applied sciences

MDPI

Article

Recognition of Hand Gesture Sequences by Accelerometers and Gyroscopes

Yen-Cheng Chu [1,†], Yun-Jie Jhang [2,†], Tsung-Ming Tai [3,†] and Wen-Jyi Hwang [1,*,†]

[1] Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei 117, Taiwan; 60647009s@ntnu.edu.tw
[2] Andro Video, Taipei 115, Taiwan; taco.chang@androvideo.com
[3] NVIDIA AI Technology Center, NVIDIA Taiwan, Taipei 114, Taiwan; ntai@nvidia.com
* Correspondence: whwang@ntnu.edu.tw
† These authors contributed equally to this work.

Received: 25 August 2020; Accepted: 16 September 2020; Published: 18 September 2020

Abstract: The objective of this study is to present novel neural network (NN) algorithms and systems for sensor-based hand gesture recognition. The algorithms are able to classify accurately a sequence of hand gestures from the sensory data produced by accelerometers and gyroscopes. They are the extensions from the PairNet, which is a Convolutional Neural Network (CNN) capable of carrying out simple pairing operations with low computational complexities. Three different types of feedforward NNs, termed Residual PairNet, PairNet with Inception, and Residual PairNet with Inception are proposed for the extension. They are the PairNet operating in conjunction with short-cut connections and/or inception modules for achieving high classification accuracy and low computation complexity. A prototype system based on smart phones for remote control of home appliances has been implemented for the performance evaluation. Experimental results reveal that the PairNet has superior classification accuracy over its basic CNN and Recurrent NN (RNN) counterparts. Furthermore, the Residual PairNet, PairNet with Inception, and Residual PairNet with Inception are able to further improve classification hit rate and/or reduce recognition time for hand gesture recognition.

Keywords: hand gesture recognition; human–machine interface; artificial intelligence; feedforward neural networks

1. Introduction

Hand gesture recognition is a technique for the mathematical interpretation of hand movements by computers. It can be used to facilitate the interaction between human and computer. Gesture recognition has been found to be effective for applications such as devices control, entertainment, health care, and education. Hand gesture recognition approaches can be separated into two classes: Vision-Based Recognition (VBR) algorithms [1–4] and Sensor-Based Recognition (SBR) algorithms [5–16]. The VBR algorithms perform gesture recognition from images captured by a camera. Although accurate classification is possible, high computation efforts may be required to extract information from images for both training and inference operations.

The SBR algorithms are based on sensors other than camera. Commonly used sensors include accelerometers [5,6], gyroscopes [7,8], photoplethysmography (PPG) [9], flex sensors [10], electromyography (EMG) [11], Radio Frequency IDentification (RFID) [12,13], WiFi Channel State Information (CSI) [14–16], and the fusion of these sensors. In some SBR-based studies, high classification accuracy for gesture recognition has been observed. However, many of these techniques do not support the recognition of a sequence of gestures. Only isolated gestures can be recognized.

Furthermore, some SBR techniques are based on Dynamic Time Warping (DTW) [7] technique for gesture recognition, which may incur high computation complexities. The Recurrent Neural Networks (RNNs) [17] and its variants such as Long Short Term Memory (LSTM) algorithm [18] have been found to be effective [8,19] for the recognition of gesture sequences with low computation costs. Nevertheless, because of the inherent gradient vanishing problem, the algorithms may not be able to effectively exploit the long term dependency of the sensory data. The dependency may be beneficial for accurate recognition when slow gesture movements are observed.

In addition to RNNs, feedforward networks such as Convolutional Neural Networks (CNNs) [17,20] can be used for hand gesture recognition. The One-Dimensional (1D) CNNs have been found to be effective for a number of applications such as the classifications of ECG signals [21], human activities [22], and internet traffic [23]. Gesture recognition based on basic 1D CNNs may achieve high classification accuracy when the kernel sizes and/or the depth of the network are large. However, in these cases, the computation complexities for inference may also be high. Although the computation load can be alleviated by lowering the kernel sizes and/or the depth of the network, the coverage area of receptive field will then become small. Consequently, the classification error may become large because the long term dependency may not be effectively exploited with a small receptive field. The WaveNet [24,25] can be used to solve the problem. It is based on dilated convolution operations for the growth of receptive field with low computational complexities. However, the WaveNet is used for signal generation with additional autoregression operations. Direct applications of WaveNet to hand gesture recognition may then be difficult.

The objective of this study is to present novel SBR algorithms and systems based on feedforward neural networks for effective recognition of a sequence of hand gestures. The proposed algorithms have the advantages of high classification accuracy and low computation complexities. They are based on basic accelerometers and gyroscopes commonly used in smart phones or devices. This may facilitate the deployment of the algorithms to large varieties of the applications in smart devices.

The proposed feedforward algorithms are based on the PairNet [26] featuring large receptive field and simple inference process. The PairNet is implemented by configuring the CNN with a kernel size of two and a stride size of two. The corresponding CNN operations can be regarded as the simple pairing operations, where the input sequences to each convolution layer can be separated into non-overlapping pairs. The operations for each pair are determined by the weights associated with the kernels. Because of the simplicity of the operations, a deep CNN based on PairNet algorithm can then be easily formed for high classification accuracy with low computation time as compared with the basic CNN.

Although the PairNet has a simple structure, it may have large depth for full coverage of a single gesture. However, the classification accuracy may be saturated as the depth increases. One approach to further improve the performance is the employment of short-cut connections [27–29] for some layers. This may be beneficial for providing a good reference for effective training and inference at these layers [27]. The resulting network, termed Residual PairNet, has superior classification accuracy over the PairNet and basic CNN. Similar to the PairNet, the Residual PairNet has low computational complexities as compared with basic CNN.

The performance of the feedforward algorithms can be further improved. This is based on the observation that sensory data produced by accelerator and gyroscope may possess both slow and rapid variations dependent on the hand movements. Consequently, the employment of inception modules [20,30–32] consisting of different sizes of kernels at the same layer may be beneficial for efficient capturing of gesture features. Furthermore, while having superior classification accuracy, the PairNet with inception is able to lower the computation costs of the PairNet. This is because dimensionality reduction can be carried out in the inception modules for lowering the number of weights [30]. Similarly, the residual PairNet can also operate in conjunction with inception. The employment of short-cut connections and/or inception modules to PairNet provides flexibilities

to the feedforward networks for varying requirements on classification accuracy, computation costs, and design complexities.

To demonstrate the effectiveness of the proposed feedforward algorithms, a smart home system was built, where the home appliances can be controlled remotely by hand gestures. The sequences of hand gestures were acquired by smart phones, and were delivered to the embedded systems in the home appliances for gesture recognition. The trained neural network models were deployed in the embedded systems so that sequences of hand gestures can be recognized in real-time. Experimental results reveal that the proposed algorithms are able to carry out real-time gesture recognition on embedded devices with only limited computation capacity. Furthermore, they have superior classification accuracy over existing works. They are therefore well-suited for the implementation of light-weight smart human–computer interfaces where the only sensors used are for hand gesture recognition.

The remaining parts of this paper are organized as follows. Section 2 reviews some advanced works for gesture recognition. Section 3 presents the proposed gesture recognition systems. The PairNet algorithm is studied in detail. The augmentation of short-cut connections and inception modules to the PairNet are also discussed. Experimental results of the proposed feedforward networks are then presented in Section 4. Finally, Section 5 includes some concluding remarks of this work.

2. Related Works

VBR approaches can be separated into two classes: data-driven and model-based. The data-driven methods [1–4] are based on the appearance of gesture images. The gestures are classified by relating the appearance of any gesture to the appearance of pre-defined gestures. Basic model-based methods [33] can be adopted for 3D hand tracking/recognition and 3D gesture estimation. Hand model fitting operations are usually required for the tracking of hand motion. As cameras are required for VBR approaches, computational complexities for these approaches may be high.

Many existing SBR methods are data-driven. A common feature of some SBR-based techniques is that they are based on wearable sensors such as PPG sensors [9], flex sensors [10], and EMG sensors [11] for attaining high recognition accuracy. However, the systems with wearable sensors demand users to take extra devices for gesture recognition. These techniques may then be intrusive, because the devices sometimes are obstructive and inconvenient for gesture actions. This could undermine the quality of user experience.

An alternative to wearable sensors is based on RFIDs for the location-sensing of hand gestures. The techniques in [12] carry out the tracking of motion patterns of RFID tags for gesture recognition. Although high accuracy can be achieved, users are still required to wear tags. To alleviate user intrusiveness, the RFID tags and readers were deployed in a fixed location in the study [13]. The recognition is then based on the fact that when gestures are performed in front of tags, the movement information of the gestures can be extracted by the resulting phases of the RFID signals. Although there is no demand for wearing sensors, gesture recognition can only take place in locations where RFID tags and readers are properly deployed.

Another approach for gesture recognition with no user intrusiveness is based on the CSI information from commercial WiFi. It is known that fluctuation of WiFi signals can be observed by human actions. Variation statics such as variance and correlation coefficients [14] have been found to be beneficial for the detection of human activities. However, the detection performance may be limited by the random noises and/or WiFi channel variations. Furthermore, it may be difficult to carry out accurate gesture classifications from the simple statics. Approaches based on deep learning in [15,16] are proposed to solve the problem. By the incorporation of RNN and its variants such as LSTM, the features of gestures can be effectively extracted, and accurate recognition can be achieved at the presence of noises. However, similar to their RFID counterparts [12,13], the performance of WiFi CSI-based techniques may be dependent on the deployments of WiFi transmitters and receivers in an indoor environment.

As compared with existing SBR-based algorithms, the proposed algorithms have a number of advantages. The first is that they are not user intrusive. Sensors commonly available in smart phones such as accelerometers and gyroscopes are adopted for gesture recognition. Therefore, only a single smart phone is needed for gesture recognition without additional wearable sensors and/or devices. Furthermore, the proposed algorithms provide higher flexibilities for the deployment of gesture recognition systems as compared with the existing RFID and WiFi techniques. The acquisition of sensory data is performed in the smart phone locally from its accelerometers and gyroscopes. The acquired sensory data can be delivered to any external devices accessible by Internet for gesture inference. Therefore, in the proposed algorithms, gesture recognition can take place in either indoor or outdoor environments, wherever Internet access is available.

3. The Proposed Gesture Recognition Algorithms and Systems

In this section, we first give an overview of the proposed gesture algorithms and systems. The four feedforward neural networks (i.e., PairNet, Residual PairNet, PairNet with Inception, and Residual PairNet with Inception) are presented separately in the subsequent subsections. We then consider the issues for the training data collection for the algorithms, which is followed by the applications of the algorithms such as the remote control systems for home appliances. To facilitate the understanding of the discussions in this study, Table 1 includes a list of frequently used symbols.

Table 1. A list of frequently used symbols in this study.

Symbol	Meaning
A	The path of the sensory data. Each individual index along the path is the index of the gesture having highest probability at its corresponding time step.
a_t	The index of the gesture having the largest probability at time step t. It is the t-th component of A, $t = 1, ..., T$.
K	Number of different gestures in the input sensory data sequence X.
M	Number of elements in each input sample x_t, $t = 1, ..., T$.
N	Length of input window X_t.
Q	Number of gesture classes.
R	Classification results.
r_k	The k-th element of classification results R, $k = 1, ..., K$.
T	Length of the input sequence X and output sequence Y.
X	Input sensory data sequence.
X_t	Input window with central component x_t.
x_t	The sample at time step t of input sensory data sequence X, $t = 1, ..., T$.
Y	Gesture spotting results.
y_t	The sample at time step t of gesture spotting results Y, $t = 1, ..., T$.
$y_{t,j}$	The j-th element of y_t, $j = 1, ..., Q$, $t = 1, ..., T$.

3.1. Overview

As shown in Figure 1, the proposed gesture recognition algorithms can be separated into two parts: feedforward neural networks and Maximum A Posteriori (MAP) estimation. The feedforward networks take sensory data as input. Examples of sensory data include the data produced by accelerometers and/or gyroscopes. Based on the input data, the feedforward neural networks carry out the gesture spotting operations. Given the spotting results, the MAP estimation is then performed to obtain the final classification outcomes.

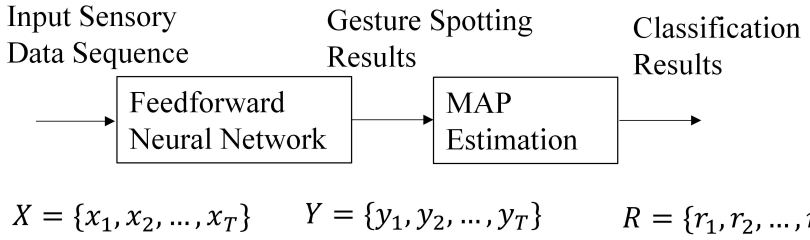

Figure 1. The overview of the proposed gesture recognition system, where X, Y, and R denote the input sensory data, gesture spotting results, and classification results, respectively.

Let $X = \{x_1, ..., x_T\}$ be an input sensory data sequence for the recognition of a hand gesture sequence containing K different gestures back-to-back. Each of the K gestures belongs to one of the pre-defined Q gesture classes. Each x_t, $t = 1, 2, ..., T$, is the sample of the sequence X acquired at time step t, and T is the length of of the sequence. There are M elements in each sample x_t of the data sequence X. The dimension M is dependent on the sensors for the gesture recognition. For example, when both 3-axis accelerometer and 3-axis gyroscope are used, each sample x_t contains six elements, and therefore $M = 6$. However, when only one of the sensors is used, $M = 3$. Let $Y = \{y_1, ..., y_T\}$ be the results produced by a feedforward neural network, where y_t is the output of the network at time step t. There are Q elements in each output y_t. Let $y_{t,j}$ be the j-th element of y_t, $t = 1, ..., Q$. The activation function associated with y_t is the softmax activation function. We then can view $y_{t,j}, j = 1, ..., Q$, as the probability of the occurrence of gesture class j at the time step t.

Figure 2 shows the operations of a feedforward network on X for producing Y. Starting from $t = 1$, the feedforward network produces y_t based on X_t for each t until $t = T$ is reached, where X_t is an input window with length N. The central component of X_t is x_t. This allows for the full exploitation of correlation among neighboring samples of x_t at expense of higher latency for acquiring the input window X_t. Although causal operations are possible, only the correlation in the past neighboring samples of x_t is exploited. Because the full utilization of correlation among neighboring samples is important for accurate gesture spotting, causal operations are not considered. Note that, when $t < N/2$ or $t > T - N/2$, parts of X_t is outside X. These parts are filled with zeros.

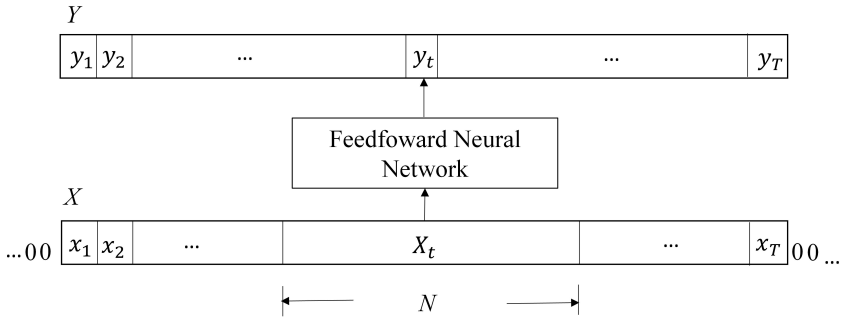

Figure 2. The feedforward operations on X for producing Y.

After $Y = \{y_1, ..., y_T\}$ is available, post-processing operations are required to produce the final classification results. With the a priori knowledge of the number of gestures K contained in the input sensory data $X = \{x_1, ..., x_T\}$, the goal of the post-processing operations is to find the set of indices $R = \{r_1, ..., r_K\}$ of the gestures, where r_k is the index of the k-th gesture appears in the sensory data $X = \{x_1, ..., x_T\}$. The selection of K is dependent on the number of different gesture sequences required for an application. Under the restriction of no occurrence of repetitive gestures, there are at most $Q \times (Q-1) \times ... \times (Q-(K-1))$ different gesture sequences.

Let a_t be the index of the gesture having the largest probability at time step t. That is,

$$a_t = \underset{1 \le j \le Q}{\operatorname{argmax}}\, y_{t,j}. \tag{1}$$

We call $A = \{a_1, ..., a_T\}$ the path of the sensory data, where each individual index along the path is the index of the gesture having highest probability at its corresponding time step. The computation of final classification outcome R is based on the probability model

$$Prob(C/A) = \prod_{k=1}^{K} Prob(c_k/A), \tag{2}$$

where $C = \{c_1, ..., c_K\}$ is a possible classification outcome, and c_k is the index of the k-th gesture of C. The probability $Prob(c_k/A)$ is estimated by

$$Prob(c_k/A) = \frac{|I_{c_k}|}{T}, \tag{3}$$

where

$$I_{c_k} = \{a_t : a_t = c_k\}, \tag{4}$$

and the size of I_{c_k} is denoted by $|I_{c_k}|$. The final classification result R is then the C maximizing $Prob(C/A)$. That is,

$$R = \underset{C \in S}{\operatorname{argmax}}\, P(C/A), \tag{5}$$

where S denotes the set of all possible classification outcomes.

From (2) and (3), the search process in (5) is equivalent to the identification of gestures having top-K occurrence. The classification results $R = \{r_1, ..., r_K\}$ are then obtained from these gestures according to their locations in the path A. Figure 3 shows an example of mapping from the path A to the classification results R for the recognition of three gestures (i.e., $K = 3$). In this example, based on the results of the feedforward neural network, the operations in (1) produces index values 1, 3, 4, 6, or 7. The locations of the corresponding intervals I_1, I_3, I_4, I_6, and I_7 are revealed in Figure 3. The top-three largest intervals are I_1, I_3, and I_6. The set of gestures having the top-three occurrences then contains Gesture 1, Gesture 3, and Gesture 6. Their order of occurrence along the path A is Gesture 3, Gesture 6, and Gesture 1. Consequently, $r_1 = 3$, $r_2 = 6$, and $r_3 = 1$.

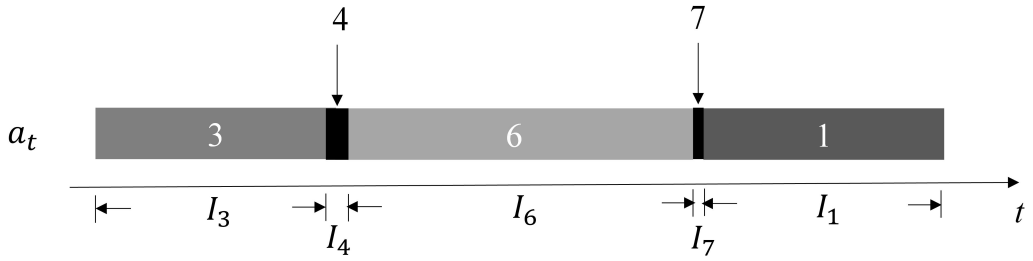

Figure 3. An example of mapping from path $A = \{a_1, ..., a_T\}$ to final classification outcome $R = \{r_1, r_2, r_3\}$ for the recognition of three gestures (i.e., $K = 3$). In this example, $r_1 = 3$, $r_2 = 6$, and $r_3 = 1$.

The feedforward neural network in the proposed system could be a basic CNN, or the PairNet, Residual PairNet, PairNet with Inception, or Residual PairNet with Inception. They are presented separately in the following subsections.

3.2. PairNet

As shown in Figure 4, the PairNet is a 7-layer CNN, where layers 1–5 are convolutional layers. Layer 1 is the convolutional layer with stride size 1 and kernel size 1×3. We can observe from Figure 4 that the input data to layer 1 is X_t with length $N = 50$. Recall that X_t is a window from the sensory data X, where each sample contains $M = 6$ elements. We therefore view X_t as a set of six 1D sequences, where the i-th 1D sequence is formed by the i-th element of samples of the sensory data in X_t, $i = 1, \ldots, 6$. In this study, each 1D sequence is termed a channel. Therefore, the number of input channels is six. Layer 1 produces a set of 128 1D sequences. The number of output channels for layer 1 is then 128. From Figure 4, we see that the length of each 1D sequence (i.e., channel) is 48.

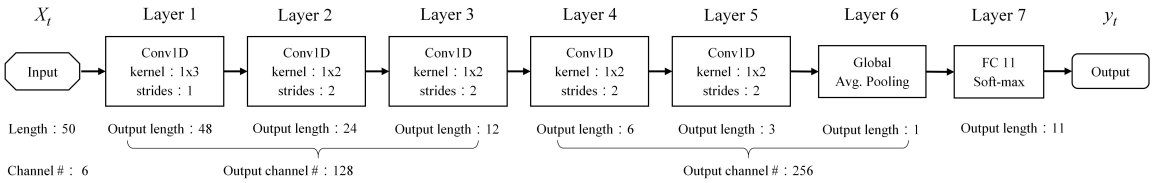

Figure 4. The parameters of the PairNet. The dimension of each sample of X_t is 6 ($M = 6$). Therefore, there are six channels. Each channel has a length of 50. Layer 1 is a convolution layer with kernel size 1×3 and stride size 1. The output length therefore is 48. Layers 2–5 are also convolution layers with kernel size 1×2 and stride size 2 for pairing operations. The output lengths of Layers 2–5 are then 24, 12, 6, and 3, respectively. Layer 6 is an average pooling layer averaging three elements to a single one. The number of gestures to be classified is $Q = 11$. Layer 7 is a fully connected layer producing 11 outputs.

The output sequences produced by layer 1 then serve as the input sequences to layer 2, which in turns propagates its computation results to subsequent layers. For layers 2–5, the stride size is 2, and kernel size is 1×2. The corresponding convolution operations can be viewed as a pairing operations, where each of the input 1D sequences are separated into disjoint pairs (due to a stride size of 2), and each pair operates independently in accordance with the kernel weights (due to kernel size 1×2). The pairing operations then produces output channels half the length of the input channels to that layer. Although the length of channels decrease as the data propagate through the network, we retain or increase the number of channels so that sufficient features can be extracted for final classification. In fact, there are 128 output channels at layers 2 and 3, and there are 256 output channels at layers 4 and 5.

When we only consider the length of channels at each layer, the structure of PairNet therefore is pyramid-like, as shown in Figure 5. The output channels produced by the layer 5 are located on the top of the pyramid. At the layer 6, the average pooling operations are then carried out over these output channels. The resulting data are subsequently flattened, and served as inputs to the fully connected layer with softmax activation function for the final classification results at layer 7.

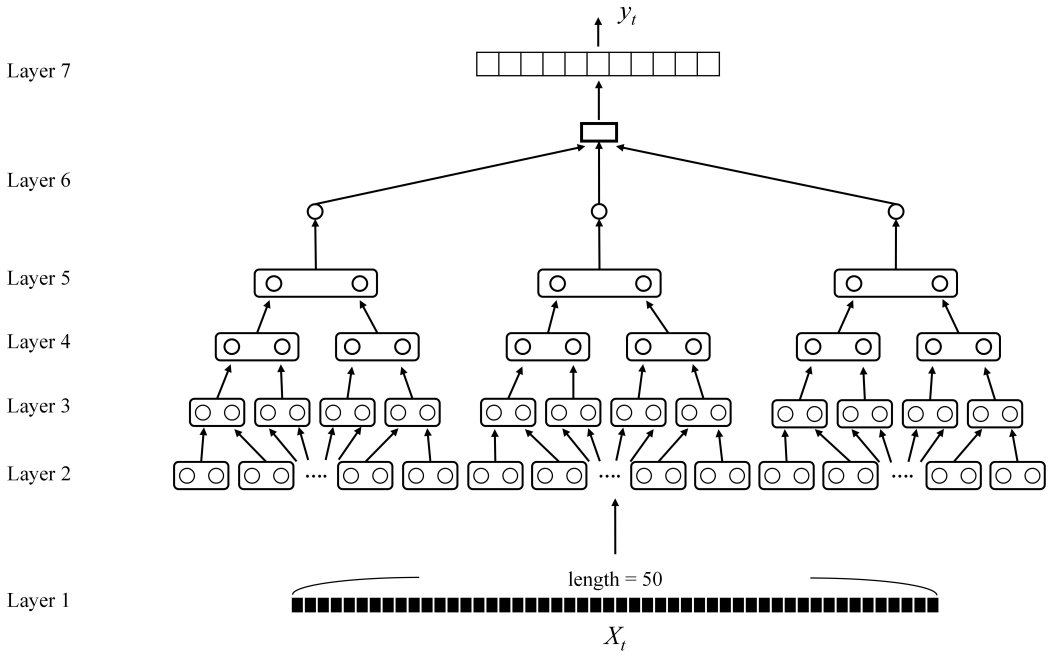

Figure 5. The structure of PairNet. In this example, the length of X_t is $N = 50$. There are seven layers in the network. Layer 1 is a basic convolution layer. Layers 2–5 are convolution layers supporting pairing operations. Layers 6 and 7 are pooling layer and fully-connected layers, respectively.

3.3. Residual PairNet

The performance of the PairNet can be improved by the employment of short-cut connections in the network. The resulting network is termed residual PairNet in this study. Figure 6 shows an example of the Residual PairNet accommodating short-cut connections. By comparing Figure 4 with Figure 6, we can see that the Residual PairNet has two short-cut connections. The first short-cut connection is from the output of Layer 1 to the output of Layer 3. The second one is from the input of Layer 4 to the output of Layer 5.

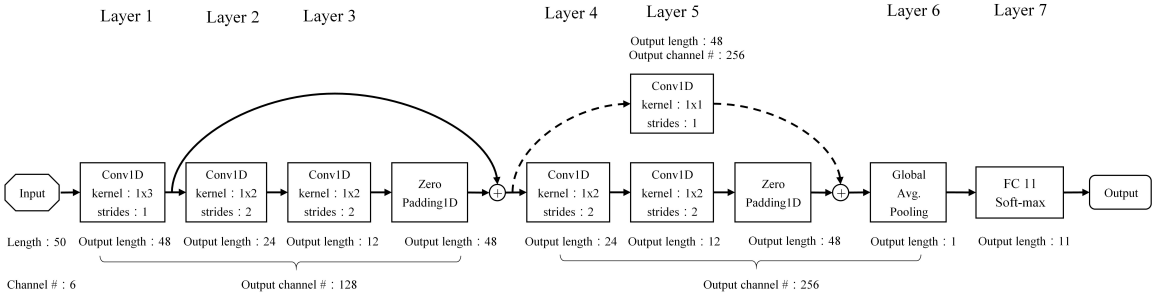

Figure 6. The parameters of the Residual PairNet. The input X_t has the same dimension and number of channels as those of the input of PairNet. The network contains seven layers and two short-cut connections. Two zero-padding modules are also included for the combination of the short-cut connections with the convolutional layers. Layers 1–5 are convolution layers. Layer 6 is the average pooling layer. The number of gestures to be classified is $Q = 11$. Layer 7 is a fully connected layer producing 11 outputs.

It can be observed from Figure 6 that the short-cut connections should be combined with the convolution layers for enhancing the performance of the network. To implement the combination, the length of the short-cut connections and the length of output of convolution layers should be identical. Because the pairing operations in the convolution layers reduce the length of their outputs, zero padding operations are included for the compensation of the length reduction in our design.

For example, the length of the first short-cut connection is 48. The length of the output of the Layer 3 is only 12. Therefore, a module is included to increase the length of the output of Layer 3 from 12 to 48 by padding zeros to the end of the output of Layer 3. In this way, the path can be combined with the first short-cut connection, as shown in Figure 6. Similarly, another zero padding module is employed at the output of Layer 5 to facilitate the combination of this path with the second short-cut connection.

In addition to the path length of the short-cut connection, we may need to take the number of channels into consideration for the combination of short-cut connection and convolution layers. This is particularly true for the second short-cut connection. It originates at the path with only 128 channels. Furthermore, it needs to be combined with the path containing 256 channels. This discrepancy can be solved by the employment of the convolution module with kernel size 1×1. There are 128 input channels and 256 output channels for the module. We can then see from Figure 6 that the number of channels at the output of the module is the same as that of the target path for combination. The integration of the short-cut connections to the PairNet can then be carried out.

3.4. PairNet with Inception

Another technique for the improvement of the PairNet technique is the incorporation of an inception module. As shown in Figure 7, the inception module is located at layer 5 of the network. The architecture of the inception module is revealed in Figure 8, which contains four paths. Each path is associated with convolution modules, average pooling module, and/or a zero-padding module. The convolution modules at different paths have different kernel sizes. In this way, features of hand gestures with slow or fast movements can be effectively captured by the modules.

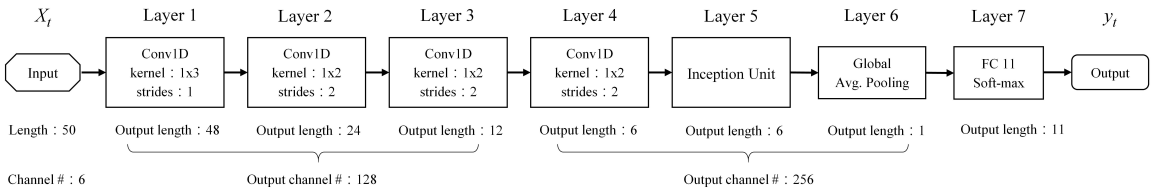

Figure 7. The parameters of the PairNet with Inception. The input X_t has the same dimension and number of channels as those of the input of PairNet. The network contains seven layers, where layers 1–4 are convolution layers. The inception module is located at layer 5. The architecture of inception module is revealed in Figure 8. Layer 6 is the average pooling layer. The number of gestures to be classified is $Q = 11$. Layer 7 is a fully connected layer producing 11 outputs.

An additional advantage of the inception module is that it has lower size of weights. Consequently, its computation complexity is lower than that of the other feedforward counterparts. The superior computation efficiency can be observed from Figure 8 that each convolution module in the inception module carries out the dimension reduction operations. That is, each convolution modules has lower number of output channels as compared with its number of input channels. In this way, the number of weights of the network may be effectively lowered. This may be beneficial for reducing the number of addition and multiplications.

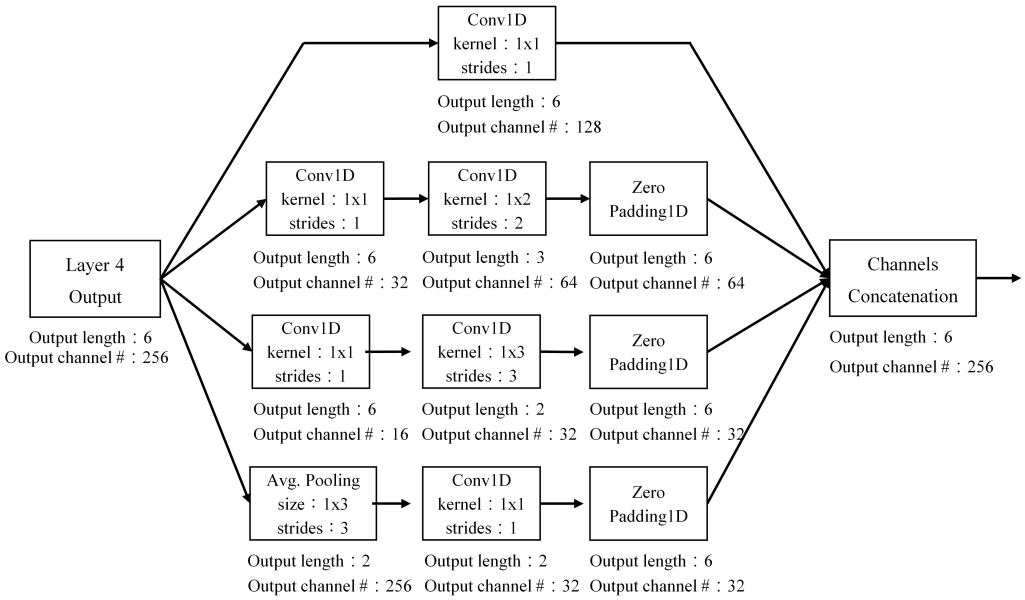

Figure 8. The parameters of the Inception module, which consists of four paths. Each path is associated with convolution modules, average pooling module, and/or a zero-padding module.

3.5. Residual PairNet with Inception

The employment of short-cut connections and inception module may have the advantages of both high classification accuracy and low computational complexities. An example of the design is shown in Figure 9, where both the inception module and short-cut connection are located at layer 5 of the network. Figure 10 shows the architecture of the inception module with short-cut. There are three paths in the module, where one of the paths serves as the short-cut. This allows layer 4 to be directly connected to layer 6. The remaining two paths contain convolution modules and a zero-padding module.

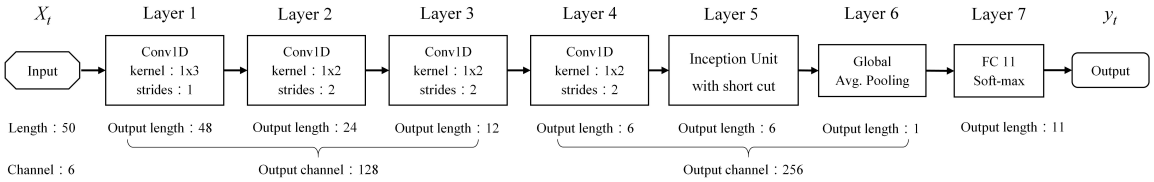

Figure 9. The parameters of Residual PairNet with inception. The input X_t has the same dimension and number of channels as those of the input of PairNet. The network contains seven layers, where the inception module with short-cut connection is located at layer 5. The architecture of inception module is revealed in Figure 10. The number of gestures to be classified is $Q = 11$. Layer 7 is a fully connected layer producing 11 outputs.

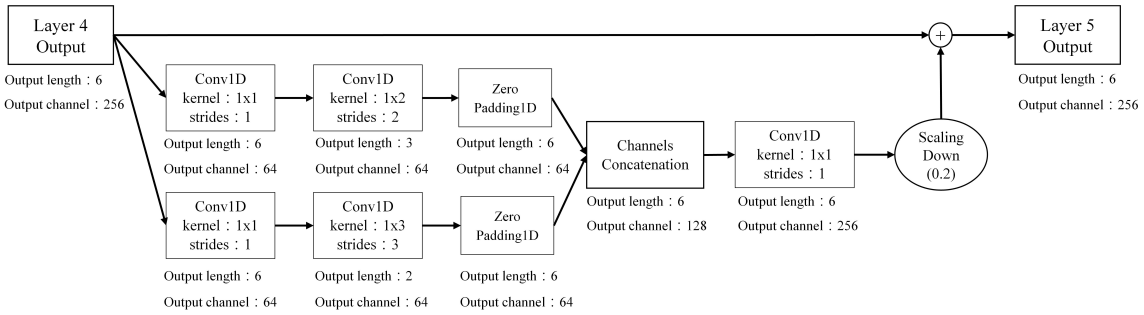

Figure 10. The parameters of the inception module with short-cut. It contains three paths. The top path is the short-cut for connecting the output of layer 4 to the input of layer 6. The remaining two paths are convolution modules and zero padding operations.

By comparing Figures 5, 7 and 9, it can be observed that the major difference among PairNet, Pairnet with Inception, and Residual PairNet with Inception is at layer 5. For the PairNet, only a single convolution module is employed at that layer. By contrast, inception module is adopted at layer 5 for Pairnet with Inception and Residual PairNet with Inception. Therefore, the PairNet may have inferior classification accuracy with larger size of weights as compared with the other two. Furthermore, because the Residual PairNet with inception contains short-cut connections, it may has highest classification accuracy among the three algorithms.

3.6. Gesture Data Collection for Training

Because this study aims to recognize a sequence of K gestures, a basic approach for the collection of training sequences for the proposed algorithms is to require each sensory data sequence for training should contain K different gestures. A drawback of this approach is that the proposed algorithms should be re-trained when the applications for other lengths of gestures K are desired. In addition, the collection of the training sets should also be carried out again when different number of gesture classes Q are adopted for the new applications. The reusability of the neural network models and/or the training data are low. Furthermore, large efforts are required for accurate labeling of K gestures for each sensory data sequence.

To improve the reusability of neural network models and training data for gesture sequences with different lengths of gestures K and/or different number of gesture classes Q, each sensory data sequence for training contains only a single gesture. Since there are Q gesture classes, the training set can be separated into Q subsets. The sensory data sequences belonging to the same subset represent the same gesture. All the training sequences in the Q subsets should be involved in the training of the proposed algorithms. The resulting neural network models can be applied to the recognition of gesture sequences with different K values, where K is known a priori. The same neural network models can then be re-used even though K varies. When the incorporation of new gesture classes or removal of existing gesture classes are desired for the new applications, the training subsets common to the new and original applications can be re-used. Consequently, the reusability for the neural network models and training sets can be improved. Moreover, because each training sensory sequence represents only a single hand gesture, the efforts for labeling can be minimized.

3.7. Gesture-Based Remote Control System for Home Appliances

The proposed algorithms can be effectively used for SBR applications. Figure 11 shows an example of the applications, where the home appliances such as a TV, air conditioner, and music player in a smart home can be remotely controlled by the sensory data of hand gestures acquired by mobile phones. It is assumed that the mobile phones are equipped with accelerators and gyroscope. We developed a mobile application (APP) for the smart phones for capturing the sensory data, and deliver the data via Wifi to external devices.

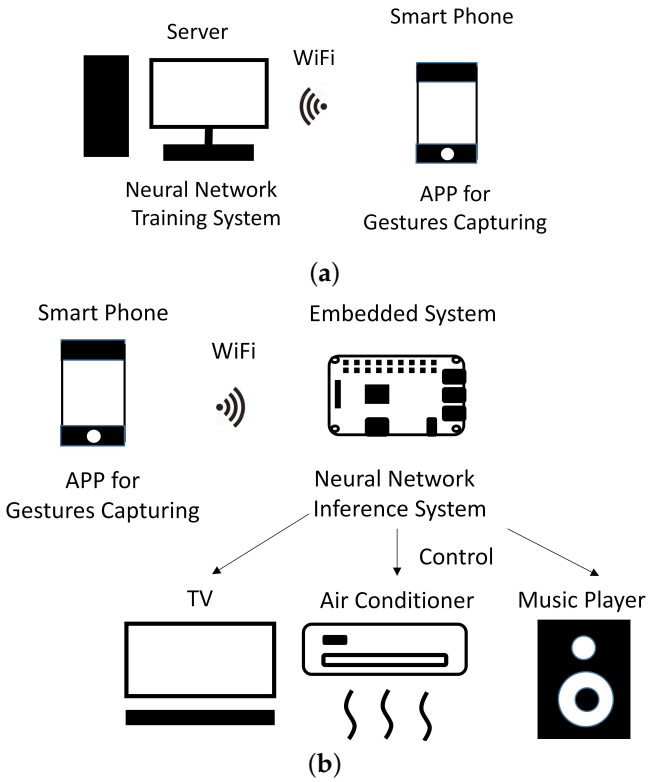

Figure 11. The proposed gesture-based remote control system for home appliances: (**a**) the training system for dynamic gesture recognition; (**b**) real-time gesture recognition system for the remote control of home appliances.

During the training phase, the sensory data acquired by smart phones serve as the training data. The data are delivered to a dedicated server for subsequent annotation and algorithm training. After the training operations are completed, the resulting neural network models are stored in an embedded system for online gesture recognition. It is preferable that the embedded system has a small size and low power consumption so that it can be easily attached to home appliances for the action control. A typical example would be a Raspberry Pi computer. During the inference phase, the sensory data captured by smart phones are sent to the embedded systems, which carry out the gesture recognition based on the trained neural models. The recognition results are then translated into action commands for home appliances.

Because the APP responsible for sampling sensory data is deployed in the smart phone, our system is able to carry out the recording of sensory data in parallel with human gesture actions. In fact, the APP is able to directly acquire the samples produced by accelerometers and gyroscopes of the smart phone, and deliver the data immediately over WiFi to the embedded system responsible for inference. After the human actions are completed, the embedded system has all the sensory data for classification. Inference time is then simply the computation time of neural networks producing the classification results. No additional waiting time for acquiring sensory data at sampling rate is required.

To implement the gesture-based remote control system in a smart home, a set of Q gesture classes needs to be pre-defined. Figure 12 shows an example of 11 gesture classes (i.e., $Q = 11$) for the implementation of the system. The gestures in each class actually involve entire arm motions, not only upper limb ones. Based on the set, Table 2 shows examples of the gesture sequences and their actions for various home appliances. For each gesture sequence, its order revealed in Table 2 should be followed. As shown in these examples, each sequence for actions contains two gestures (i.e., $K = 2$). In addition, the sequences used as Personal Identification Numbers (PINs) for home appliances authentication contain three or four gestures (i.e., $K = 3$ or $K = 4$), dependent on the home appliances.

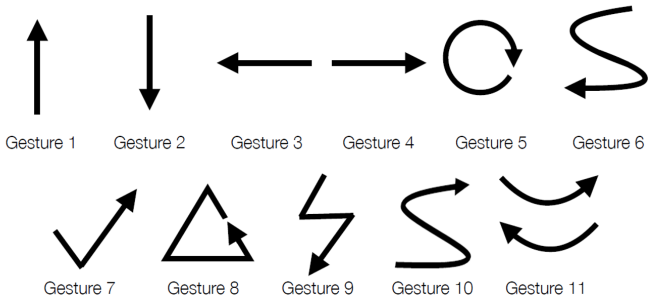

Figure 12. The eleven gesture classes considered in the smart home system in this study.

Table 2. Examples of the gesture sequences and their actions for various home appliances. Each sequence of actions contains two gestures. Each sequence has a personal identification number (PIN) for home appliances' authentication contains three or four gestures. The definition of gestures are shown in Figure 12.

	Gesture Sequences	**Actions**	**Gesture Sequences**	**Actions**
TV	Gest. 5+1	Volume Up	Gest. 5+2	Volume Down
	Gest. 5+3	Prev. Chan.	Gest. 5+4	Next Chan.
	Gest. 5+6	Power On/Off	Gest. 5+7	Record On/Off
PIN for TV	Gest. 11+ $i+j, i \neq j \neq 11$	Authentication		
Air	Gest. 8+1	Temp. Up	Gest. 8+2	Temp Down
Cond.	Gest. 8+3	Air Vol. Up	Gest. 8+4	Air Vol. Down
(AC)	Gest. 8+6	Power On/Off	Gest. 8+7	Func. Sel.
PIN for AC	Gest. 3+ $i+j+k, i \neq j \neq k \neq 3$	Authentication		
Music	Gest. 9+1	Volume Up	Gest. 9+2	Volume Down
Player	Gest. 9+3	Prev. Song	Gest. 9+4	Next Song
(MP)	Gest. 9+6	Power On/Off	Gest. 9+7	Source Sel.
PIN for MP	Gest. 11+ $i+j, i \neq j \neq 11$	Authentication		

4. Experimental Results

This section presents some experimental results for the proposed gesture recognition algorithms and systems. We implemented a gesture-based remote control system for home appliances, shown in Figure 11. The evaluation is then based on the system. There are eleven gesture classes (i.e., $Q = 11$), where each gesture class is defined in Figure 12. The gesture sequences to be classified are listed in Table 2. They have lengths of two, three, or four (i.e., $K = 2, 3$, or 4) dependent on the types of home appliances, actions, or PIN. To facilitate the evaluation, a JAVA-based APP was built on the smartphones for gesture capturing and delivery. We adopted a Samsung Galaxy S8 and HTC ONE M9 for the experiments.

The training of feedforward neural networks was carried out offline by Keras [34] with backend TensorFlow. The server for the training was a personal computer with an Intel I7 CPU and NVIDIA GTX 1070 GPU. Raspberry Pi 3 computers are adopted as the embedded systems attached to the home appliances. A python-based inference system is deployed in each embedded system. It is built from the neural network model acquired from Keras after training operations are completed. The inference system is capable of receiving the sensory data from smart phones for real-time dynamic gesture recognition.

All the gesture sequences for training and testing were captured by accelerometers and gyroscopes associated with the smart phones. The sensors were capable of measuring acceleration and angular velocity in three orthogonal axes, respectively. Therefore, the dimension of each sample x_t was $M = 6$. Figures 13 and 14 show the samples of waveforms produced by gyroscopes and accelerometers of the

smart phones for each gesture class in Figure 12, respectively. The sampling rate was 50 samples/s. Although a large number of classes was considered in our experiments, it can be observed from Figures 13 and 14 that different gesture classes have different waveforms produced from gyroscopes and/or accelerometers. The employment of the sensors would then be beneficial for the accurate classification of the gestures. Furthermore, we can also see from Figures 13 and 14 that some of the waveforms exhibit small and fast vibrations. These may be due to unstable and/or shaking hands. It would then be essential for the proposed algorithm to accurately classify the gestures with the presence of the vibrations.

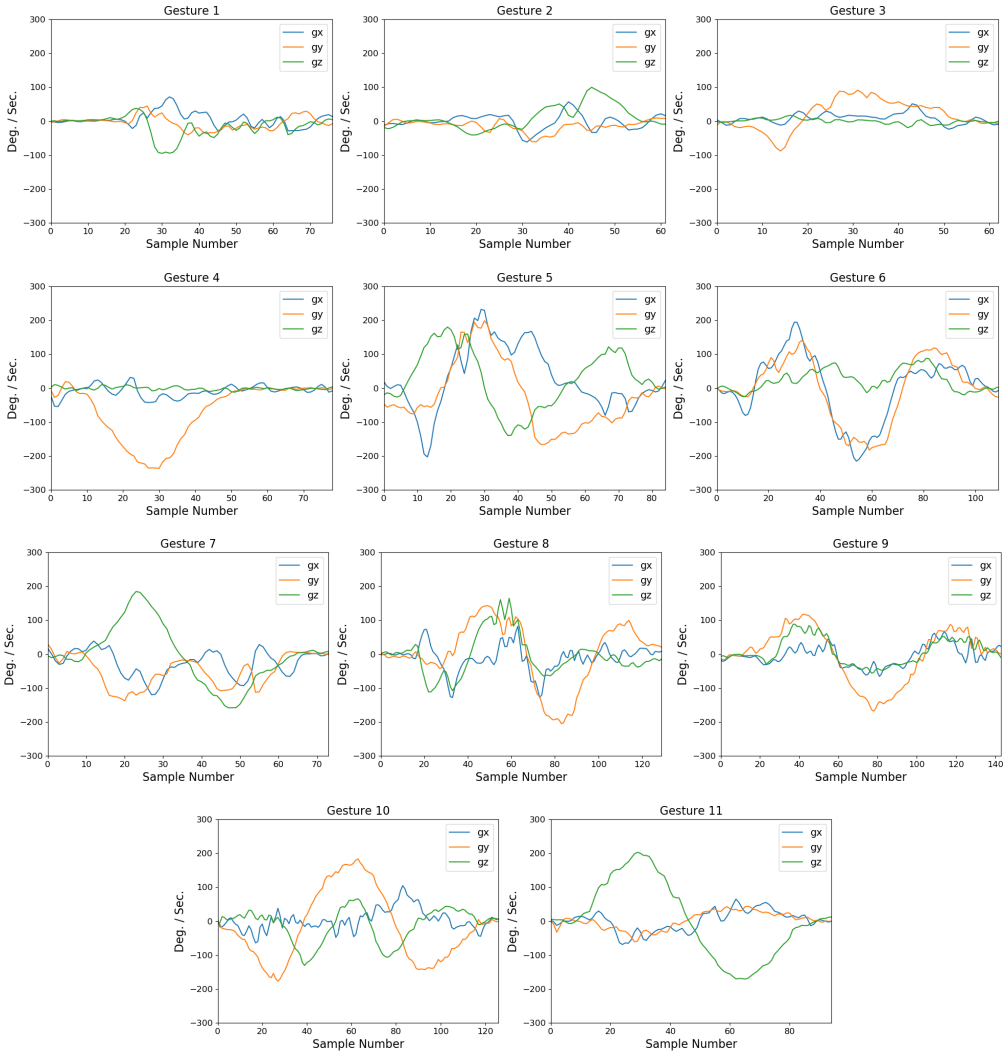

Figure 13. Samples of waveforms produced by gyroscopes in three axes (gx, gy, and gz) for each gesture.

Figure 14. Samples of waveforms produced by accelerometers in three axes (gx, gy, and gz) for each gesture.

In the experiments, each training gesture sequence represents only a single gesture (i.e., $K = 1$). There were 100 training sequences for each gesture class. Because the number of gesture classes is 11 (i.e., $Q = 11$), the training set of the experiments consisted of 1100 training gesture sequences. They were acquired from two participants. The sequences were collected on different days for capturing different variations in gesturing, such as variations in lengths of gestures and/or amplitudes of samples. The testing set was different from the training set. It contained 3404 gestures from six participants. For some test sequences, vibrations due to shaking or unstable hands were included for testing the robustness of the proposed algorithm. The initial orientation of smart phones for data acquisition of both training and testing sequences was in the portrait orientation. Each test sequence may contain two, three, or four hand gestures (i.e., $K = 2$, 3, or 4). Therefore, although the proposed neural networks were trained by sequences with $K = 1$, they can be applied to sequences with larger K values. The high reusability of the proposed models for different K values is an advantage of the proposed algorithms.

Examples of a testing sequences captured by gyroscopes consisting of three (Gesture 11, Gesture 8, and Gesture 10) and four hand gestures (Gesture 3, Gesture 7, Gesture 5, and Gesture 11) are revealed in Figures 15 and 16, respectively. The results of gesture spotting by PairNet are annotated in the bottom of the figures. For the sake of brevity, the waveforms produced by the accelerometers are not included. Small and fast vibrations on some of the waveforms can be found in the figures because of unstable

and/or shaking hands. Therefore, gesture spotting may be difficult even by direct visual inspection. Nevertheless, it can be observed from the bottom of Figure 15 that the size of sets I_{11}, I_8, and I_{10} are the largest as compared with other sets. The results are consistent with the ground truth. Furthermore, the sets I_3, I_7, I_5, and I_{11} have the largest size in Figure 16. Accurate classification is then possible based on the gesture spotting results provided by the PairNet.

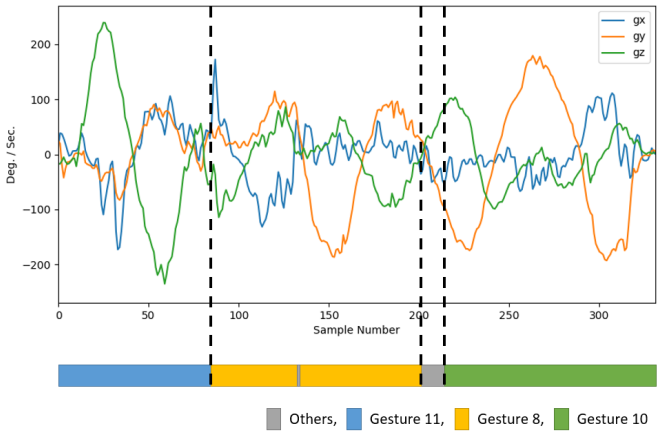

Figure 15. An example of test sequence produced by a gyroscope containing three gestures (Gesture 11, Gesture 8, and Gesture 10). The bottom of the figure reveals the gesture-spotting results by PairNet.

Figure 16. An example of test sequence produced by a gyroscope containing four gestures (Gesture 3, Gesture 7, Gesture 5, and Gesture 11). The bottom of the figure reveals the gesture-spotting results by PairNet.

To evaluate the performance of the proposed feedforward algorithms, we first consider the classification accuracy of each gesture class for the algorithms. Let H_i of an algorithm be the hit rate of gesture class i for the algorithm in the testing set. In the experiments, the hit rate H_i of an algorithm was equal to the number of gestures in class i, which are correctly classified by the algorithm divided by the total number of gestures in class i in the testing set. Tables 3 and 4 contain the model summary of the proposed algorithms and their classification accuracy, respectively. For comparison purposes, the 1D CNN [22], LSTM [8], and Bidirectional LSTM (Bi-LSTM) [19] algorithms are also considered. For each algorithm, models from 20 independent training operations were acquired. The hit rates of the 20 models of each given algorithm over the testing set were measured. The hit rates of the best model of each algorithm are reported in Table 4.

Table 3. The model summary of various algorithms.

Model	Summary
PairNet	5 Conv. layers, 1 Avg. polling layer, 1 FC layer Stride Size 1 and kernel size 1 × 3 for Conv. layer 1 Stride Size 2 and kernel size 1 × 2 for Conv. layers 2–5 Softmax output
Residual PairNet	5 Conv. layers, 1 Avg. polling layer, 1 FC layer Stride Size 1 and kernel size 1 × 3 for Conv. layer 1 Stride Size 2 and kernel size 1 × 2 for Conv. layers 2–5 2 Short-cut connections, Softmax output
PairNet with Inception	4 Conv. layers, 1 Inception layer 1 Avg. polling layer, 1 FC layer Stride Size 1 and kernel size 1 × 3 for Conv. layer 1 Stride Size 2 and kernel size 1 × 2 for Conv. layers 2–4 Softmax output
Residual PairNet with Inception	4 Conv. layers, 1 Inception layer 1 Avg. polling layer, 1 FC layer Stride Size 1 and kernel size 1 × 3 for Conv. layer 1 Stride Size 2 and kernel size 1 × 2 for Conv. layers 2–4 1 Short-cut connection, Softmax output
CNN	5 Conv. Layers, 2 Max. polling layer, 1 FC layer Stride Size 1 and kernel size 1 × 3 for Conv. layers 1–5 Softmax output
LSTM	Hidden states and mem. cells for single direction Hidden states and mem. cells dimension = 32 1 FC layer with Softmax output
Bi-LSTM	Hidden states and mem. cells for dual directions Hidden states and mem. cells dimension = 32 1 FC layer with Softmax output

We can see from Table 3 that all five feedforward algorithms (i.e., CNN, PairNet, Residual Pairnet, PairNet with inception, and Residual PairNet with inception) have five convolution/inception layers. However, it can be observed from Table 4 that the CNN has a lower classification accuracy when compared to the other four models. In particular, the average hit rates of CNN and PairNet are 90.42% and 92.36%, respectively. The PairNet outperforms the CNN in average hit rate by 1.94%. The CNN is inferior because it is based on convolution operations with a stride size of 1 for all the convolution layers. As a result, it has smaller receptive field for classification. By contrast, the convolution layers of the proposed algorithms have a stride size of 2. They may then have a larger receptive field for attaining better classification accuracy.

Table 4. The comparisons on hit rates (in percentage) of various algorithms. Both gyroscope and accelerometer are used for the corresponding implementations.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	Ave.
PairNet	70.35	90.19	72.14	86.43	99.12	97.93	98.23	98.16	94.05	100.00	99.54	92.36
Res. PairNet	81.86	96.98	82.66	97.29	99.12	97.93	98.99	96.63	92.57	100.00	99.54	95.30
PairNet w. Incep.	75.66	91.70	75.85	91.09	99.12	99.11	97.73	99.69	92.57	100.00	99.54	94.00
Res. PairNet w. Incep.	77.88	95.47	75.23	92.25	98.95	98.82	98.99	97.55	94.05	100.00	99.54	94.10
CNN [22]	69.03	74.72	70.90	85.66	99.12	97.34	96.97	96.93	92.94	99.07	99.54	90.42
LSTM [8]	70.35	72.83	65.63	78.29	98.60	97.04	89.14	93.25	79.55	99.54	99.07	86.87
Bi-LSTM [19]	70.80	88.68	73.07	81.01	97.90	97.34	87.12	90.18	84.01	97.22	99.54	88.66

In addition to larger stride sizes, it is revealed from Table 4 that the incorporation of short-cut connections and/or inception modules are also beneficial for the improvement of classification accuracy. That is, Residual Pairnet, PairNet with inception, and Residual PairNet with inception have superior hit rates over those of PairNet. In fact, the experimental results show that the average hit rates of PairNet, Residual Pairnet, PairNet with inception, and Residual PairNet with inception are 92.36%, 95.30%, 94.00%, and 94.10%, respectively. The improvement in average hit rate over the PairNet by Residual Pairnet, PairNet with inception, and Residual PairNet with inception are then 2.94%, 1.64%, and 1.74%, respectively. The improvements are due to the facts that the employment of short-cut connections may be able to provide more reliable references for training. Furthermore, the inception modules are able to effectively capture both slow and fast hand movements in a single layer.

Note that the hit rates in Table 4 are the best hit rates for the corresponding algorithms. To further assess Residual Pairnet, PairNet with inception, and Residual PairNet with inception algorithms, the statistical evaluations over the hit rates for each algorithm are included in Table 5. The evaluation includes the measurements of highest, mean, and lowest average hit rates over the 20 models associated with each algorithm. It can be observed from Table 5 that these statistical measurements for PairNet are inferior to those for Residual Pairnet, PairNet with inception, and Residual PairNet with inception algorithms. These results confirm the superiority of the proposed algorithms over their baseline PairNet counterpart.

Table 5. Statistical evaluation on the hit rates (in percentage) of various algorithms.

Algorithm	Highest Hit Rate	Mean Hit Rate	Lowest Hit Rate	Standard Deviation
PairNet	92.36	90.19	87.39	1.48×10^{-2}
Res. PairNet	95.30	93.60	91.36	1.22×10^{-2}
PairNet w. Incep.	94.00	91.37	89.62	1.54×10^{-2}
Res. PairNet w. Incp.	94.10	91.93	89.86	1.34×10^{-2}

An additional advantage of the proposed algorithms is that they outperform recurrent algorithms such as LSTM and Bi-LSTM, as shown in Table 4. The model summary of the LSTM and Bi-LSTM in our experiments can also be found in Table 3. The LSTM has inferior hit rates because of the gradient vanishing problem. The performance can be improved by bidirectional training and inference.

However, it can be observed in Table 4 that the proposed algorithms still has superior hit rates over the Bi-LSTM.

To further elaborate the effectiveness of the proposed algorithms, the confusion matrix of the Residual PairNet is revealed in Table 6. The confusion matrix reveals information about actual and predicted gesture classifications by the proposed algorithm. In the confusion matrix, the cell located at row i and column j represents the percentage in which the gesture in the row i is classified as the gesture in the column j. The hit rate H_i is then the value of the cell at row i and column i of the confusion matrix. It can be shown in Table 6 that simple gestures such as Gesture 1 and Gesture 3 have slightly lower hit rate because they may be misclassified as a part of other more complicated gestures. Nevertheless, the hit rates of these classes are still above 80%. For complicated gestures such as Gesture 5, Gesture 7, Gesture 10, and Gesture 11, the corresponding hit rates are above 99%.

Table 6. The confusion matrix of the Residual PairNet over the testing dataset. The cell located at row i and column j of the matrix represents the percentage in which the gesture in the row i is classified as the gesture in the column j.

	Gest. 1	**Gest. 2**	**Gest. 3**	**Gest. 4**	**Gest. 5**	**Gest. 6**	**Gest. 7**	**Gest. 8**	**Gest. 9**	**Gest. 10**	**Gest. 11**
Gest. 1	81.86	0.00	0.40	0.00	0.00	0.40	4.40	5.83	1.81	0.00	5.30
Gest. 2	0.00	96.98	0.00	0.38	0.00	0.77	0.79	0.00	0.00	0.00	1.08
Gest. 3	0.00	0.62	82.66	0.91	0.00	5.31	0.00	2.19	4.60	3.71	0.00
Gest. 4	0.00	0.39	0.35	97.29	0.00	0.75	0.00	0.00	1.21	0.00	0.00
Gest. 5	0.00	0.00	0.18	0.00	99.12	0.00	0.00	0.35	0.00	0.35	0.00
Gest. 6	0.00	0.00	0.29	0.00	0.00	97.93	0.00	0.00	0.89	0.89	0.00
Gest. 7	0.04	0.23	0.03	0.00	0.00	0.24	98.99	0.01	0.00	0.22	0.23
Gest. 8	0.00	0.00	0.00	0.00	0.00	2.11	0.62	96.63	0.00	0.00	0.64
Gest. 9	0.00	0.35	0.00	0.00	0.00	5.15	0.00	1.93	92.57	0.00	0.00
Gest. 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.0	0.00
Gest. 11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.46	0.00	99.54

As compared with the basic CNN, the proposed algorithms may have both the advantages of higher classification accuracy, lower storage overhead, and lower computation complexity. In this study, we define the storage overhead and computation complexity of a neural network as the parameter size and the number of multiplications for the inference operations of that network, respectively. Table 7 shows the average hit rate, parameter size, and the number of multiplications of the algorithms considered in Table 3. It can be observed from the figure that all the proposed feedforward models have higher average hit rate, lower weight size, and lower number of multiplications as compared with basic CNN. This is because all the proposed models have a common feature that the intermediate convolution layers are with kernel size 1×2 and a stride size of 2. By contrast, all the convolution layers of the CNN have kernel size 1×3 and a stride size of 1. Smaller kernel sizes adopted by the proposed networks are beneficial for lowering the parameter sizes and computation complexities. Furthermore, larger stride sizes could enhance classification accuracy.

Table 7. The comparisons on average hit rates, number of weights, and computation complexity of various neural networks. The computation complexity of a neural network is the total number of multiplications required for the inference operations of the neural network.

	PairNet	CNN [22]	LSTM [8]	Bi-LSTM [19]	Res. PairNet	PairNet w. Incep.	Res. PairNet w. Incep.
Average Hit Rate	92.36%	90.42%	86.87%	88.66%	95.30%	93.63%	94.10%
Weights Size	271,116	426,764	5,388	10,764	304,396	200,844	244,876
Number of Multiplications	2,079,488	10,064,640	256,363	506,304	6,011,648	1,988,352	2,130,176

The improvement in classification accuracy of the PairNet and its variants over basic CNN is due to the employment of both a stride size of 2 and a kernel size of 1×2. The improvement would be marginal when only a stride size of 2 is adopted, while the kernel size retains the same as 1×3. To elaborate this fact, a new CNN with a stride size of 2 is considered, where its kernel sizes are 1×3 for all the convolution layers. To achieve meaningful comparisons, the new CNN and PairNet have the same number of convolution layers, and the same number of output channels associated with each convolution layer. Furthermore, 20 models of the new CNN algorithms were trained, and the best model only achieved the average hit rate of 90.78%. By contrast, the best average hit rate of PaiNet is 92.36% . The new CNN does not perform well because the large kernel size of 1×3 introduces a large number of parameters (i.e., 400,369) so that overfitting may be likely. To provide better generalization for the training, smaller kernel size with short-cut connections and/or inception modules would be more effective.

Among the proposed feedforward algorithms, it can be observed from Table 7 that the PairNet with Inception has lowest parameter size and computation complexity. This is because the dimension reduction is carried out by front kernel in each path of the inception module shown in Figure 8. The total number of kernels can then be effectively reduced. This in turn may lower the parameter size and computation complexity. Because inception modules may be beneficial for reducing the model complexity, the Residual PairNet with inception also has lower complexity as compared with its Residual PairNet counterpart without inception, as shown in Table 7.

It can also be observed from Table 7 that the LSTM algorithm has the lowest computational complexity. Although there are four pairs of matrices used in the cell and hidden state calculations in the LSTM, the dimension of cells and hidden states is only 32. This is beneficial for maintaining small number of multiplications for the algorithm. However, the inference operations of the LSTM should be carried out in a recurrent fashion. Therefore, the reduction in computation time may not be significant as compared with the feedforward algorithms. To elaborate this fact, we have measured the computation time of LSTM, PairNet and CNN on the Raspberry Pi 3 platform, where the computation time of a network is the average CPU time for the inference of a single gesture from the testing set of that network. Based on the experiments, the computation time of LSTM, PairNet, and CNN are 102, 162, and 410 ms, respectively. The computation time of PairNet is only slightly longer than that of LSTM. This is because the inference operations of the feedforward networks can be computed in parallel. Therefore, the real-time inference may still be possible even the system is deployed in the embedded systems with Raspberry Pi 3 platform.

Finally, we evaluate the performance of the proposed algorithms when only one of the sensors is adopted. Table 8 shows the results of the experiments for the PairNet. We can see from Table 8 that the hit rates of most gesture classes are degraded without the employment of both sensors. These results show that the employment of both gyroscope and accelerometer is beneficial for hand-gesture recognition.

Table 8. The hit rates (in percentage) of the proposed PairNet algorithm with the employment of only accelerometer, gyroscope, or both.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	Ave.
Both	70.35	90.19	72.14	86.43	99.12	97.93	98.23	98.16	94.05	100.00	99.54	92.36
Gyroscope	69.03	74.72	70.90	85.66	99.12	97.34	96.97	96.93	92.94	99.07	99.54	90.42
Accelero.	70.35	72.83	65.63	78.29	98.60	97.04	89.14	93.25	79.55	99.54	99.07	86.87

5. Conclusions

The proposed feedforward algorithms have been found to be effective for dynamic hand gesture recognition. In our experiments, smart phones equipped with accelerometers and gyroscopes were adopted for the data acquisition of hand gestures. The resulting implementations can be deployed for the remote control of devices such as home appliances. It is observed from the experiments that the PairNet algorithm attains average hit rate of 92.36% for 11 gesture classes over 3404 test gestures. The hit rate of the PairNet is 1.94%, 5.49%, and 3.70% higher than those of the basic CNN, LSTM, and Bi-LSTM, respectively. Furthermore, the proposed Residual PairNet, PairNet with inception, and Residual PairNet with inception have superior performance over PairNet. In fact, the improvement in average hit rate over the PairNet by Residual Pairnet, PairNet with inception, and Residual PairNet with inception are then 2.94%, 1.64%, and 1.74%, respectively. The proposed feedforward algorithms have superior hit rate because they have a large receptive field for accurate gesture spotting. Furthermore, the algorithms also have lower weight sizes as compared with its basic CNN counterpart. In particular, the average computation time on the Raspberry Pi 3 platform for the inference of a single gesture is only 162 ms for PairNet, while the basic CNN needs 410 ms. The proposed work is therefore beneficial for human–machine interface applications where reliable continuous hand gesture recognition with real-time computation on the embedded platform is desired.

Author Contributions: Conceptualization, Y.-C.C., Y.-J.J., and T.-M.T.; methodology, Y.-C.C. and Y.-J.J.; software, Y.-C.C. and Y.-J.J.; visualization, Y.-C.C., Y.-J.J., and T.-M.T.; validation, Y.-C.C., Y.-J.J., and T.-M.T.; supervision, T.-M.T. and W.-J.H.; project administration, W.-J.H.; writing—original draft, W.-J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, under Grant MOST 107-2221-E-003-001-MY2.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

Bi-LSTM	Bidirectional Long Short Term Memory
CNN	Convolution Neural Network
CSI	Channel State Information
LSTM	Long Short Term Memory
MAP	Maximum A Posteriori
PIN	Personal Identification Number
RNN	Recurrent Neural Network
SBR	Sensor-Based Recognition
VBR	Vision-Based Recognition

References

1. Wang, C.; Liu, Z.; Chan, S.-C. Superpixel-based hand gesture recognition with kinect depth camera. *IEEE Trans. Multimed.* **2015**, *17*, 29–39. [CrossRef]
2. Bobic, V.; Tadic, P.; Kvascev, G. Hand gesture recognition using neural network based techniques. In Proceedings of the 2016 13th Symposium on Neural Networks and Applications, Belgrade, Serbia, 22–24 November 2016.
3. Zhu, G.; Zhang, L.; Shen, P.; Song, J. Multimodal gesture recognition using 3-D convolution and convolutional LSTM. *IEEE Access* **2017**, *5*, 4517–4524. [CrossRef]
4. Oyedotun, O.K.; Khashman, A. Deep Learning in Vision-Based Static Hand Gesture Recognition. *Neural Comput. Appl.* **2017**, *28*, 3941–3951. [CrossRef]
5. Xu, R.; Zhou, S.; Li, W.J. MEMS Accelerometer Based Nonspecific-User Hand Gesture Recognition. *IEEE Sens. J.* **2012**, *12*, 1166–1173. [CrossRef]
6. Xie, R.; Sun, X.; Xia, X.; Cao, J. Similarity Matching-Based Extensible Hand Gesture Recognition. *IEEE Sens. J.* **2015**, *15*, 3474–3483. [CrossRef]
7. Gupta, H.P.; Chudgar, H.S.; Mukherjee, S.; Dutta, T.; Sharma, K. A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors. *IEEE Sens. J.* **2016**, *16*, 6425–6432. [CrossRef]
8. Tai, T.M.; Jhang, Y.J.; Liao, Z.W.; Teng, K.C.; Hwang, W.J. Sensor-Based Continuous Hand Gesture Recognition by Long Short-Term Memory. *IEEE Sens. Lett.* **2018**, *2*, 6000704. [CrossRef]
9. Zhao, T.; Liu, J.; Wang, Y.; Liu, H.; Chen, Y. PPG-Based Finger-Level Gesture Recognition Leveraging Wearables. In Proceedings of the IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 1457–1465.
10. Pathak, V.; Mongia, S.; Chitranshi, G. A Framework for Hand Gesture Recognition Based on Fusion of Flex, Contact and Accelerometer Sensor. In Proceedings of the Third International Conference on Image Information Processing, Waknaghat, India, 21–24 December 2015; pp. 312–319.
11. Zhang, X.; Chen, X.; Li, Y.; Lantz, V.; Wang, K.; Yang, J. A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans* **2011**, *41*, 1064–1076. [CrossRef]
12. Asadzadeh, P.; Kulik, L.; Tanin, E. Gesture Recognition Using RFID Technology. *Pers. Ubiquit. Comput.* **2012**, *16*, 225–234. [CrossRef]
13. Zhou, Y.; Xiao, J.; Han, J.; Wu, K.; Li, Y.; Ni, L.M. GRfid: A Device-Free RFID-Based Gesture Recognition System. *IEEE Trans. Mob. Comput.* **2017**, *16*, 381–393. [CrossRef]
14. Lv, J.; Yang, W.; Gong, L.; Man, D.; Du, X. Robust WLAN-based indoor fine-grained intrusion detection. In Proceedings of the 2016 IEEE Global Communications Conference, Washington, DC, USA, 4–8 December 2016; pp. 1–6.
15. Chen, Z.; Zhang, L.; Jiang, C.; Cao, Z.; Cui, W. WiFi CSI Based Passive Human Activity Recognition Using Attention Based BLSTM. *IEEE Trans. Mob. Comput.* **2019**, *18*, 2714–2724. [CrossRef]
16. Ding, J.; Wang, Y. WiFi CSI-Based Human Activity Recognition Using Deep Recurrent Neural Network. *IEEE Access* **2019**, *7*, 174257–174269. [CrossRef]
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
18. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
19. Lefebvre, G.; Berlemont, S.; Mamalet, F.; Garcia, C. Inertial Gesture Recognition with BLSTM-RNN. In *Artificial Neural Networks, Springer Series in Bio-/Neuroinformatics*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 4, pp. 393–410.
20. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
21. Li, D.; Zhang, J.; Zhang, Q.; Wei, X. Classification of ECG Signals Based on 1D Convolution Neural Network. In Proceedings of the 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), Dalian, China, 12–15 October 2017.

22. Lee, S.M.; Yoon, S.M.; Cho, H. Human Activity Recognition From Accelerometer Data Using Convolutional Neural Network. In Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, Korea, 13–16 February 2017; pp. 131–134.
23. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.
24. Van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. Wavenet: A Generative Model for Raw Audio. *arXiv* **2016**, arXiv:1609.03499.
25. Engel, J.; Resnick, C.; Roberts, A.; Dieleman, S.; Norouzi, M.; Eck, D.; Simonyan, K. Neural audio synthesis of musical notes with WaveNet autoencoders. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
26. Jhang, Y.J.; Chu, Y.C.; Tai, T.M.; Hwang, W.J.; Cheng, P.W.; Lee, C.K. Sensor-Based Dynamic Hand Gesture Recognition by PairNet. In Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 14–17 July 2019; pp. 994–1001.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. *arXiv* **2016**, arXiv:1603.05027.
29. Wu, Z.; Shen, C.; van den Hengel, A. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *Pattern Recognit.* **2019**, *90*, 119–133. [CrossRef]
30. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.
31. Szegedy, C.; Ioffe, S.; Vanhoucke, V. Inception-v4, Inception-Resnet and the impact of residual connections on learning. *arXiv* **2016**, arXiv:1602.07261.
32. Sun, Y.; Liang, D.; Wang, X.; Tang, X. DeepID3: Face Recognition with Very Deep Neural Networks. *arXiv* **2015**, arXiv:1502.00873.
33. Chua, C.-S.; Guan, H.; Ho, Y.-K. Model-Based 3D Hand Posture Estimation form a Single 2D Image. *Image Vis. Comput.* **2002**, *20*, 191–202. [CrossRef]
34. Chollet, F. Keras. Available online: http://github.com/fchollet/keras (accessed on 3 August 2020).