{c}^t + \overleftarrow{\Gamma}_f \odot \overleftarrow{c}^{t-1} \tag{22}$$

$$\vec{\Gamma}_o = \sigma(\vec{W}_{oa} \cdot \vec{a}^{t-1} + \vec{W}_{ox} \cdot x^t + \vec{W}_{oc} \cdot \vec{c}^t + \vec{b}_o) \tag{23}$$

$$\overleftarrow{\Gamma}_o = \sigma(\overleftarrow{W}_{oa} \cdot \overleftarrow{a}^{t-1} + \overleftarrow{W}_{ox} \cdot x^t + \overleftarrow{W}_{oc} \cdot \overleftarrow{c}^t + \overleftarrow{b}_o) \tag{24}$$

$$\vec{a}^t = \vec{\Gamma}_o \odot \tanh c^t \tag{25}$$

$$\overleftarrow{a}^t = \overleftarrow{\Gamma}_o \odot \tanh c^t \tag{26}$$

the output of the two-layer BRNN is defined as follows:

$$\hat{y} = \text{softmax}\left(\vec{W}_y \cdot \sum_{t=1}^T \vec{a}^t + \overleftarrow{W}_y \cdot \sum_{t=1}^T \overleftarrow{a}^t + b_y\right) \quad (27)$$

where, \rightarrow represents the forward Recurrent Neural Network and \leftarrow represents the backward Recurrent Neural Network. The scales of these four RNNs are the same.

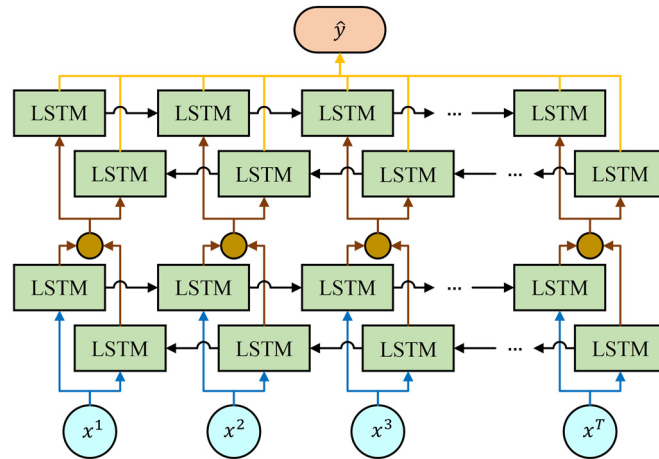


Figure 5. The two-layer Bidirectional Recurrent Neural Network. The outputs of the two corresponding LSTM units are added up in the first layer and sent to the second layer.

2.4. Model Details

2.4.1. Loss Function

The loss function is defined as below:

$$L^t(\hat{y}^t, y^t) = -y^t \cdot \log(\hat{y}^t) \quad (28)$$

$$L(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^T L^t(\hat{y}^t, y^t) \quad (29)$$

where, \hat{y} is prediction of the sample X , y is the label of the sample X , and y^t is the label of the X_t in the sample X . In addition, m is the number of samples and T is the length of one sample based on the time sequence. Further, \cdot is the inner product. This formulation is obtained by referring to and modifying the cross-entropy proposed in [30].

2.4.2. Learning Rate

Learning rate is the most important hyper-parameter in all kinds of neural networks. Consequently, Cyclical Learning Rate method was adopted in this work, which can eliminate the need to vary the learning rate yet achieve near optimal classification accuracy. Instead of monotonically decreasing the learning rate, CLR lets the learning rate vary cyclically between reasonable boundary values which are achieved through, linearly, increasing the learning rate for a few epochs. The corresponding codes are the given as:

$$s = \frac{\text{Iterations}}{\text{Batchsize}} \quad (30)$$

$$c = \left[1 + \frac{en}{2 \cdot s}\right] \quad (31)$$

$$p = \left| \frac{en}{s} - 2 \cdot c + 1 \right| \quad (32)$$

$$LR = LR_{min} + (LR_{max} - LR_{min}) \cdot \max(0, 1 - p) \quad (33)$$

where $[x]$ is to take the largest integer less than x , $|x|$ is to take the absolute value of x , and $\max(x, y)$ is to take the larger of x and y . The en is the sequence number of epochs. More details can be found in [31].

3. Experimental Results and Discussion

This section shows a series of dynamic hand gesture recognition experiments. The systematic framework includes six parts: dynamic gesture datasets, selection of the optimal dropout rate on two datasets, experiment results on ASL dataset, experiment results on Handicraft-Gesture dataset, k-fold and Leave-One-Out cross-validation, and comparisons between our results with the works found in the literature.

3.1. Dynamic Gesture Datasets

Here, we created two new datasets composed of 12 hand gestures and 10 hand gestures, respectively. They are called ASL Dataset (Figure 6) and Handicraft-Gesture Dataset [21] (Figure 7). In particular, the ASL Dataset is picked from the American Sign Language and the other dataset is chosen from the paper [21].

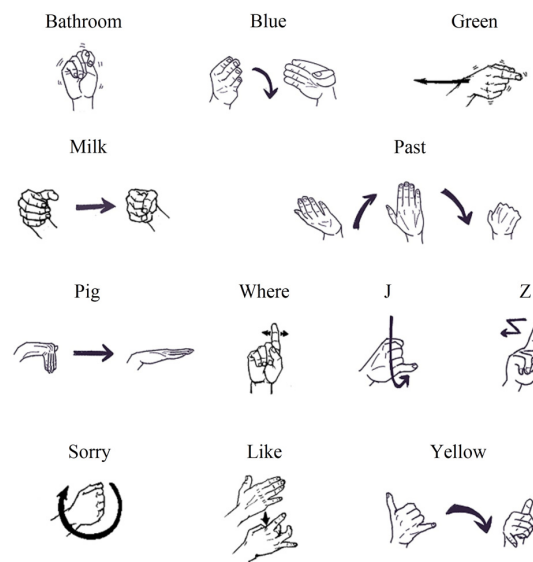


Figure 6. The illustration of twelve gestures of the ASL dataset.

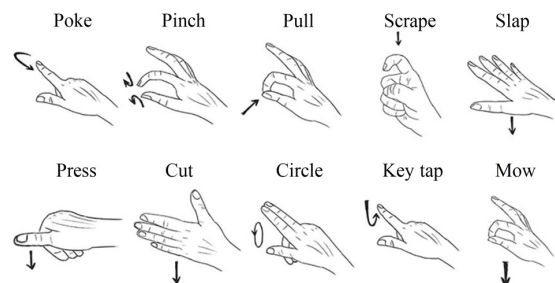


Figure 7. The illustration of ten gestures of the Handicraft-Gesture dataset.

3.1.1. ASL Dataset

Currently, most of datasets are based on images. In order to test the performance of the model we proposed, we built this dataset with an LMC. This dataset contains a series of gestures from ASL. The 12 gestures are bathroom, blue, green, j, like, milk, past, pig, sorry, where, yellow, and z. The arrows are the motion trajectories of the hand or the fingers.

This work is mainly focused on single-hand dynamic gestures. In order to compare with other works, like [21,23] which has the dataset containing bathroom, blue, finish, green, hungry, milk, past, pig, store, where, j, and z, we replaced finish, hungry, and store with sorry, like, and yellow which are nearly equivalent to them in terms of difficulty.

3.1.2. Handicraft-Gesture Dataset

In order to evaluate the proposed model with more different dynamic hand gestures, we introduced the Handicraft-Gesture dataset. This dataset includes 10 gestures, namely poke, scrape, circle, pull, pinch, slap, cut, key type, press, and mow.

All gestures are dynamic and based on a single hand, coming from four different people aged 25. In order to compare with other literatures, there are two types of ASL datasets which contain 360 and 480 samples, respectively. In particular, the dataset with 480 samples is from [23] which has a dataset composed of 720 static hand sequences and 480 dynamic hand sequences. In the one that contains 360 samples, each type of gesture has 30 samples that are performed by four different persons. According to [21], 70% of the data are used as the training set and 30% are used as the testing set. In the other one, each type of gesture has 40 samples that are performed by four different persons. According to [23], 65% of the data are used as the training set and 35% are used as the testing set. The Handicraft-Gesture dataset is composed of 300 samples and each type of gesture has 30 samples that are performed by four different persons. According to [21], 70% of the data are used as the training set and 30% are used as the testing set.

3.2. Selection of the Optimal Dropout Rate on Two Dataset

Although BRNN has shown excellent ability to capture deep information between sequences, BRNN architectures are prone to overfit [32,33]. In order to avoid the problem of overfitting, one of the most useful techniques in neural networks, namely dropout regularization, was taken. Several experiments on the selection of the dropout rate were conducted.

The Figure 8a shows that the best dropout rate for the ASL dataset is between 0.3 and 0.4, and the Figure 8b indicates that the best dropout rate for the Handicraft-Gesture dataset is between 0.4 and 0.5. However, as the number of iterations increases, the dropout rate still needs to be tuned properly in these two ranges.

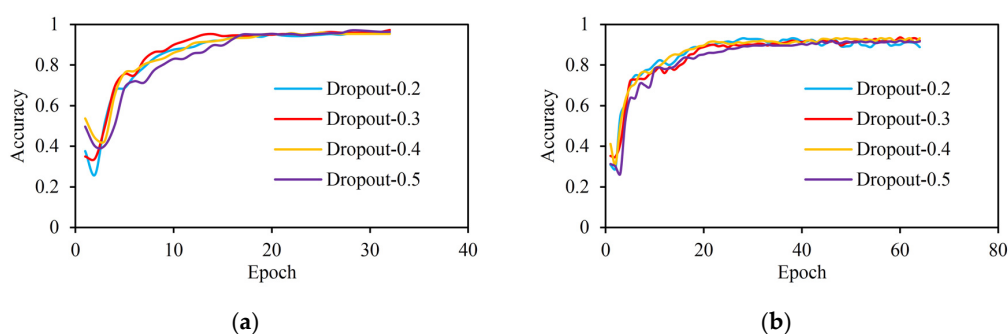


Figure 8. Accuracies on datasets by varying the dropout rate. (a) Accuracies on the ASL dataset; (b) Accuracies on the Handicraft-Gesture dataset.

3.3. Experiment Results on ASL Dataset

Using an LMC and a computer with an Intel i5 3.2GHz CPU and 8GB RAM, experiments were performed to estimate the performance of the proposed two-layer BRNN model. The model was trained using cross-entropy loss function, Adam gradient descent algorithm [34], and varying learning rate. The model was implemented by using a framework in C++, written by ourselves.

Figure 9a–f shows changes in the recognition accuracy of the model on the training and testing sets as the number of iterations increases and loss curves which represent the sum of the errors

provided for each training and test sample. On the ASL dataset with 360 samples, after 630 epochs (18,900 iterations), the curve of the train accuracy converges to 100% and the curve of the test accuracy converges to 96.2963%, respectively. On the ASL dataset with 480 samples, after 1170 epochs (70,200 iterations), the curve of the train accuracy converges to 100% and the curve of the test accuracy converges to 95.238%, respectively. In Figure 9c,d, a sudden and unexpected leap of value happens over the epoch of 340 for accuracies and losses on both training and testing sets. To explore the cause we did another three types of experiments, each performed five times. The first experiment used the CLR but did not use the dropout, the second experiment used the dropout but did not use the CLR, and the third experiment did not use any of them. The result shows that the leap happens randomly when the model uses the dropout and CLR. It seems to be a random and transient instability because this leap only happens when the learning rate varies during the first period (0–400 epoch), and after this the model will consistently converge with smaller learning rate. This leap happens infrequently, and it does not appear in experiments on the other two datasets.

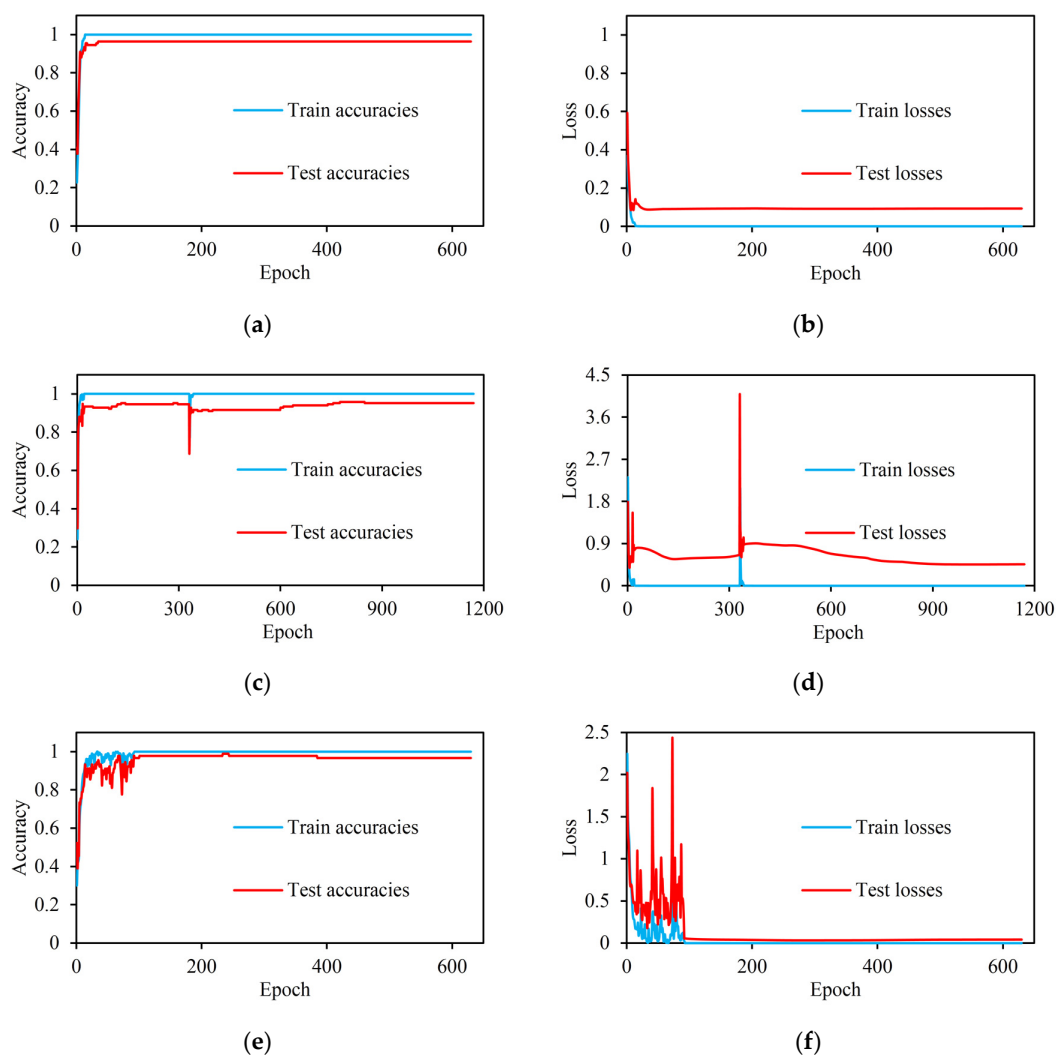


Figure 9. Curves for the accuracies and losses of the training and testing sets. (a) The accuracies of the ASL dataset with 360 samples; (b) The losses of the ASL dataset with 360 samples; (c) The accuracies of the ASL dataset with 480 samples; (d) The losses of the ASL dataset with 480 samples; (e) The accuracies of the Handicraft-Gesture dataset with 300 samples; (f) The losses of Handicraft-Gesture dataset with 300 samples.

In order to better evaluate the proposed approach, three significant metrics were taken, namely Precision, Recall, and F1-score. As a standard, these indexes can be used to measure the performance of the model [35]. The results are presented in Table 1.

Table 1. The performance of the proposed approach on the ASL dataset.

Type	Accuracy	Precision	Recall	F1-Score
360 samples	96.2963%	96.8182%	96.2963%	96.2674%
480 samples	95.238%	95.546%	95.238%	95.274%

In addition, heat-maps of the confusion matrix were also computed and drawn (Figure 10). Each column of the confusion matrix represents the predictive value while each row represents the real value. In the confusion matrix, each diagonal element represents the prediction accuracy of the corresponding row identifier. The elements below or above the diagonal are misclassified. As depicted in Figure 10, for each gesture, there is a low accuracy for gesture ‘green’ (78%) on the dataset with 360 samples. However, the accuracy of each gesture is above 93% on the dataset with 480 samples, which is a good performance.

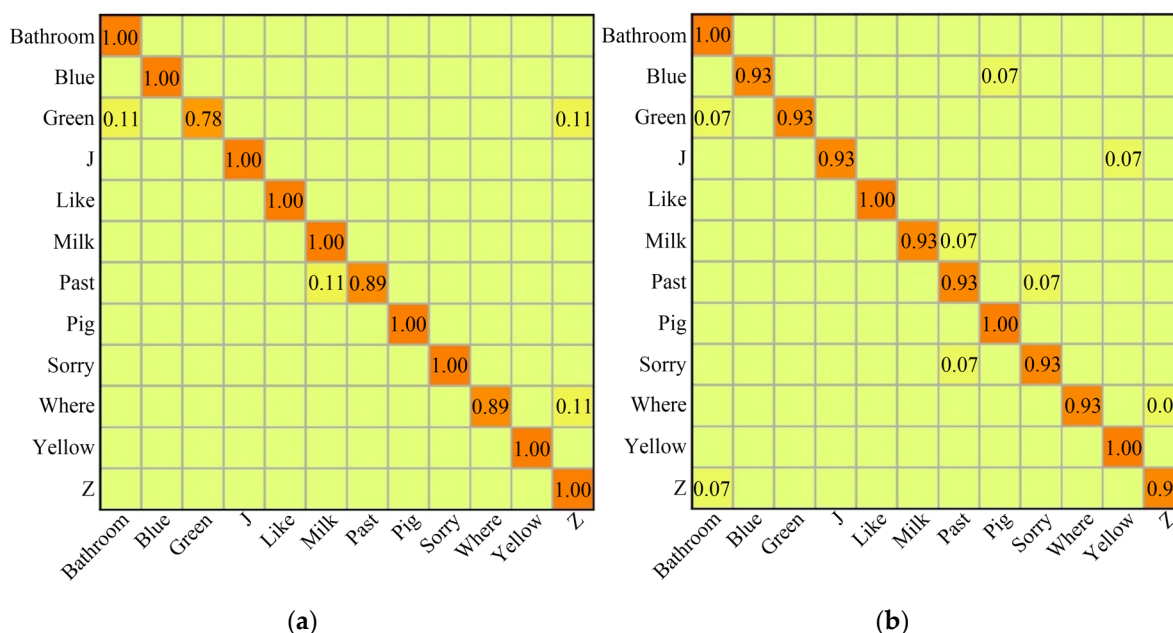


Figure 10. Heat-maps of the confusion matrix for the recognition of the ASL dataset. (a) The heat-map of the ASL dataset with 360 samples; (b) The heat-map of the ASL dataset with 480 samples.

3.4. Experiment Results on Handicraft-Gesture Dataset

Similar experiments and evaluations were executed on the Handicraft-Gesture dataset. As shown in Figure 9, after 630 epochs (15,750 iterations), the curve of the train accuracy converges to about 100% and the curve of the test accuracy converges to about 96.6667%, respectively. In addition, Table 2 shows the F1-score of 96.6563% and Figure 11 describes the heat-map of the confusion matrix for recognizing hand gestures on this dataset.

Table 2. The performance of the proposed approach on the Handicraft-Gesture dataset.

Accuracy	Precision	Recall	F1-Score
96.6667%	96.75%	96.6667%	96.6563%

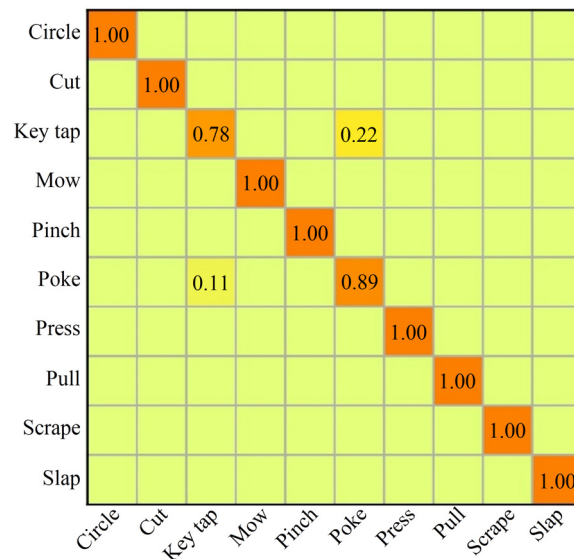


Figure 11. The heat-map of the confusion matrix for the recognition of the Handicraft-Gesture dataset.

Except for low accuracies for gesture ‘key tap’ (78%) and ‘poke’ (89%), all other accuracies are 100%. It is clear that, at some moments, key tap and poke have similar hand shapes and the same motion trajectories in different directions. Different people make this gesture with different force and directions. For this reason, the key tap and poke are misclassified in some cases.

3.5. K-Fold and Leave-One-Out Cross-Validation

As a statistical analysis method, cross-validation is commonly used to verify the performance of the recognition model. In the work, we adopted the 5-fold cross-validation, 10-fold cross-validation, and Leave-One-Out cross-validation to verify the model. In the 5-fold cross-validation, the dataset is divided into 5 subsets. In each subset, there are 72 samples for the ASL dataset with 360 samples, 96 samples for the ASL dataset with 480 samples, and 60 samples for the Handicraft-Gesture dataset. In 10-fold cross-validation, the dataset is divided into 10 subsets. In each subset, there are 36 samples for the ASL dataset with 360 samples, 48 samples for the ASL dataset with 480 samples, and 30 samples for the Handicraft-Gesture dataset. In the Leave-One-Out cross-validation, for each dataset, one sample is used as the testing set and the others are used as the training set. Leave-One-Out cross-validation is robust, but it is computationally expensive. In order to reduce the computation time, we chose to use pre-training (Transfer Learning) [36] to conduct Leave-One-Out cross-validation. In our experiments, for each dataset, pre-training starts with a model resulted from about 200 epochs of computation on the same dataset. We noticed that the model converges quickly on these datasets. Therefore, we only ran nearly five epochs (nearly 150 iterations) for the Leave-One-Out cross-validation experiments to obtain comparably good results. The results are shown in Tables 3–5.

Table 3. Recognition accuracies (%) for the three datasets by using the 5-fold cross-validation method.

	ASL Dataset (360 Samples) (%)	ASL Dataset (480 Samples) (%)	Handicraft-Gesture Dataset (%)
1	95.83	90.63	93.33
2	95.83	94.79	83.33
3	84.72	90.63	90
4	94.44	94.79	83.33
5	95.83	97.93	93.33
Average:	93.33	93.75	88.66

Table 4. Recognition accuracies (%) for the three datasets by using the 10-fold cross-validation method.

	ASL Dataset (360 Samples) (%)	ASL Dataset (480 Samples) (%)	Handicraft-Gesture Dataset (%)
1	94.44	97.92	100
2	88.89	93.75	93.33
3	91.67	95.83	96.67
4	91.67	89.53	86.67
5	97.2	91.67	86.67
6	97.2	87.5	83.33
7	86.1	91.67	76.67
8	100	97.92	96.67
9	97.2	97.92	86.67
10	97.2	91.67	93.33
Average:	94.1	93.5	90

Table 5. Recognition accuracies (%) for the three datasets by using the Leave-One-Out cross-validation method.

	ASL Dataset (360 Samples) (%)	ASL Dataset (480 Samples) (%)	Handicraft-Gesture Dataset (%)
Average:	98.33	98.13	92

3.6. Comparisons

Firstly, we compared the proposed method with several significant works [21,23,37,38] on the ASL dataset. American Sign Language is one of the most important communication tools between the deaf–mute and contains many challenging dynamic hand gestures. Most of works chose a subset of the American Sign Language to build dataset and tested recognition systems.

Table 6 shows that the proposed method surpasses the accuracies of three works on the ASL dataset and, compared to [23], we obtained similar results on the dataset. In particular, the overall accuracy in [23] is 96.4102%, but the dataset contains 18 static hand gestures (720 samples) and 12 dynamic hand gestures (480 samples). We removed the static hand gesture, and the accuracy of the dynamic hand gesture is calculated to be 95.2%. The recognition of static hand gestures has matured in recent years and, sometimes, the accuracy is up to 100%. Therefore, it is reasonable for us to compare with it. The other works are all based on the same or a smaller sample size, and they have different models and collection approaches of dynamic hand gestures.

Table 6. Comparison of the accuracy among significant works on the ASL dataset.

Method	Dataset	Accuracy
This paper’s method	ASL (360 samples)	96.3%
This paper’s method	ASL (480 samples)	95.2%
Deep Recurrent Neural Network on Depth Features [23]	ASL (480 samples)	95.2%
Hidden Conditional Neural Field on Depth Features [21]	ASL (360 samples)	89.5%
Action Graph on Silhouette Features [37]	ASL (360 samples)	87.7%
SVM on Random Occupancy Pattern Features [38]	ASL (334 samples)	88.5%

Furthermore, this work also compared the proposed method with the work in [21] on Handicraft-Gesture dataset and, as shown in Table 7, our method achieved better performance. In addition, in its confusion matrix, it has low accuracies in the recognition of gesture ‘pinch’ (82.5%) and ‘pull’ (87.5%). However, our work only gets a low accuracy in the recognition of ‘key tap’ (78%) (Figure 11).

Table 7. Comparison of the accuracy among significant works on the Handicraft-Gesture dataset.

Method	Dataset	Accuracy
This paper's method	Handicraft-Gesture (300 samples)	96.7%
Hidden Conditional Neural Field on Depth Features [21]	Handicraft-Gesture (300 samples)	95%

The computation time of our experiments is shown in the Table 8.

Table 8. The computation time of the experiments.

Dataset	Item	Epoch	Time(h)
ASL dataset with 360 samples	single experiment (7:3)	620	2
	5-fold cross-validation (4:1)	612	9.7
	10-fold cross-validation (9:1)	612	29.52
	Leave-One-Out cross-validation (359:1)	5	5.4
ASL dataset with 480 samples	single experiment (13:7)	1170	4.4
	5-fold cross-validation (4:1)	720	13.1
	10-fold cross-validation (9:1)	810	31.3
	Leave-One-Out cross-validation (479:1)	5	9.5
Handicraft-Gesture dataset with 300 samples	single experiment (7:3)	630	1.8
	5-fold cross-validation (4:1)	630	8.8
	10-fold cross-validation (9:1)	810	24.9
	Leave-One-Out cross-validation (299:1)	5	3.8

4. Conclusions

In this work, a recognition system of dynamic hand gesture recognition method is presented. It is based on a two-layer Bidirectional Recurrent Neural Network. In addition, an efficient dynamic gesture acquisition framework is proposed, and 26 discriminative features based on angles, positions, and distances between fingers are used. The system was able to obtain high accuracy results and our method demonstrated similar or better performance than several works on ASL and Handicraft-Gesture datasets. Future work will be focused on larger and more difficult dynamic hand gestures, combining two or more LMCs to capture dynamic hand gesture sequences. Furthermore, we are also working on creating a new dataset that includes more challenging dynamic hand gestures from ASL and other sign languages which supports multiple formats of data, such as time-based sequences and depth images. This dataset will serve as benchmark for testing on different models.

Author Contributions: Conceptualization, L.Y., J.C. and W.Z.; methodology, J.C. and L.Y.; software, J.C.; validation, L.Y., J.C. and W.Z.; formal analysis, L.Y. and J.C.; writing the paper, L.Y., J.C. and W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: W.Z. was partially funded by National Science Foundation, grant number EEC-1855147.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Parimalam, A.; Shanmugam, A.; Raj, A.S.; Murali, N.; Murty, S.A.V.S. Convenient and elegant HCI features of PFBR operator consoles for safe operation. In Proceedings of the 4th International Conference on Intelligent Human Computer Interaction (IHCI), Kharagpur, India, 27–29 December 2012; pp. 1–9.
2. Minwoo, K.; Jaechan, C.; Seongjoo, L.; Yunho, J. IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces. *Sensors* **2019**, *19*, 3827–3839.
3. Cheng, H.; Yang, L.; Liu, Z.C. Survey on 3D hand gesture recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *26*, 1659–1673. [[CrossRef](#)]

4. Cheng, H.; Dai, Z.J.; Liu, Z.C. Image-to-class dynamic time warping for 3D hand gesture recognition. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, 15–19 July 2013; pp. 1–6.
5. Hachaj, T.; Piekarczyk, M. Evaluation of Pattern Recognition Methods for Head Gesture-Based Interface of a Virtual Reality Helmet Equipped with a Single IMU Sensor. *Sensors* **2019**, *19*, 5408. [[CrossRef](#)] [[PubMed](#)]
6. Rautaray, S.S.; Agrawal, A. Real time hand gesture recognition system for dynamic applications. *IJU* **2012**, *3*, 21–31. [[CrossRef](#)]
7. Hu, T.J.; Zhu, X.J.; Wang, X.Q. Human stochastic closed-loop behavior for master-slave teleoperation using multi-leap-motion sensor. *Sci. China Tech. Sci.* **2017**, *60*, 374–384. [[CrossRef](#)]
8. Bergh, M.V.D.; Carton, D.; Nijs, R.D.; Mitsou, N.; Landsiedel, C.; Kuehnlentz, K.; Wollherr, D.; Gool, L.V.; Buss, M. Real-time 3D hand gesture interaction with a robot for understanding directions from humans. In Proceedings of the Ro-Man, Atlanta, GA, USA, 31 July–3 August 2011; pp. 357–362.
9. Doe-Hyung, L.; Kwang-Seok, H. Game interface using hand gesture recognition. In Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), Seoul, Korea, 30 November–2 December 2010; pp. 1092–1097.
10. Rautaray, S.S.; Agrawal, A. Interaction with virtual game through hand gesture recognition. In Proceedings of the International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT), Aligarh, India, 17–19 December 2011; pp. 244–247.
11. Kim, T.; Shakhnarovich, G.; Livescu, K. Fingerspelling recognition with semi-Markov conditional random fields. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 1–8 December 2013; pp. 1521–1528.
12. Myoung-Kyu, S.; Sang-Heon, L.; Dong-Ju, K. A comparison of 3D hand gesture recognition using dynamic time warping. In Proceedings of the 27th Conference on Image and Vision Computing New Zealand (IVCNZ), Dunedin, New Zealand, 26–28 November 2012; pp. 418–422.
13. Nobutaka, S.; Yoshiaki, S.; Yoshinori, K.; Juno, M. Hand gesture estimation and model refinement using monocular camera-ambiguity limitation by inequality constraints. In Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition (FG), Nara, Japan, 14–16 April 1998; pp. 268–273.
14. Youngmo, H. A low-cost visual motion data glove as an input device to interpret human hand gestures. *IEEE Trans. Consum. Electron.* **2010**, *56*, 501–509.
15. Guna, J.; Jakus, G.; Pogačnik, M.; Sašo, T.; Jaka, S. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors* **2014**, *14*, 3702–3720. [[CrossRef](#)] [[PubMed](#)]
16. Zhang, Z.Y. Microsoft kinect sensor and its effect. *IEEE Multimed.* **2012**, *19*, 4–10. [[CrossRef](#)]
17. Xu, Y.R.; Wang, Q.Q.; Bai, X.; Chen, Y.L.; Wu, X.Y. A novel feature extracting method for dynamic gesture recognition based on support vector machine. In Proceedings of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 28–30 July 2014; pp. 437–441.
18. Karthick, P.; Prathiba, N.; Rekha, V.B.; Thanalaxmi, S. Transforming Indian sign language into text using leap motion. *IJIRSET* **2014**, *3*, 10906–10910.
19. Schramm, R.; Jung, C.R.; Miranda, E.R. Dynamic time warping for music conducting gestures evaluation. *IEEE Trans. Multimed.* **2014**, *17*, 243–255. [[CrossRef](#)]
20. Kumar, P.; Saini, R.; Roy, P.P.; Dogra, D.P. Study of text segmentation and recognition using leap motion sensor. *IEEE Sens. J.* **2016**, *17*, 1293–1301. [[CrossRef](#)]
21. Lu, W.; Tong, Z.; Chu, J.H. Dynamic hand gesture recognition with leap motion controller. *IEEE Signal Process. Lett.* **2016**, *23*, 1188–1192. [[CrossRef](#)]
22. Zhang, Q.X.; Deng, F. Dynamic gesture recognition based on leapmotion and HMM-CART model. In Proceedings of the 2017 International Conference on Cloud Technology and Communication Engineering (CTCE), Guilin, China, 18–20 August 2017; pp. 1–8.
23. Avola, D.; Bernardi, M.; Cinque, L.; Foresti, G.L.; Massaroni, C. Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures. *IEEE Trans. Multimed.* **2018**, *21*, 234–245. [[CrossRef](#)]
24. Zeng, W.; Wang, C.; Wang, Q.H. Hand gesture recognition using Leap Motion via deterministic learning. *Multimed. Tools Appl.* **2018**, *77*, 28185–28206. [[CrossRef](#)]

25. Mittal, A.; Kumar, P.; Roy, P.P.; Balasubramanian, R.; Chaudhuri, B.B. A Modified LSTM Model for Continuous Sign Language Recognition Using Leap Motion. *IEEE Sens. J.* **2019**, *19*, 7056–7063. [[CrossRef](#)]
26. Deriche, M.; Aliyu, S.O.; Mohandes, M. An Intelligent Arabic Sign Language Recognition System Using a Pair of LMCs With GMM Based Classification. *IEEE Sens. J.* **2019**, *19*, 8067–8078. [[CrossRef](#)]
27. Marin, G.; Dominio, F.; Zanuttigh, P. Hand gesture recognition with leap motion and kinect devices. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 1565–1569.
28. Marin, G.; Dominio, F.; Zanuttigh, P. Hand gesture recognition with jointly calibrated leap motion and depth sensor. *Multimed. Tools Appl.* **2016**, *75*, 14991–15015. [[CrossRef](#)]
29. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [[CrossRef](#)]
30. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; pp. 665–667.
31. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
32. Semeniuta, S.; Severyn, A.; Barth, E. Recurrent dropout without memory loss. In Proceedings of the 26th International Conference on Computational Linguistics (COLING), Osaka, Japan, 11–16 December 2016; pp. 1757–1766.
33. Gal, Y.; Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 1019–1027.
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
35. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inform. Process. Manag.* **2009**, *45*, 427–437. [[CrossRef](#)]
36. Liang, H.; Fu, W.L.; Yi, F.J. A Survey of Recent Advances in Transfer Learning. In Proceedings of the IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 16–19 October 2019; pp. 1516–1523.
37. Kurakin, A.; Zhang, Z.Y.; Liu, Z.C. A real time system for dynamic hand gesture recognition with a depth sensor. In Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Bucharest, Romania, 27–31 August 2012; pp. 1975–1979.
38. Wang, J.; Liu, Z.C.; Chorowski, J.; Chen, Z.Y.; Wu, Y. Robust 3d action recognition with random occupancy patterns. In Proceedings of the European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; pp. 872–885.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).